



HAL
open science

Design of novel visual representations and tools applied to plant pangenome visualization

Éloi Durant

► **To cite this version:**

Éloi Durant. Design of novel visual representations and tools applied to plant pangenome visualization. Genetics. Université de montpellier, 2022. English. NNT : . tel-03959992

HAL Id: tel-03959992

<https://hal.science/tel-03959992>

Submitted on 27 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En Génétique et Génomique

École doctorale GAIA

Unité de recherche DIADE

Design of novel visual representations and tools applied to
plant pangenome visualization

Présentée par Éloi DURANT
Le 29 Septembre 2022

Sous la direction de François SABOT

Devant le jury composé de

Tobias ISENBERG, Directeur de Recherche, INRIA, Université Paris-Saclay, Orsay, France

Kay NIESELT, Full Professor, IBMI, University of Tübingen, Tübingen, Allemagne

Anne-Françoise ADAM-BLONDON, Directrice de Recherche, INRAE, Versailles, France

Matthieu CONTE, Chef d'équipe, SYNGENTA SEEDS, Saint-Sauveur, France

Mohammad GHONIEM, Directeur de Recherche, LIST, Esch-sur-Alzette, Luxembourg

Eugénie HEBRARD, Directrice de Recherche, IRD, Montpellier, France

Éric RIVALS, Directeur de Recherche, LIRMM, CNRS, Montpellier, France

François SABOT, Directeur de Recherche, IRD, Montpellier, France

Mathieu Rouard, Chef d'équipe, BIOVERSITY INTERNATIONAL, Montpellier, France

Rapporteur

Rapporteuse

Examinatrice

Co-encadrant de thèse

Examinateur

Examinatrice, Présidente du jury

Examinateur

Directeur de thèse

Co-encadrant de thèse, Invité



UNIVERSITÉ
DE MONTPELLIER

Résumé en Français

La démocratisation des technologies de séquençage lors des vingt dernières années a entraîné une explosion du nombre de génomes séquencés. Cette évolution a permis l'apparition de plusieurs génomes de bonne qualité, multipliant le nombre de génomes de références disponibles au point d'en avoir plusieurs pour une même espèce.

La diversité des génomes de référence disponible a mis en évidence les biais induits par l'utilisation d'une unique référence, qui n'est pas suffisante pour donner accès à la diversité au sein d'une espèce. Des efforts de séquençage de génomes humains d'origine africaine¹ ont par exemple mis en évidence les manques de l'actuel génome de référence GRCh38, principalement construit à partir d'un seul individu. Ce manque de diversité se traduit par une grande quantité de fragments d'ADN humain manquant dans la référence, biaisée en faveur de génomes d'origine européenne. La sous-représentation d'autres populations pose un problème, par exemple pour l'identification de maladies spécifiques ou la compréhension de la réponse métabolique à l'administration de différents traitements².

Chez les plantes, la diversité génétique des espèces cultivées ainsi que la taille des génomes sont bien plus importantes que chez l'humain. De plus, de nombreux exemples de variations intraspécifiques ont été recensés, notamment de la variation en présence / absence ou en nombre de copies de gènes. Ces variations peuvent exercer une forte influence sur le phénotype des plantes, par exemple chez le riz où la présence d'un gène *Sub1A* est associée à une tolérance à l'inondation.

Pour une meilleure intégration de ces variations en génomique, le concept de pangénome s'est progressivement développé. Un pangénome est un concept qui regroupe l'information de plusieurs génomes en une seule entité, pour faciliter le stockage d'information, la comparaison entre individus, ou encore les étapes d'assemblage et d'alignement pour de nouveaux génomes. Un pangénome peut être construit aussi bien pour recenser des gènes (comme c'est le cas pour des études sur bactéries) que pour tous types de fragments génomiques (séquences d'ADN ou protéiques), et est utile pour comparer la répartition de ces éléments entre plusieurs individus. Plusieurs catégories d'éléments existent selon le taux de présence, les deux principales recensent les éléments présents chez tous les individus (les éléments 'core') et ceux présents seulement chez certains (les éléments 'variable').

Étant un concept encore relativement récent—apparu en 2005, son application aux plantes date des années 2010—la pangénomique souffre encore d'un manque d'outils, aussi bien pour la création de pangénomes que pour leur analyse, et d'autant plus leur visualisation. Ce manque est particulièrement vrai pour les eucaryotes (dont les plantes), aux génomes plus gros et complexes que les bactéries (premier domaine d'application des pangénomes) et dont les outils ne supportent facilement pas le passage à l'échelle.

Mes travaux de thèse ont donc porté sur la création de nouvelles représentations visuelles ainsi que la création d'outils de visualisation utilisables pour la visualisation de pangénomes de plantes, et d'eucaryotes en général. Ces travaux s'inscrivent dans le cadre de la thèse Cifre n°2018/1475 en collaboration entre trois organismes : Syngenta France (partenaire privé), l'Institut de Recherche pour le Développement (IRD, un Établissement Public à caractère Scientifique et Technologique), et l'Alliance Bioversity International-CIAT (organisation internationale de recherche). Ils font suite à des travaux de Master 2 ayant mis en évidence le manque d'outils utilisable pour visualiser des pangénomes de plantes. Les codes produits dans le cadre de cette

¹ Sherman, R.M., Forman, J., Antonescu, V. *et al.* Assembly of a pan-genome from deep sequencing of 910 humans of African descent. *Nat Genet* **51**, 30–35 (2019). <https://doi.org/10.1038/s41588-018-0273-y>

² Popejoy, A., Fullerton, S. Genomics is failing on diversity. *Nature* **538**, 161–164 (2016). <https://doi.org/10.1038/538161a>

thèse Cifre ainsi que les outils en résultant sont disponibles en Open Source sur la plate-forme GitHub.

Dans ce manuscrit de thèse, je présente l'état de l'art de la pangénomique, introduisant les différents concepts et formats de données associés. J'y fais la distinction entre la notion de pan-gene atlas et de pangénome, le premier étant un inventaire de gènes présents ou absents et le second étant l'ensemble du matériel génomique partagé entre génomes, étendu à tous fragments de séquence et leurs successions. Ces pangénomes peuvent être représentés par des graphes de séquences, où chaque nœud est une séquence génomique et les liens sont autant de successions de séquences observées dans différents génomes. Les graphes de pangénomes contiennent l'information originelle de la succession des séquences en représentant chaque génome par un chemin au sein du graphe.

Je développe également une catégorisation des outils de visualisation utilisables en pangénomique. J'identifie ainsi des outils non spécifiques (génériques, non spécifiques des pangénomes), qualitatifs (donnant des informations générales sur certaines propriétés des pangénomes, principalement concentrés sur les pangénomes de gènes), positionnés (utilisant un système de coordonnées pour positionner les éléments pangénomiques, soit sur un génome existant soit sur une panréférence), structuraux (mettant l'accent sur la succession des séquences d'ADN dans les différents génomes), et enfin composites (combinant plusieurs propriétés des catégories précédemment listées, offrant plusieurs représentations et/ou approches pour l'analyse). Je mets en évidence le manque d'outils interactifs utilisables pour une analyse poussée de pangénomes.

J'y présente également quelques principes utilisés en datavisualisation pour les lecteurs qui ne seraient pas familiers de cette notion. Je reviens principalement sur l'importance de visualiser ses données et sur les principes de Gestalt qui régissent notre façon instinctive de grouper des éléments visuels.

Mon premier chapitre recueille dix conseils pour créer un outil de visualisation de données génomiques, à l'attention de futurs chercheurs et chercheuses en biologie ou bio-informatique qui s'intéresseraient à la data visualisation. Ces conseils sont issus de mon expérience personnelle acquise lors de cette thèse partagée et confrontée à celle d'autres chercheurs ayant travaillé sur la création de logiciels pour l'analyse de données génomiques et avec qui j'ai eu l'opportunité de collaborer.

Ces « règles », soumises sous la forme d'un article « *Ten simple rules for...* » en révision au journal PLOS One, indiquent dix bonnes pratiques à suivre tout au long du développement d'un outil de visualisation en génomique : du design des représentations visuelles jusqu'à leur implémentation dans un outil utilisable, incluant également la question du maintien et de la vie sur la durée de tels outils. Ces dix règles ont été éprouvées à plusieurs reprises lors de ma thèse, et je les utilise comme prisme pour les conclusions des deux chapitres suivants ainsi que pour la conclusion générale.

Dans mon second chapitre, je décris mon premier outil de visualisation de pangénome, publié dans le journal Bioinformatics sous le titre « *Panache : a Web Browser-Based Viewer for Linearized Pangenomes* ». J'y explique comment j'ai imaginé la représentation visuelle utilisée dans Panache, jusqu'à la création d'une application web développée en JavaScript permettant l'exploration dynamique de données pangénomiques.

Panache (Pangenome Analyzer with Chromosomal Exploration) se présente sous la forme d'un navigateur pour pangénomes, incluant une vue principale affichant une 'heatmap' de la présence / absence de blocs pangénomiques (en colonnes) entre différents génomes (en ligne). Cette heatmap est accompagnée de lignes d'information supplémentaires, donnant des détails sur la position de ces blocs au sein du système de coordonnées linéaires utilisé, leur taille, leur taux de présence et catégorisation entre **core** et **variable**, ou encore la répartition de leur potentielles autres occurrences au sein du pangénome.

Cet outil dispose de plusieurs fonctionnalités implémentées à la suite de sessions d'échanges avec des utilisateurs. Il dispose par exemple d'un algorithme permettant de déplacer automatiquement la vue sur des zones d'absence successive de blocs de pangénome, d'une représentation des annotations de gènes associées au système linéaire de coordonnées, ou encore de multiples options de tri des génomes. Ces options de tris ont été implémentées dans l'outil dans le cadre d'un stage d'IUT que j'ai supervisé, et permettent de réarranger visuellement les lignes de génomes selon différents critères. Une personne utilisant Panache pourrait ainsi par exemple choisir d'afficher les génomes selon leur ordre dans une phylogénie préétablie ou encore selon un algorithme de groupement hiérarchique appliqué à une sous-région.

Panache a été utilisé en interne pour la représentation de données de pangénome du bananier, disponible sur le Banana Genome Hub de la plateforme de bioinformatique SouthGreen³. Il a aussi été adopté par d'autres équipes de recherche, donnant lieu à une publication sur le pangénome du blé (« *Wheat Panache : a pangenome graph database representing presence-absence variations across sixteen breadwheat genomes* ») et de futures utilisations pour l'orge.

Le troisième et dernier chapitre détaille le travail de conception d'un outil composite de visualisation de pangénomes, appelé SaVanache, permettant la navigation entre plusieurs niveaux d'échelle pangénomique. Cet outil sera divisé en plusieurs vues entre lesquelles un utilisateur ou une utilisatrice pourra naviguer, correspondant à différents niveaux de zoom entre les différentes échelles. Quatre vues ont été identifiées : une vue de la diversité globale de génomes assemblés, illustrant les génomes les plus similaires ou distincts ; une vue des réarrangements chromosomiques et variations structurales entre plusieurs des génomes d'un pangénome ; une vue dédiée à la variation en présence / absence de gènes ou blocs pangénomiques sur une échelle plus restreinte ; et une dernière vue dédiée aux variations nucléotidiques et à l'identification d'haplotypes.

Mon travail s'est focalisé principalement sur les deux premières vues et la conception des représentations visuelles associées. La troisième vue correspond au travail déjà réalisé pour Panache, qui pourrait y être intégré sous couvert de quelques ajustements logistiques. La dernière vue fut laissée de côté car plusieurs groupes de recherche ont déjà proposé différents outils pour la visualisation et comparaison de plusieurs génomes à l'échelle des nucléotides, par exemple avec la représentation d'alignement de séquences multiples.

Je propose une première vue dynamique, représentant les génomes assemblés dans un graphique en nuage de points et permettant de choisir un pangénome composé d'un sous-ensemble spécifique de génomes, permettant de réduire la taille des données à manipuler. Deux systèmes de colorations y coexistent. Le premier permet d'identifier les génomes partageant les mêmes métadonnées qu'un autre génome (survolé avec la souris). Le deuxième est utilisé pour mettre en avant les génomes qui seront inclus dans les vues suivantes de l'outil. Ce système proposé permet de rapidement identifier les génomes, potentiellement d'intérêt, absent d'un pangénome, afin de choisir au mieux le pangénome à conserver dans le reste de l'analyse. Un encodage coloré différencie les génomes qui sont absents dans le pangénome survolé de ce qui y sont présents, avec une surcouche d'information selon si ces génomes sont marqués comme étant important pour la personne utilisatrice ou si leur statut de présence est différent entre deux pangénomes disponibles.

La seconde vue est dédiée à la visualisation de variations structurales entre génomes. Je propose une nouvelle approche pour l'annotation de variations structurelles au sein d'un graphe de pangénome.

³ <https://banana-genome-hub.southgreen.fr/content/panache>; comme présenté dans l'article de Droc G, Martin G, Guignon V, Summo M, Sempere G, Durant E, Breton C, Cenci A, Baurens FC, Shah T, Aury JM, Ge XJ, Helsen Harrison P, Yahiaoui N, D'Hont A, Rouard M. *The Banana Genome Hub: a community database for genomics in the Musaceae*. Soumis à Horticulture research.

Je propose d'axer cette vue autour d'un chemin pivot, c'est-à-dire une certaine succession de nœuds au sein du graphe, servant de système de coordonnées principal pour toutes les comparaisons. Ces comparaisons d'un-contre-plusieurs génomes peuvent être visualisées dans une matrice représentant chaque variation structurale par un glyphe, ou une combinaison de glyphes dédiés. Comme chaque divergence entre deux nœuds d'un graphe de pangénome peut être caractérisée selon les différences de trajet des chemins la traversant, j'identifie différents types de variations. En effet, deux chemins au sein d'un graphe de pangénome peuvent différer à cause de : une insertion ou délétion (l'un des chemins contient des nœuds que l'autre n'a pas), une inversion ou chaîne d'inversion (l'un des chemins possède le même nœud ou la même succession de nœud que l'autre, mais dans leur sens complémentaire), un échange (les deux chemins contiennent des nœuds différents avant de se réunir). En complément, je présente le principe de cooccurrences d'un nœud, qui sont autant de copies de la séquence associée que nécessaire, présentes dans le graph selon le nombre de répétitions ou de positions différentes entre les génomes contenus. Ces cooccurrences, combinées à l'information de présence / absence, peuvent être catégorisées en translocations ou duplications. Chaque catégorie identifiée peut ainsi être symbolisée visuellement par un glyphe, variable en forme, position en couleur selon la catégorie.

Ces glyphes peuvent être disposés dans une matrice où chaque ligne est une comparaison du pivot avec un autre génome, et chaque colonne un nœud du chemin pivot. Une surcouverte d'interaction avec l'interface permet de grouper plusieurs comparaisons en une ligne résumée, ou d'afficher des diagrammes en barres pour comptabiliser la quantité observée de variations structurales par ligne de comparaison.

De plus cette représentation visuelle peut être complétée par une visualisation du détail des variations structurales pour un nœud donné du pivot, illustrant la taille des segments de séquence impliqués. Une vue générale inspirée des représentations Circos et 'violin plots' affichant les cooccurrences existantes entre tous les chromosomes d'un pangénome est également disponible. L'interface de cette vue permet de filtrer les variations structurales sur une région donnée, selon leur type, ou selon leur taille, pour affiner l'affichage en fonction de paramètres entrés par un l'utilisateur ou utilisatrice.

Cette vue pourrait ensuite rediriger sur une vue dédiée aux variations de présence absence ou de nombre de copies pour des fragments pangénomique de plus petite échelle.

Je conclus cette thèse en analysant l'apport de ces nouvelles représentations visuelles et les manques encore à combler pour la visualisation de pangénomes. J'ouvre la discussion sur l'intérêt d'autres disciplines pour la création d'outils de visualisation plus faciles à prendre en main pour des biologistes.

Acknowledgements

I would first like to address my sincere thanks to all the people and colleagues who helped me during my PhD, either through collaborative work or simple discussion.

Thank you to all the jury members who accepted to read this dissertation for their time and attention, I am eager to discuss it with them on the day of my defense and on future occasions.

I would like to acknowledge both Theresa Harbig and Simon Heumos for the discussions we had, speaking to other PhD students about (pangenome) datavisualization was a relief and provided some fresh air during these times of isolation. I wish we can meet properly someday.

This work would not have been the same without the help of all the developers and contractors that worked with me: Joffrey Gallais, Romain Basset, Mel Florance, Alexandre Bousquet, Laure Lebon, Allan Bertide and Chris Simpson. Thank you all for your motivation and patience while enduring my debut as a manager, and for the fruitful exchanges we had.

Thank you to all my colleagues from Bioversity, Syngenta and IRD, for their presence and naive points-of-view which often helped me during my work. Thank you to all the team at Bioversity International for welcoming me during my MSc's internship, in particular my office-mates Alberto, Cathy, and Valentin.

Thank you very much to all people at Syngenta Saint-Sauveur for welcoming me the few times I have been able to come and for carsharing. I will have my license someday, I promise. Thank you especially to the extended bioinformatics team both in France and in the US: Abdel, Aïcha, Allan, Andrew, Cécile, Cédric, Charles, Chris, Eric, Guillaume, Josh, Laurence, Steve, Véronique and all with whom I did not have as many opportunities to discuss. A special thanks to Cécile and Cédric for giving me insightful headaches and bearing with my trial-and-error progression.

Again, thank you, to everyone who interacted with me in some way at the CIRAD and especially IRD where I spent most of my time during my PhD, when I was not quarantined at home. Thank you to all members of Southgreen whom I interacted with, especially Gaëtan, there from the very beginning of this project and whose help was crucial in deploying Panache to the Genome Hub. I also am thankful to the teams RICE and DYNADIV (there are too many of you to put all of your names, but you know who you are) which hosted me through the years, and the people from the plateau I-Trop (Aurore, Julie, Ndomassi, Valérie & Co.), it was nice to have some geeks around. A special thought goes to all who shared my office, in particular Emma, Nguyet, Pierre, and Sonia who bore with me during the final year, it was a pleasure sharing my office with you.

Moreover, I could not thank enough all three of my supervisors who helped me during these three (four?) years, both as scientists and as people. I am grateful for the time and efforts they spent with me on this work and all the fruitful discussions that we made happen despite their chaotic time schedules. I am especially happy that we could work together on the directions taken by this project despite different objectives and backgrounds. Many thanks for working together with me for so long without fighting, and for getting me through these years of PhD and COVID.

Matthieu, thank you for including me within your team at Syngenta and for fighting so that the tools could be Open Source, for funding the developers who worked with me and for making your best to ensure that this PhD could happen in the best conditions. It was an awesome opportunity and I have never felt limited on that end. It always is a pleasure to discuss with you, and I will always be available for some beers after the work hours.

Mathieu, thank you so very much for supporting me from the very beginning, this PhD would not have been possible without you and the time you take to convince me. I am glad you did it. You helped me to grow more confident throughout the internship and PhD and was always there when I needed help. I am grateful for all you did, your patience, your empathy, and many useful advice. I am genuinely happy to have met you as both my supervisor and a person.

François, you once told me at the beginning of the PhD that we might seriously fight at some point and that it would be normal, I truly hope that you are not waiting until the defense for this. Thank you too for embarking me in this adventure, for the time you took to answer my naïve questions, for the experiences at JOBIM, for reassuring me on my skills, for showing me that research could be fun, for so many things and for the graphic tablet. A wonderful investment!

I wish to dedicate a special thanks to Christine, Clothilde and Nguyet, my favorite co-minions, whose communicative craziness helped me to stay sane during these years. I had wonderful times thanks to you, and it was nice to have some people around to let off steam.

I finally wish to thank all the friends I have made in the meantime and those who supported me from the beginning.

Faustine, Camille, you made my time at BioComp a bazillion time better. I will hold you accountable for making a bioinformatician out of me until the end of times. From the ENSAT still, I wish to thank Camchou, Etienne and Nina for helping me to have a social life in Montpellier.

Thank you to Aliénor, Aurélien, Charlène, Charlotte, Dimitri, Georges, Martin, Maxime, Ninon, Solène and Soline, who truly welcomed me in Montpellier when I arrived and gave me wonderful times. Special shoutout to Aurélien, Martin and Maxime, for I never played Ascension.

I wish to acknowledge all people involved in all extra-work activities I had, especially from the choir Ecume and its after-rehearsal drinks, and the MAO sessions. In particular, thank you to Antoine D., Antoine H., Claire, Jules, Kévin, Lisa, Louise, Nathalie, Saele, Sébastien D., Sébastien P., and all the tenors. A special thanks the crew at the Castors which opened shortly before my PhD, best boardgame bar ever!

Thank you too, to all IRD youngsters whom I had the pleasure to encounter post-quarantine. Cécile, Franca, Laura, Marine, Margot, Pablo, Patrick, Rémi, Stella, Thomas, Tran, Vincent et al.: thanks for the boardgames, bouldering, cakes, coffee, discussions, and karaoke sessions!

Last, but not least: Anaïs, I hope that you are proud of your Singing Meerkat. Camchou, how can you be so consistently lovable? Daniela: I hope I watered the plants enough. Marc: thanks for being you and for being there. Ninon: guess who has time for Armello now!

Table of Contents

RÉSUMÉ EN FRANÇAIS	II
ACKNOWLEDGEMENTS	VI
TABLE OF CONTENTS	VIII
TABLE OF FIGURES.....	XII
TABLE OF TABLES	XV
TABLE OF ABBREVIATIONS	XVI
INTRODUCTION.....	1
STATE OF THE ART	5
I. PANGENOMES.....	6
A. Context, interest & applications	6
B. Multiple definitions.....	7
1. Pan-Gene Atlas.....	8
2. (Structural) Pangenome	8
C. Significant properties.....	8
D. Challenges specific to plant pangenomes.....	10
E. Desirable features.....	11
F. Status of the pangenomic landscape.....	11
1. Lack of standard approaches.....	12
a. Different concepts.....	12
b. Different building strategies.....	16
2. ...hence a lack of standard tools.....	16
a. Tools creating pangenomes	16
b. Tools using and manipulating pangenomes	17
3. ...and a lack of standard formats	17
a. Generic formats.....	17
b. Specialized formats	18
c. Pangenome Graph formats	19
II. OVERVIEW OF TOOLS WITH PANGENOME VISUALIZATION	20
A. Visual representations, visualization tools	20
B. List of tools usable for visualizing pangenomes.....	21
1. Unspecific.....	23
a. Cytoscape	23
b. Gephi	24
c. neo4j	24
d. UpSet.....	24
2. Qualifying	25
a. anvio.....	25
b. Coinfinder.....	26
c. Harvest	26
d. Hierarchical Sets	26
e. Microreact.....	26
f. PanGeT	26
g. Pan-Tetris	26
h. PanViz.....	26
i. panX	27
j. Phandango	27
3. Positioned	28
a. Chromatiblock	28
b. GCV – Genome Context Viewer.....	29
c. PanACEA.....	29
d. Panache	29
e. PGV.....	30
f. TASUKE+	30
4. Structural.....	30

a.	Bandage.....	31
b.	Crop-Haplotypes.....	32
c.	GenomeRing.....	33
d.	gfaestus.....	33
e.	GfaViz.....	33
f.	graphgenomeviewer.....	33
g.	IGGE -The Immersive Graph Genome Explorer.....	33
h.	ODGI.....	33
i.	Panaconda.....	35
j.	panGraphViewer.....	35
k.	pantograph.....	35
l.	plotsr.....	36
m.	Sequence Tube Map.....	36
n.	vg toolkit.....	37
5.	Composite.....	37
a.	MoMI-G.....	38
b.	PGAP-X.....	39
C.	<i>Platforms and (repurposed) genome browsers</i>	39
D.	<i>Tools not included</i>	39
III.	INTRODUCTION TO DATA VISUALIZATION.....	40
A.	<i>Why do datavis?</i>	40
B.	<i>One concept, many visual representations</i>	41
C.	<i>Some principles of data visualization and UI design</i>	43
1.	Tufte's data-to-ink ratio and chartjunk.....	43
2.	The principles of Gestalt.....	43
3.	How datavis lies.....	44
4.	The principle of least astonishment.....	45
5.	Shape distinguishability.....	45
D.	<i>What are design studies?</i>	46
	TEN RULES ON GENOMIC VISUALIZATION TOOL DEVELOPMENT.....	49
I.	MOTIVATION.....	50
II.	TEN SIMPLE RULES FOR DEVELOPING VISUALIZATION TOOLS IN GENOMICS.....	52
A.	<i>Introduction</i>	52
B.	<i>The Rules</i>	53
1.	Rule 1: Articulate the need for new visualization tools.....	53
2.	Rule 2: Involve others early on.....	53
3.	Rule 3: Think about visual scalability and resolution.....	54
4.	Rule 4: Be creative, be bold.....	55
5.	Rule 5: Make data complexity intelligible.....	55
6.	Rule 6: Let your inner nerd shine, when needed.....	56
7.	Rule 7: Benchmark with diverse datasets.....	57
8.	Rule 8: Stay tuned to the genomic tool ecosystem and promote interoperability.....	57
9.	Rule 9: Keep up to date with related work.....	58
10.	Rule 10: Grow and support your user community.....	58
C.	<i>Conclusion</i>	59
D.	<i>References</i>	59
E.	<i>Supporting information</i>	63
III.	AUTHORS' CONTRIBUTIONS.....	64
IV.	ADDENDUM.....	64
	PANACHE.....	67
I.	WHY CREATE A NEW PANGENOME VISUALIZATION TOOL.....	68
A.	<i>Context</i>	68
B.	<i>Tools benchmarked</i>	68
C.	<i>Identified gaps</i>	69
II.	DESIGN OF THE VISUAL REPRESENTATION.....	70
A.	<i>Panache, a PAV browser</i>	70
1.	Visual representation of a PAV matrix and tracks of information.....	70
2.	View dedicated to conserved blocks.....	72

3.	Visual representation of gene annotations	72
4.	Hollow Areas	73
B.	<i>Identification of desirable features</i>	73
1.	Face to face discussions	73
2.	Public survey	74
III.	DEVELOPMENT OF A WEB-BASED TOOL	74
A.	<i>Techniques</i>	74
1.	Data files and formats	74
a.	PAV matrix.....	74
b.	Others.....	75
2.	Technical choices	75
a.	Programming language and libraries.....	75
b.	Choosing a JS framework	75
c.	Deployment.....	75
B.	<i>Implementation</i>	76
1.	File parsing and companion script.....	76
2.	Filtering Objects to what can be seen	76
3.	Conception of the geometric zoom.....	76
4.	Canvas and SVG for the miniature	76
5.	Hollow Area Finder.....	76
6.	Sorting options.....	77
7.	Performance evaluation.....	77
8.	Identified missing functionalities	77
IV.	PUBLISHED APPLICATION NOTE.....	79
V.	DISCUSSION	82
A.	<i>Outside reach</i>	82
B.	<i>Envisioned improvements</i>	82
C.	<i>Application of the Ten rules</i>	83
SAVANACHE	85
I.	STRUCTURAL VARIATIONS AND PANGENOMES.....	86
A.	<i>SV nomenclature</i>	86
B.	<i>SV visual representations</i>	86
C.	<i>SV within genome graphs</i>	88
II.	SAVANACHE'S DESIGN	89
A.	<i>Multiple scales</i>	89
1.	Overall diversity	90
2.	Structural variations.....	90
3.	Presence Absence	90
4.	Haplotypes	90
B.	<i>SaVanache's UI</i>	91
C.	<i>View 1 – Overall diversity</i>	91
1.	From existing PCA representation.....	91
2.	...to SaVanache's redesign	92
a.	Identification of interesting assemblies	93
b.	Selection of identified assemblies	94
c.	Choice of a pangenome including the selected assemblies	94
d.	Matching pangenome barchart.....	96
D.	<i>View 2.1 – Visualization of SVs intra pangenome</i>	96
1.	Division into panchromosomes.....	96
2.	PanCircos.....	98
3.	One-versus-all tabular representation of a PanCircos	99
a.	The main panchromosome.....	99
b.	The comparative table of panchromosomes.....	100
c.	Preliminary implementation and abandon	101
E.	<i>View 2.2 – Visualization of SVs inter genomes</i>	102
1.	PanCircos as an optional overview.....	103
2.	Selection of a 'pivot' genome.....	103
3.	Glyphs for representing SVs in pangenome graphs	104
a.	Proposed SV nomenclature in a pangenome graph	104
b.	A glyph system for the visual representation of SVs	105

c.	Implementation of the glyph system	107
d.	Feedback on the glyphs	109
4.	Dedicated UI	109
5.	Alternative representation of glyphs for groups	109
6.	Summary of SVs within the selected region	110
7.	Details on SVs from a selected intersection	111
III.	SAVANACHE'S IMPLEMENTATION	112
A.	<i>Building a composite visualization tool</i>	112
1.	Development strategy	112
2.	Chosen technologies	113
3.	Overall diversity view	113
4.	Integration within a SPA and <i>Structural variations</i> view	113
a.	Special format of pangenome graph file	113
b.	Annotating SVs from the JSON file	114
c.	Storage of the SV annotations	115
d.	Implementation done	115
B.	<i>Future developments</i>	115
IV.	DISCUSSION	116
A.	<i>Design validation and extension</i>	116
B.	<i>Development of an integrated solution</i>	116
C.	<i>Feedback and general discussion</i>	117
D.	<i>Application of the 10 rules</i>	118
	CONCLUSION	119
I.	SUMMARY	120
II.	AN ANSWER	120
III.	DISCUSSION	121
IV.	FOOD FOR THOUGHTS	122
V.	PERSPECTIVES	122
VI.	PERSONAL CONCLUSION	123
	REFERENCES	125
	TABLE OF APPENDICES	137
	APPENDIX	141
I.	APPENDICES FROM THE STATE OF THE ART	142
II.	APPENDICES FROM PANACHE'S CHAPTER	170
III.	APPENDICES FROM SAVANACHE'S CHAPTER	229
	RÉSUMÉ	252
	ABSTRACT	252

Table of Figures

Figure 1: "Pangenome" took over "Pan-genome" in early 2022.....	6
Figure 2: Different sets of genomes share different genomic elements.....	9
Figure 3: The more exhaustive a pangenome is, the closer to a "closed" pangenome it becomes	10
Figure 4: Pan-gene atlases can be described as a listing of the presence statuses of genes in multiple genomes.....	12
Figure 5: With position-dependent twins, categorizing panBlocks can be difficult.....	13
Figure 6: Graphs are mainly composed of Nodes and Edges.....	14
Figure 7: Directed Acyclic Graphs duplicate the nodes present at multiple positions	15
Figure 8: colored de Bruijn graphs encode the origin of nodes through colors	15
Figure 9: I identified five categories of pangenome visualization tools.....	22
Figure 10: UpSet offers an alternative to Venn diagrams by displaying barcharts and summary statistics for each set intersection.....	24
Figure 11: panX provides multiple visual representations detailing the core genes	27
Figure 12: PanACEA's pan-chromosome view displays genes positioned on a circular bacterial coordinate system.....	29
Figure 13: Bandage gives details on the GFA's node succession, their ID, and size in bp.....	32
Figure 14: <i>odgi viz</i> draws genome graphs with ordered nodes on top and edges below	34
Figure 15: The <i>odgi viz</i> representation of raw graphs can be messy.....	35
Figure 16: pantograph represents all genomes within a pangenome as linear paths through panBlocks represented as PAV matrices at a nucleotide level.....	36
Figure 17: Sequence Tube Maps represents genomes within a pangenome as successive Sankey diagrams	37
Figure 18: MoMI-G combines multiple views dedicated to structural variations.....	38
Figure 19: The Datasaurus dozen shows that highly different datasets can share the same summary statistics.....	41
Figure 20: Depending on the chosen projection, a visual representation can highlight different properties of a same concept.....	42
Figure 21: The Gestalt principles explain why some visual elements are understood as being grouped together.....	44
Figure 22: The Rainbow color scale does not accurately represent distances from a linear space	45
Figure 23: Huang proposes that shapes are distinguishable based on the three properties constituting the SCI space.....	46
Figure 24: Munzner ranked the effectiveness of visual channels depending on the type of data that should be represented	47
Figure 25: The Venn Banana chart, both praised and criticized	50

Figure 26: Panache’s UI is divided between a main view and a menu panel	70
Figure 27: panBlocks are built from common fragments found in different genomes	71
Figure 28: The PAV matrix can fit into an encapsulating UI, with interaction available for navigation.....	71
Figure 29: Annotation Cards display details on a gene annotation	73
Figure 30: Structural Variations are labeled based on the differences between a genome and a reference.....	86
Figure 31: Translocations can be depicted with various layouts.....	87
Figure 32: plotsr represents different types of SVs as colored ribbons between lines of genome assemblies.....	88
Figure 33: An SV between two genomes will create alternative connections between nodes in a sequence graph	89
Figure 34: The overall diversity view of SaVanache displays an interactive scatterplot of genome assemblies.....	92
Figure 35: On hovering, dots of assemblies with similar metadata are colored accordingly.....	93
Figure 36: The pangenomes are ordered depending on the percentage of selected assemblies they possess.....	94
Figure 37: When hovering a pangenome, dots are colored depending on their presence status within	94
Figure 38: The coloring pattern of a dot for the pangenome-hovering interaction evolves depending on selection and presence status and the chosen pangenome.....	95
Figure 39: Each observed chromosome sequence can be described as a linear succession of nodes within a panchromosome	97
Figure 40: Representing panchromosome as violin plots can be thought of as putting a graph in a sock to see its silhouette	97
Figure 41: The shape of violin plots for panchromosome can be computed from the connections spanning successive nodes	98
Figure 42: Interactivity on PanCircos would enable dynamic filtering of SVs depending on their sizes	99
Figure 43: The selected region of the main panchromosome is delimited by two dynamic handles that can be moved left and right.....	100
Figure 44: Showing cooccurrences as connecting ribbons between a source and targets scales badly	100
Figure 45: Encoding the position on the source with color makes it possible to hide the ribbons completely.....	101
Figure 46: The interface for the visualization of SV is divided into multiple connected subparts	102
Figure 47: A PanCircos visual representation can be opened by clicking the PanCircos icon within the interface	103

Figure 48: Each comparison to a pivot path reveals several patterns that can be linked with common SV types.....	105
Figure 49: Six different glyphs would be enough to encode all identified SV types.....	105
Figure 50: The glyph system depicts SV from a pangenome graph, built from linear genomes.	107
Figure 51: Five colors were chosen for SaVanache UI with the glyph system on display	107
Figure 52: All glyphs can fit within one comparison unit and still be identifiable	108
Figure 53: Glyphs can be colored as a heatmap to depict the distribution of SV annotations within a group of assemblies	110
Figure 54: A summary panel can show bar charts of the percentage of nodes with a certain SV glyph within an assembly	111
Figure 55: Multiple SVs can be depicted simultaneously in the detail view.....	112
Figure 56: All Nodes from the panSkeleton are described as Steps when included within a path	114

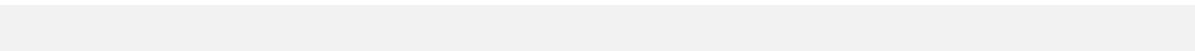
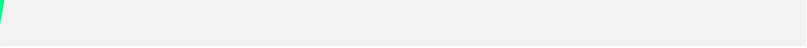
Table of Tables

Table 1: PAV matrix is a broad term for various format specifications encoding presence absence	18
Table 2: Visualization tools for pangenomics are diverse	22
Table 3: Unspecific tools are generic tools not dedicated to (pan)genomics.....	23
Table 4: Qualifying tools provide (summary) information on pan-gene atlases without any coordinate system.....	25
Table 5: Positioned tools anchor fragmented pangenomics data on a coordinate system	28
Table 6: Most of the structural tools are based on genome graphs	30
Table 7: Composite visualization tools leverage multiple pangenome visual representations and visualization categories.....	37
Table 8: Out of the four visualization tools tested, none was deemed suitable for our pangenomes	69
Table 9: Panache PAV matrix can be written with integers or strings alike	75

Table of Abbreviations

Abbreviation	Meaning
BAM	Binary Alignment Map
BED	Browser Extensible Data
CNV	Copy Number Variation
CSV	Comma Separated Value
DAG	Directed Acyclic Graph
Datavis	Data visualization
DNA	DesoxyriboNucleic Acid
FASTA	FAST-All
fGI	Flexible Genomic Island
fGR	Flexible Genomic Region
GAM	Graph Alignment/Map
GAF	Graph Alignment Format
GFA	Graphical Fragment Assembly
GFF	Generic Feature Format; General Feature Format; Gene Finding Format
GO	Gene Ontology
GWAS	Genome Wide Association Study
HCI	Human Computer Interaction
InDel	Insertion Deletion
IP	Internet Protocol
JS	JavaScript
JSON	JavaScript Object Notation
LCB	Locally Colinear Block
MAF	Multiple Alignment Format; Mutation Annotation Format
MSA	Multiple Sequence Alignment
PAV	Presence Absence Variation
PCA	Principal Component Analysis
pgAtlas	Pan-gene Atlas
pggb	PanGenome Graph Builder
RDF	Resource Description Framework
SCI	Segmentability, Compactness, and Spikiness
SNP	Single Nucleotide Polymorphism
SPA	Single Page Application
SV	Structural Variation

SVG	Scalable Vector Graphic
TSV	Tab Separated Value
UI	User Interface
UX	User eXperience
VCF	Variant Call Format
VR	Virtual Reality
YAML	YAML Ain't Markup Language; or Yet Another Markup Language



Introduction

What if we had access to many genomes?

What if they were merged?

What if someone wanted to see that?

That first question is not far-fetched: the steady improvement of sequencing technologies along their cost reductions enabled many scientific studies on multiple assembled genomes [1]. The drastic expansion of large-scale sequencing projects—like the 1000 Genomes Project [2], the 3,000 rice genomes project [3], or the *Anopheles gambiae* 1000 Genomes Project [4]—is leaving scientists with enormous amounts of genomic data, with the drawback of being difficult to handle with the existing analysis workflows. Storing and comparing these genomes has a high computational cost, and it becomes harder to make sense of all the available information. Besides, most studies are based on a single high-quality reference genome, from a species of interest or the closest one evolutionary speaking.

It induced a paradigm shift as it became apparent that one reference genome per species was not enough to capture its whole diversity [5]. Intra-species variations cannot be captured by a unique reference genome, while having potentially big effects on phenotypes. As an example, the gene *Sub1A* found in *Oryza sativa* (African rice) is completely absent from some of the individuals of that species but is positively linked to submergence tolerance [6]. Scientists therefore started to shift away from single references, towards *pangenomes* instead.

Pangenomes are the answer, or rather one answer, to the second question above: abstract entities that hold the genomic content from multiple genomes from a same species, group, or clades.

First described by Tettelin in 2005 for bacteria [7], they reached a broader audience as their scope was expanded to other organisms, in particular eukaryotes like human or plants. International collaboration arose in the past years, such as the Human Pangenome project [8] (initiated for improving the current reference human genome through pangenome approaches) or the PANGAIA project⁴ (about novel methods for the creation of pangenomes). Though pangenomes have multiple shapes and use cases, they could be described as *exhaustive inventories of unique and shared genomic items within a group of related individuals*. The genomic items in question can be genes as in the original study, or simply genomic sequences, depending on the studies.


The advent of pangenomics is still recent as it has not been popularized until the late 2010s⁵. Consequently, there is a slowly resorbing lack of tools and standards approaches for its creation, analysis, and visualization, especially for eukaryotes as they embraced this field of study later.

Data visualization—or simply '*datavis*'—is especially important as it empowers researchers during analyses, helping them to identify patterns and outliers or to compare data more easily than by looking directly at the datasets. *Datavis* tools use visual representations that encode data, transforming concepts into tangible visible objects that are easier for the human brain to make sense of. They are useful at different stages, from the exploratory analysis steps to the communication of results, for publication, for other scientists, or even non specialists.

For pangenomes, visualization tools could help users during their analysis workflows: as the number of sequenced genomes rapidly increases, there is a pressing need for visual representations and tools that would handle pangenomic datasets built from hundreds if not thousands of genomes. Such tools should provide interaction, enabling the exploration of different parts of the underlying data to answer both low- and high-level questions about the overall dataset or specific subsections. More specifically, one could be interested in genomic content that is unique to a plant with a phenotypic trait of agronomical importance, in the comparisons of

⁴ <https://www.pangenome.eu/>

⁵ For example, with a major publication by the Computational Pan-Genomics Consortium 9. Computational Pan-Genomics Consortium, *Computational pan-genomics: status, promises and challenges*. Brief Bioinform, 2018. **19**(1): p. 118-135.



nucleotide sequences for identified gene of interest between multiple individuals, or even in the identification of stable genomic regions that would be good candidates for a gene introgression.

This leads to the final question out of the three above: what if someone wanted to visualize pangenomes? Previous work⁶ highlighted a lack of usable visualization tools and visual representations for plant pangenomes. Existing pangenome visualization tools were either not working properly, not scalable to plant (pan)genomes or working specifically on genes without including intergenic sequences.

My PhD has therefore been focused on the creation of new approaches to both the visual representation of eukaryote (and especially plant) pangenomes and the design of dedicated visualization tools. In this dissertation, I propose ten guidelines for the creation of datavis tools in genomics and introduce two designs of tools (Panache and SaVanache) for the visualization of plant pangenomes, providing an answer to the question “*What would scalable and meaningful visualization tools for plant pangenomes be like?*”.

⁶ done as part of my Master 2 internship



State of the Art

Pangenome visualization is at a crossroad between different disciplines: biology, bioinformatics, software development, data visualization... The following chapter explains the concepts involved in this thesis and its context—the “*pangenomes*”, the related visualization tools, and notions of data visualization design.

I. Pangenomes

Before getting into details, here are some precisions on the terminology used within this PhD thesis. Due to the relative novelty of pangenomics (mid 2000s) [7], its exact definitions and name are still unclear, with variations between studies. Some spell it ‘*pan-genome*’ with a hyphen [10-13] similarly to its recognized original spelling [7], while others prefer to write it ‘*pangenome*’, as a single word [14-20], as showed in Figure 1. Some authors like researchers working with Dr. David Edwards even changed their spelling from one publication to another [10, 21], with no apparent pattern.

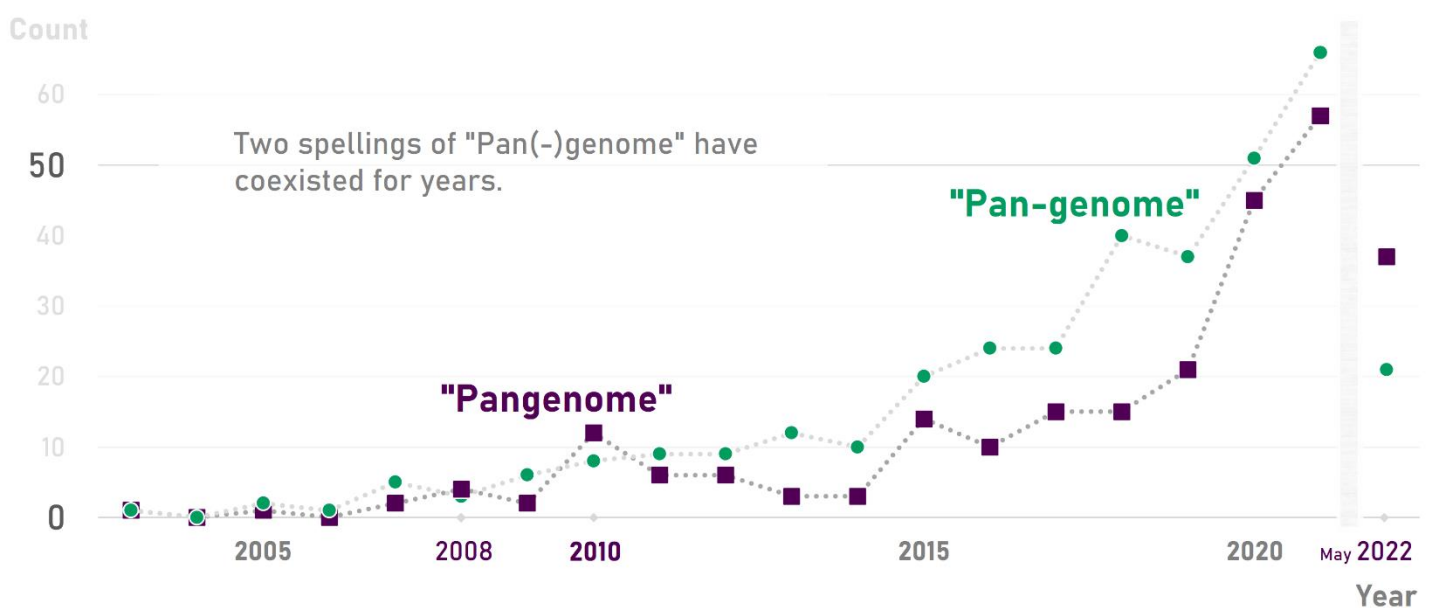


Figure 1: “*Pangenome*” took over “*Pan-genome*” in early 2022; While both spellings have been in use since the first publications and “*Pan-genome*” seemed slightly more popular overall, “*Pangenome*” became more popular in early 2022. Whether this trend will continue remains to be seen. Results were extracted from PubMed on 3 June 2022, by listing, filtering and counting all papers whose title contained “pangenom*” vs “pan-genom*” using search criteria as the following: (((“pangenome”[Title]) OR (“pangenomes”[Title])) OR (“pangenomic”[Title])) OR (“pangenomics”[Title]).

In this manuscript I will use the concatenated spelling ‘*pangenome*’, as it is more consistent with other spellings of -omics fields (e.g., ‘*metagenomics*’), has currently more popularity than the hyphenated version and suits my personal preferences better. Besides, it is worth mentioning that Dr. Hervé Tettelin edited in 2020 a book titled “*The Pangenome*” [22], abandoning the hyphenated version that he coined 15 years earlier.

A. Context, interest & applications

Variations and diversity between individuals are important factors for crop development and led to many of the fruits and vegetables we know and eat today [23-26]. At a genomic level, these variations can happen on small locations which is the case of **Single Nucleotide Polymorphisms (SNPs)** and **Insertions and Deletions (InDels)**, with known effects on phenotypes [27-30]. There are also larger genomic variations, within and between chromosomes, known as **Structural Variations (SVs)** when they span more than 50bp [31].

They could be major chromosomal rearrangements [32-34], or duplication, insertions, deletions, translocations... of various sizes. Deletions and duplications which have an influence on the number of occurrences of a given genomic pattern are called **Presence Absence Variations (PAVs)**, a pattern can be present or absent), or more generally **Copy Number Variations (CNVs)**, a pattern can be present 0, 1, 2+ times) [35, 36]. These variations can have important effects on (plant) phenotypes: flowering time modified by CNVs of two genes in *Triticum aestivum* (wheat) [37]; tolerance to submergence in *Oryza sativa* (rice) depending on PAV of the Sub1 gene [6]; CNV-induced resistance to nematodes in soybean [38]... among others [16, 39]. For example, in maize more than hundreds of PAVs and CNVs have been detected, with potential impact on phenotypes (disease resistance, coloration...) and heterosis [36]. In *Brassica napus*, 38% of the genes showed PAVs, often influencing agronomic traits of interest (resistance to blackleg infection and other defense responses, acyl lipid metabolism...) [40].

One reference genome is not enough to access all this diversity [5], thus restraining to a single reference prevents the research of interesting agronomic features from reaching its full potential. As the latest generations of sequencing technologies along with their price reduction enabled the production of tremendous amounts of genomic data [1], including multiple genomic references for a single species [41-43], scientists are not restricted to one reference genome per genus, group or species anymore.

Pangenomics is an integrative approach which aims at the assessment of every possible genomic variation within a group of closely related individuals. Although it has already often been applied to bacteria as it is its first application field [7, 44], its use with eukaryotes, including plants, is still recent and slowly gets more popular [16, 45], especially with efforts made on human genomes [8, 46, 47]. Therefore, there is still a lot to explore concerning its methods, file formats, related tools, applications [48]...

Pangenomes could be used for different use cases: diversity characterization, optimized storage of genome comparisons, mapping on exhaustive references [9, 39]... Diversity characterization is essential for crop improvement: identifying genomic regions linked to phenotypes of interest or conserved through selection and domestication could lead to improved quantity of production, resistance, taste, ... through breeding processes [18, 39, 45, 49-51]. Example use cases of pangenomes have been documented for rice [52, 53], cucumber [54], tomato [55, 56], soybean [57], melon [58] and a broad range of other crops.

Particularly, characterization of wild versus domesticated crop could bring many benefits, with the detection of variants (gene, transcripts...) not present within single references and the introgression of genes from wild relative [9, 52, 59-61].

Besides, this PhD dissertation is plant-oriented but pangenomes are also useful for other organisms, like human for health purpose or cattle [62, 63]. I focused my work on plant pangenomes, but it could be broadened to eukaryotes in general.

B. Multiple definitions

There is no proper, single definition of what a pangenome is, but multiple coexisting definitions instead [18, 45, 49]. These definitions might be split into different names in future work, but as of 2022 they still are called the same while dealing with different concepts. There is no standard yet, and the definition in use highly depends on the research group and the organisms it is applied to. Some scientists are more interested in the presence/absence of genes and gene families, which might be the case of breeders or bacteriologists for instance, while others might have more interest in SVs related to evolutionary events (duplications, translocations...), therefore focusing on the structure of the different genomes.

Even though the gene definition is the oldest one, it is still in use nowadays [64, 65] and coexists with its structural equivalent, which has recently gained popularity [19, 60].

For disambiguation, I call '*pan-gene atlas*' (**pgAtlas**) the genic definition and keep '*pangenome*' for the structural one.

1. Pan-Gene Atlas

A pgAtlas only focuses on the functions shared within a group of individuals. It therefore is a repertoire of genes, gene families, or transcripts that are shared between individuals, or unique. This definition made sense in the early days of pangenomics, which was then focused on bacteria and prokaryotic genomes. Since they are mostly made of gene clusters—*operons*—with about only 12% of non-coding DeoxyriboNucleic Acid (DNA) (suspected of being promoter regions) [66], focusing on functions was not leaving much of the sequences out. However, while still in wide use this definition does not fit well eukaryotes as genes only make a small part of their genomes (around 8% for plant genomes [67]). Sometimes both the structural and functional definitions are applied to the study of a eukaryotic organism, in which case the functional approach can be presented as a '*pantranscriptome*' [49, 68] (built as the repertoire of all transcripts) or simply '*functional pangenome*' [60].

I first came across the phrase “Pan-Gene Atlas” while participating in an AgBioData conference call about pangenomes [69]. Participants were confused by the concept of ‘pangenomes’, trying to precise every time whether they were talking about a '*gene pangenome*' or a '*structural pangenome*'. Pr. Pankaj Jaiswal proposed to distinguish the two by calling the gene pangenomes “*pan-gene atlases*” instead, which I became quite fond of. I have chosen to keep this name for I think that its phrasing is far enough from '*pangenome*' (compared with '*genic pangenome*' as proposed by Sherman and Salzberg [49]) and that it reflects well the definition used.

2. (Structural) Pangenome

A '*structural pangenome*' or '*pangenome*' for short is similar to a pgAtlas in that it is created to embrace the variability within a group of individuals. The difference is that it does so in a more ambitious fashion as it is supposed to take most (if not all) of the genomic information from the DNA sequences into account, and not only the genes. For example, Tranchant-Dubreuil et al. defines it as “*the complete set of non-redundant sequences approximately 100 base pairs (bp) in length or more (except for few SNP, and InDels [...]) within a given group of individuals*” [18]. In this definition, '*pangenome*' is more faithful to its etymology, as genetics focuses on genes while genomics embraces a broader understanding of DNA and its study.

Some plant pangenomic studies only use a pgAtlas [17, 68], but integrative studies of shared elements have now become more popular, and the structure is more often taken into account [9, 10, 60, 62], even for prokaryotes [70].

Even though I worked on both definitions, my PhD focused principally on pangenomes rather than pgAtlases, as this structural definition makes more sense for plant pangenomics, attracts more scientists every day and introduces more visualization challenges.

C. Significant properties

Pangenomes and pgAtlases alike have special properties that make them more than just a sum of genomes.

The main interesting one lies in which elements are shared, and which are not. Pangenomes are traditionally divided in two parts, originally called the '*core genome*' and the '*dispensable genome*' [7]. Such categorizations are debated since pangenomics has been lacking standard nomenclature, and both their names and limits have been criticized. While the notion of '*core genome*' is usually kept, some may talk about '*accessory genome*' or '*variable genome*' instead of '*dispensable genome*', which they find misleading and easily confusing since it may be interpreted as '*useless*'. Others use additional categories, with different combinations: '*core, dispensable &*

rare" [71], 'core, softcore, shell & cloud' [11, 72], 'core, softcore, dispensable & private' [10, 60], 'persistent, softcore, shell & cloud' [70], 'core, dispensable & specific' [42]...

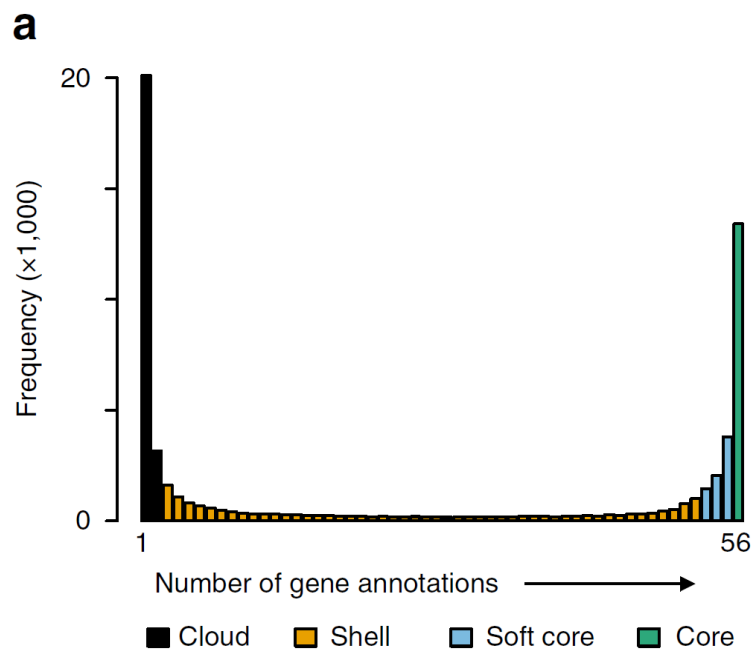


Figure 2: Different sets of genomes share different genomic elements; Genes, or chunks of DNA sequences are not homogeneously shared between genomes of a given species: some of them may be unique to a single genome while others may be present in every single genome. Depending on how much shared they are, they can be classified as belonging to different categories—*Cloud*, *Shell*, *Soft core*, and *Core* in the case of the study by S. Gordon et al. (2017) [72] on 56 *Brachypodium distachyon* genomes, which originally featured this Figure.

While using different terminologies, all of those convey a sense of variable frequencies of genomic items within a group of individuals, as shown in Figure 2. Without taking their names into consideration, distinctions are often made to qualify items present in:

- A) **every** individual of a considered group
- B) **most of** the individuals of a group—*might be useful in case sequencing errors did not detect an item within one individual for example, or when considering the evolution of a population*
- C) **some** individuals only
- D) really **few** individuals or even only one—*often labelled as 'orphan'*

In this thesis I refer to **core** and **variable** genomes: only two categories for simplicity, and *variable* instead of *dispensable* so that no difference in usefulness can be implied.

Such categorizations of shared genomic content can be useful for pangenome and population analysis, for example for identifying candidates with rare agronomic traits of interest for hybridization. Another useful property is the completeness of the pangenome (related to representativeness), which is interesting during the construction of a pangenome.

A pangenome can either be '*open*' when the addition of a new individual to the group keeps adding new genomic material to the pangenome, or '*closed*' when new material ceases to be added [7, 12, 45], as illustrated in Figure 3 below.

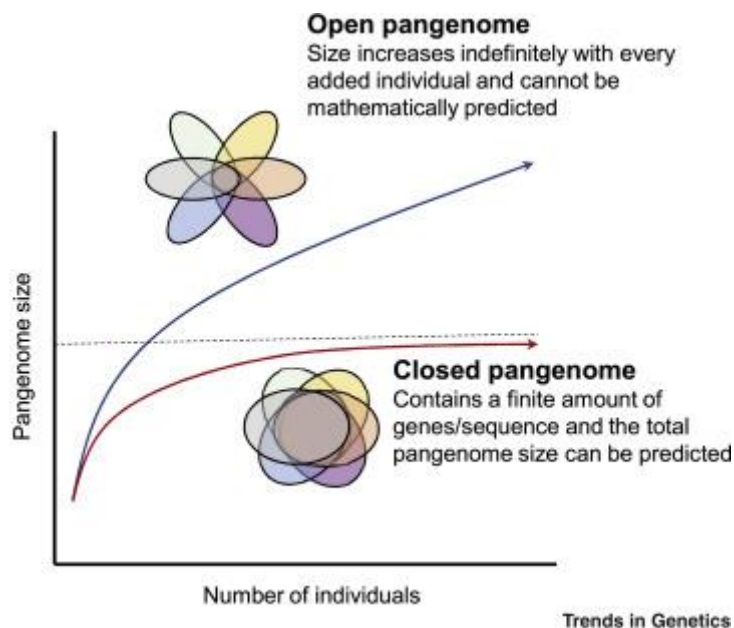


Figure 3: The more exhaustive a pangenome is, the closer to a "closed" pangenome it becomes; When the addition of genomes within a pangenome does not add any new material, the pangenome is considered 'closed', as opposed to 'open'. Figure from [45].

In order to have a pangenome as representative as possible of the variations within a group, a closed pangenome should be favored. If new material keeps being added by expanding the pangenome to other individuals, this means that there are still undiscovered variations that are not taken into account yet. Including more genomes or doing a better sampling would ways to come closer to a closed pangenome. Depending on the order of integration of new genomes within a pangenome this 'closed' status can be reached sooner or later.

Moreover, pangenomes are mostly used for groups of closely related genomes but some studies focus on 'super-pangenomes' spanning more diverse genomes [59]. Every vaguely distant genome will add its own diversity which can create a sudden size gain of the pangenome. That might happen when adding a genome from a wild species into a pangenome built with domesticated species, for example [52, 59, 60].

D. Challenges specific to plant pangenomes

Pangenomes can be applied to many organisms, as illustrated by the numerous existing studies on Bacteria (*Escherichia coli*, *Streptococcus pneumoniae*, ...), Fungi (*Saccharomyces cerevisiae*, *Cryptococcus neoformans*, ...), Protists (*Emiliana*), Plants (rice, corn, ...), and Animals (human, pig...) [45]. In this work however I focused on eukaryotes, and plants more specifically. Prokaryotic pangenomes have been studied earlier, and a small, dedicated tool ecosystem already existed (see State of the Art II.B.2) when I started working on pangenomics in 2018, while there was nearly no available tool for eukaryotes. Among eukaryotes, my work focused on Plants (bananas, rice, soybean, wheat...) as they are the main applications for my different affiliated organisms.

Plant pangenomes, and their visualization, face specific challenges that make them stand out from the other eukaryotes, and their prokaryotic counterparts [9].

Plant genomes, as all eukaryotes, are divided into multiple chromosomes, which is a first challenge when it comes to visualization as it is a major difference from prokaryotes, and the related visualization tools do not scale well. They are of highly diverse sizes (from about 60 Mbp in *Genlisea aurea* to more than a hundred Gbp in *Paris japonica*) [73, 74], complex, often polyploids, and might be constituted of multiple subgenomes inherited from ancient hybridization events (e.

g. in the case of domesticated species). This happened for example in the genus *Brassica* (with species originating in hybridization between three genomes: A, B, and C [75, 76]), and cultivated bananas *Musa spp.* (with four identified original genomes: A, B, S, and T [77]). Moreover, they are subject to multiple structural variations, with repeated elements [78-81] but also large genomic rearrangements such as chromosomal inversions, often related to traits of interest for the plants [82]. These genomic specificities result in challenges for visualization: how to manage these large scales? ...compare between different chromosomes or subgenomes? ...highlight both small and large structural variations?

Plant pangenomes are of high interest for breeding programs: for identifying seeds with the best combination of agronomic traits and therefore choosing the best candidates for crossing; for narrowing down the genomic regions likely to be related to a given phenotypic character... Providing an improved access to the intra-clade diversity can enhance the quality of genome mappings and guide the related genomic analyses better. Tools to make these complex pangenomes more manipulable and understandable for biologists are greatly needed.

E. Desirable features

To better pave the way toward popularized and widely spread pangenomics, The Computational Pan-genomics Consortium drew up a list of desirable features for pangenomes and their use in bioinformatics [9]. These features are mostly of interest for pangenome building tools but can provide the readers with a better understanding of their (expected) inherent properties.

Completeness

As mentioned earlier, a pangenome used for analyses should be as closed as possible. The more elements and the more varied they are, the more a pangenome can be useful as a replacement for reference genomes and for studying a sequence's distribution within a population, enabling researchers to think of better hypotheses.

Stability

As reproducibility is an important stake of scientific research, a pangenome should have a layer of stability, enabling proper comparisons between teams and/or time points. Identification of similarities and differences between two states are at the heart of scientific analyses and should therefore be applicable here.

Comprehensibility

As the definitions suggest, pangenomes should be expendable to large amounts of individuals and/or species, enabling analyses at different genomic resolutions.

Efficiency

Pangenomics data should be organized, with clear and clever specifications as to make further analyses faster and easier.

Regarding visualization in particular, the design goal of a pangenome data structure is, according to the consortium, to enable multi-scale visualization with access to all the available information, including the “*visualization of global genome structure, SVs on genome level and local variants on nucleotide level, but also biological features and other computational layers*” [9].

F. Status of the pangenomic landscape

Pangenomics' popularity increased during the last two decades (as illustrated in Figure 1) but it still is a young field of study, with lots of opportunities and things to uncover. Due to its novelty

however, there is a lack of standards and established procedures, as reflected by its multiplicity of definitions, as explained in State of the Art I.B.

1. Lack of standard approaches...

The two definitions, pan-gene atlases versus pangenomes, led to highly different approaches for building pangenomes. Since the two use different concepts, the ensuing building strategies could not be the same.

a. Different concepts

Pan-gene atlases or pangenomes, function or structure, discrete blocks or connected sequences: the two pangenomic definitions involve concepts that seem hard to reconcile. The related mental representations evolved with this dichotomy at their very basis.

- **Presence Absence Matrices**

As repertoire of genes, pgAtlases can be thought of as sets of elements that can be present or absent from genomes, as in Figure 4. These **presence absence matrices** (or '*PAV matrices*') can be built from individual gene names, gene clusters, or shared gene families for example, in which case they do not encode any sequence information per se. However, PAV matrices can be applied on pangenomes when they are considered as collections of fragments of nucleotide sequences.

In such matrices, multiple copies of items can be listed present or absent as a whole, or the presence status can be refined and attributed on a case-by-case basis. Which copy should be marked as present or absent depends on the methods used.



Figure 4: Pan-gene atlases can be described as a listing of the presence statuses of genes in multiple genomes; For each genome (here the bacteria have one dedicated row each) a presence or absence status can be registered for every item (one item per column, presence is here encoded with colored discs, one hue per item). Note how here the bottom bacteria has the "blueberry" item twice but registered it with a single presence status.

- **Locally Colinear Blocks and panBlocks**

A hybrid approach, taking from both worlds of discrete items and connected sequences, and inherited from **Multiple Sequence Alignments (MSA)** considers genomes as **successive chunks of sequences**. These chunks can be sorted into blocks of sequences free from internal rearrangements, called **Locally Colinear Blocks (LCBs)** [83-85]. Similar chunks of sequence from two genomes would be gathered into one LCB.

In this PhD manuscript, I use the notion of *panBlocks* that I define as any sequence unit derived from the fragmentations of genomes in a pangenomic context. They are conceptually close to LCBs but with small internal variations possible. This looser object leaves more flexibility regarding

how the blocks and level of similarity should be defined exactly. If one sequence is labelled as being *'the same'* than one in another genome, they would be part of a same panBlock, no matter how that similarity is measured. The freedom of choosing any similarity threshold that should divide chunks between being *'the same'* or *'different'* is left to researchers creating their pangenome.

Moreover, panBlocks can be defined as containing sequences, genes, or any kind of information whose presence status should be compared between genomes. In the case of genes, panBlocks could therefore share partially overlapping coordinates, as annotations can overlap. For example, one panBlock could correspond to the annotation of a gene on the forward strand, overlapping a panBlock with the annotation of a gene on the backward strand.

They could also be unique or duplicated into multiple copies (or *'cooccurrences'*) instead. A chunk of sequence present in two different positions (either within a genome or relative to the same fragment in another genome) could be considered as belonging to a unique panBlock merging both occurrences, or to one version of a panBlock, position dependent. For example, the green (■) block in Figure 5 which appears twice in the genome C could be described by a unique panBlock merging all possible positions or by two panBlocks, one for each potential position.

The exact definition used for a panBlock (what it contains and whether the copies are merged) would therefore have an influence on the **core** / **variable** genome categorization, as illustrated in Figure 5.

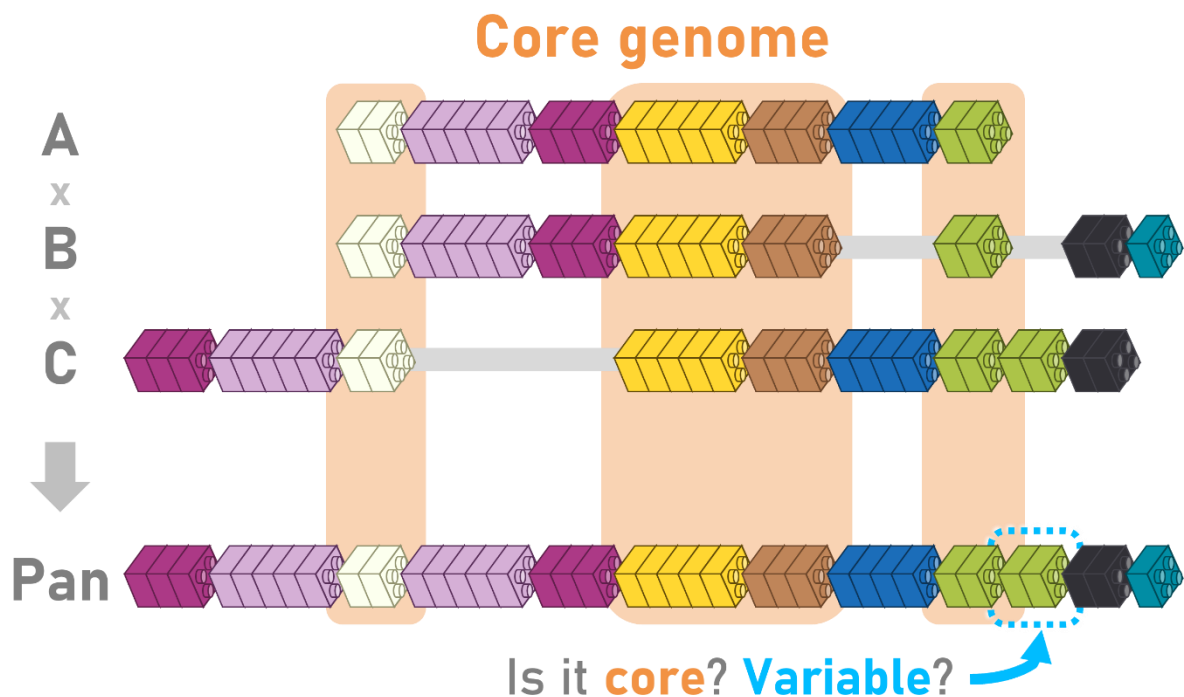


Figure 5: With position-dependent twins, categorizing panBlocks can be difficult; Genomes A, B, and C have been segmented into chunks, each belonging to a panBlock depending on their content (encoded by the color and size) and position. PanBlocks have been laid out on a linear string corresponding to the pan-gene atlas or pangenome coordinate system. The magenta (■) chunk is stored within two panBlocks, corresponding to two different possible positions. Should the rightmost green (■) panBlock be considered as part of the core genome since all genomes have a green (■) chunk, following a functional definition? Or should it belong to the variable genome as only one genome has this chunk in this position, following a structural definition instead?

- **Panreferences**

Shared fragments of genomes are useful but need to be hosted in a macro-entity to really form a pangenome. This led to the concept of **panreference**, which is a certain layout of some or all of these fragments, involving relative positions and a sort of coordinate system. The Pan-Genomics Consortium refers to panreferences as pangenomics data structures centralizing read mappings and used for variant calling [9]. Tranchant-Dubreuil et al. describe panreferences as reference sequences (implied as linear) extended with additional genomic fragments from multiple individuals⁷ [18].

These panreferences can be mosaic linear sequences—built upon an existing reference genome and extended with external genomic material or completely reference free instead—but they also accept other shapes like graphs as detailed in State of the Art I.F.3.c. When linear, the additional material can be anchored on the existent, or packed in abstract regions instead if no suitable place can be found. Panreferences are therefore highly context dependent, but their common purpose is to serve as a pangenomic skeleton, which organizes and hosts all the genomic fragments of a pangenome.

- **Genome/sequence/variation/pangenome graphs**

The most recent concept used to represent pangenomes is that of **pangenome graphs**. Graphs are malleable entities that can be used to store information of connections between different sub-elements, such as DNA sequences.

As graphs work with a specific vocabulary, I introduce here the main concept that a reader should before pursuing further without entering the depths of graph theory. Graphs are composed of different entities (see Figure 6) with specific names that will be used throughout this manuscript. There primarily are the '**nodes**' (also called '**vertices**', '**vertex**' in its singular form), which are points corresponding to the entities being connected. The connection, or links between two nodes is called an '**edge**', or '**arc**' when it is directed—that is the case when $u \rightarrow v \neq v \rightarrow u$. A succession of nodes and edges is called a '**walk**'. If no node is repeated within that succession, it can be called '**path**' instead [86]. For convenience and consistency with other works I use **edge** whenever I talk about connections (directed or not) and do not use **arc**—most of the time the context is enough to infer which kind of **edges** are discussed.

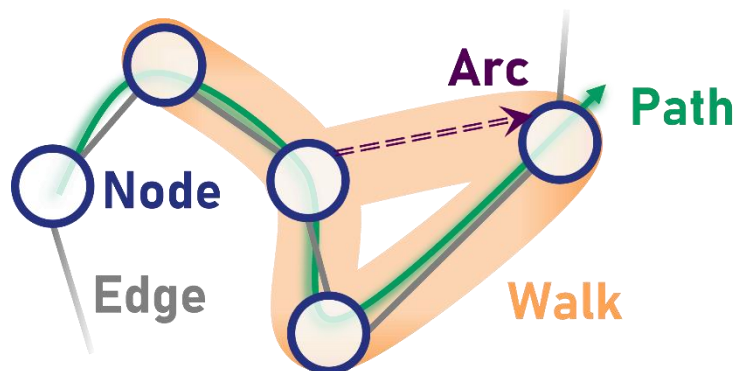


Figure 6: Graphs are mainly composed of Nodes and Edges; 'Arc' is rarely encountered, it is more common to see '*directed edge*' instead. Depending on the researchers, path and walk can at times be used interchangeably too.

Graphs are common entities in bioinformatics as de Bruijn graphs for example are often used for genome assemblies and alignments [87, 88]. They also have recently been used to represent genomes in a more compact way than a linear succession of nucleotides, within **genome graphs**. This term encompasses all kind of graphs using sequences to represent genomes: directed—

⁷ See also C. Tranchant-Dubreuil, C. Chenal, M. Blaison, L. Albar, V. Klein, C. Mariac, R. A. Wing, Y. Vigouroux, F. Sabot, "*FrangiPANE, a tool for creating a panreference using left behind reads*", submitted to NAR Bioinformatics and Genomics, 2022

either *vertex*-labeled as in de Bruijn graphs (where the *nodes* represent sequences) or *edge*-labeled (where the *edges* represent sequences instead, *nodes* being intersections)—but especially bidirected (where sequences are oriented, enabling one item to represent forward and backward strands depending on how they are traversed) [89]. The genomes used to create these graphs can be retrieved by *walks* through the graph, providing the original observed succession of sequences.

A **sequence graph** is a bidirected *vertex*-labeled graph, meaning a graph where *nodes* represent nucleotide sequences and are oriented as to represent the possible DNA strands. Each orientation encodes the reverse complement of the opposite one. To this already complex nomenclature, Garrison et al. added the **variation graphs**, which are *sequence graphs* with paths used to represent the linear sequences of a pangenome: each genome of a pangenome has a related path within the variation graph, which can be used to anchor a coordinate system for annotations and others [19, 90, 91].

Finally, **pangenome graphs** can be described as *genome graphs* containing sequences from different genomes. All *variation graphs* are pangenome graphs, but not all *sequence graphs* are pangenome graphs. They have branches whenever two genome sequences differ, creating “bubbles” of sort. Pangenome graphs can be represented as **Directed Acyclic Graphs (DAGs)** which is their most human readable form, with repeated subsequences being duplicated within the graph instead of being encoded by a unique node (see Figure 7).

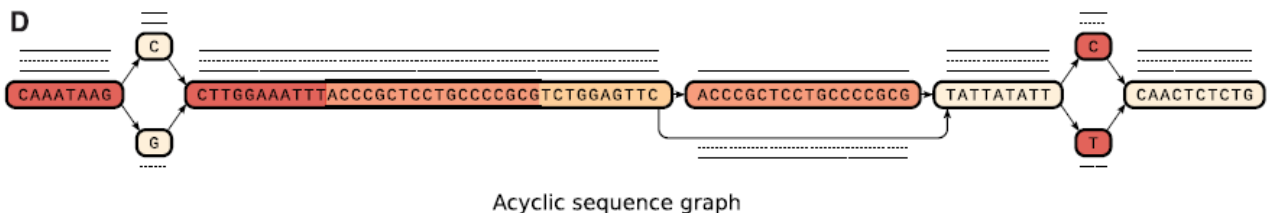


Figure 7: Directed Acyclic Graphs duplicate the nodes present at multiple positions instead of creating loops toward the same node again and again. They are more human readable than cyclic graphs, but also take more space to store. Here, note how the long beige (■) sequence is present twice, with an edge skipping it entirely at the second occurrence. In a cyclic graph this node would have been present once, with an edge looping on it instead. Figure from [9].

Others for example explored the possibilities offered by colored and compacted *de Bruijn* Graphs [92-95], with colors embedding the origin of different parts of the graphs as in Figure 8 below.

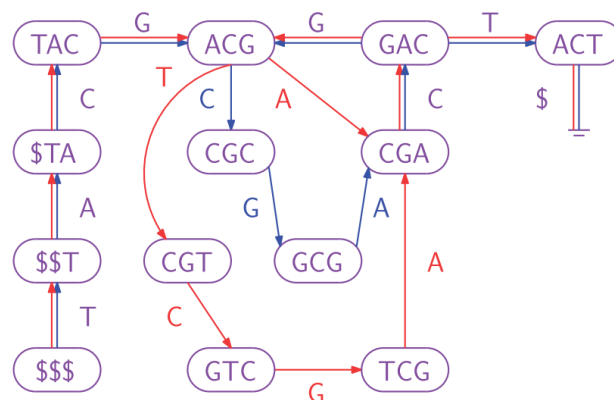


Figure 8: colored de Bruijn graphs encode the origin of nodes through colors; In this example two de Bruijn graphs (red and blue) have been merged into one. Nodes are violet as they supposedly appear in both. Figure from [92].

To summarize, genomes can be embedded in graphs by turning bits of their sequences into either *nodes* or *edges*. Connections are then represented by the remaining graph entity (*edges* or *nodes*, respectively). Moreover, such graph could be *directed* or not, *cyclic* or *acyclic* (meaning with or without possible loops within the graph), with nodes of uniform lengths or not, with or without

sequence overlaps... Thus, a great variety of possible graph representations for genomes exist [9, 19, 88, 89, 91], with the main one being *sequence graphs*.

b. Different building strategies

Multiple concepts involve multiple strategies for building pgAtlases and pangenomes [96]. I do not detail them here as they are not at the core of my PhD work. However, they can be summarized into different categories: clustering genes from multiple individuals; having a collection of shared blocks through MSA; creating a linear panreference by anchoring reads on a reference genome; building a graph from assembled genomes...

The important property to keep in mind is that some strategies are heavily dependent on the order of addition of new genomes into a pangenome (when they build sequentially), meaning that there could be different pangenomes built from the same sequences. It essentially comes down to the difference between tools with incremental strategies (order dependent) against those with integrated strategies instead (reference-free).

2. ...hence a lack of standard tools...

The diversity of pangenomic approaches is echoed by the diversity of related tools, for both the creation and analysis of pgAtlases and pangenomes [19, 97, 98].

a. Tools creating pangenomes

Since pangenomes have many shapes, there are multiple tools dedicated to their construction.

First there are tools focused on the creation of clusters (of genes, gene families) for pgAtlases: BPGA [99], ClustAGE [100], GET_HOMOLOGUES [101], GET_HOMOLOGUES_EST [102], LS-BSR [103], OrthoMCL [104], PanACoTA [105], PanOct [106], PGAP [107], Roary [108]...

Others focus more on the sequences and the creation of panBlocks instead, through Multiple Sequence Alignment for example: Harvest [109], Mugsy [110], PanCake [15], progressiveMauve [84], seq-seq-pan [111], SibeliaZ [85]...

Finally, there is the newest category of tools, those creating (pan)genome graphs: Cactus [112], minigraph [113], NovoGraph [114], Panaconda [115], PanGraph [116], PanTools [117], PPanGGOLiN [70], pggp [118], Progressive Cactus [119], seqwish [120], SplitMEM [121], the vg toolkit [90, 122], svaha2 [123]... the trendiest being minigraph, pggp and the vg toolkit.

minigraph takes multiple genomes as an input and incrementally creates a pangenome graph by adding new parts as branches on the base graph/genome. The origin of every new branch is stored, as well as its position on the genome it comes from. It is dedicated to “big enough” SVs and therefore might not include small variations such as SNPs and those below 50bp.

The pangenome graph builder (*pggp*) is a pipeline used for the creation of locally directed and acyclic variation graphs from sequences. It combines three steps—sequence alignment, graph induction, and graph normalization—and the resulting pangenome graphs can be used with other tools enabling pangenome graph manipulation (like ODGI [124]).

The *vg* toolkit contains an extensive set of tools that can be used to either create, update, convert or merge variation graphs from a variety of conventional formats

These three tools are still under active development (during the first half of 2022, *minigraph*, *pggp*, and the *vg* toolkit respectively had 4, 2, and 4 releases) and additional features and improvements are to be expected.

b. Tools using and manipulating pangenomes

Creating pgAtlases and pangenomes is only a first step towards pangenomic analysis—including data filtering or extraction, statistics, comparison, visualization, and many others—which can be done by downstream tools, quite diverse again.

As an example, for graph pangenomes only (indexing, haplotyping, mapping...) there are BGREAT [125], BrownieAligner [126], deBGA [127], GBWT [128], GCSA2 [129], GenomeMapper [130], GraphAligner [131], gfapy [132], gfatools [133], HISAT2 [134], ODGI [124], PLAST [135], the vg toolkit [90], V-MAP [136]...

Visualization tools, which are of particular interest for this PhD, are detailed later in a dedicated section, see State of the Art II.

3. ...and a lack of standard formats

Unsurprisingly, the landscape of pangenome files formats is as varied as the tools enabling their creation or using them. Some are well-known file formats, some are a bit more obscure, and others even are *ad hoc* combinations of different formats.

a. Generic formats

Among the different formats, some of them are common in bioinformatics, describing genomic annotations, sequences, variations, or default data structures.

- **BED**

The **B**rower **E**xtensible **D**ata (**BED**) format [137] stores genomic features (*i.e.* “linear region[s] of a chromosome with specified properties”) placed on a linear coordinate system, as a table (one row per feature, columns for details). Normally used within the UCSC Genome Browser, it can be derived to create PAV matrices of pangenomic items.

- **FASTA**

Short for “FAST-All”, FASTA is a standard file format used to write nucleotide or amino-acid sequences from the FASTA program [138]. It is used in pangenomics either to store sequences from multiple individuals, or as the base for the creation of panreferences and genome graphs, for example.

- **GFF**

Generic **F**eature **F**ormat, **G**ene-**F**inding **F**ormat, or even **G**eneral **F**eature **F**ormat, **GFF** is another format used to describe genomic features. Similar to BED, it also is a table-like format, with columns describing properties of features, one per row. Its third version, GFF3, has been created as an attempt to standardize all the *ad hoc* tab-separated formats used in bioinformatics into one universal format [139].

- **VCF**

The **V**ariant **C**all **F**ormat (**VCF**) [140] is widely used to describe observed variations of sequences compared with a reference genome. In pangenomics it can be used to extend pangenome graphs with additional nodes and/or edges depending on the variations already stored.

- **JSON**

The **J**ava**S**cript **O**bject **N**otation (**JSON**) format [141] is a language-independent format used to describe dictionaries of objects, with their properties and related values. It is a versatile format that can be used to store all kind of information without taking the order of the entries into account, useful to transfer data between different tools.

b. Specialized formats

- **PAV matrices and *ad hoc* formats**

Pan-Genome atlases are most often found as PAV matrices, and it is a format that can be used for panBlocks too. PAV matrices are tabular files, where pangenomic items (generally on the rows) are associated with a presence status (count, Boolean, gene name if present...) per genome (generally on columns) as illustrated in Table 1. Most of the time these PAV matrices are formatted similarly to BEDs, within tab- or comma-separated files from a variety of sources like gene clustering algorithms.

Table 1: PAV matrix is a broad term for various format specifications encoding presence absence; The PAV matrices presented here are examples of how the PAV matrix from Figure 4 could be written in a text file. **A)** Presence status is encoded for each combination of item (column) and bacteria (row) by a numeral, for example 0 when there is an absence, and 1 or 2 depending on the number of copies found. **B)** Similarly, presence and absence could be respectively encoded by True and False Booleans instead. **C)** There could also be no header, but the name(s) of the item(s) found within each bacterium if any (or an empty field otherwise), each column being a cluster.

	Bacter	■	■	■	■	■	■
A)	BacteriA	1	0	1	1	1	0
	BacteriB	1	1	1	1	0	1
	BacteriC	1	2	0	1	0	1
	Bacter	■	■	■	■	■	■
B)	BacteriA	True	False	True	True	True	False
	BacteriB	True	True	True	True	False	True
	BacteriC	True	True	False	True	False	True
C)	BacteriA	Sprin_1		Steel_1	Oran_1	Brick_1	
	BacteriB	Sprin_2	Berry_2	Steel_2	Oran_2		Rasp_2
	BacteriC	Sprin_3	Berry_3.1 Berry_3.2		Oran_3		Rasp_3

There is no standard way for writing a PAV matrix, but multiple *ad hoc* versions instead. Some are relatively easy to reproduce, for example by adding as a note the names of the genomes owning a certain feature within a GFF3 file [142]. Others are more obscure like the pan-genome map files used by Pan-Tetris [143], created by the unpublished in-house software *PanGee*. These files have position information of pangenomic items for each row, and multiple columns per strain—id, start, end, length, strand, gene symbol, gene description—with empty fields for encoding an absence.

In Panache for example, I chose to use PAV matrices built with BED fields, additional columns, and one column per genome for storing the presence status as detailed in Panache III.A.1.

- **From Multiple Sequence Alignment**

MSA is used in tools like seq-seq-pan [111] to create blocks of similar sequences from multiple sequences. A common format for these alignments is the **Multiple Alignment Format (MAF [144]**, not to be confused with the **Mutation Annotation Format [145]**): each block of multiple alignment is stored with a score, the coordinates, length and strand of the sequences and the detail of the alignment. An alternative is the **eXtended Multi FastA (XMFA)** generated by Mauve [83, 146], which stores the position of a primary sequence, and the FASTA sequences involved in the corresponding alignment.

c. Pangenome Graph formats

Graph formats are currently the trendiest for pangenomes, with similarities between pangenome graph building tools. Some formats are specific to their tool of origin, especially for the *vg toolkit* [91, 122] which uses a combination of index files (the *succinct graph index* XG [147]) and coordinates files.

However, there is one format that stands out, with some derivatives: the **Graphical Fragment Assembly (GFA)** format [148], designed to be a standard file format for *sequence graphs*. This format has two canonical versions with slight differences, mainly associated to the anchors of an edge on a node (which can be within a node instead of at its endpoint in GFA v2). A subset of GFA v1 has been proposed by Dr. Heng Li for his own tool minigraph [113] for creating a reference pangenomic graph, which has been strongly debated in a series of blog posts with Dr. Garrison [149-151].

While different by nature, GFA v1, GFA v2 and rGFA (Dr. Li's take on the GFA format) share common conceptions:

- each record within the file represents one graph object, with a line header precisizing its nature (i.e. whether the object is a node, an edge, a path...)
- they encode sequence graphs, therefore nodes represent 'segments', sequences of nucleotides and edges represent succession between two nodes (including the orientation and strands of the nodes)
- each object may have an ID for further identification and reference
- additional tags similar to SAM optional tags [152] may be added to describe supplementary information

Based on GFA v1, rGFA (short for '*reference GFA*') makes use of these additional tags to anchor stable coordinates keeping track of the origin of new elements whenever they are added to the graph. GFA v2 changes parts of its terminology compared to GFA v1 and adds new graph objects (**F**ragment lines, **O**rdered vs **U**nordered groups...). An annotated visual example of such a GFA v2 file is presented in Appendix I.

Somehow GFA v2 is not supported by the current state-of-the-art pangenome tools, minigraph [113] and ODGI [124], which rest on GFA v1 instead. This led GFA v1 to keep on being upgraded with a v1.1 adding **W**alk lines (more detailed versions of GFA v1's **P**aths, introduced on February 11, 2022) and v1.2 adding **J**ump lines (typically used to represent Gaps⁸ between indirectly connected segments, introduced on June 3, 2022).

Other formats exist on top of GFA: the **Graph Alignment/Map (GAM)** [153] format—an equivalent of the **Binary Alignment Map (BAM)** [154] format applied to variation graphs—and **Graph Alignment Format (GAF)** [155] are both formats storing mappings on GFA files.

Moreover, there are tools enabling conversions between pangenome graphs and more classic files. For example, *hal2maf* from the HAL package [156] is used within Progressive Cactus to transform the graph into MAF files. The same goes for *maffer* [157], which work on variation graphs. Another example: PARROT (in preparation), which is a tool dedicated to conversions between pangenomic formats.

⁸ Funnily enough Gaps were already present in GFA v2, back in 2018.

II. Overview of tools with pangenome visualization

The lack of standard definitions, tools, and formats in pangenomics prevented the wide development of related visualization tools. When I started working on pangenomics in 2018, there were really few dedicated tools, mostly for bacterial pangenomics (anvi'o [158], PanViz [159], panX [160]...). In this section, I provide an overview of the existing pangenomic visualization tools I am aware of, categorize them according to the concepts they use and detail the most representative for each category.

A. Visual representations, visualization tools

A visual representation alone is not a visualization tool; it is a collection of visual encodings (i.e. the visual channel that will convert information from the data domain to something visually perceptible, for example: hue, shape, or size of a graphical mark) used to represent something visually—simply put, it is the way something is showed. A visualization is an applied visual representation, the result of a chain of information conversion from a set of data. In datavis, it should moreover be tailored for a communication purpose.

On the other hand, a “tool” implies some usage and manipulation, some level of interaction. I therefore differentiate the visualization tools (tools that enable the customization, dynamic exploration and more generally interactivity with visualizations) from the tools *with* visualizations (tools lacking a dedicated **User Interface (UI)** or with little to no emphasize on visualization or interactivity, for example with static charts created merely for illustration).

For example, out of the 40 pangenomic tools cited by Vernikos in the chapter “*A Review of Pangenome Tools and Recent Studies*” [98] of *The Pangenome* [22], 14 were described as *having* visualizations (their descriptive texts contained at least one word matching ‘*visualiz(ation)*’, ‘*representation*’, ‘*venn*’, ‘*graph*’, ‘*plot*’, or ‘*chart*’).

Among these, four rely on other tools for the visualization:

- **BGDMdocker** [161]: the advertised visualization is made by another tool, panX [160]. BGDMdocker is a pipeline that enables a connection between the building and visualization tools, but do not do any visualization directly.
- **LS-BSR** [103]: it creates matrices that can be visualized as heatmaps or clusters through third party tools, as said in their GitHub manual⁹. It also provides a script compatible with PanGP [162] for the visualization of statistics related to the pgAtlases created.
- **PANINI** [163]: again, the visualization available was not created by PANINI, but by Microreact [164], PANINI making the connection between the data created and the actual visualization platform.
- **PanTools** [117]: said to “*supports the construction and visualization of pangenomes hosting online tools and algorithms*” with a “*visual representation of the pangenome (...) based on generalized De Bruijn graphs*” [98], it creates pangenome databases as De Bruijn graphs but the visualization is entirely handled by Neo4j¹⁰—a generic graph visualization tool rather than a (pan)genome graph visualization tool.

Four of the remaining tools correspond to what I call tools *with* visualizations:

- **EDGAR** [165]: offers a variety of mostly static charts and plots (pie charts of core and variable genomes, venn diagrams, synteny plots...) for pre-defined datasets, divided between multiple sections. The users can select their genomes of interest and create plots

⁹ <https://github.com/jasonsahl/LS-BSR/blob/master/manual.md>

¹⁰ <https://neo4j.com/>

on-the-fly that appear after a few seconds of communication with the servers. While offering a high variety of visual representations, it lacks an integrative and unified UI that would enable an interactive exploration and/or comparison of these visualizations. It would therefore be best described as a collection of charts rather than a visualization tool.

- **Panaconda** [115]: generates ‘pan-synteny’ graphs in a format compatible with Gephi [166], NetworkX¹¹, and Cytoscape¹², generalist network visualization tools. It includes an in-house visualizer written in JavaScript (**JS**) and based on Gexf-JS¹³ which can display the layout created by Gephi. Compared with Gexf-JS, they added path highlighting within the graph. The visualization component is therefore mostly based off another tool and completes the actual main interest of Panaconda which is the creation of the pan-synteny graphs.
- **PanGeT** [167]: is a tool suite that computes clusters of genes to distinguish those that belong to the core genome from those that belong to the variable genome. The tool then outputs an interactive flower plot, with the core genes placed at the heart and the genes specific to one accession at the petals. Clickable hyperlinks either redirect to a database or download subsequent data (core genomes, annotations...). Since there is no UI and the hyperlinks are the only interactivity present, I therefore categorize it as a tool *with* visualization.
- **PanWeb** [168]: serves as a graphical interface for PGAP, a tool performing pangenome analyses, and produces static plots (for example, phylogenetic trees and the evolution of the number of genes within the pangenome for each additional genome). There is a proper UI, but visualization is only a minimal part of it and has no interaction, which is why I would not call it a visualization tool.

From this review, only **Harvest** [109], **Pan-Tetris** [143], **PanViz** [159, 169], **PanACEA** [170], **panX** [160] and **PGAP-X** [171] fit my definition of a visualization tool: having a UI dedicated to the creation and/or exploration of visualizations.

As all visual representations of pangenomes are of interest for my PhD, **I list in the following sections both visualization tools and tools with visualizations** but with a focus on the visualization tools. This section constitutes the only available review dedicated to pangenome visualization tools to my knowledge—though the awesome genome visualization list [172] does have a filter tag for pangenomes—and aims at their better characterization depending on the representations used (both conceptual and visual).

B. List of tools usable for visualizing pangenomes

During my PhD, I identified five main categories of visualization tools applied to pangenome datasets, depending on the type of pangenome definitions used and how they visually represent them: **Unspecific**, **Qualifying**, **Positioned**, **Structural**, and **Composite**, as summarized in Figure 9.

¹¹ <https://networkx.org/>

¹² <https://cytoscape.org/>

¹³ <https://github.com/raphv/gexf-js>

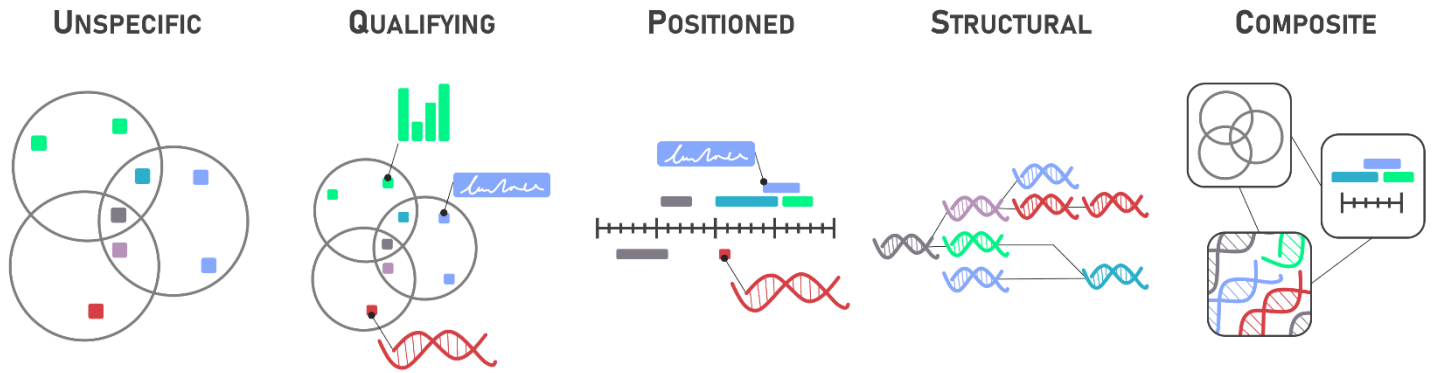


Figure 9: I identified five categories of pangenome visualization tools; Unspecific—generic and not dedicated to pangenome datasets. Qualifying—with pangenomic information, but no position. Positioned—anchored information on (pan)genomic coordinates. Structural—focus on sequences and their continuity. Composite—multiple categories within one tool.

In this section I list the visualization tools identified during my PhD, as showed in the Table 2 below, and associate each to one of the aforementioned categories depending on the implemented visual representations. Each category is then detailed with a subsequent comparative table of the related tools, and representative examples are highlighted. Modified versions of genome browsers and platforms are detailed in a dedicated complementary section (State of the Art II.C).

Table 2: Visualization tools for pangenomics are diverse; Some tools provide only limited visualization capacities. These 'tools with visualizations' are indicated by the * after their name. Tools are listed in alphabetical order, along with their main author(s) from publication or public GitHub repositories, date, reference, and category: one out of Unspecific (U), Qualifying (Q), Positioned (P), Structural (S) and Composite (C). Special implementations with genome browsers are detailed in the subsequent section State of the Art II.C.

Tool name	Reference	Category
anvi'o	Eren et al., 2016 [173]	Q
Bandage	Wick et al., 2015 [174]	S
Chromatiblock	Sullivan & van Bake, 2019 [175]	P
Coinfinder *	Whelan et al., 2020 [176]	Q
Crop-Haplotypes	Brinton et al., 2020 [177]	S
Cytoscape	Shannon et al., 2003 [178]	U
GCV	Cleary & Farmer, 2018 [179]	P
GenomeRing	Herbig et al., 2012 [180]	S
Gephi	Bastian et al., 2016 [166]	U
gfaestus	Fischer, 2021 [181]	S
GfaViz	Gonnella, 2018 [182]	S
graphgenomeviewer	Diesh, 2022 [183]	S
Harvest	Treangen et al., 2014 [109]	Q
Hierarchical Sets *	Pedersen, 2017 [184]	Q
IGGE	Kusnetsov et al., 2021 [185]	S
Microreact	Argimón et al., 2016 [164]	Q
MoMI-G	Yokoyama et al., 2019 [186]	C
neo4j	--, --, [187]	U
ODGI *	Guarracino et al., 2022 [124]	S
PanACEA	Clarke et al., 2018 [170]	P
Panache	Durant et al., 2021 [188]	P
Panaconda *	Warren et al., 2017 [115]	S
PanGeT *	Yuvaraj et al., 2017 [167]	Q

panGraphViewer	Yuan	2021	[189]				S
Pan-Tetris	Hennig et al.,	2015	[143]		Q		
pantograph	Seaman et al.,	2020	[190]				S
PanViz	Pedersen et al.,	2017	[159]		Q		
panX	Ding et al.,	2018	[160]		Q		
PGAP-X	Zhao et al.,	2018	[171]				C
PGV	Liang & Lonardi	2021	[191]			P	
Phandango	Hadfield et al.,	2017	[192]		Q		
plotsr	Goel & Schneeberger	2022	[193]				S
Sequence Tube Map	Beyer et al.,	2019	[194]				S
TASUKE+	Kumagai et al.,	2019	[195]			P	
UpSet	Lex et al.,	2014	[196]		U		
vg toolkit *	Garrison et al.,	2018	[90]				S

1. Unspecific

I define the **unspecific** visualization tools as tools that are not dedicated to pgAtlases, pangenomes, or even sometimes genomes in general, and uses generic files or files formats destined to other use cases. They could be used in a wide range of situations, and often lack access to biological metadata of interest for pangenomics. Tools creating Venn diagrams or generic network visualization tools are examples of tools that can fit in this category), as listed in the Table 3.

Table 3: Unspecific tools are generic tools not dedicated to (pan)genomics; Additional information is provided in the dedicated subsections. (*) highlight that complementary details are available. Visualization tools are marked with a (•) before their name, to better distinguish them from tools with visualization(s). The columns provide a tool's name, reference, year of publication, compatible type(s) of datasets, visual representations used, language of development, level of interactivity. 'vis.' stands for 'visualization'.

Vis. tool	Name	Year	pgAtlas or Pan.	Vis. representation	Language	Interactive vis.
■	Cytoscape [178]	2003	Pan.	Network	Java*	Yes
■	Gephi [166]	2009	Pan.	Network	Java, OpenGL	Yes
■	neo4j [187]	NA	Pan.	Network	Java*	Yes
■	UpSet [196]	2014	pgAtlas	Barcharts of set intersections*	JS*	Yes*

a. Cytoscape

Cytoscape [178] is a network visualization library, whose JS version (Cytoscape.js) has been used within multiple pangenomic tools: metaPGN [197] (see an example in Appendix II), panaconda [115], Panakeia [198], pangraphviewer [189]...

The JS library is meant to complete other applications and would not constitute a full web application on its own. Visualizations of network datasets created with the desktop application Cytoscape can be exported to Cytoscape.js, which can run on all modern web browsers.

Automation for integration within Python and R workflows [199] has been developed through the years and Cytoscape is now a well-known tool for visualizing biological networks.

b. Gephi

Gephi [166] is another tool for visualizing network graphs, which has been used for the visualization of pangenomes with Coinfinder [176], Panaconda [115], and PPanGGOLiN [70] (as illustrated in Appendix III). This tool has been built for network manipulation and exploration.

c. neo4j

neo4j [187] has been used by PanTools [117] for the representation of network graph (see Appendix IV). This tool has drivers that make it compatible with multiple languages, including Java, JS and Python among others.

d. UpSet

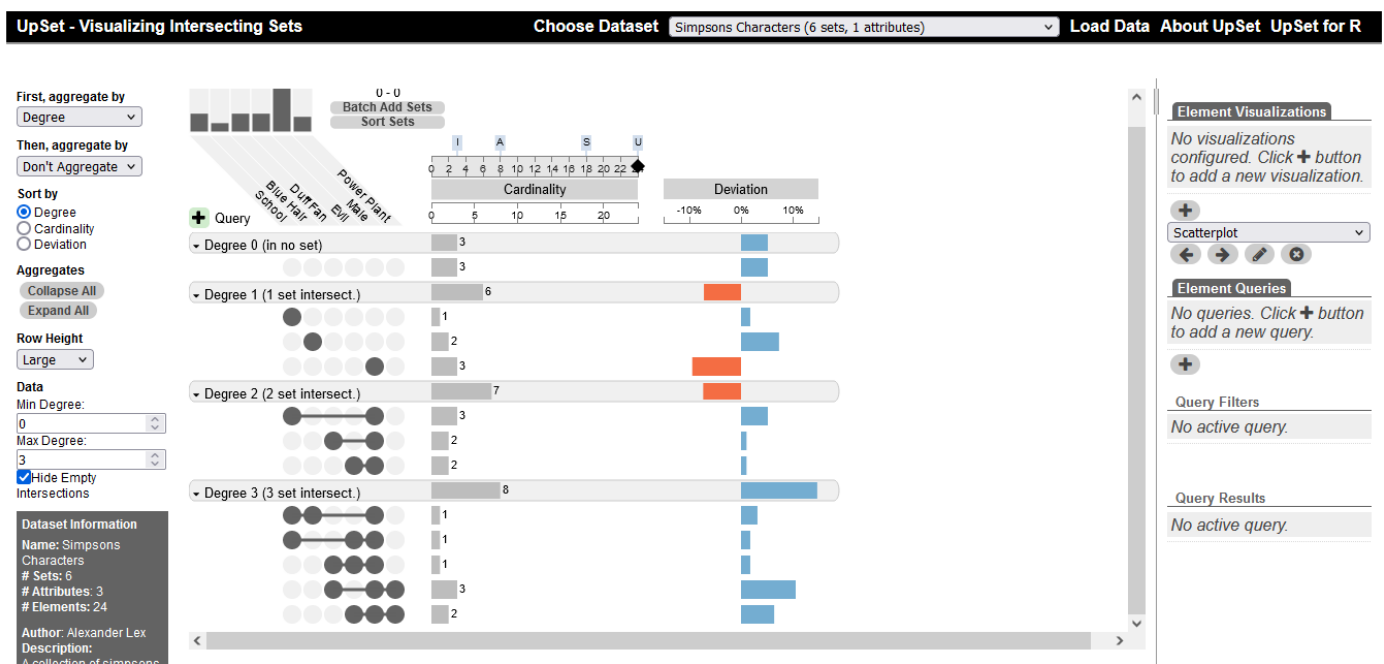


Figure 10: UpSet offers an alternative to Venn diagrams by displaying barcharts and summary statistics for each set intersection; Screenshot from UpSet v1, with the default “Simpson characters” dataset: <http://vcg.github.io/upset/>

UpSet is the implementation of an alternative visual representation of sets, compared with Venn diagrams which are limited in the number of sets they can accurately represent—more than three or four sets and they become hard to read as well as poorly representative of the number of elements contained by each set intersection. Instead, it shows barcharts for each set intersection, with additional derived data as shown in Figure 10, and provides interactivity through filtering, labelling, and many others. UpSet has been derived with R [200], Python¹⁴, and JS¹⁵ among other versions¹⁶, and a second implementation (UpSet2¹⁷ [201]) adds undo/redo actions and the possibility to download Comma Separated Value (CSV) files from sets of items.

¹⁴ <https://github.com/ImSoErgodic/py-upset>

¹⁵ <https://upset.js.org/>

¹⁶ <https://upset.app/implementations/>

¹⁷ <https://vdl.sci.utah.edu/upset2/>

2. Qualifying

Every visualization tool that serves for the high-level representation of pgAtlases falls into the **qualifying** category, listed in Table 4. They are tools that list global properties from a pangenome or pgAtlas, for example with plots about summary statistics, without providing any information of position within and between genomes. However, compared with unspecific tools, they do provide biological information related to the datasets that they display and/or specific to pangenomics (gene ontology, core and variable genomes, DNA sequence alignments, other biological metadata...).

Table 4: Qualifying tools provide (summary) information on pan-gene atlases without any coordinate system; As in Table 3: additional information is provided in the dedicated subsections. (*) highlight that complementary detail are available. Visualization tools are marked with a (*) before their name, to better distinguish them from tools with visualization(s). The columns provide a tool's name, reference, year of publication, compatible type(s) of datasets, visual representations used, language of development, level of interactivity. 'vis.' stands for 'visualization'.

Vis. tool	Name	Year	pgAtlas or Pan.	Vis. representation	Language	Interactive vis.	
■	anvi'o	[173]	2015	pgAtlas	Circular PAV matrices & dendrogram	Python, JS	No*
	Coinfinder	[176]	2020	pgAtlas	Gene association networks, PAV heatmaps	C++, R	No
■	Harvest (<i>Gingr</i>)	[109]	2014	pgAtlas	Dendrogram, sequence alignments	C++	Yes
	Hierarchical Sets	[184]	2017	pgAtlas	Dendrogram, icicle plots	R	No
■	Microreact	[164]	2016	pgAtlas	Dendrogram, scatterplot, maps, swarmplot...	JS	Yes
	PanGeT	[167]	2017	pgAtlas	Flower plot	LaTeX	No*
■	Pan-Tetris	[143]	2015	pgAtlas	Heatmap-like PAV matrix	Java	Yes
■	PanViz	[159]	2017	pgAtlas	PCA Scatterplot, treemap...	JS	Yes
■	panX	[160]	2018	pgAtlas	Dendrogram, pie chart, sequence alignments...	JS	Yes
■	Phandango	[192]	2017	pgAtlas	Dendrogram, heatmap	JS	Yes

a. anvi'o

anvi'o [173, 202] has a workflow for pangenomics [158], dedicated to gene clusters extendable with contextual information and aimed at prokaryotes. It has been used for pangenomic and even metapangenomic studies [203] and displays PAV matrices in a semi-circular heatmap (cf Appendix V) along with a dendrogram for the phylogeny of the genomes involved.

The result visualization is not directly interactive: its interface enables fine tuning and modifications of the display (group by cluster, customization of the element that should appear...),

with plenty of possible options¹⁸. It acts as a control panel for zooming, panning, refreshing the view... with sorting options and others, but the visual representation is not interactive in itself. Hovering events, while available, do not display tooltips but print values in a ‘mouse tab’ within the menu instead. It is an actively maintained tool with regular updates so far.

b. *Coinfinder*

Coinfinder [176] produces static plots of association/dissociation networks (one node being one gene family, proximity encodes frequency of cooccurrence) with Gephi, and PAV heatmaps (see Appendix VI). It works on gene clusters and is still maintained.

c. *Harvest*

Gingr, a dedicated visualization module used by **Harvest** [109] for the phylogeny of core genes. It provides a fair number of details on the underlying **core** pan-genes, with a phylogeny, sequence alignments and switches between visual representations depending on the zoom level, as illustrated in Appendix VII. Conceived to work with *Parsnp*, another Harvest module, it still can accept standard file formats from other tools (multi-FASTA, xMFA, Newick, and VCF). Written in C++, it offers interesting visualization functionalities (Fisheye, semantic zoom...) and had its latest release in October 2016, which may indicate that the tool is not actively maintained.

d. *Hierarchical Sets*

Hierarchical Sets [184] have been created to bypass the limitations of UpSet [196] regarding big datasets with numerous sets, which are plenty in pangenomics. Its algorithm computes and then displays clusters of genes as Dendrograms and Icicle plots (see Appendix VIII) through an R package. It claims to bypass UpSet’s limitations by showing only intersections between closely related branches of a family tree, hiding the information about intersection of loosely related sets.

e. *Microreact*

Microreact [164] has been used to display pangenomes with PANINI [163]. It provides scatterplot representations of dispensable genes (clustered with t-SNE for example), built with JS. Contrary to many other *qualifying* tools, it provides information on **variable** genes too. Interactivity is achieved through multiple linked views (Dendrograms, maps, swarmplots... as illustrated in Appendix IX) within a fast interface. The integration with PANINI seemed unnecessary and a bit dysfunctional (I could not use the tutorial and the transitions between PANINI and Microreact were slow).

f. *PanGeT*

PanGeT [167] is an oddball, as it is written in LaTeX. Its flower plots (simplified Venn diagrams, as seen in Appendix X) show the distribution of **core** and unique genes. It has limited interactivity, through hyperlinks that enable the download of cluster data.

g. *Pan-Tetris*

Pan-Tetris [143] represents genic PAV matrices, with information on the genes’ strands, meters for the presence rate, and the possibility to merge complementary cluster lines (see Appendix XI). Built in Java for an in-house database (the ‘*SuperGenome*’), it is best suited for pangenome files from PanGee, an unpublished tool, but can handle generic tab-separated formats too though this did not bear good results during benchmarks (see more about it in Panache I).

h. *PanViz*

PanViz [159] is a JavaScript application that displays a scatterplot, dendrogram, table and *Circos*-like [204] plots, the latest being swappable with two alternative representations (see Appendix

¹⁸ <https://merenlab.org/2016/02/27/the-anvio-interactive-interface//#using-the-anvio-interactive-interface>

XII). This representation shows explorable clusters of genes, categorized depending on a dynamic core threshold and **Gene Ontology (GO)**. Data files can be built from a companion R tool called *PanVizGenerator* [169]...

i. *panX*



Figure 11: panX provides multiple visual representations detailing the core genes; In this screenshot taken from the example instance set for *S. pneumoniae* by Croucher et al. [205], panX displays a pie chart with a customizable cutoff between core and variable ('accessory') genome, densities of gene count rank and length, a filterable table of the core genes, sequence alignments, and dendrograms for both the strains and genes. On small computer screens all these visual representations cannot appear at once, which results in mandatory scrolling interactions and harder comparisons.

panX [160] is an interactive interface built with JS and dedicated to bacteria. It enables a dynamic visualization of pgAtlases, with multiple (linked) views and representations (as illustrated in Figure 11) supporting various tasks. It comes with a clean interface and a high level of interactivity for fine tuning the visualization as well as a numerous example dataset. This visualization tool however focuses on **core** genes but not the **variable** ones, and its interface cannot fit nicely on a single screen which can hinder the **User eXperience (UX)**.

j. *Phandango*

Phandango is a JS interface which can be used for displaying results from Roary [108]. It has been applied to at least two pangenomic studies, on *Salmonella* [206] and *Streptococcus* [205] (as showed in Appendix XIII). It accepts trees, CSV metadata, or gff3, and displays multiple views, with a dendrogram and a heatmap of present blocks, whose order depends on precomputed files and do not reflect actual annotation.

As for interactivity, a user can modify the sizes of the sub-panels, zoom in and out, pan, and access tooltips and details on certain portion of the visualization but not on the PAV matrix itself. Moreover, it is mentioned that there is a risk of crash when the files outputted by Roary become too big. The latest tag has been put on 2016, and there has been no commit since 2018 which may indicate a lack of maintenance.

3. Positioned

Positioned visualization tools, showed in Table 5 below are tools that have a focus on pangenomic blocks (LCBs, panBlocks...), anchored onto a coordinate system. That coordinate system could be an existing reference genome or a panreference (either built by anchoring additional material within a base reference or stacking it at the end, see State of the Art I.F.1.a). Compared with *qualifying* tools they have the additional information of position within genome(s). However, they lack the information on sequence continuity and succession that would be needed within a *structural* tool, mainly because of the fragmented nature of pangenome blocks.

Table 5: Positioned tools anchor fragmented pangenomics data on a coordinate system; As in Table 3: additional information is provided in the dedicated subsections. (*) highlight that complementary details are available. Visualization tools are marked with a (*) before their name, to better distinguish them from tools with visualization(s). The columns provide a tool's name, reference, year of publication, compatible type(s) of datasets, visual representations used, language of development, level of interactivity. 'vis.' stands for 'visualization'.

Vis. tool	Name	Year	pgAtlas or Pan.	Vis. representation	Language	Interactive vis.
■	Chromatiblock [175]	2019	Pan.	Alignment of core or variable blocks	Python	Yes
■	GCV [179]	2018	pgAtlas	'Beads-on-a-string' genes, Circos, Dotplots	TypeScript	Yes
■	PanACEA [170]	2018	pgAtlas	Dendrogram, circular pan-chromosome, fGR view	Perl*	Yes*
■	Panache [188]	2021	Pan. & pgAtlas	PAV heatmap-like in a browser	JS	Yes
■	PGV [191]	2021	Pan.	Dotplot, block browser	Python, JS	No*
■	TASUKE+ [195]	2019	Pan.	SNPs heatmap and annotation track	HTML5, MySQL	Yes

a. Chromatiblock

Chromatiblock [175] compares syntenic haplotype blocks in multiple prokaryote genome alignments and offers two views (as seen in Appendix XIV), built with Python from MAF files. One of the views shows **core** blocks both color-coded and aligned based on their position on the first genome. Non-core and unique blocks (displayed as patterned rectangle) are then positioned relatively to these core blocks. The second view focuses on the alignment differences: **variable** blocks are displayed in a PAV matrix, one column per block.

It offers options for zooming, panning, and interactive highlighting of common regions across genomes, as illustrated in their available demo for *C. difficile*¹⁹. Both its latest release and commit date back to September 2020.

¹⁹ https://mjsull.github.io/chromatiblock/C_difficile.html

b. GCV – Genome Context Viewer

The Genome Context Viewer [179] is a pangenome synteny viewer written in TypeScript which focuses on the ‘genome contexts’, the relative ordering and orientation of the gene annotations. It has three different displays: the micro-synteny view represents genes as ‘beads-on-a-string’, colored depending on gene families and order based on any track; dotplots can show pairwise comparison of gene loci; the macro-synteny view shows the macro position of synteny blocks in a circos-like plot (see Appendix XV).

It has multiple interactive options (of navigation, highlight on hovering, linked views...) within a clean interface, along with a clear documentation and explanations. The latest release was made in 2021 but the tool is still actively maintained and improved, with new versions incoming.

c. PanACEA

PanACEA [170], built in Perl and outputting JS files, focuses on flexible **Genomic Regions (fGRs)** and their smaller flexible **Genomic Islands (fGIs)**. They are inscribed within a bacterial ‘*pan-chromosome*’ or ‘*pan-scaffold*’ built from core genes, hence a circular display as illustrated in the Figure 12.

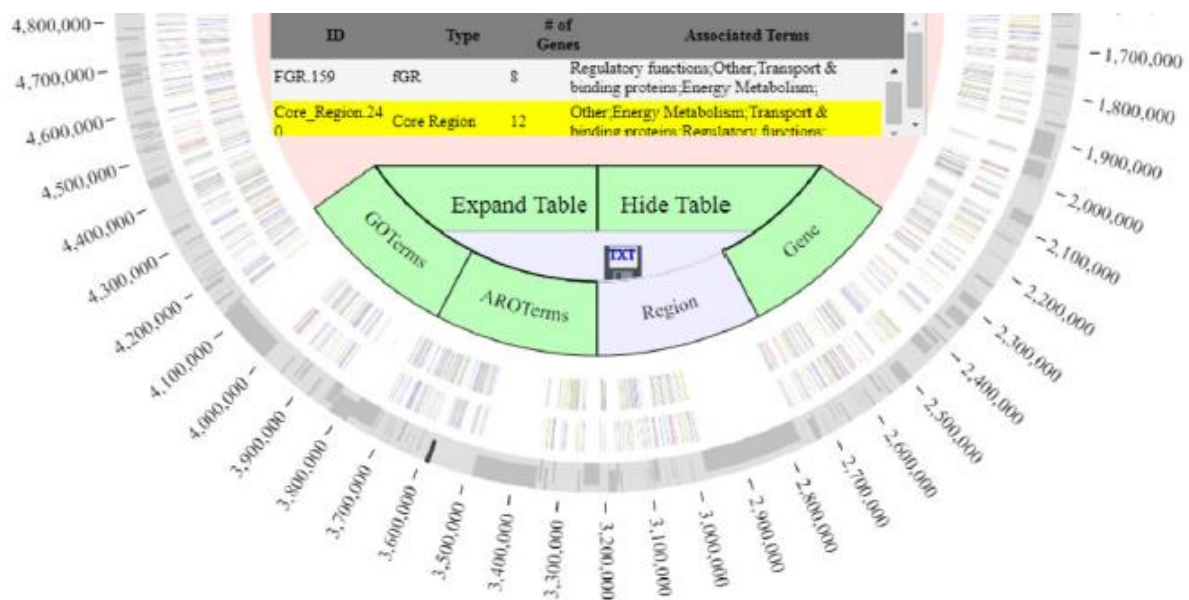


Figure 12: PanACEA's pan-chromosome view displays genes positioned on a circular bacterial coordinate system; Only the bottom half of the pan-chromosome view is showed here; variable regions are showed in dark gray and core in light gray, with core genes colored by protein functions. The center of this representation has buttons and a table for exploring the underlying data and clicking on the visualization can redirect to other visual representations, one being dedicated to the linear comparison of fGIs between genomes. Figure from [170]

It has multiple visual representations available and takes outputs from usual clustering tools (like *PanOct* [106]) but limited interactivity: a user can customize the colors, choose which fGI to display by clicking and use text search and export functions from the central buttons within the pan-chromosome view. It lacks a demo version and does not have much documentation. The last commit dates to 2018, without any tagged release, indicating a lack of maintenance.

d. Panache

Panache [188] is the first tool I developed during my PhD, as detailed in Panache's chapter. Built in JS, it represents a PAV matrix of positioned panBlocks as a heatmap within a browser-like interface.

e. PGV

The **PGV** Genome Browser is a tool written in Python and JS which has been created to represent large eukaryotic genomes and considers both gene and non-coding sequences. Its browser (shown in Appendix XVI) represents genomic blocks of alignment (built with *progressiveMauve* [84]), colored depending on their presence status (**core**, **variable**, unique...) in the pangenome and laid out following a ‘*consensus ordering*’ which is not based on a single reference.

It has limited interaction: double-clicking can show block names and connections across genomes, and the genome order can be modified by click-and-drag events.

f. TASUKE+

TASUKE+ [195] is a multi-genome web browser, built primarily for **Genome Wide Association Studies (GWAS)** and resequencing data (in FASTA, GFF, VCF...). It has been used in a pangenomic context with the *Brassica napus* database BnPIR²⁰ [207], for displaying sequence variation information.

It represents a heatmap of SNPs density on sequences, along with gene annotations (see Appendix XVII). The coordinate system is based on one reference. It offers multiple interactive options: data export and filtering, (semantic) zooming, panning, tooltip on click, navigation via buttons...

4. Structural

Structural pangenome visualization tools emphasize the continuity and successions of sequences within the pangenome and their structural variations. They do not rely on panreferences as *positioned* tools do. Tools displaying genome graphs are good examples of this category, among others listed in the Table 6 below.

Table 6: Most of the structural tools are based on genome graphs; As in Table 3: additional information is provided in the dedicated subsections. (*) highlight that complementary detail are available. Visualization tools are marked with a (·) before their name, to better distinguish them from tools with visualization(s). The columns provide a tool’s name, reference, year of publication, compatible type(s) of datasets, visual representations used, language of development, level of interactivity. ‘vis.’ stands for ‘*visualization*’.

Vis. tool	Name	Year	pgAtlas or Pan.	Vis. representation	Language	Interactive vis.	
■	Bandage	[174]	2015	Pan.	Genome graph	C++	Yes
■	Crop-Haplotypes	[177]	2020	Pan.	Shared blocks within browser	JS	Yes
■	GenomeRing	[180]	2012	Pan.	Genome as lines through blocks in circular rings display	Java	Yes*
■	gfaestus	[181]	2021	Pan.	Genome graph	Rust	Yes
■	GfaViz	[182]	2018	Pan.	Genome graph	C++	Yes
■	graphgenomeviewer	[183]	2022	Pan.	Genome graph	JS	Yes
■	IGGE	[185]	2021	Pan.	3D VR layout of genome graph	(Unity Game Engine)	Yes

²⁰ http://chi.hzau.edu.cn/bnapus/tasuke-plus_20200305/tasuke_www/

	ODGI	[124]	2022	Pan.	Table-like genome graphs: nodes on top, edges below	C++	No
	Panaconda	[115]	2017	Pan.	Graph	Python	Yes*
■	panGraphViewer	[189]	2021	Pan.	Genome graph with customizable node shapes	Python, Perl, JS	Yes
■	pantograph	[190]	2020	Pan.	Connected PAV matrices	JS	No*
	plotsr	[193]	2022	Pan.	Genomes as lines, with strokes in between representing SVs	Python	No
■	Sequence Tube Map	[194]	2019	Pan.	Genomes as lines through block in linear display	JS	Yes
	vg toolkit	[90]	2018	Pan.	Short graph of sequences	C++	No

a. Bandage

A most famous tool in graph pangenomics is **Bandage** [19, 174], written in C++, which displays a raw graph (or multiple subgraphs) from an assembly graph file which can be in different formats: LastGraph (*Velvet* [208]), FASTG (*SPAdes* [209]), Trinity.fasta (*Trinity* [210]), ASQG (*String Graph Assembler* [211]) or GFA. It has been popularized in pangenomics for the visualization of GFA files [63], as it can provide an overview of the whole genome graph or be used on smaller regions. It makes for a great first impression of a gfa file but gives no information about the sequences within each node, and only shows the overall layout of the graph, with no access to the detail (as seen in Figure 13).

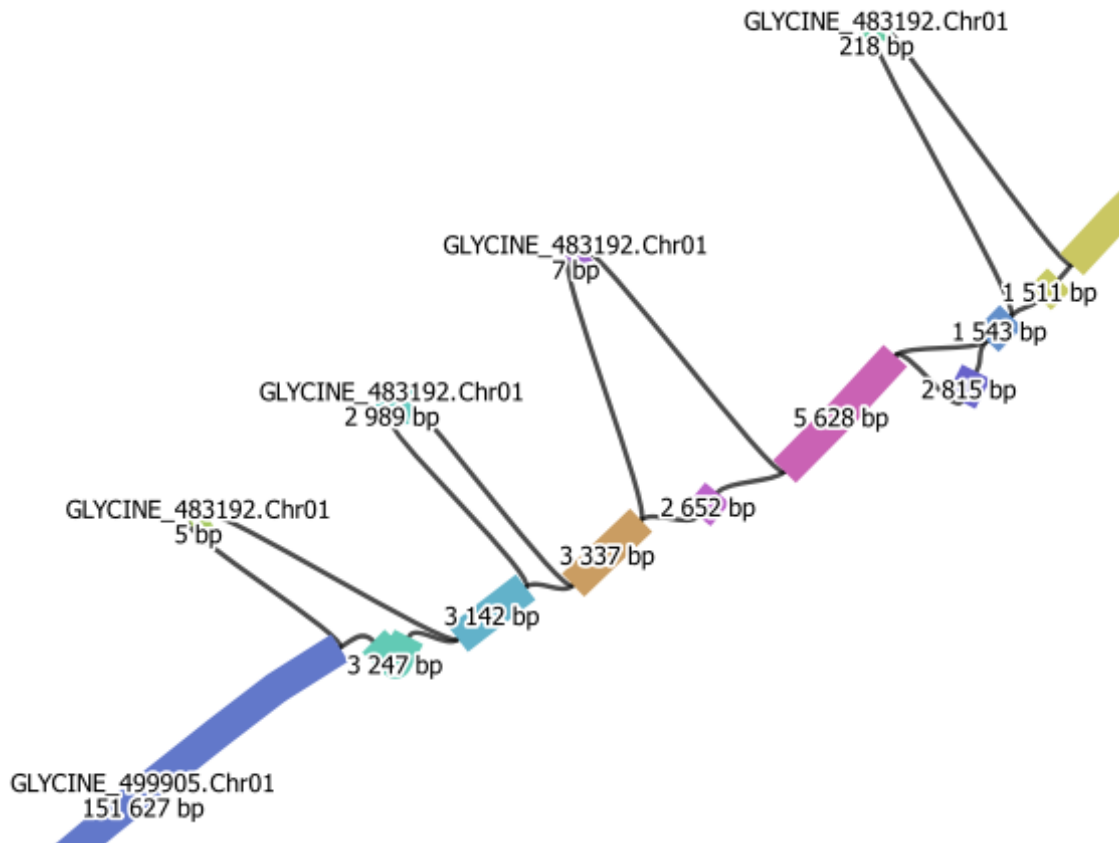


Figure 13: Bandage gives details on the GFA's node succession, their ID, and size in bp; Close-up view of an in-house *Glycine soy* pangenome (unpublished) written as a GFAv1 file and visualized with Bandage. The information on node sequences for example is not available through the interface, which overall lacks pangenome-dedicated features.

Furthermore, it can take a long time when loading graphs with numerous nodes, and it even sometimes crashes when this number is too high, as it is the case for pangenome graphs. A last oddity is that said layout might change at every attempt to look at it: nodes colors, absolute positions and curvatures change whenever the visualization is loaded again, even though no changes are applied to the graph file. This can be confusing, especially since big graphs can quickly be tangled, and it makes it harder to find the same interesting location on multiple occasions by using visual clues only.

The original Bandage reached maintenance stage, and a newer version, **Bandage-NG** [212], is being developed and claims to greatly improve the memory usage and drawing speed of Bandage, with additional features for path visualization with GFAs.

b. Crop-Haplotypes

Crop-Haplotypes²¹ is a JS interface that has been featured at the ISMB/ECCB 2021 BioVis conferences²². Built for the 10+ wheat genomes project [177], it displays the haplotype blocks obtained from 15 chromosome-level to scaffold-level genome assemblies, one chromosome at a time. Each assembly is represented as a succession of haplotype blocks, ordered according to their positions in each genome.

Common blocks are encoded with color and can be highlighted across genomes by hovering them, and the exact equivalent positions on other assemblies are displayed in real time (as showed in

²¹ <http://crop-haplotypes.com/>

²² https://www.youtube.com/watch?v=ViihSiAmO_c

Appendix XVIII). Other interactive functions include on click fixation of the coordinates on display, exploration and filtering options, zooms...

c. GenomeRing

GenomeRing [180], written in Java, offers an interesting visual representation of pangenomes. Each species is attributed a line which traverses (if present) or avoids (if absent) pangenomic blocks divided into two 'rings', corresponding to the backward and forward strands (as illustrated in Appendix XIX). It works on in-house pangenomes and is compatible to their other in-house tool Mayday [213] which can link the visualization to a browser with gene annotation for example. Interactivity is limited to zooming, rotating, and panning the view.

d. gfaestus

gfaestus [181] is a Rust implementation of a genome graph visualizer, similar to Bandage (see Appendix XX), and is used by the people that created ODGI [124]. It enables zooming, panning, and hovering.

e. GfaViz

GfaViz [182] enables the visualization and manipulation of sequence graphs in GFA format, as illustrated in Appendix XXI. The latest commit was made in February 2019, potentially indicating a lack of maintenance.

f. graphgenomeviewer

graphgenomeviewer [183] is a project, initiated during the Bioinformatics Community Conference CoFest 2020, which handles the visualization of GFAv1 and a subset of GFAv2. Its simple interface (see Appendix XXII) enables some color customization, along with zooming, panning, and manually moving graph nodes.

g. IGGE -The Immersive Graph Genome Explorer

The **Immersive Graph Genome Explorer** [185] is set in **Virtual Reality (VR)** for the 3D exploration of genome graphs (featured in Appendix XXIII) from successively converted GFA format. Usable with Oculus, it aims at clearer graph layouts and enables layout transformations (both single-node and multi-node movements are available), genome sequence data extraction from nodes, bookmark of regions of interest... An official version is in the works to expand this proof-of-concept project.

h. ODGI

ODGI [214] is a collection of tools usable on variation graphs. Two of its commands will display genome graphs: *odgi draw* can create a static 2D graph layout, and *odgi viz* creates a special static representation with nodes on top in a PAV-like display and edges below, as showed in Figure 14.

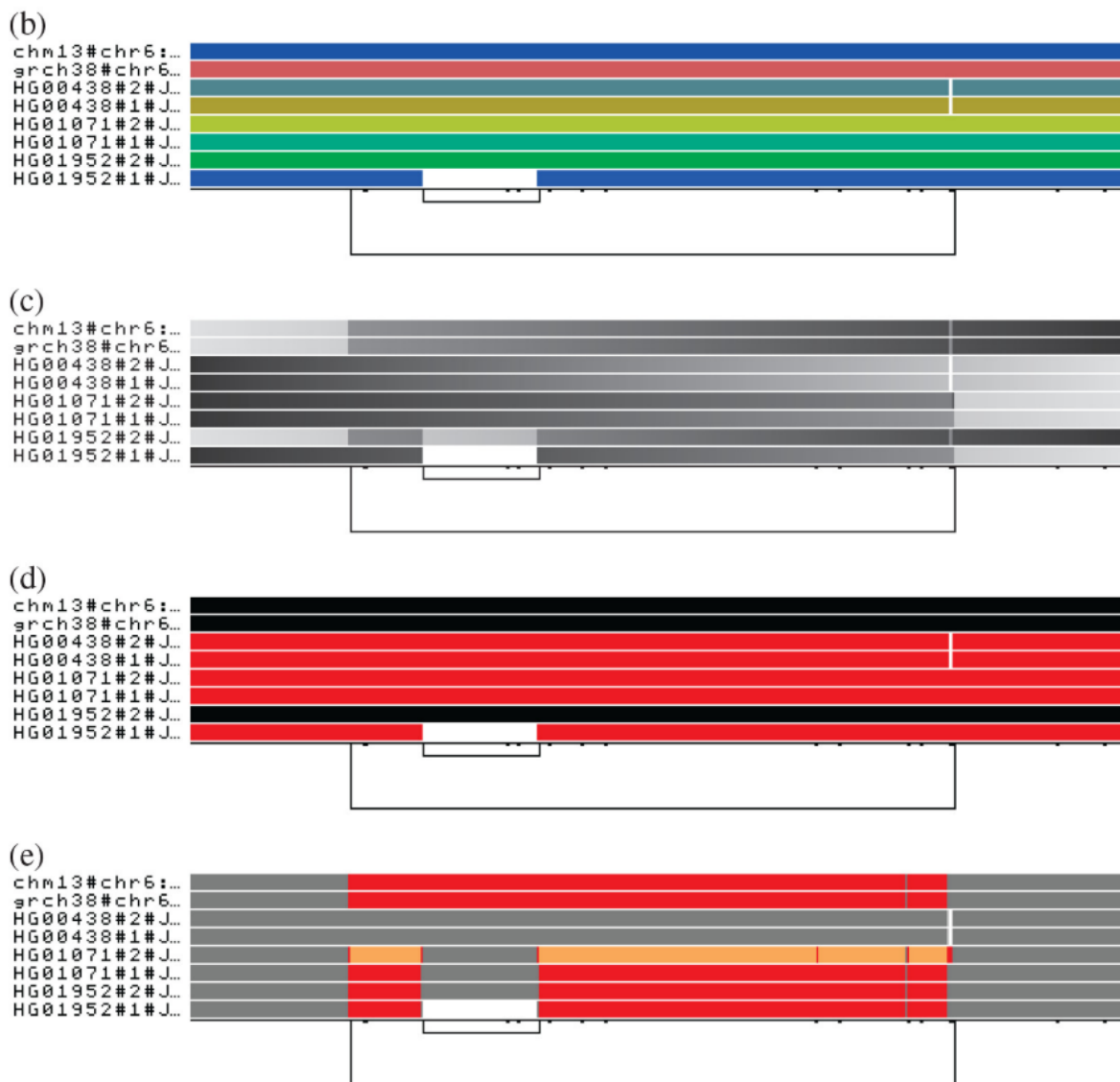


Figure 14: *odgi viz* draws genome graphs with ordered nodes on top and edges below; Different sets of color codes can be used: color per genome, light-to-dark gray gradient depending on position, black/red encoding of strand, color encoding of repeat number (red = one repeat, orange = two repeats). Figure from [124]

It is better suited for small graphs or portions of graphs, for example focusing on a genic region. When used on whole graphs they can become hard to read depending on the overall linearity of the graphs, or if no cleaning and ordering step was made, as illustrated in the Figure 15 below.

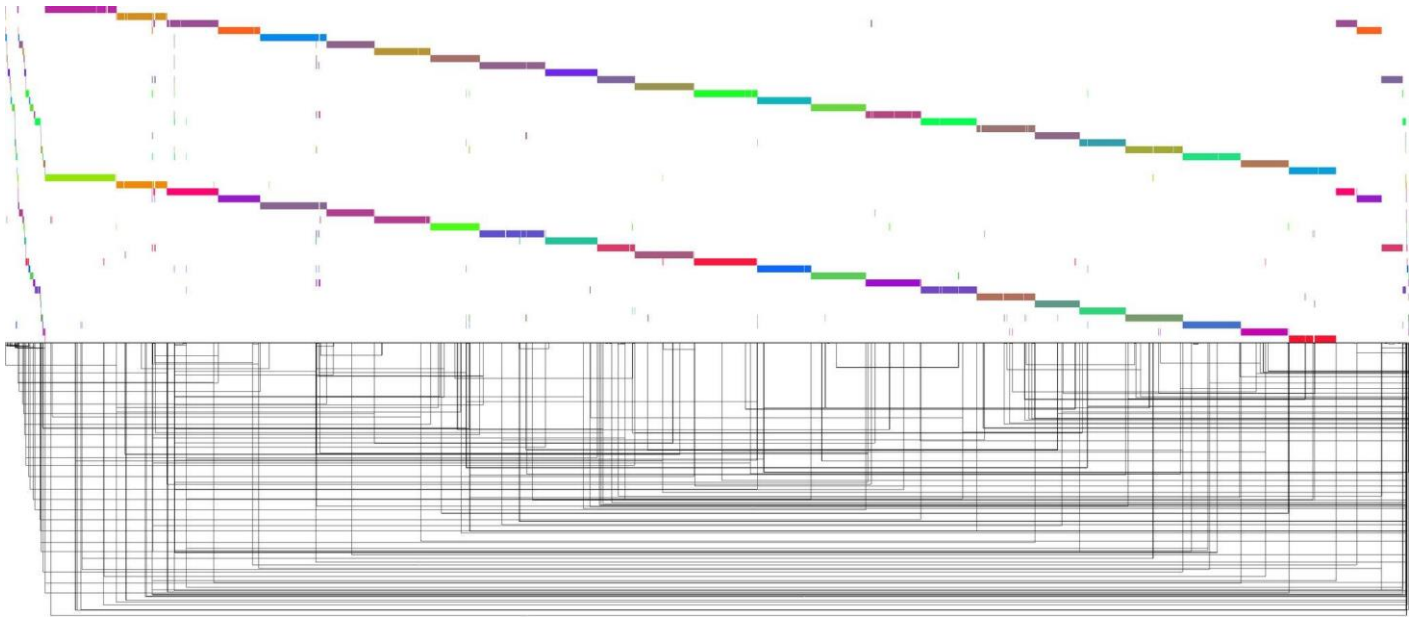


Figure 15: The odgi viz representation of raw graphs can be messy; Here the graph has been built from two *Medaka* genomes. Figure taken from Erik Garrison’s Twitter thread, 2019

i. Panaconda

As described in State of the Art II.A, **Panaconda** displays graphs built with Gexf-JS and graph layouts from Gephi. Each node is associated with an annotation, and details are available on a left panel, as showed in Appendix XXIV.

j. panGraphViewer

panGraphViewer [189], written in Python, has been built as an alternative to Bandage and focuses on subgraphs rather than whole genome graphs. Based on vis.js or Cytoscape.js depending on the graph’s size, it offers the possibility to attribute different shapes to nodes depending on the SVs (SNPs, Deletion, Insertion, Inversion, Duplication, Translocation) they depict compare with ‘backbone nodes’, as seen in Appendix XXV.

k. pantograph

pantograph [190, 215] (not to be mistaken with Pantograph [216] which is a method for metabolic model reconstruction) proposes an alternative representation of pangenome graphs as connected PAV matrices, where the actual succession of blocks within genomes can be deduced from arrows connecting the different pangenomic blocks (as seen in the Figure 16). Unfortunately, the original version built for the virtual BioHackathon 2020 has been abandoned and the related demo is not available anymore.

Nested Inversions - Read Left to Right

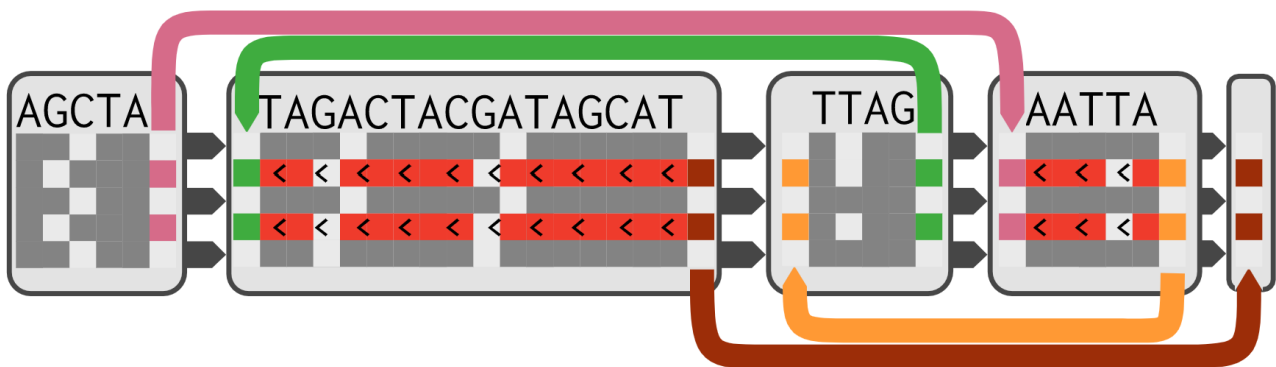


Figure 16: *pantograph* represents all genomes within a pangenome as linear paths through panBlocks represented as PAV matrices at a nucleotide level; Individual paths always stay on one line instead of going up and down as in *Sequence Tube Map* [194]. Colored arrows flanking the panBlocks show where rearrangements disrupting the order of block succession occur. This indicates for a given path whether the following sequence is that of the next block on the right or if it jumps somewhere else within the visual representation. Image taken from <https://graph-genome.github.io/pantograph.html>

A new version being developed by Computomics [217] is the successor of this collaborative project, but it is not widely available yet. A demo of this C++ version, featured in Appendix XXVI is available on a new dedicated website²³. The exact pricing of this newest version is unknown.

waragraph [218] has been described²⁴ as a potential spiritual successor to *pantograph*, but there is not enough documentation as of June 2022 to have a good idea of its capacities.

l. plotsr

plotsr [193] is a tool written in Python that enables the creation of static plots representing pairwise comparisons of the structures of successive genome assemblies. It displays genomes as lines, and rearrangements (inversions, translocations, duplications) as colored connections between the lines, linking the equivalent positions (see Figure 32). Multiple plotting options are available to refine the resulting visualization.

m. Sequence Tube Map

Sequence Tube Maps [194] is a JS tool which aims at showing SVs within variation graphs. Its display is similar to a Sankey diagram [219], with genomes' paths being packed together and flowing through successive sequence boxes (that would be the node of a sequence graph), as illustrated in Figure 17.

²³ <https://pantograph.computomics.com/>

²⁴ <https://github.com/graph-genome/graph-genome.github.io/issues/30#issuecomment-1161680179>

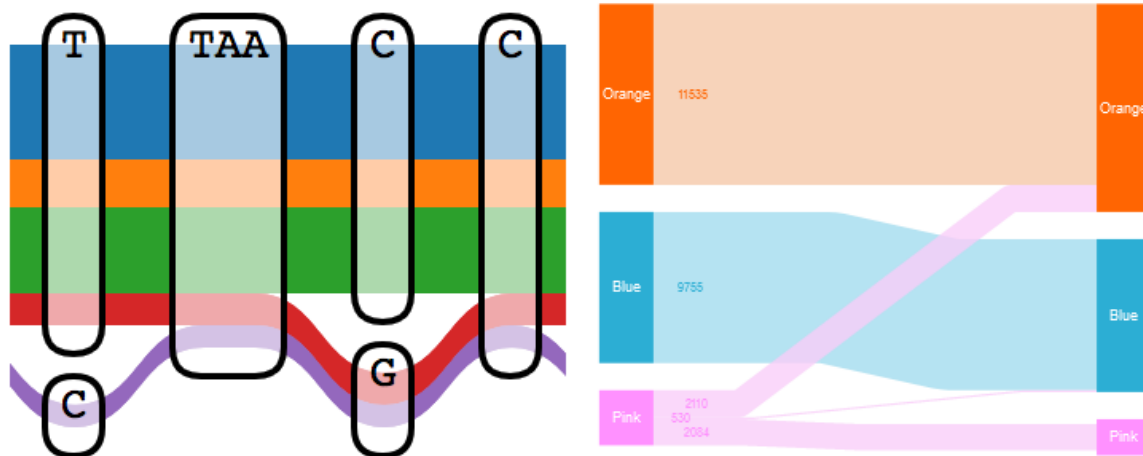


Figure 17: Sequence Tube Maps represents genomes within a pangenome as successive Sankey diagrams; Left is a schematic version of a Sequence Tube Map, as introduced on its GitHub repository²⁵; right is a simple example of a Sankey diagram taken from the internet²⁶

This representation can be thought of as a linear version of the visual representation used by *GenomeRing* [180], and a simpler version of the connected PAV matrices of *pantograph* [190]. However, it does not scale well to datasets with numerous paths as they are stacked under each other (see Appendix XXVII), which can make areas with complex rearrangements hard to read, possibly creating a hairball effect (see Ten rules Rule 3: Think about visual scalability and resolution).

This representation supports zooming, panning, hovering, and on-the-fly change of reference ordering of the blocks.

n. vg toolkit

The **variation graph toolkit** [90] mainly focuses of the creation of variation graphs but has a wiki section²⁷ dedicated to visualization and a command for displaying the graph built. This static representation displays a small region of a graph along the paths going through, as showed in Appendix XXVIII. It is actively maintained but is a *tool with visualization* rather than a *visualization tool*.

5. Composite

The **composite** category gathers tools that are harder to categorize as they display multiple visual representations from two or more of the previous categories—sometimes within the same view. There are few examples so far, as listed in Table 7.

Table 7: Composite visualization tools leverage multiple pangenome visual representations and visualization categories; As in Table 3: additional information is provided in the dedicated subsections. (*) highlight that complementary detail are available. Visualization tools are marked with a (*) before their name, to better distinguish them from tools with visualization(s). The columns provide a tool’s name, reference, year of publication, compatible type(s) of datasets, visual representations used, language of development, level of interactivity. ‘vis.’ stands for ‘visualization’.

²⁵ <https://github.com/vgteam/sequenceTubeMap>

²⁶ <https://www.yworks.com/pages/interactive-sankey-diagram-visualization>, accessed 2020

²⁷ <https://github.com/vgteam/vg/wiki/Visualization>

Vis. tool	Name	Year	pgAtlas or Pan.	Vis. representation	Language	Interactive vis.
■	MoMI-G [186]	2019	Pan.	Circos, Sequence Tube Maps...	TypeScript, JS	Yes
■	PGAP-X [171]	2018	Pan.	Multi-genome browser, pangenome profiles	C++, (Qt)	Yes*

a. MoMI-G

MoMI-G [186] is an interface that combines multiple view modules, including a Circos for translocations between chromosomes in a single coordinate system (which could correspond to a *positioned* representation), SV card within the *Interval Card Deck*, detailing information on the SV stored, and an implementation of *Sequence Tube Maps* [194] for detailed view of the nucleotide sequences (which falls into the *structure* category). These three view modules are represented in Figure 18 below.

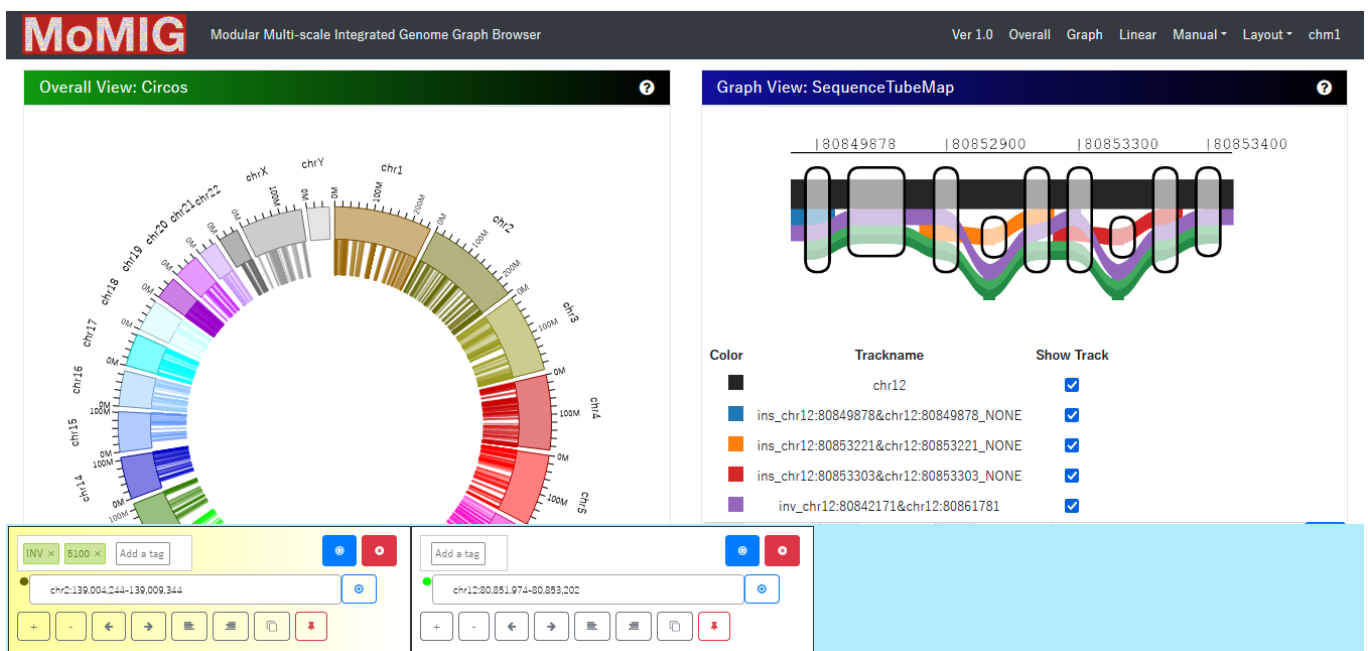


Figure 18: MoMI-G combines multiple views dedicated to structural variations; Here the Circos, Sequence Tube Map, and Interval Card Deck are visible, but other modules can be added and moved at will. Figure taken from MoMI-G’s demo²⁸, after editing the visible modules to display the overall and graph views within the same screen.

It takes files from various formats (genome graph, variation graphs, GAM, GFF, FASTA, VCF...) and is intended for large genome graphs. The view modules are customizable and can be toggled on or off, with various interactions available on the different visual representations. As for *panX* [160], too many view modules would not fit within one screen, though a navigation pane at the top enables users to jump directly to the desired view module.

²⁸ <http://demo.momig.tokyo/>

b. PGAP-X

PGAP-X [171] is an extension to *PGAP* [107] dedicated to visualization, built with C++ and Qt, and implemented within *PGAweb* [220]. It offers four representations, related to four analyses tasks (genome alignment, ortholog analysis, variation analyses, pangenome profile) applied to positioned gene orthologs. It combines mostly qualifying and positioned visualizations, but do show successions onto different genome coordinate system, like *Crop-Haplotypes* [177], which could make it a structural tool too (see Appendix XXIX). It is built for prokaryotes, and PGV claims that it could not scale to their eukaryote data.

It has a detailed documentation and provides limited interaction: customization of plots through menus, basic navigation through buttons on top of tracks, choice of genome to display and their order through the menu too. The plots themselves are static overall, except for a double click interaction that realigns the visualization.

C. Platforms and (repurposed) genome browsers

Alongside these visualization tools and tools with visualization(s), there are alternative tools that are either platforms dedicated to some species (with few visualization capacities or depending on implementations of other tools adapted to their own data) or repurposed genome browsers.

From plant research teams there are for example **BrachyPan** [72], **PepperPan** [221], **RPan** [222], **GBrowse** implementation for Brassica genomes [142] and wheat²⁹ [17]; **Persephone** browser used for 26 maize accession [223]... In the same manner I have found a case of pgAtlas visualization for bacteria, where the scientists used an altered version of **JBrowse** for browsing flexible **Genomic Islands (fGI)** based on a reference genome [224].

While interesting, such *ad-hoc* browsers are mostly one-time attempts at visualizing pgAtlases, and they cannot be used to visualize information from genomes other than the one they were specifically built for. Some of them even are not available anymore, the web servers hosting them being down (that is the case for *PepperPan*). They are therefore not suitable for the visualization of general plant pangenomes.

As for other pangenomic platforms I could cite **EDGAR** [165] (access to a diversity of mostly static charts built from pre-computed datasets), **Panoptes** [225] (multi-view browser that should be highly customizable but is hard to use), **PanWeb** [168] (a graphical interface for **PGAP**, which produces basic static plots from user-provided data), and the **Pan-Genome Explorer**³⁰ (like *EDGAR*, it integrates various pangenomes with multiple visual representations, but with more interactivity).

D. Tools not included

Some other tools described as doing pangenome visualization by various papers did not make it within this review, for a variety of reasons:

- **AGB**, the Assembly Graph Browser [226]: cited by Eizenga et al. [19], it is dedicated to assembly graphs and is presented as an alternative to *Bandage* [174] but struggled to “show fine details within the graph”, which is edge-labelled rather than node-labelled.
- **Augmented Graph Viewer** [227]: too little information and no news for some years—not much since it won the best BioVis poster award in 2017, see Appendix XXX—it has allegedly evolved into the *Sequence Tube Maps* and *pantograph* projects where computomics was involved.

²⁹ <https://appliedbioinformatics.com.au/cgi-bin/gb2/gbrowse/WheatPan/>

³⁰ <https://panexplorer.southgreen.fr/cgi-bin/home.cgi>

- **CINTENY** [228]: creates static plots of pairwise synteny comparisons between species, it not really is pangenomics but comparative genomics.
- **Gobe** [229]: a Flash application used for the comparison of orthologous genes between species.
- **gGnomes** and **gGnomes.js**: listed as graph visualization tools by the awesome-genome-visualization list [172], it seems to work on genome graphs, but I could not explore them into detail.
- **GView** [230]: seems to work on single genomes only.
- **Linx**³¹: an annotation, interpretation, and visualization tool for SVs, which is part of the tool suit used by the Hartwig Medical Foundation. It provides complex Circos-like representations of SVs (see Appendix XXXI) that I could not study in detail but seems interesting.
- **Multiple Experiment Viewer** [231]: used by *LS-BSR* [103], it creates static plots from generic data; a recent web version called *WebMeV* [232] exists.
- **MetaPGN** [197]: similar to PPanGGOLiN [70], it uses Cytoscape to represent bacterial pgAtlases.
- **MSAViewer** [233]: Useful for representing MSA, not much for pangenomes on non-nucleotidic scales.
- **Panakeia** [198]: uses Cytoscape for its visualization.
- **ppsPCP** [234]: supposedly enabling the visualization of PAVs according to Danilevicz et al. [48], though ppsPCP's paper does not mention visualization at all.
- **SGTK** [235]: Cited by Eizenga et al. [19], it can be used to visualize assembly graph, for scaffolding within assembly steps for example. Bandage [174] gained more popularity.
- Corteva's **TagDots** and **PANDA**: static representations (examples showed in Appendix XXXII) used by a private company
- **Shasta** [236]: toolkit for long read assembly, uses third party visualizations including a dotplot viewer

Moreover, it is always possible that some tools were not included simply because I did not notice them or because they were not already available at the time of writing this review. The tools listed in this manuscript still represent a good overview of the possible visualizations and visual representations for pangenomes, which has been valuable for the design phases of both tools detailed in Panache's and SaVanache's chapters.

III. Introduction to data visualization

Data visualization is the art of representing data visually, to give a sense of their underlying patterns and interactions. Datavis for scientific purposes is aimed at the faithful representation of these patterns for analysis. Artistic datavis, on the other end, is based on data too but is designed for beauty and/or feelings, rather than legibility and reasoning. Scientific datavis involves different principles and concepts; this section serves as an introduction to these.

A. Why do datavis?

Datavis takes a significant space in analysis workflows, as the human mind is used to make sense of shapes and colors rather than tables of numbers. Plotting data before diving deep into analysis is a good practice, as it can reveal unexpected outliers or patterns of interest that would be hard to detect through numbers alone.

A most famous example of the importance of representing data visually is Anscombe's quartet, illustrated in Appendix XXXIII, which shows how datasets with the same summary statistics can contain really different patterns. Researchers further detailed this idea with additional datasets

³¹ https://github.com/hartwigmedical/hmftools/blob/master/linx/README_VIS.md

representing recognizable shapes, including the Datasaurus, as illustrated in Figure 19 below. These additional datasets emphasize the importance of representing data visually, with more engaging and obvious examples.



Figure 19: The Datasaurus dozen shows that highly different datasets can share the same summary statistics; The Datasaurus Dozen is a new take on famous Anscombe's Quartet: obviously different graphs may still produce identical statistics. This perfectly illustrates how visualization can add an important layer of information in such a way that numbers alone could never do. Figure is adapted from <https://www.autodesk.com/research/publications/same-stats-different-graphs>

Moreover, scientific datavis serves different purposes: exploration and analysis of data, but communication to others too. Researchers might want to use datavis to explore their data, trying to find patterns of interest when comparing different parts. It could also be used with a specific research question in mind, for a more targeted approach, for example on a specific subsection of the dataset. Finally, it is useful too to communicate results to others, and is therefore important in the transmission of knowledge, between researchers but to broader audiences too.

B. One concept, many visual representations

There is no universal way of visually representing data, or concepts, and there is no purely neutral way to represent something either. Every visual representation is a projection onto a two-dimensional (or even three-dimensional) space, and therefore cannot properly convey all internal interactions and properties. A datavis designer *chooses* how to visually encode and convey the information that is considered as important. Multiple visual representations could be made from a same concept, without one being less truthful than the other³² as illustrated by Matthieu Robert-Ortis's work in Figure 20.

³² However, some can be misleading, as discussed in State of the Art III.C.3.



Figure 20: Depending on the chosen projection, a visual representation can highlight different properties of a same concept; Matthieu Robert-Ortis is a French sculptor focusing on anamorphosis. His wire sculptures can represent multiple distinguishable shapes depending on the direction from which they are observed. Here, his sculpture “L’homme crabe” can be both seen as a person (shadow) or a crab (front view). None of these projections fully represent all the internal relations between wires in the three dimensions. Taken from <https://cargocollective.com/matthieu-robert-ortis/Ombre-portee>

A widespread example of visualization is the writing system used in English. It is easy to take for granted that words are constituted of letters, separated with spaces, and following a certain

order³³, but it is the result of a long-matured evolution of visual encodings of language throughout history!

Ancient Egyptians wrote hieroglyphs, which combined drawings to convey names and sentences. Boustrophedon (see Appendix XXXIV) is another ancient approach to writing, where each line has a succession of glyphs (associated with a determined meaning), and every other line is mirrored upside-down. As another example, *Scriptio continua*—a way of writing without space or punctuation for separating letters into words³⁴—has been in use in Ancient Greek and Classical Latin.

Even nowadays there still are multiple systems coexisting throughout the world, sometimes within a same language. Japanese for example uses four different writing systems: Kanji where a glyph or combination of glyphs represent a word; Katakana and Hiragana where each glyph describes a syllable instead; Romaji which is a roman alphabetization of Japanese words, mostly in use for foreigners and typing on a computer. We are therefore used to the coexistence of multiple visual representations, even if we may not be conscious of it.

However, there are ways to visually encode information that are more efficient than others in the context of scientific datavis, as explained below.

C. Some principles of data visualization and UI design

There are no formal rules or principles, rather guidelines for good practices, that can be ignored but not without risks. It is a common point of view among datavis scientists today, though some share Edward Tufte's elitist vision that distinguishes good and bad datavis depending on rational criteria.

1. Tufte's data-to-ink ratio and chartjunk

Edward Tufte worked on the effective design of visualizations. One of the main takeaways from his book "*The Visual Display of Quantitative Information*" (originally published in 1983) [237] is that datavis designers should tend to a maximal '*data-to-ink*' ratio. Simply put, he means that ideally every graphic element displayed should be linked to data, with as little superfluous content as possible. As an example, he would encourage the removal of background gridlines of bar charts, as they add nothing to what can already be seen.

He also expressed strong thoughts against 'chartjunk', ie the visual decorations that can be found within graphs for aesthetics or as bits of humor. Again, his idea is that most-if-not-all the visual elements should convey insights about data.

His extreme minimalist approach has since been nuanced by later studies that showed that even though the impact of visual embellishments on memorizing the data was questionable, they do make datavis more likeable and memorable overall [238, 239].

2. The principles of Gestalt

A fundamental concept of datavis and UI/UX design in general is the psychology of Gestalten (the German word for '*shapes*'), exploring how the human brain processes what is visually one entity

³³ "*Aocdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mtttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae.*" As illustrated by this sentence widely shared on the internet, even the order of letters might not be that important. Matt Davis from the University of Cambridge provides an interesting analysis of this claim at <https://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/>

³⁴ Wearenotusedtosuchawayofwritinganymore

or different. Since it was first explored by Wertheimer and Köhler in the 1920s³⁵, an extensive amount of literature has studied how shape recognition works and what its main drivers are [240-242]. The ‘*Gestalt principles*’ (or sometimes ‘*Gestalt Laws*’) are nowadays common concepts when working on visual representations.

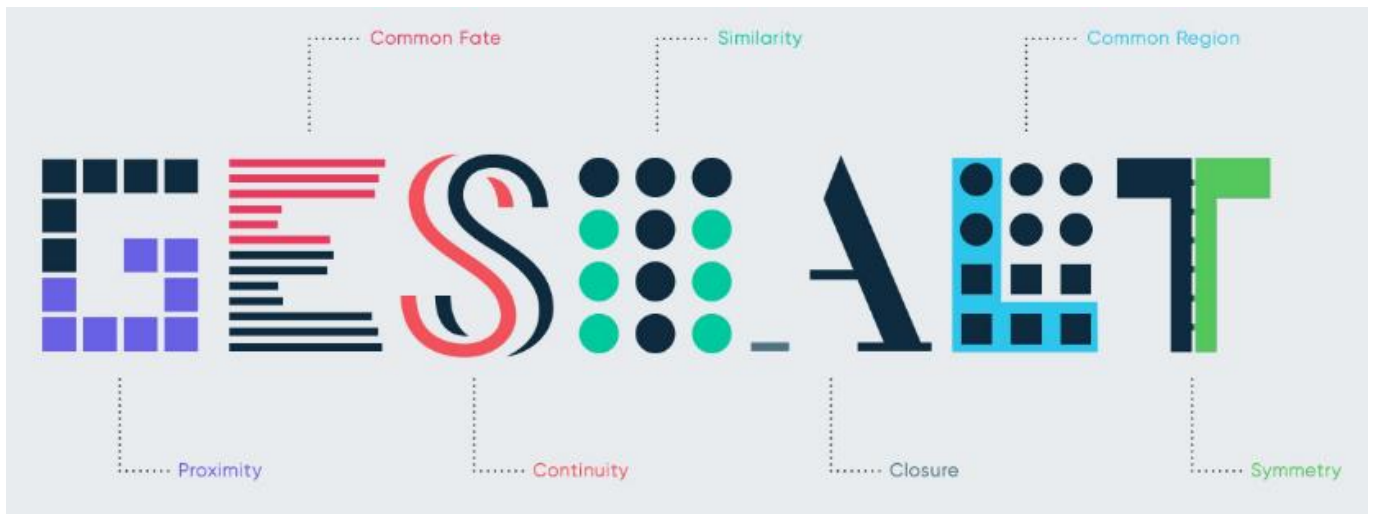


Figure 21: The Gestalt principles explain why some visual elements are understood as being grouped together; Taken from <https://www.north-47.com/knowledge-base/gestalt-principles/>

Some of these principles are illustrated in Figure 21 above³⁶.

- **Proximity:** elements that appear visually close are grouped together (here the squares at each end of the letter ‘G’ are closer to their direct neighbors than to each other).
- **Common Fate:** elements that experience common changes tend to be grouped.
- **Continuity:** if two shapes appear as if they could be in the continuity from one another, we assimilate them to a unique shape split in two parts.
- **Similarity:** groups are easily made between things that look alike.
- **Closure:** similarly, to *continuity*, our mind can abstract visual elements that are not visibly present (working as an autocompletion).
- **Common Region:** elements encapsulated within a same area are grouped together.
- **Symmetry:** symmetrical shapes are likely to be associated.

3. How datavis lies

Datavis uses different visual channels (position, color, shape, orientation, size...) to encode information, and each channel has its own strength and weakness, and can even mislead if not used properly, as described by Alberto Cairo in his book “*How charts lie*” [243].

Especially, color is a difficult channel to manipulate [244]. It encompasses the three different concepts of **hue** (whether a color looks more red, green, blue, or other), **saturation** (if a color is vivid or rather grey-ish), and **luminance** (how dark a color appears, ranging from black to white). However, linear gradients of these values do not end up in linearly perceived difference, as illustrated for the rainbow color scheme in Figure 22.

³⁵ The notion of Gestalt was already used for perception of musical patterns, but not for visualization yet.

³⁶ Other principles include Parallelism, Synchrony, Element Connectedness and Uniform Connectedness.

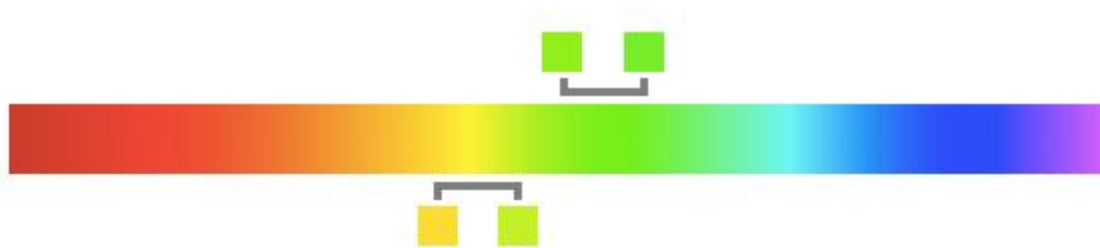


Figure 22: The Rainbow color scale does not accurately represent distances from a linear space; This typical example of rainbow color map associates a linear gradient of values with color varying in hues. However, for a same interval the perceived differences are not representative of the underlying data. The difference between the two-colored squares on top seems smaller (they both look green) than the difference between the two colors at the bottom (one is yellow, the other one is green), while their distance intervals are in fact the same.

Moreover, people can have various forms of colorblindness, preventing them from accurately identifying hues. Taking the different types of colorblindness (Protanopia, Deuteranopia, Tritanopia) into account would make a tool more accessible and user-friendly.

Among other considerations, one should keep in mind that color is understood relatively to its direct context and neighbors. This effect has been illustrated in many optical illusions, including Adelson’s checker featured in Appendix XXXV. It plays a role for visual representations like heatmaps where colors encode numeric values, accurately identifying the exact value would be made difficult as the signal of every colored square is disturbed by the color of its direct neighbors.

The human brain has its own ways of understanding and translating what it sees. More visual channels are more powerful for making comparisons, or for making something visually ‘pop out’. In scientific datavis it is therefore important to pay special attention to the faithfulness of a visual representation to the underlying data, through careful design.

4. The principle of least astonishment

Also called principle of least surprise, this UI/UX design and software development principle states that some behaviors are expected from some interactions and should be kept. A tool would be easier to adopt if these expectations are met rather than avoided. For instance, scrolling down with a mouse wheel is usually associated with vertical panning. The UX would be hindered if it suddenly triggered the opening of new tabs within a browser instead!

Simply put, a tool that feels intuitive will offer a better experience to the users, and it will favor the adoption of the tool. It can be extended to a general principle of reluctance to embrace novelty, as learning how to use something always comes with a cost as most people have little energy to invest in it.

5. Shape distinguishability

An important part of datavis for many scientists is the distinguishability of shapes within the plots they can make on a daily basis (like scatterplots from a PCA or line charts showing an evolution over time).

Some scientists worked on the identification of factors that help to differentiate two shapes. Notably, Huang [245] proposed a three-dimensional space based on Segmentability, Compactness, and Spikiness (**SCI**), with a set of new shapes (featured in Figure 23) that could be more distinguishable than the ones currently in use with classic plotting tools (like Microsoft Excel or Tableau).

c. (S)egmentability-(C)ompactness- Sp(i)kiness
(SCI) Shape space (plotted in one panel)

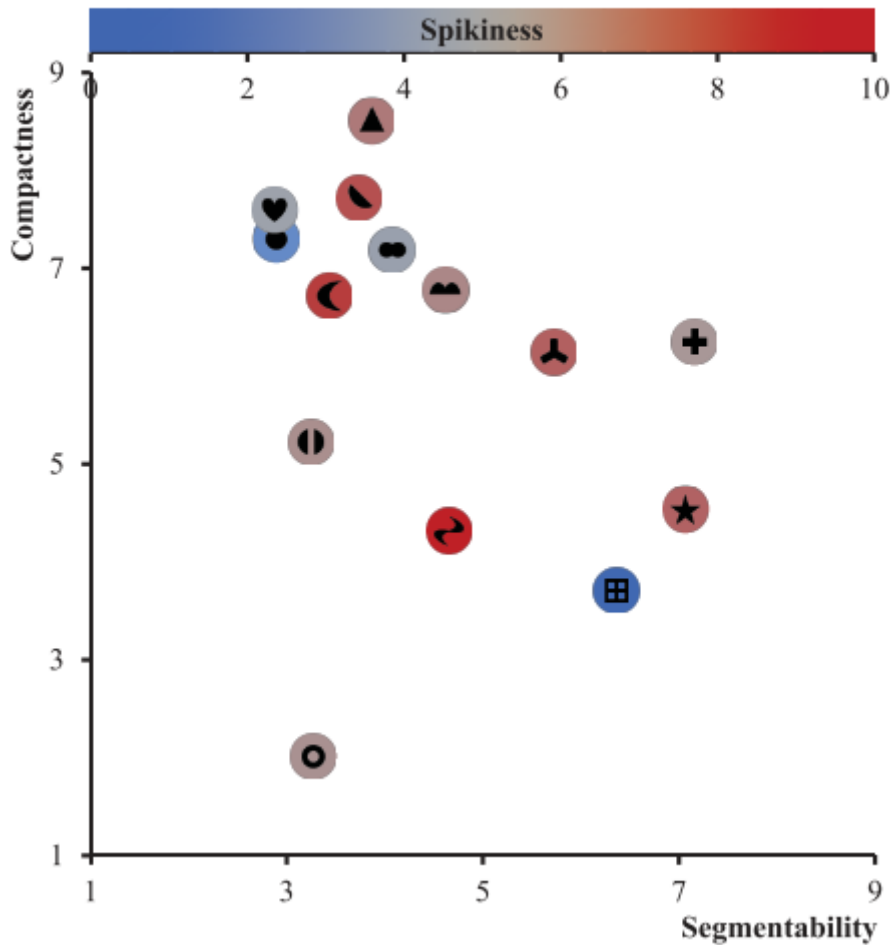


Figure 23: Huang proposes that shapes are distinguishable based on the three properties constituting the SCI space; Figure from [245]

D. What are design studies?

For datavis, a design study is the exploration of the possible visual representations that could be created to tackle a certain problem and their integration within a visualization tool. It also encompasses the analysis of how well a design performs compared with others for accomplishing user-defined tasks. They should include a description and discussion of the visual representations used and their performance with users.

Some researchers (like Tamara Munzner [246-248] or the Visualization Design Lab³⁷), provide workflows and guidance for successful design studies. These examples and advice are full of interesting ideas that better explain what can or cannot work when building a visualization tool.

Munzner's book "*Visualization Analysis and Design*" [248] was particularly insightful, for example with her work on the effectiveness of the different visual channels in encoding different types of data, as showed in Figure 24.

³⁷ <https://vdl.sci.utah.edu/>

Channels: Expressiveness Types and Effectiveness Ranks

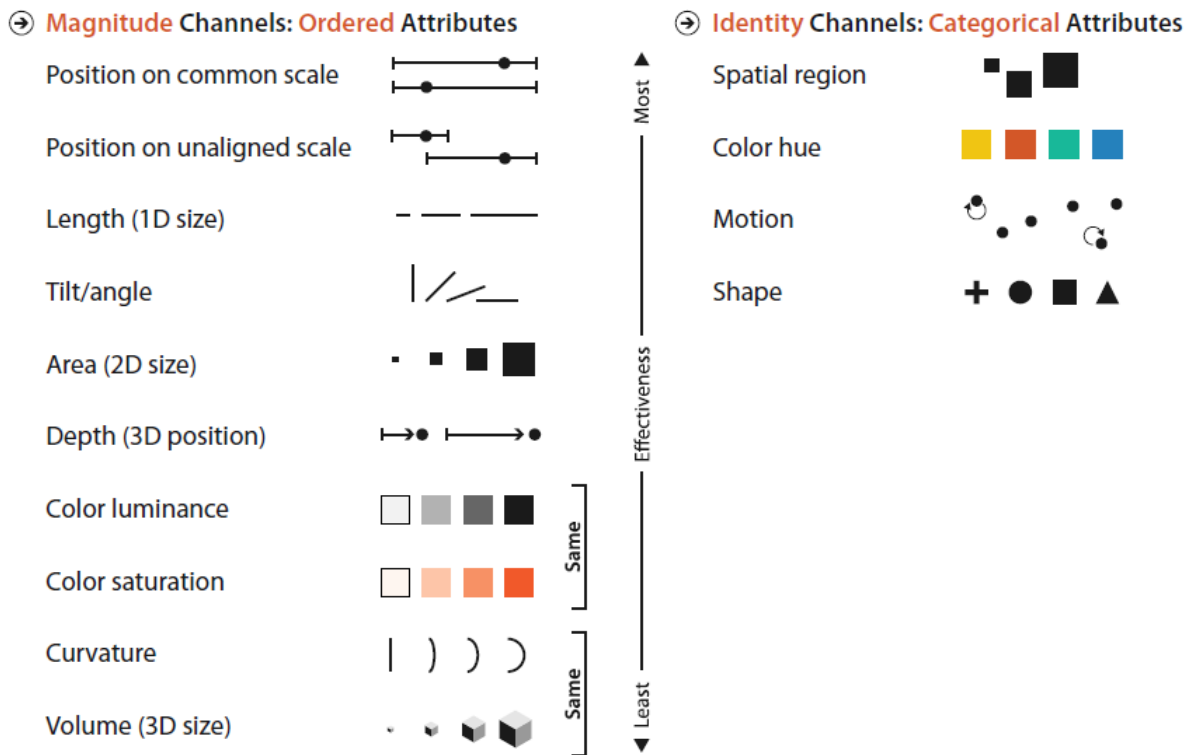


Figure 24: Munzner ranked the effectiveness of visual channels depending on the type of data that should be represented; Her main takeaway was that position was the most efficient channel for both quantitative and qualitative data. Figure from [248]



Ten rules on Genomic Visualization Tool Development

Developing an -omic visualization tool (or any visualization tool, for that matter) is no trivial task, and implies a harmonious cooperation between biology, bioinformatics, software development and datavis skills. In this chapter I introduce our article “10 Simple Rules for developing genomic visualization tools”³⁸ written from hands-on experience and knowledge acquired through my PhD and previous genomic projects of all contributors. I comment on some of these rules, providing an addendum to those that had to be shortened in the final paper. These ten rules are referred in the subsequent chapters of this PhD dissertation.

I. Motivation

When I first worked on pangenomes back in 2018 for my MSc2 internship, I had experience from my biology background and specialization in bioinformatics (a few months out of my overall cursus) but knew little about software development and datavis. My supervisors were knowledgeable in biology, bioinformatics, and software development to an extent (they had built tools without being from a proper tool development background) but had little to no experience with datavis. One of them is even to blame (affectionately) for being behind the now infamous Banana “Vennster”³⁹ chart Figure 25.

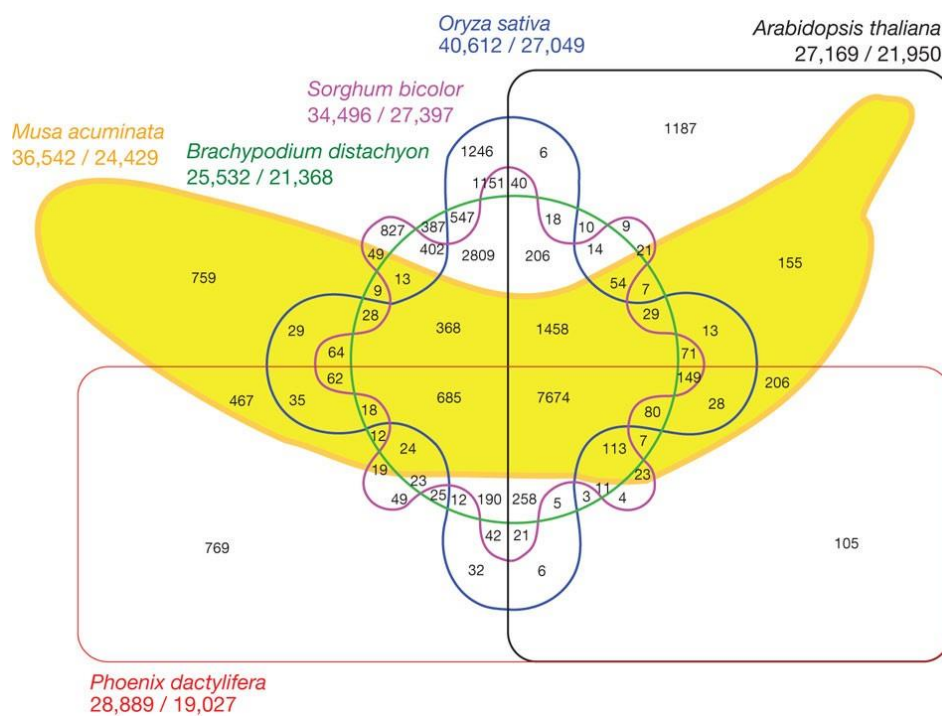


Figure 25: The Venn Banana chart, both praised and criticized; taken from [250]. This altered Edwards-Venn diagram represents sets of shared gene families between six plant species, featuring a distinguishable banana shape for *Musa acuminata* (a banana species).

This chart started many discussions between scientists, some praising its inventiveness and novelty, others criticizing its lack of readability and practicality. It created a heated debate among datavis practitioners and was one of the motivations for the creation of UpSet [196], a visualization technique designed for efficiently visualizing set intersections even for large numbers of sets (i.e., 4 and more).

³⁸ In review at PLOS Computational Biology

³⁹ I.e. the intersection of a Venn diagram and a monster, as coined by McDermott 249. McDermott, J.E., M. Partridge, and Y. Bromberg, *Ten simple rules for drawing scientific comics*. PLOS Computational Biology, 2018. 14(1): p. e1005845.

Its main detractors' argument is that the area dedicated to intersections are not proportionate to the count of gene families contained in them, making it barely useful as a visualization. While receiving backlash from a part of the scientific community, it also gathered positive comments from many biologists who simply *enjoyed* it, making it a successful graph if we consider only engagements with the host paper. More broadly, this reflects the opposition of minimalist researchers banning “chartjunks” (as first coined by Edward Tufte [237], see State of the Art III.C.1) to those in favor of chart “embellishments”—usefulness versus attractiveness [251-255].

Oddly shaped bananas apart, it illustrates the existing interest (if not the academic skills) we all shared for datavis before my PhD project even existed. This PhD therefore unfolded with close to no prior datavis knowledge, which proved difficult at times but proportionally instructive. We decided to build from these back-and-forths between datavis theory and practice and came up with ten simple rules for building a genomic visualization tool, written with collaborators who already faced the same challenges. Our goal with these rules was to offer a “starter pack” of sorts to other bioinformaticians and biologists eager to build their own visualization tool, with advice and numerous references to datavis papers as an attempt to limit the number of hasty enthusiasts diving headfirst in development.

As I discovered during my PhD, datavis is a whole scientific field, linked to research on **Human Computer Interaction (HCI)**, cognitive and vision sciences with a lot of applications since visualization can be used for communication as well as scientific exploration and analysis (if we stick to serious use cases). I personally think that scientists would gain a lot by using even basic knowledge of datavis in their everyday work. These rules therefore include some of the things about datavis I wish I had known earlier, in hope that they will be of use to others.

II. Ten simple rules for developing visualization tools in genomics

Eloi Durant^{1,2,3,4*}, Mathieu Rouard^{3,4}, Eric W. Ganko⁵, Cedric Muller², Alan M. Cleary⁶, Andrew D. Farmer⁶, Matthieu Conte², Francois Sabot^{1,4*}

¹ DIADE, University of Montpellier, CIRAD, IRD, Montpellier, France

² Syngenta Seeds SAS, Saint-Sauveur, France

³ Bioversity International, Parc Scientifique Agropolis II, Montpellier, France

⁴ French Institute of Bioinformatics (IFB)—South Green Bioinformatics Platform, Bioversity, CIRAD, INRAE, IRD, Montpellier, France

⁵ Seeds Research, Syngenta Crop Protection, LLC, Research Triangle Park, Durham, North Carolina, United States of America

⁶ National Center for Genome Resources, Santa Fe, New Mexico, United States of America

* Corresponding authors

E-mail: eloi.durant@ird.fr (ED); francois.sabot@ird.fr (FS)

A. Introduction

Visualization is key for expanding and communicating knowledge to both specialized and broad audiences—after all, “*a picture is worth a thousand words*,” right? That much has become clear during the SARS-CoV2 outbreak when “*Flatten the curve!*” [1] turned into a catchier watchword than the usual “*Wash your hands!*”, by referring to the related graphics rather than the health discourses themselves. COVID-19 visualizations, good and bad [2], became omnipresent in the public debates, with interactive and dynamic visualization platforms like Nextstrain [3] gaining a lot of popularity.

We are now used to seeing graphs and charts in our everyday lives and creating them for scientific papers as part of research. Unfortunately, most of the time classic visual representations are insufficient to effectively communicate data complexity, and as S. O’Donoghue puts it, “*often dedicated communication approaches need to be developed to address specific data challenges, especially when conveying complex or unfamiliar ideas*” [4]. Given the gap between creating static figures and building a visualization tool from the ground up, it can be easy to get lost in this non-trivial journey.

Our following ten simple rules are dedicated to biologists and bioinformaticians who, while already being at the crossroads of many fields, want to venture further into the land of Data Visualization (“datavis” or “dataviz” for short). They combine tips and advice that we would have wanted when we first started our own journeys, gathered from our experiences in building genomic and/or datavis tools, and the time spent with related communities. Additionally, they address current challenges in computational biology and the needs of the community.

We aim these rules at bioinfo-to-datavis novices looking for guidelines, particularly regarding genomics visualization tools, but experienced practitioners may find it useful to see them gathered in one place too.

For better reading comfort, we organized the rules chronologically depending on when they would matter the most during a visualization tool's life cycle, starting with the design (rules 1 to 5) then development (rules 5 to 8-ish) phases until it is shared with the rest of the world. Please note however that they are still relevant during the whole gestation and that creating a visualization tool is rarely a fully linear process—it does not even end after the initial release, as continued development and support is a cyclic process to be sustained over the years.

B. The Rules

1. Rule 1: Articulate the need for new visualization tools

For the past few decades, most genomics data have been visualized through genome browsers (Jbrowse [5], Ensembl [6], IGV [7], etc.). With the advent of Next Generation Sequencing technologies, the number of draft or high-quality assembled reference genomes exploded, and the need to federate (gen)omic data across assemblies within and between species and to visualize them has become a major challenge.

All these data types and tools lead to critical questions, such as “**What am I trying to visualize?**” and “**Why?**”.

More specifically: What data do I have? Am I interested in structural variations? in epigenetic changes? at genome scale or at a defined locus? Are there gene annotations available? Can I get sequence alignments? Is there already a tool fulfilling all my expectations?...

Such questions must drive your quest for the appropriate datavis tool. A good first step would be to explore the related bibliography and tool reviews. Some reviews focus on the use cases [8] or layouts [9] of genomic visualization tools in general, while other reviews are specific to certain subjects, such as structural variations [10] or Hi-C data [11]. Your data and visualization needs may already be compatible with an existing tool, or additional development of an existing tool could bring the feature you need. Getting a feature added to a tool can be done for example with GitHub by making a feature request or even adding the feature yourself with a pull request—consulting the dev team beforehand is usually a good idea, to be sure that what you propose is in line with their vision and community.

If after all these steps nothing matches your needs, it then appears necessary to develop a new visualization tool. If so, your big challenge will be to **clearly identify the scope of your visualization tool** and to **keep your end goal in mind**.

2. Rule 2: Involve others early on

There are many “others” that you could and often should work with when creating a datavis tool. Sedlmair et al. [12] identified six non-mutually-exclusive collaborator roles of interest for design studies of such tools, with one of the main roles being the **front-line analysts**.

Front-line analysts are your end users, people who will use your tool and need it to complete certain tasks, and who should therefore be handled with particular care. It is crucial to engage them in discussion as early as possible since they are the ones who will (hopefully) adopt your tool. **Front-line analysts** can help you to properly define the tool tasks (i.e. both high- and low-level tasks that your tool should achieve) and provide test data as well as valuable feedback during both the design and development phases. There are many good ways to engage with your end users, including face-to-face interviews, surveys, design sprints, and even hands-on sessions

which can often reveal edge cases and sometimes bugs that would have been hard to find on your own—beta-testing at its finest.

Moreover, there are fields dedicated to datavis: understanding how they are perceived by the human brain (visual perception and cognitive vision science), improving how they can be used with machines (Human Computer Interaction) and growing communities of **datavis designers**, full of people who could help build a visualization tool. They can be rather generalist, like the Data Visualization Society (<https://www.datavisualizationsociety.org>), or more dedicated to the visualization of life-sciences data, such as the BioVis community (<http://biovis.net>). Look around, there might be someone there willing to give you a hand!

3. Rule 3: Think about visual scalability and resolution

If the history of genomics teaches us one thing, it is that what was once a huge dataset can be considered a toy set 5 years later [13]. The exponential growth of genomic datasets leaves us no choice but to consider the scalability of our tools' design and the efficiency of their visual encodings (i.e. how well the chosen visual representations reflect the underlying data). Some designs might be good visual representations for small datasets, but scale poorly for larger datasets.

For example, Venn diagrams are fine for up to 3 sets, but with more sets they become messy at best and a nightmare in worst case scenario. UpSet bypasses this issue by focusing on the set intersections and presenting them in an ordered “table”, but it still suffers a loss of readability with increasing sets (there is simply too much to see) and is not designed for visualizing hundreds of sets [14]. Similarly, networks and graphs do not adapt well to big datasets with many nodes and edges, resulting in the infamous “hairball effect” [15] (Fig. 1). **Your data will come with edge cases that you need to anticipate if you want your design to work properly at scale** [16]. Try varied configurations (good ol' pen & paper are still useful to quickly draft multiple layouts), get familiar with your design's limitations, and offer alternatives if you have the resources for it.

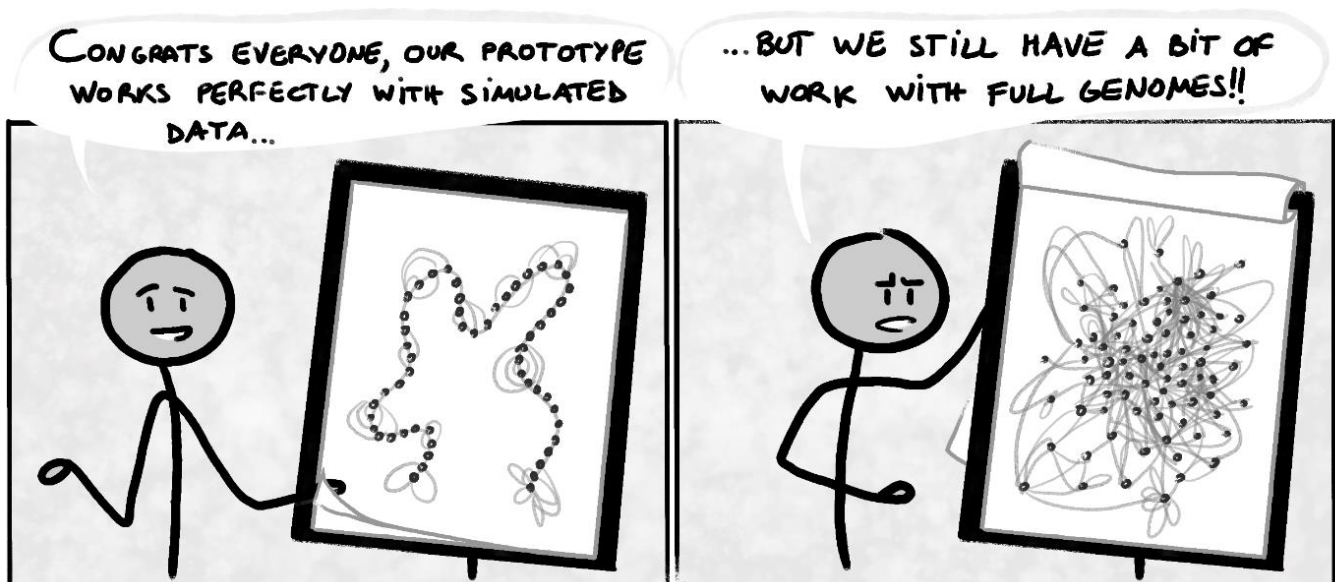


Fig. 1. Your chosen visual encoding's clarity can depend on the size and complexity of your datasets, which could also hold unexpected characteristics compared with simulated or real data of limited scope.

Genomic data has specificities and layers that you need to consider at different resolutions. Your organism(s) may have multiple chromosomes (linear or circular) to consider, heterozygosity or even polyploidy, and may present macro structural rearrangements at a chromosome level that will not interest you at the nucleotide sequence level. Along with the genomic sequences, you may

also have access to additional data types, such as Hi-C, epigenomic signatures, or detections of transcription factor binding sites, all of which can blur the respective message in all-in-one visualizations. Instead, consider different visual representations for each data type and how they can complement each other through comparisons and interactions.

4. Rule 4: Be creative, be bold

To develop a successful visualization application, you will have to find the right mix of technology, usability, and aesthetics.

Regarding technology, you can envision that scientists will soon have access to equipment similar to what Tom Cruise used in *Minority Report*, that will interact with virtual screens to perform multidimensional genomic data analysis. Virtual reality headsets and augmented reality devices offer a first, more affordable step in this unfamiliar 3D environment for datavis—some have already been put to use for GWAS [17] and visualization of 3D structure of chromatin [18] or genome graphs [19]. However, while technically feasible, such technologies are not popularized yet and that choice would likely reduce the number of end users. There are plenty of technology options available, and you will need to keep up to date, but make sure that your audience can effectively and correctly use them!

As for aesthetic and artistic ways to present data, it is partly subjective. Something nice and eye-pleasing to one person may not be appreciated and understood the same way by another person. However, there is a good amount of literature providing advice for efficient datavis design [20–22]. One example is to pay special attention to how you use colors [23]. Make your tool more accessible by accommodating visually impaired people [24], who make up more than 3% of the global population [25]. Overall, **be creative, do not be afraid to create designs that will not make it to the final cut**, and test different graphical strategies following user experience (UX) design processes [26]. Finally, even though too much novelty could be a barrier to adoption, a smart and beautiful design will encourage your users to engage more by offering them visual clarity of their data.

If art is really your thing, though, you could totally find a place among the “*Xenographics*” (<https://xeno.graphics>) or create your own science-inspired pieces in your free time (check out Martin Krzywinski’s π -art gallery: <http://mkweb.bcgsc.ca/pi/art>)!

5. Rule 5: Make data complexity intelligible

A familiar way of making complex data accessible is to compute derived measures and statistics or to apply dimension reduction techniques. Still, visualization is recommended to detect patterns (Fig. 2) that would not be found with these means alone, as illustrated in Anscombe’s quartet [27] and the more recent Datasaurus Dozen [28] (based on Alberto Cairo’s eponymous dataset). Unfortunately, the human mind cannot make sense of everything that it perceives at once and our attention is instead focused on fragments of what we see.

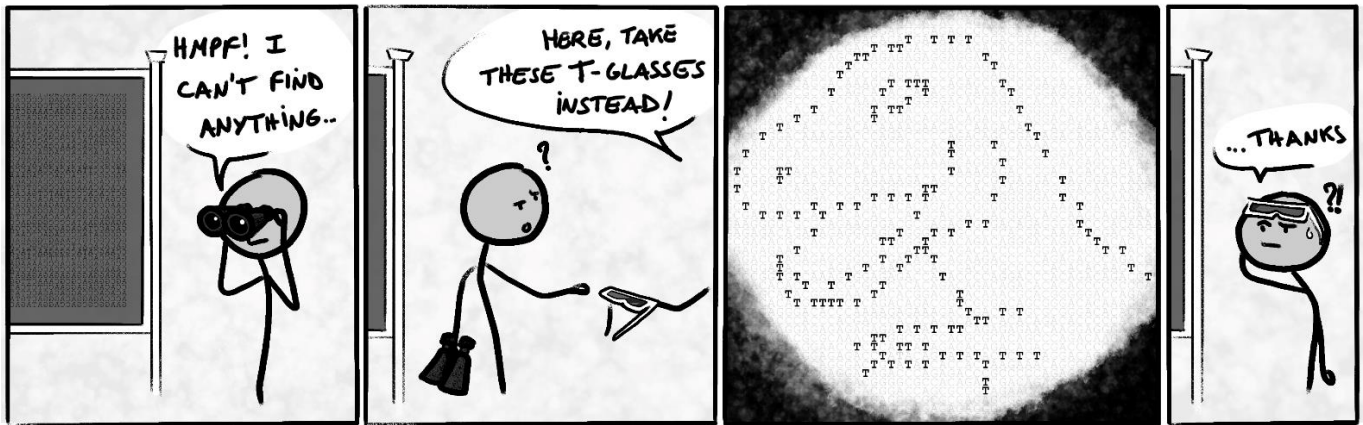


Fig. 2. A good datavis tool should make the identification of patterns easier (feat. the Datasaurus)

In practice, our understanding of graphical representations comes in part from the immediate identification of salient elements that visually “pop out” (i.e. *preattentive processing* [29]) but it mainly comes from active exploration [30]. Therefore, a key component of datavis is to make exploration easier, and to reduce the workload for a user’s brain starting with careful design as to relieve their working memory [21].

A universally praised rule of thumb to this end is Shneiderman’s mantra: “**Overview first, zoom and filter, then details-on-demand**” [31]. This approach encourages visualization developers to make clever use of the main advantage of visualization tools over visual representations: interactivity. With such a divide and conquer approach it becomes easier to make sense of the displayed visual information as a user. For example, adding details with dynamic tooltips or including dedicated linked views enables users to build a deep understanding of the data provided without hiding the overall patterns.

6. Rule 6: Let your inner nerd shine, when needed

Technology matters, but it is not an end in itself. If you can design and produce an efficient tool using reliable methods instead of the latest and trendiest ones, there is no reason not to. Using the latest framework or published method can be interesting when they add significant advantages, but they may come with stability issues and will not always be the safest option for a robust and long-term tool (as an example, we can think of the infatuation with Objective-C in the mid-2000s leading to the creation of the BioCocoa framework (<http://bioinformatics.org/biococoa>), without any evolution since 2011). Moreover, accumulating niche technologies and methods would make it difficult to find someone with the exact skillset needed to maintain or help develop your tool; consider the consequences of that choice carefully.

On the other hand, it is also important to recognize that software technologies evolve quickly, and once-common approaches to user interface development may lose favor or become inviable (consider applets and other rich-client alternatives to browser-based development; or the genomic visualization tool Gobe [32] which unfortunately came out after the first signs of Adobe Flash’s downfall). Sometimes a well-conceived and popular tool may need to be completely overhauled in order to remain relevant and usable. This is what happened with genome browsers implemented as imagemaps generated server-side which transitioned to later generations implemented using client-side JavaScript frameworks.

You will need to use relevant technologies for your tool: using LaTeX to produce pangenomics visualizations is fun but needlessly complicated [33] and there are better, dedicated technologies out there. Common tools in datavis differ from common tools in bioinformatics in general, and you will have to try and get accustomed to unfamiliar technologies and concepts. For web-based

visualization, you should first familiarize yourself with the differences between Scalable Vector Graphics (**SVG**, geometrically defined images) versus **HTML5 Canvas** (pixel-defined images) elements. A must-know library for working with visualization is **D3.js**, which enables the manipulation of (graphic) elements on a webpage. Finally, if your goal is to display numerous elements at a time (say thousands or millions), you should consider working with **WebGL** (an API using the graphic card for faster display)—check out the Three.js library or the GenomeSpy [34] or Gosling [35] visualization grammars if interested...

7. Rule 7: Benchmark with diverse datasets

While it is important to have a well-thought-out design, you cannot ignore performance, as a smooth user experience is key for your tool's adoption [36]. Benchmarking with different datasets is critical for your visualization tool, and each type has its own use.

- **Small simulated** datasets can and should be used during development, as they are better suited for fast iteration while debugging. Make sure that what you see is faithful to the data you have. Handcrafted files can be useful to make extra sure that the resulting visuals match your expectations.
- **Big simulated** datasets are meant to be used for stress tests, to assess scalability and performance: how much data can you feed to your tool until it breaks? until the tool starts slowing down too noticeably? Performance is an important aspect of your tool that should be determined, at least to let users know what to expect if they want to use their data with your tool.
- **Real** data, finally, to make sure that your tool is working properly with real-life data. Look out for unexpected behaviors and edge cases, related to your design or data format loopholes that may hinder the user experience.

With the decrease of sequencing costs, it is quite frequent to have huge projects with hundreds if not thousands of samples: humans, *Arabidopsis*, mosquitoes, rice, bacteria, viruses and so on. A modern tool would be expected to manage hundreds of datasets, with as few lags as possible and no memory crashes. Keep in mind that “hundreds of genomes” may not correspond to the same size depending on the studied organisms: one thousand HIV genomes represent ~9.2 Mb, which is less than 0.0014 the size of a single human genome. You should therefore consider performance through two axes: count capacity (number of genomes or measures that can be added) and size capacity (efficient and maximum file sizes that can be used).

You may also need to consider characteristics such as the number of chromosomes (or scaffolds) within a genome as well as the distribution of sizes among these elements (for example, an abnormally large chromosome in an otherwise “normally-sized” genome could break the assumptions of your data structures).

8. Rule 8: Stay tuned to the genomic tool ecosystem and promote interoperability

Your application will likely need to work among an existing ecosystem of databases and tools that vary by community, even when developed as a stand-alone instance. To lower your tool's barriers to adoption, **pay special attention to implementation, distribution, documentation and deployment.**

Regarding implementation, your tool should be as light as possible, OS agnostic [37], and if possible, run client-side. Implementing a standard Genomics API [38] to consume input datasets will also enable seamless integration and better interoperability with other databases and applications. Moreover, some users may want to include your visualizations in automated workflows, so the ability to get output via a command-line and companion scripts can be a nice additional feature to provide.

For distribution, consider multiple options—a Docker or Singularity container can make complex setups much easier to install, but providing users with the details to install with dependencies on other systems is also valuable. Web applications (for example, those built with JavaScript) are great for avoiding multiple setup issues as they can work with any web browser.

When time and resources allow, making a popular tool available on several platforms and programming languages is a good option for sustainability. JBrowse is a good illustration since it is available in both web and desktop versions [5], can be embedded in large genome portals as a Drupal module [39], has an R markdown and R Shiny compliant version (JBrowseR [40]), and exists as a Jupyter package [41]

Hosting an example of your software on a website can be useful for trial purposes but can be difficult to maintain over time. Always seek to make it easy for potential users to try your tool on their systems – include clear instructions and a straightforward way to download and run the tool. Good trial data helps users understand the tool and is also a good test to make sure the tool is running properly following installation. Moreover, data formats in bioinformatics can be tricky [42] so clearly document the formats your tool needs and provide examples.

9. Rule 9: Keep up to date with related work

Genomics has been quickly evolving in the past years and shows no sign of slowing down in the future. Envisioned future developments and challenges are linked to more diverse and applied -omics approaches (especially in health), data ethics and security, reference-free studies and others, all with ever more data and a growing need for integrated solutions [43].

With fast evolving needs and many people working on these subjects, it can be easy to be the needle in a haystack and be forgotten in favor of another tool: as of May 30, 2022, there are 434 visualization tools listed in the *awesome-genome-visualization* list [44] and 151 research articles (already 13 out in 2022 and counting) found via Europe PMC whose listed keywords matched “*visualization*” and “*genomics*”. Make sure you keep up to date with data and technologies in genomics so that your ideas and tools stay fresh and relevant. Do not let your focus wander off too much either: a polished and thoroughly executed idea is better than a hybrid monstrosity of hastily (re)defined goals.

Moreover, you should try to **ride the wave rather than fight against it**: make your work known and alive (in conferences, articles, Twitter...) and people will start noticing and using it. After all, you could very well make the new state-of-the-art tool!

10. Rule 10: Grow and support your user community

Gaining users is the final key to a great visualization tool. To do that you need to communicate your tool to prospective users as well as those who are actively working with it. **Communication can take time, but it is rewarding**—not only to ensure that people are using your tool, but also to gather ideas for improvements.

User documentation is a simple starting point – make sure you have an up-to-date README or manual for your software, including multiple examples on how to actually use your software. GitHub is an easy way to centralize this information and engage with those that have downloaded your project and are attempting to use it, especially through the “Discussions” and “Issues” features. Projects with no activity or updates are a warning sign to potential users, whereas those with activity are more enticing. Plus, you can centralize answers to frequent questions and, if you have an active user base, GitHub Discussions now supports polls, making it a convenient way to get input on new ideas. Furthermore, as reproducible research and FAIR-sharing principles are applicable to software products [45,46] it may encourage users to cite a released version of your tool by using tools like Zenodo to issue persistent Digital Object Identifiers (DOI) against your releases.

When presenting at a conference try to end with 1-2 questions about new feature requests or biggest issues, along with contact details. Also consider setting up a google search notifications around the name of your tool – you might find mentions on Biostars, publications, and other sites you were not expecting.

Finally, do not forget to use social media to cast a wider net by advertising releases of the tool as well as occasional examples of its use [47].

C. Conclusion

Our 10 simple rules for developing genomic visualization tools will not replace the experience gained by trial-and-error but will hopefully make the development process less painful for future bio-datavis practitioners and other datavis enthusiasts. Moreover, these rules merely cover the tip of the iceberg, and we strongly encourage our readers to take a look at the references included in this paper to further improve their understanding. Other interesting matters that did not make it through the introductory cut include visual design validation [48,49], inclusion and accessibility matters [50,51], and visual representation of uncertainty [52,53], which are subfields on their own.

Most of the elements presented here would also be applicable to non-genomic datavis tools; what may be more specific to genomics here is that the datasets are increasing at fast pace [13], with a target audience of scientists that have specialized questions covering a wide range of data types and scales. Depending on your goal (see Rule 1) you may have to create a multi usage tool that could work for pre-established analysis workflows and exploratory usages as well as for capturing printable versions of your visual representations at a given state.

These static snapshots could be used for publication, as you should also considering publishing papers about your tool and the design choices behind it (datavis oriented columns are gaining in popularity, for example with the datavis section in *Frontiers in Bioinformatics*).

To wrap up these ten rules, here is a summary of what we deem most important, our take-home message for those who did not have time to read the article in detail:

- Going blindly into building a visualization tool is a bad idea; take time to learn about datavis principles first (we highly recommend Nature Methods' "Points of View" column [54]) and to carefully consider your tool's visual representations and interface designs.
- Genomics comes with multiple challenges, due in part to its diversity and scale of data. Make sure you have correctly identified the tasks your tool should complete and consider implementing interactions between different visual representations.
- Interact with your community at all times: during the design phase to correctly assess your end goal; during development to use real data and make sure you are going in the right direction; and on an ongoing basis once your tool is available for all to enjoy, to prevent it from ending in the "*oubliettes de l'histoire*."

D. References

1. Brian É. "Flatten the Curve!" But Which Curve? *Histoire & mesure*. 2020;XXXV: 233–246. doi:10.4000/histoiremesure.13544
2. Helena Klara Jambor. Martin Krzywinski: A pandemic of bad charts. 2022. Available: https://www.youtube.com/watch?v=_YGmfsKL8N8
3. Hadfield J, Megill C, Bell SM, Huddleston J, Potter B, Callender C, et al. Nextstrain: real-time tracking of pathogen evolution. *Bioinformatics*. 2018;34: 4121–4123. doi:10.1093/bioinformatics/bty407

4. O'Donoghue SI. Grand Challenges in Bioinformatics Data Visualization. *Frontiers in Bioinformatics*. 2021;1. Available: <https://www.frontiersin.org/article/10.3389/fbinf.2021.669186>
5. Buels R, Yao E, Diesh CM, Hayes RD, Munoz-Torres M, Helt G, et al. JBrowse: a dynamic web platform for genome visualization and analysis. *Genome Biol*. 2016;17: 66. doi:10.1186/s13059-016-0924-1
6. Cunningham F, Allen JE, Allen J, Alvarez-Jarreta J, Amode MR, Armean IM, et al. Ensembl 2022. *Nucleic Acids Research*. 2022;50: D988–D995. doi:10.1093/nar/gkab1049
7. Thorvaldsdóttir H, Robinson JT, Mesirov JP. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Brief Bioinform*. 2013;14: 178–192. doi:10.1093/bib/bbs017
8. Pavlopoulos GA, Malliarakis D, Papanikolaou N, Theodosiou T, Enright AJ, Iliopoulos I. Visualizing genome and systems biology: technologies, tools, implementation techniques and trends, past, present and future. *GigaScience*. 2015;4: 38. doi:10.1186/s13742-015-0077-2
9. Nusrat S, Harbig T, Gehlenborg N. Tasks, Techniques, and Tools for Genomic Data Visualization. *Computer Graphics Forum*. 2019;38: 781–805. doi:10.1111/cgf.13727
10. Yokoyama TT, Kasahara M. Visualization tools for human structural variations identified by whole-genome sequencing. *J Hum Genet*. 2020;65: 49–60. doi:10.1038/s10038-019-0687-0
11. Yardımcı GG, Noble WS. Software tools for visualizing Hi-C data. *Genome Biology*. 2017;18: 26. doi:10.1186/s13059-017-1161-y
12. Sedlmair M, Meyer M, Munzner T. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Transactions on Visualization and Computer Graphics*. 2012;18: 2431–2440. doi:10.1109/TVCG.2012.213
13. Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ, et al. Big Data: Astronomical or Genomical? *PLOS Biol*. 2015;13: e1002195. doi:10.1371/journal.pbio.1002195
14. Lex A, Gehlenborg N, Strobel H, Vuillemot R, Pfister H. UpSet: Visualization of Intersecting Sets. *IEEE Trans Vis Comput Graph*. 2014;20: 1983–1992. doi:10.1109/TVCG.2014.2346248
15. Kosara R. Graphs Beyond the Hairball. In: *eagereyes* [Internet]. 2012 [cited 24 May 2022]. Available: <https://eagereyes.org/techniques/graphs-hairball>
16. Walny J, Frisson C, West M, Kosminsky D, Knudsen S, Carpendale S, et al. Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff. *IEEE Transactions on Visualization and Computer Graphics*. 2020;26: 12–22. doi:10.1109/TVCG.2019.2934538
17. Westreich ST, Nattestad M, Meyer C. BigTop: a three-dimensional virtual reality tool for GWAS visualization. *BMC Bioinformatics*. 2020;21: 39. doi:10.1186/s12859-020-3373-5
18. Tang B, Li X, Li G, Tian D, Li F, Zhang Z. Delta.AR: An augmented reality-based visualization platform for 3D genome. *Innovation (N Y)*. 2021;2: 100149. doi:10.1016/j.xinn.2021.100149
19. Kuznetsov M, Elor A, Kurniawan S, Bosworth C, Rosen Y, Heyer N, et al. The Immersive Graph Genome Explorer: Navigating Genomics in Immersive Virtual Reality. 2021 IEEE 9th International Conference on Serious Games and Applications for Health (SeGAH). 2021. pp. 1–8. doi:10.1109/SEGAH52098.2021.9551857
20. Munzner T. *Visualization analysis and design*. CRC press; 2014.

21. Franconeri SL, Padilla LM, Shah P, Zacks JM, Hullman J. The Science of Visual Data Communication: What Works. *Psychol Sci Public Interest*. 2021;22: 110–161. doi:10.1177/15291006211051956
22. Wilke CO. *Fundamentals of Data Visualization*. O'Reilly Media. O'Reilly Media; 2019. Available: <https://clauswilke.com/dataviz/index.html>
23. Hattab G, Rhyne T-M, Heider D. Ten simple rules to colorize biological data visualization. *PLoS Computational Biology*. 2020;16: e1008259. doi:10.1371/journal.pcbi.1008259
24. Kim NW, Joyner SC, Riegelhuth A, Kim Y. Accessible Visualization: Design Space, Opportunities, and Challenges. *Computer Graphics Forum*. 2021;40: 173–188. doi:10.1111/cgf.14298
25. Ackland P, Resnikoff S, Bourne R. World blindness and visual impairment: despite many successes, the problem is growing. *Community Eye Health*. 2017;30: 71–73.
26. Babich N. The 15 Rules Every UX Designer Should Know | Adobe XD Ideas. In: Ideas [Internet]. 21 Feb 2020 [cited 29 May 2022]. Available: <https://xd.adobe.com/ideas/career-tips/15-rules-every-ux-designer-know/>
27. Anscombe FJ. Graphs in statistical analysis. *The American Statistician*. 1973;27: 17–21.
28. Matejka J, Fitzmaurice G. Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery; 2017. pp. 1290–1294. doi:10.1145/3025453.3025912
29. Treisman A. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*. 1985;31: 156–177. doi:10.1016/S0734-189X(85)80004-9
30. Boger T, Most S, Franconeri S. Jurassic Mark: Inattentional Blindness for a Datasaurus Reveals that Visualizations are Explored, not Seen. 2021 IEEE Visualization Conference (VIS). 2021. doi:10.1109/VIS49827.2021.9623273
31. Shneiderman B. The eyes have it: a task by data type taxonomy for information visualizations. *Proceedings 1996 IEEE Symposium on Visual Languages*. 1996. pp. 336–343. doi:10.1109/VL.1996.545307
32. Pedersen BS, Tang H, Freeling M. Gobe: an interactive, web-based tool for comparative genomic visualization. *Bioinformatics*. 2011;27: 1015–1016. doi:10.1093/bioinformatics/btr056
33. Yuvaraj I, Sridhar J, Michael D, Sekar K. PanGeT: Pan-genomics tool. *Gene*. 2017;600: 77–84. doi:10.1016/j.gene.2016.11.025
34. Lavikka K, Oikkonen J, Lehtonen R, Hynninen J, Hietanen S, Hautaniemi S. GenomeSpy: grammar-based interactive genome visualization. *F1000Research*. 2020;9. doi:10.7490/f1000research.1118237.1
35. LYi S, Wang Q, Lekschas F, Gehlenborg N. Gosling: A Grammar-based Toolkit for Scalable and Interactive Genomics Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 2022;28: 140–150. doi:10.1109/TVCG.2021.3114876
36. Galletta DF, Henry R, McCoy S, Polak P. Web Site Delays: How Tolerant are Users? *Journal of the Association for Information Systems*. 2004;5. doi:10.17705/1jais.00044
37. Mangul S, Mosqueiro T, Abdill RJ, Duong D, Mitchell K, Sarwal V, et al. Challenges and recommendations to improve the installability and archival stability of omics computational tools. *PLoS Biology*. 2019;17: e3000333. doi:10.1371/journal.pbio.3000333

38. Swaminathan R, Huang Y, Moosavinasab S, Buckley R, Bartlett CW, Lin SM. A Review on Genomics APIs. *Computational and Structural Biotechnology Journal*. 2016;14: 8–15. doi:10.1016/j.csbj.2015.10.004
39. Staton M, Cannon E, Sanderson L-A, Wegrzyn J, Anderson T, Buehler S, et al. Tripal, a community update after 10 years of supporting open source, standards-based genetic, genomic and breeding databases. *Briefings in Bioinformatics*. 2021 [cited 13 Jul 2021]. doi:10.1093/bib/bbab238
40. Hershberg EA, Stevens G, Diesh C, Xie P, De Jesus Martinez T, Buels R, et al. JBrowseR: an R interface to the JBrowse 2 genome browser. *Bioinformatics*. 2021;37: 3914–3915. doi:10.1093/bioinformatics/btab459
41. Martinez TDJ, Hershberg EA, Guo E, Stevens GJ, Diesh C, Xie P, et al. JBrowse Jupyter: A Python interface to JBrowse 2. *bioRxiv*; 2022. p. 2022.05.11.491552. doi:10.1101/2022.05.11.491552
42. Niu YN, Roberts EG, Denisko D, Hoffman MM. Assessing and assuring interoperability of a genomics file format. *Bioinformatics*. 2022; btac327. doi:10.1093/bioinformatics/btac327
43. Cheifet B. Where is genomics going next? *Genome Biology*. 2019;20: 17. doi:10.1186/s13059-019-1626-2
44. Diesh C. awesome-genome-visualization. [cited 6 May 2022]. Available: <https://cmdcolin.github.io/awesome-genome-visualization>
45. Lamprecht A-L, Garcia L, Kuzak M, Martinez C, Arcila R, Martin Del Pico E, et al. Towards FAIR principles for research software. *Data Science*. 2020;3: 37–59. doi:10.3233/DS-190026
46. Katz DS, Gruenpeter M, Honeyman T. Taking a fresh look at FAIR for research software. *Patterns*. 2021;2: 100222. doi:10.1016/j.patter.2021.100222
47. Social media for scientists. *Nat Cell Biol*. 2018;20: 1329–1329. doi:10.1038/s41556-018-0253-6
48. Carpendale S. Evaluating Information Visualizations. In: Kerren A, Stasko JT, Fekete J-D, North C, editors. *Information Visualization: Human-Centered Issues and Perspectives*. Berlin, Heidelberg: Springer; 2008. pp. 19–45. doi:10.1007/978-3-540-70956-5_2
49. Meyer M, Sedlmair M, Munzner T. The four-level nested model revisited: blocks and guidelines. *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors - Novel Evaluation Methods for Visualization*. New York, NY, USA: Association for Computing Machinery; 2012. pp. 1–6. doi:10.1145/2442576.2442587
50. Zong J, Lee C, Lundgard A, Jang J, Hajas D, Satyanarayan A. Rich Screen Reader Experiences for Accessible Data Visualization. 2022 [cited 29 May 2022]. Available: <http://vis.csail.mit.edu/pubs/rich-screen-reader-vis-experiences/>
51. Elavsky F, Bennett C, Moritz D. How accessible is my visualization? Evaluating visualization accessibility with Chartability. *Chartability*. Rome, Italy: John Wiley & Sons Ltd; 2022. Available: <https://www.frank.computer/chartability/>
52. Kale A, Kay M, Hullman J. Visual Reasoning Strategies for Effect Size Judgments and Decisions. *IEEE Transactions on Visualization and Computer Graphics*. 2021;27: 272–282. doi:10.1109/TVCG.2020.3030335

53. Weiskopf D. Uncertainty Visualization: Concepts, Methods, and Applications in Biological Data Visualization. *Frontiers in Bioinformatics*. 2022;2. Available: <https://www.frontiersin.org/article/10.3389/fbinf.2022.793819>

54. Evanko D. Data visualization: A view of every Points of View column. *Protocols & Methods Community*. 2013 [cited 29 May 2022]. Available: <https://protocolsmethods.springernature.com/posts/43650-data-visualization-a-view-of-every-points-of-view-column>

E. Supporting information

S1 File. Extraction processes and data used in rule 9. Text file including the extraction method and request string used to count tools from the awesome genome visualization list and publications related to genomic visualization from PMC.

III. Authors' contributions

All the contributors involved (Alan CLEARY, Matthieu CONTE, Éloi DURANT, Andrew FARMER, Eric GANKO, Cédric MULLER, Mathieu ROUARD and François SABOT) discussed and proposed rules (content and title) to include. My supervisors M. C., M. R. and F. S. are to be credited for the original idea leading to this paper.

First drafts for the rules were attributed as follows:

1. Articulate the need for new visualization tools – C. M.
2. Involve others early on – É. D.
3. Think about visual scalability and resolution – É. D.
4. Be creative, be bold – M. C.
5. Make data complexity intelligible – É. D.
6. Let your inner nerd shine, when needed – F. S.
7. Benchmark with diverse datasets – A. C.
8. Stay tuned to the genomic tool ecosystem and promote interoperability – E. G., M. R.
9. Keep up to date with related work – É. D.
10. Grow and support your user community – E. G.

É. D. supervised the writing process, revised and enriched all rules with appropriate references, set a consistent writing tone and wrote the introduction and conclusion. É. D. and M. R. imagined the figures, É. D. drew them. É. D., A. F., M. R., and F. S. added supplementary paragraphs to the drafted rules. A. C., É. D., A. F., and M. R. proofread the manuscript in its entirety. F. S. was the corresponding author for the article's submission.

IV. Addendum

The 10 rules evolved on multiple occasions, from their initial definition through the first drafts and the successive following versions. Some material did not make it to the final cut to avoid overloading the rules, or to stay within the scope. Below are some of these scrapped ideas, which further detail the rules presented above.

First, the original scope of this paper was limited to pangenomics, which explains the list of authors as we were all involved in research on pangenomes, to various extent. The first rule originally featured a whole paragraph about the interest of pangenomics and the existing visualization tools, which was removed to expand the scope to genomics in general. Many principles that applied to the development of tools for the visualization of pangenomes could be applied to the broader genomics field, and the authors agreed that it would be more of interest for the readership of the PLOS Computational Biology journal, resulting in higher chances to be accepted, in particular considering the scope of the "Ten Rules" papers published until now.

Among the rules I worked on, some elements were removed from Rule 2 – Involve others early on and Rule 5 – Make data complexity intelligible.

The paper by Sedlmair et al. [246] identified more roles than listed in the final Rule 2, and I originally proposed to add the role of datavis designer explicitly, as a different audience could read our 10 rules, presumably unfamiliar with the world of datavis. I also included a part emphasizing the importance of communication at all steps of the design process, to avoid common pitfalls.

The original text described these additional roles as follows:

Gatekeeper – People with decision-making power who can choose whenever a project starts or stops and have the last say on the final product. Identifying them early can save a

lot of time and trouble by continuously making sure that they agree with the directions taken rather than during a final rush before production.

The remaining roles are described as useful while not mandatory, but it can help to know that such roles might exist: **Connectors** (who can help you contact other useful people easily); **Translators** (who can clearly explain and act as links between domain-experts and a non-specialist audience such as developers); **Co-authors** (who could help to write about the tool in a scientific paper); **Fellow tool builders** (who might have worked on the project before visualization was considered and have useful information but should not be mistaken for front-line analysts).

To these we would add the role of the actual **Datavis designer**, as many people creating tools do not directly come from the wonderful world of datavisualization and “you” as a reader might be one of the previously mentioned roles instead.

Particular attention should be given to the communication between these roles, especially for larger projects where they might be split between different individuals. Walny et al. [256] highlighted the importance of such communication for faithful data representation, with concrete examples of how things could go wrong and challenges to overcome: “adapting to changing data, anticipating edge cases in data, understanding technical challenges, articulating data-dependent interactions, communicating data mappings, and preserving the integrity of data mappings across iterations.”

The original Rule 5 included an introductory paragraph providing more omics context, referencing one of the “Points of View” column, by Shores & Wong [257]. This column emphasized the importance of data visualization for analyzing biological data. This paragraph was written as follows:

Biological data come in all shapes and sizes and vary with the studied organisms, the types of analysis, the technologies used...: short reads for de novo sequencing of multiple bacterial genomes would not produce the same data than long reads for structural variation characterization in a human genome. Through visual representations, a visualization tool is used for communication (either for presentation or exploration purposes) of processed information from this digital primordial soup [257].

On a side note, a special attention has been brought to the titles of the 10 rules, so that they would all correspond to actions that anyone could do to improve their process of creating a visualization tool. They all start with a verb and are positive rather than negative. I wanted the rules to highlight possibilities rather than pitfalls. This proved difficult for the Rule 6, as its core message is about restricting the development fancies to when it is actually useful. The original phrasing that I found was “*Outnerd yourself, when needed*”, with the idea of making use of all of one’s informatic skills—*be even more nerd than you already are*. Unfortunately, the meaning of this title was too obscure to be kept.



Panache

I. Why create a new pangenome visualization tool

A. Context

During my Master 2 internship in 2018, prior to my PhD project, I benchmarked different visualization tools, trying to find one that we could use with plant pgAtlases and pangenomes, and investigating the different file formats in use. The majority of the available tools at the time was built for pgAtlases only—as illustrated in Table 2, most tools available then fall into the Unspecific or Qualifying categories—and provided no information on structure. Moreover, genome graphs were not as common as they can be today, *Bandage* [174] was not popularized yet, and the first versions of *Sequence Tube Maps* [194] were not suited for many genomes which would be stacked together, with crossing lines between blocks. We were looking for tools that could scale to plant pangenomes, both from a hardware and readability points-of-view.

The available pgAtlases tools had limitations: the first was that they were not able to scale up to plant pangenomes as the number of genes increased. The second was that we could not use it for pangenomes, while the inter-genic material is of interest within plants. Moreover, we observed that some of these tools were not functioning properly, often because of a lack of maintenance [258]. Still, they were useful for figuring out what would be interesting features to have within a pgAtlas/pangenome visualization tool, as well as possible designs and their limitations.

B. Tools benchmarked

The four tools benchmarked during this internship were, in the order they were tested, UpSet [196], Pan-Tetris [143], PanViz [159] and Panoptes [225]. Sequence Tube Map [194] was excluded as it had already been identified as unfit for our goals. I did not have enough time to properly test panX [160] with our data. GenomeRing [180] and anvi'o [173] among others were not identified as pangenome visualization tools then. We also excluded existing platforms for pangenomics which were unpractical at the time and were not easily adaptable to our data.

For testing purposes, the tools were given (when compatible) a PAV matrix of genes from five banana genomes, grouped with GET_HOMOLOGUES [101] into 71,725 clusters.

Table 8 summarized the results obtained with the four tools tested at the time.

UpSet was the most functional tool out of the four but is limited to the visualization of sets (it could only work with pgAtlases), and started to get hardly readable with more than a dozen of sets (i.e. genomes) as the displayed number of intersections grew exponentially. Moreover, being Unspecific, it did not provide much biological information. In its online version, it uses PAV in **Tab Separated Value (TSV)** format and a JSON file for configuration.

Pan-Tetris can use two different kinds of inputs: files from the in-house unpublished tool *PanGee* which produces TSVs with multiple columns per genome, or simpler TSV in association with TIGRFAM codes, linking biological functions to gene identifiers. When tested with our banana data (only a TSV file, with no TIGRFAM associated) or example files (example output of *PanGee* provided on Pan-Tetris' website), multiple bugs occurred: the genome name were not displayed, the pan-gene names were not updated when scrolling the PAV matrix, resizing froze the display (as seen in Appendix XXXVI). Those bugs, associated with the poor support for user-provided data, disqualified Pan-Tetris from being a visualization tool that we would use with our data.

PanViz uses files created with a companion R tool called *PanVizGenerator* [169]. It takes PAV written as CSV files completed with GO terms per row detailing the associated biological functions. Easy to play with, and with multiple interactive options, its interface proved to be inappropriate

for normal-sized screens and lacked documentation both within and outside of the tool. More importantly, it lost responsiveness and eventually froze with our banana data.

Panoptes was the last tool I tested during my internship, described as a prototype still under development at the time. It is designed to be highly customizable, with the drawback of being difficult to set and manage, with many options to manually provide with each file import. For example, when setting the genome browser representation Panoptes would need a FASTA with the nucleotide sequence of a reference genome, a GFF3 file for the position of genes, exons, etc, both linked to a VCF file through multiple configuration files in **YAML** (YAML Ain't Markup Language, formerly **Yet Another Markup Language**) format. Completed with a difficult installation and some bugs when the files were not exactly formatted or with minor modifications (an empty line in the middle for example), it made the tool hard to use correctly. Being based on a reference genome, it was not properly fit for pangenomes either and was limited by the then-current state of web technology for plotting millions of data points. We therefore abandoned it because of its impracticality and complexity.

Table 8: Out of the four visualization tools tested, none was deemed suitable for our pangenomes; Each visualization tool tested with our data had disadvantages that prevented us from adopting it. UpSet did not provide enough pangenome specific information. Pan-Tetris was not working properly, especially with user-provided data. PanViz had bugs that hindered the UX when used with too many data. Panoptes was too difficult to install and expand to the use of our data. Table recreated from my MSc2 report, made in mid-2018.

	UpSet	Pan-Tetris	PanViz	Panoptes
Year of publication	2015	2015	2015	2015
Last update	2016	2015: Version 0.9	2017	Actively maintained
Language	JS, HTML, CSS	Java	JS, HTML, CSS, R	JS, HTML, CSS...
Input formats	PAV + JSON	PAV + TIGRFAM	PAV	VCF, GFF, FASTA, YAML...
Deployment	Online or local	Local	Local	Local
Interactivity	++	-	+	+
Ease of use	++	-	+	--
Scalability	+	+	-	+
Available details	-	+	++	++
Working condition	+	--	-	-

C. Identified gaps

The benchmarked tools were not considered as suitable for plant pangenomics: the visual representations lacked biological information or would not scale well to dozens of genomes, the implementations would bug, or would not be user-friendly enough. We therefore chose to design and develop a new pangenome visualization tool, which is how my PhD project came to be.

We wanted a tool that would be light, easy to install (web application preferably, to avoid problems of compatibility between setups), and usable by various organisms. This meant that the format had to be standard enough, and that it could work for both pgAtlases and pangenomes. As *positioned* and *structural* visualization tools were underrepresented then, there was a gap to fill for the visualization of pangenomes. Moreover, the tool had to be scalable to plant genomes, both in visual design (providing efficient visual representation of large datasets is a complex task) and computational performances (a tool should be responsive enough to enable interactions and exploration of its visual representations). Furthermore, the tool should provide an alternative to Graphs and Circular-based visual representations, already heavily investigated by other groups and prone to visual clutter.

II. Design of the visual representation

A. Panache, a PAV browser

After exploring different visual representations (as exemplified by Appendix XXXVII and Appendix XXXVIII), I focused on the visual representation of a PAV matrix, explorable through navigation similarly to a genome browser, as illustrated in Figure 26.

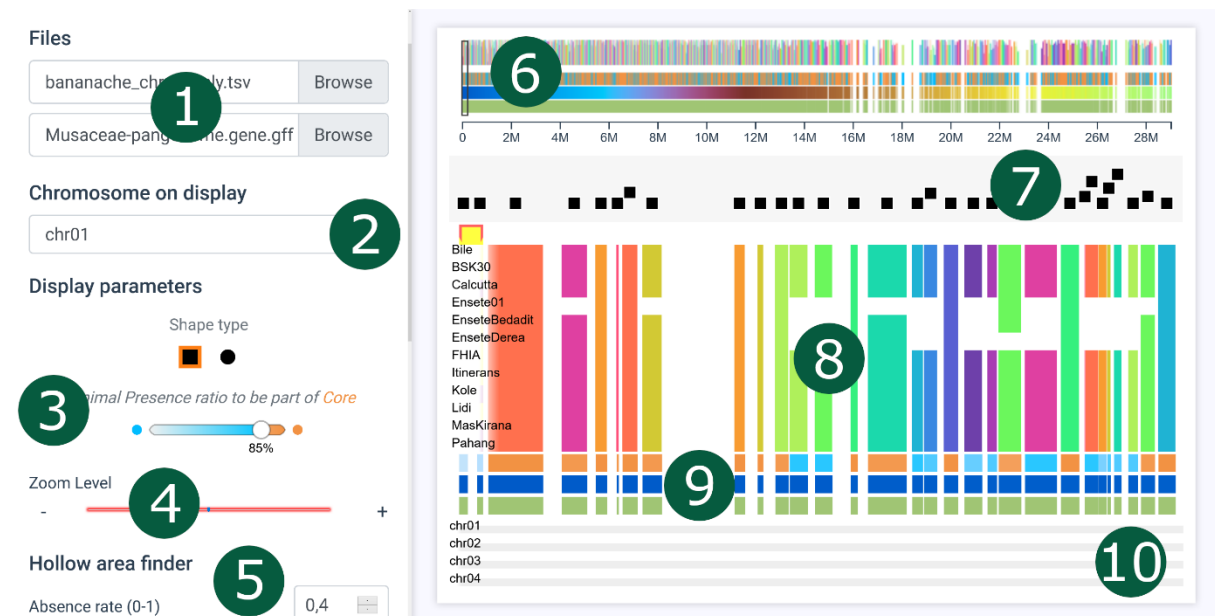


Figure 26: Panache’s UI is divided between a main view and a menu panel; The menu contains multiple interactive elements: 1) File loaders; 2) Choice of Panchromosome to display; 3) Core threshold; 4) Geometric zoom level; 5) The Hollow Area Finder (see Panache II.A.4); ...all of which dynamically update the main view composed of: 6) A miniature overview of the whole PAV matrix; 7) A beeswarm plot of gene annotations; 8) The PAV matrix displayed as panBlocks; 9) Tracks summarizing panBlocks; 10) Similar panBlocks across panchromosomes, when available.

1. Visual representation of a PAV matrix and tracks of information

Panache’s main visual representation is based on a binary heatmap representing a PAV matrix of panBlocks positioned on a linear coordinate system (see Figure 27). These panBlocks could either be genes or fragments of sequences, or other, depending on what users want to study.

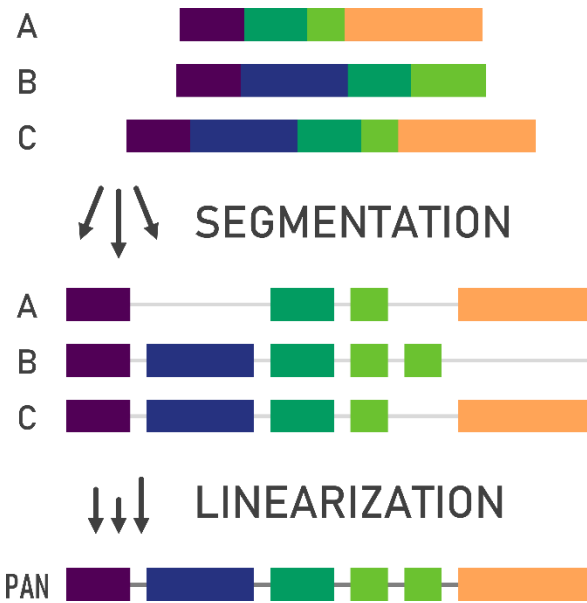


Figure 27: panBlocks are built from common fragments found in different genomes; They can then be distributed on a linear string, following an existing genome's coordinate system or a linear panreference.

The heatmap represents genomes as lines and panBlocks as columns, each intersection is filled with a color when that panBlock is present within the genome. Only a part of the full matrix is displayed at a time, a user can pan through the whole matrix by using a miniature overview displayed as a histogram (see Appendix XXXIX) at the top of the UI, schematized in Figure 28. Each panBlock can have a dedicated color for its column, encoding for example a biological function (e.g. Gene Ontology (GO)) associated with it. If no information of function is available, a pseudo-rainbow pattern is applied to distinguish each column.

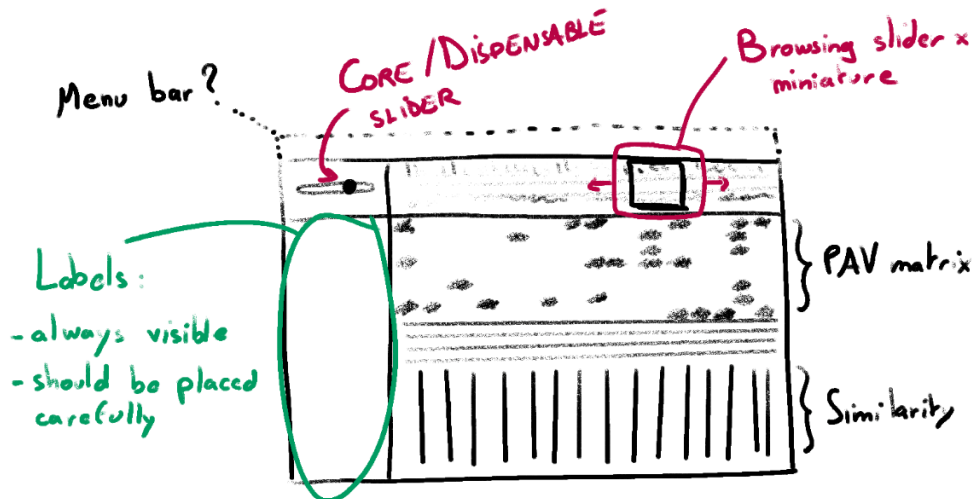


Figure 28: The PAV matrix can fit into an encapsulating UI, with interaction available for navigation; A miniature of the whole matrix at the top can be used to pan through the whole matrix. A slider enables users to choose the threshold to apply for the definition of the core genome, along with other options placed in a panel on the left of the UI.

Additional tracks of information are located below the PAV matrix. The tracks encode different summary information, such as the **core** or **variable** status of a panBlock, its position and size within the linear coordinate system, and the number of repeats found elsewhere in the pangenome (see Panache II.A.2).

Each track is color encoded. The track for the presence status dynamically encodes the category of the panBlock in orange (■) or blue (■) depending on the value of a threshold set by the user,

distinguishing the **core** and **variable** categories, respectively. The combination of orange and blue is colorblind-friendly.

The position track is encoded with an alternative to rainbow maps as a sequential color palette varying in hue and luminance (see Appendix XL). Hues are chosen as to be easily distinguishable, enabling users to quickly identify an approximative location for a block.

This main view representing the PAV matrix is integrated within a UI composed of different elements, schematized through Appendix XLI to Appendix XLVI.

2. View dedicated to conserved blocks

Though the choice of the method for the creation is left to the users, I assumed that cooccurrences (see State of the Art I.F.1.a) could exist in the files provided to the visualization tool, encoding multiple possible positions of panBlocks and further refining their PAV status. This track has evolved from the first drafts which featured circles whose area encoded the number of repetition (as illustrated in Appendix XLVII). These circles were not kept as the visual channel of areas is inefficient for comparisons and were distorted when I encoded panBlocks of different widths. I chose rectangles instead, as their length is more easily compared between lines (as schematized in Appendix XLVIII).

This track is divided in two parts, one with a heatmap encoding the number of repeats found within the whole pangenome for a given panBlock, and the detail below of the distribution of these repetitions within the different (pan)chromosomes.

I drafted additional visual representation to detail the repetitions of specific panBlocks, as illustrated in Appendix XLIX to Appendix LIII. They have not been implemented within the developed tool as this information was not available within the files used at the time and it was not a priority during the development of Panache compared to other features.

3. Visual representation of gene annotations

Following user expectations, gene annotations were added as a beeswarm plot, on top of the PAV matrix. This plot displays marks at the position of each annotated gene according to the linear coordinate system used for the PAV matrix, vertically distributed so that no mark overlaps another one. Users can access the detailed information by hovering on a mark, which will display an 'annotation card' as showed in Figure 29 below.

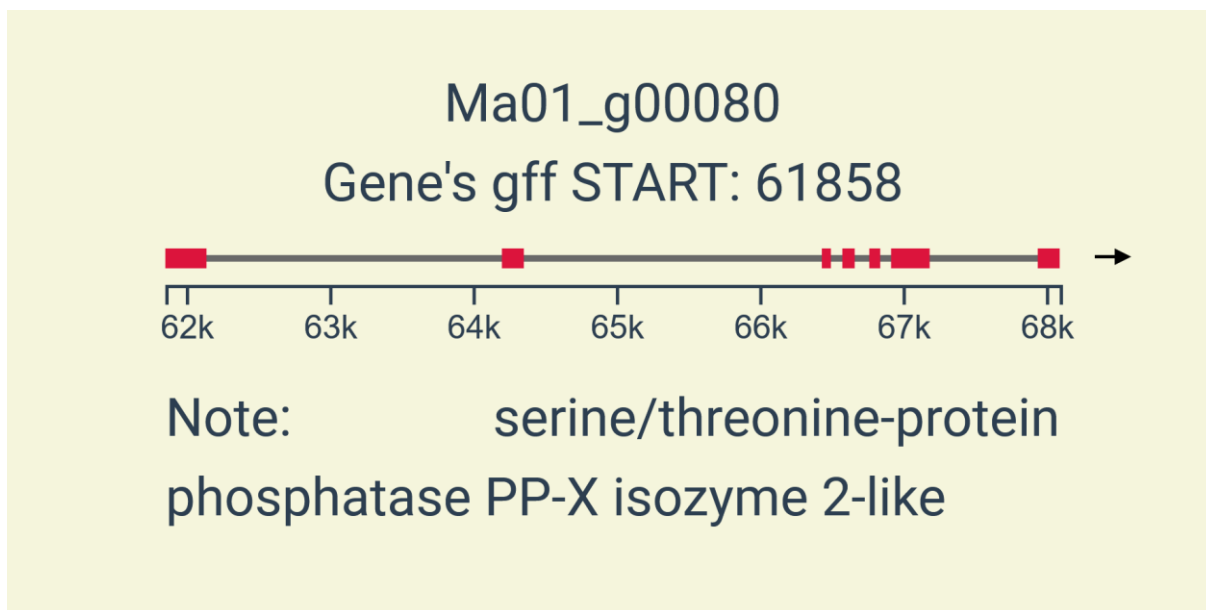


Figure 29: Annotation Cards display details on a gene annotation; From top to bottom: a gene's ID, its position on the linear coordinate system, the positions of exons on this coordinate system, along with the strand represented as an arrow (forward here as the arrow points right), the functional annotation available for this gene. Not shown is the information of overlapping genes, as this cannot be directly visible within the beeswarm plot as all annotations are attributed a universal width.

4. Hollow Areas

One of the features that to be included to the original prototype was an automated system enabling users to 'jump' at the locations of consecutive absences to facilitate the automatic discovery of large PAV along the panchromosomes. The details of the implementation of that 'Hollow Area Finder' are available at Panache III.B.5.

These areas are highlighted within the PAV Matrix by a yellow background, with brackets delimiting the position of every different area (see Appendix LIV). One panBlock can be included within multiple hollow areas if the consecutive absence patterns from different genomes overlap in one place.

B. Identification of desirable features

1. Face to face discussions

I discussed with identified target end users from Syngenta, IRD, and Bioversity International to identify the features that should be included within a pangenome visualization tool. Most discussions oriented the drafts of visual representations towards multiple directions. One of these discussion sessions for example (whose notes are available at Appendix LV) highlighted the need for a representation of the positions of panBlocks within the linear coordinate system. We envisioned that these blocks could be filtered out and / or reordered, modifying their order of appearance within the display. Keeping the information of their original positions could highlight patterns, for example regions most likely to be core.

Among these discussions, I conducted interviews with group project leaders at Syngenta, in order to gather ideas for a larger survey for the scientific community. These discussions helped me to identify elements to explain within the survey as well as questions to ask and the first missing features of interest.

2. Public survey

To collect the needs and expectations, I created a survey shared both within my research organisms and with outside scientists (through Twitter and conference posters). This survey (whose layout is available in Appendix LVI) was divided in three parts; an introductory part giving details about pangenomes and asking how knowledgeable the respondents were about pangenomes (to adapt later questions), a section with hands-on experience of a draft version of Panache built around a *Brassica napus* pangenome [142], and a final section asking respondents to order missing functionalities to their liking.

The results of this survey, available in Appendix LVII to Appendix LXVII, highlighted the missing functions that had to be prioritized during development, both for the improvement of Panache's prototype and general functionalities that should be added into a pangenome visualization tool.

The survey gathered answers from 97 respondents, including 37 who reached the final question. From this survey, I highlighted that respondents were interested in both pgAtlases and pangenomes and that Panache's prototype was missing interactivity (as revealed by the heatmap of click events within the UI, see Appendix LXVIII). Missing features for Panache's prototype included links to gene annotations, reordering the genomes within the PAV matrix and zooming to see the nucleotides, among others. The first two have been added to Panache since, and the latter has been addressed with another design (see SaVanache's chapter).

The identification of needs and expectations continued even after the survey closed, through discussions during PhD committees, conferences, or through GitHub issues when users identified features that they wanted to see added within Panache.

III. Development of a web-based tool

We wanted Panache to be lightweight and easy to install, with as little configuration as possible. We therefore chose to develop it as a web application, available on GitHub⁴⁰ along with a detailed documentation⁴¹.

A. Techniques

1. Data files and formats

a. PAV matrix

Panache's main file is a PAV matrix, inspired by various formats. It is a TSV which, as shown in Table 9, combines BED-like columns for position on the chosen linear coordinate system (0-based), additional columns of metadata, and one column per genome, encoding the presence or absence of each panBlock (one row being one panBlock). Each column is named after a header dedicated to this format.

The PAV matrix can adapt to multiple syntaxes: either binary matrix written with 0 and 1, count matrix where 0 encodes the absence and any other integer the presence, matrix of gene names where the name of a gene annotation found within a genome encodes presence, and 0 encodes absence. In all cases, 0 always encodes the absence of a panBlock, and any other value encodes its presence.

The additional columns of metadata can contain a nucleotide sequence, information on contained biological function, or the coordinates of the identified repeats of that panBlock.

⁴⁰ <https://github.com/SouthGreenPlatform/panache>

⁴¹ <https://github.com/SouthGreenPlatform/panache/wiki>

Table 9: Panache PAV matrix can be written with integers or strings alike

Chromosome	FeatureStart	FeatureStop	...	Geno1	Geno2	Geno3
[String]	[Integer]	[Integer]	...	[String Integer]		
Chr2	21	230	...	0	1	1
Chr5	454	123	...	73	0	4
ChrUnknown	0	1234	...	GeneID_A	0	GeneID_B

b. Others

Panache can also use gff3 files of annotation to display the Annotation Cards (see Panache II.A.3) or files in Newick format to display genomes sorted according to a phylogenetic tree.

2. Technical choices

a. Programming language and libraries

Panache has been developed in JavaScript for its wide availability and compatibility with multiple systems. It was also chosen in order to use D3.js, a powerful library widely used in datavis, enabling the manipulation of elements within a webpage, including **Scalable Vector Graphic (SVG)** elements. Its dynamic handling of variable data is useful to make interactive visualization, including common types of interactions (zooming, panning, brushing-and-linking...) or more complex transitions.

For Panache, D3 is especially useful for triggering on-click events while capturing the mouse position at the same time and comes with a useful 'scale' system enabling parameterizable conversions between different coordinate systems, for example from nucleotide coordinates to pixel positions or to a color palette.

b. Choosing a JS framework

Panache's first prototype was written in a single JavaScript file with thousands of lines. This quickly revealed to be a limitation to the addition of new functionalities and dynamic variables as all relations had to be manually rewritten each time.

I therefore decided to work with the framework Vue.js, with which web applications can be written as multiple components, one per file. Vue was with React and Angular one of the three main JavaScript available, which implied that web developers could be familiar with already. I chose it out of the three as it was advertised as more intuitive than Angular and easier to learn than React.

With the help of an external web developer, we adapted Panache's code to the framework Vue, which made the addition of new features easier.

c. Deployment

For easy installation, Panache comes with two possible deployment methods.

It is available through a Docker container that can be run to create and run a version of Panache either locally or on any **Internet Protocol (IP)** address of choice.

It is also possible to build the production files and deploy them within a web server, for example to host pre-formatted instances of Panache.

B. Implementation

In this section, I highlight some of the choices made when building Panache which necessitated special conception.

1. File parsing and companion script

Panache files are parsed on upload, converting them into JS objects with additional properties like the presence counter or the distribution of repeats along panchromosomes. These JS objects can either be obtained by uploading a normal PAV file (see Panache III.A.1) within Panache's default docker instance, or by using a companion Python script which enables the conversion of files from Panache format to JSON used within the web application. Conversions are available for both the PAV matrix and the gff3 files and can be used to set up pre-loaded instances of Panache where a user cannot choose the dataset to display.

2. Filtering Objects to what can be seen

In order to improve Panache's performances, the drawing functions only create SVG elements when they are visible on the main display. Scripts filter out all the panBlocks not located within the window encompassing the current region on display. The selection of this region is detailed in Appendix LXIX and Appendix LXX.

3. Conception of the geometric zoom

The zoom system evolved throughout the different iterations of Panache. Initially divided into two conversion scales, one providing a global vision and another focusing on less than a hundred blocks on screen I turned it into a more restrictive version. Visualizing all the panBlocks at once does not bring any more information than the miniature overview and leads to bad performances as the web page cannot handle more than some hundreds of SVGs at once.

The current zoom is based on the mean size of blocks and allows users to see between a dozen and a hundred panBlocks at once.

4. Canvas and SVG for the miniature

Creating high amounts of SVGs within one web application demands a lot of resources and is not viable for a smooth UX. As the miniature has to represent all panBlocks, I use a combination of canvas and SVG for the creation of this component. HTML5 canvas enables the creation of static images within a single web element and therefore takes less memory than multiple SVGs. However, it does not provide easy interaction within the images drawn.

In Panache, the miniature is drawn on a canvas element, and interactive SVG elements are overlaid in order to provide the panning system where a user can click on a position within the miniature to actualize the main view.

5. Hollow Area Finder

The Hollow Area Finder locates areas with consecutive absences. A user can use two parameters as input: the minimum absence rate across genomes desired and the number of consecutive blocks that should be absent. Panache can compute the positions of areas matching these parameters, and stores these areas within a sparse matrix which can be easily parsed to find matching regions located before or after the region currently on display.

As simply identifying consecutive panBlocks with a minimum absence rate would not target areas where the absences are consecutive within one genome, Panache works on successive PAV profiles instead. A PAV profile is created for each panBlock whose presence rate matches the input parameter. If the next n panBlocks (n being the other input parameter) match that rate too, then Panache checks if the absences are consecutive.

This background computation was initially always on but was using a lot of resources as the sparse matrices were updated every time the users panned. Later development enabled users to toggle the Hollow Area Finder on or off at will, the default being off.

6. Sorting options

As sorting the genomes was one of the main missing functionalities asked by users, I designed multiple sorting options. These sorting options were implemented after Panache's initial public release with the help of an intern under my supervision. Detailed within Panache's documentation⁴², these sorting options included:

- Alphabetical and reverse alphabetical order
- Phylogenetic-based order, using a Newick file
- Decreasing order based on a compliance score to certain PAV patterns of genes
- Order following a **H**ierarchical **C**lustering **A**lgorithm (**HCA**) built from local PAV patterns

7. Performance evaluation

Panache is limited by the size of files used for the PAV matrix⁴³. Benchmarking highlighted that this limitation was due to the overall file size rather than the number of genomes or blocks involved. These limitations also proved to be machine-dependent during later development sessions.

To bypass some of these limitations, static stores have been added to Panache to reduce the number of elements dynamically watched within the store management system used by Vue.

8. Identified missing functionalities

Panache does not include all missing functionalities yet, and new ones have been identified with additional discussions during conferences or after presentations, or even through GitHub's issue system where users can directly contact the developers of a tool⁴⁴.

These functionalities include:

- Zooming in to see DNA sequences
- Having sorting and filtering options for the panBlocks (instead of the genomes only)
- Customization of the display (color schemes, elements to hide or show...)
- Panning through the PAV matrix when scrolling with the mouse
- Modifying the resolution of panBlocks at will
- Switching between horizontal and vertical display
- Providing a view dedicated to the repeats
- Comparison of groups of genomes
- Adding supplementary categories for the **core** threshold (not just **core** and **variable**)
- Visualizing multiple regions at once
- Export of data and images

⁴² <https://github.com/SouthGreenPlatform/panache/wiki/Sorting-options>

⁴³ <https://github.com/SouthGreenPlatform/panache/wiki/Performance>; not updated with the latest improvements yet

⁴⁴ Brett Chapman for instance proposed many improvements:
<https://github.com/SouthGreenPlatform/panache/issues/32>

- ...

This missing functionalities spans features useful for exploration and analysis purposes as well as quality of life improvements.

IV. Published Application Note

Bioinformatics, 37(23), 2021, 4556–4558

doi: 10.1093/bioinformatics/btab688

Advance Access Publication Date: 2 October 2021

Applications Note

OXFORD

Genome analysis

Panache: a web browser-based viewer for linearized pangenomes

Éloi Durant ^{1,2,3,4,*}, François Sabot ^{1,4}, Matthieu Conte² and Mathieu Rouard ^{3,4,*}

¹DIADÉ, Univ Montpellier, CIRAD, IRD, Montpellier 34830, France, ²Syngenta Seeds SAS, Saint-Sauveur 31790, France, ³Bioversity International, Parc Scientifique Agropolis II, Montpellier 34397, France and ⁴French Institute of Bioinformatics (IFB)—South Green Bioinformatics Platform, Bioversity, CIRAD, INRAE, IRD, Montpellier 34398, France

*To whom correspondence should be addressed.

Associate Editor: Tobias Marschall

Received on May 3, 2021; revised on July 28, 2021; editorial decision on September 24, 2021; accepted on September 24, 2021

Abstract

Motivation: Pangenomics evolved since its first applications on bacteria, extending from the study of genes for a given population to the study of all of its sequences available. While multiple methods are being developed to construct pangenomes in eukaryotic species there is still a gap for efficient and user-friendly visualization tools. Emerging graph representations come with their own challenges, and linearity remains a suitable option for user-friendliness.

Results: We introduce Panache, a tool for the visualization and exploration of linear representations of gene-based and sequence-based pangenomes. It uses a layout similar to genome browsers to display presence absence variations and additional tracks along a linear axis with a pangenomics perspective.

Availability and implementation: Panache is available at github.com/SouthGreenPlatform/panache under the MIT License.

Contact: eloi.durant@ird.fr or m.rouard@cgjar.org

1 Introduction

The widespread use of fast and affordable sequencing technologies unveiled how much genomic information was lost when relying on a single and unique reference genome. For instance, it was found that about 10% of additional DNA was not captured by the current human reference genome (Sherman *et al.*, 2019). By leveraging data of multiple references instead, a new era of genomics emerged: Pangenomics. This is now being applied from bacteria to eukaryotes and has been increasingly used in more complex genomes such as humans and plants. As reviewed in Golicz *et al.* (2020), some studies handle this approach through a gene or functional annotation lens while others extend it to DNA sequences, especially when the studied organisms are eukaryotes.

Still, more tools are needed to help pangenomics to reach a broader audience within the scientific community (Computational Pan-Genomics Consortium, 2018; Golicz *et al.*, 2016b; Tranchant-Dubreuil *et al.*, 2019). While recent progress has been made to compute and store pangenomes (Garrison *et al.*, 2018; Li *et al.*, 2020), the tool landscape is particularly barren when it comes to visualization. Tools such as Pan-Tetris (Hennig *et al.*, 2015), PanViz (Pedersen *et al.*, 2017) or PanX (Ding *et al.*, 2018) were designed for gene-based pangenomes and do not scale well to large-scale

eukaryotic studies. Indeed, this gene-centric definition does not take into account positions within genomes, thereby blending paralogs together, and ignoring non-coding sequences despite their crucial influence on phenotypes (Maston *et al.*, 2006). The current trend for sequence-based pangenomes is to use graph visualization software like Bandage (Wick *et al.*, 2015), a general tool for navigating assembly graphs, but alternatives dedicated to pangenomes and their inner properties are yet to be refined and adopted. For instance, Sequence Tube Maps (Beyer *et al.*, 2019) and MoMi-G (Yokoyama *et al.*, 2019) both focus on structural variations from individual genomes but lack information on the pangenome itself. They indeed do not have direct visual cues for the identification of the most represented parts of a pangenome and lose clarity when more than a dozen genomes are involved.

As useful as graph representations may be, they can easily be overloaded with content, resulting in a ‘hairball’ effect that is hard to read and explore (Yoghoudjian *et al.*, 2020). Linear representations of genomes have their own weaknesses (Nielsen and Wong, 2012) but are widely used in a variety of genome browsers, and most efficient when it comes to exploration tasks. Former attempts such as UCSC’s snake tracks (Nguyen *et al.*, 2014) focused on sequence alignments on one reference, which lack clarity when numerous genomes are involved. Here we introduce Panache—the

© The Author(s) 2021. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

4556

PANgenome Analyzer with CHromosomal Exploration—a web browser-based viewer which renders interactive linear representations of pangenomes as successions of pangenomic blocks, one panchromosome at a time.

2 Features

Panache is designed to display a linear representation of pangenomes. This representation is based on the idea that every genome within a pangenome can be divided into multiple blocks, with each block potentially shared with other genomes. Such blocks could be either DNA sequences (extracted from the nodes of a graph pangenome for example) or genes. Blocks from all genomes could be laid out and ordered along a single string, which would then serve as a flattened pan-reference, as illustrated in Figure 1A. One could also imagine ordering the blocks according to an existing genome instead, using its own linear coordinate system as a reference. Every pangenomic block can therefore be represented in an easy-to-browse visualization, with additional tracks of summarized information such as a block's presence/absence status or whether it belongs to the core genome (most present blocks) or the variable genome (also referred to as dispensable genome).

While a graph representation might give a better sense of structural variations or of a genome's full sequence within a pangenome, a linear representation allows more reproducibility when exploring data thanks to its fixed and ordered coordinate system. A fixed order allows users to experience the same exploration between visualization sessions, contrary to graphs that may be represented differently every time a file is loaded. Moreover, missing information can be visually hinted at even when not directly available. For example, an additional track can specify which blocks are repeated elsewhere in the pangenome.

The tool comes with a variety of navigation and exploration related functionalities: choosing which panchromosome to display, browsing through it and sorting related individuals with various options based on known phylogenetic information or presence/absence status (gene list or pattern in a selected region). In addition, users can jump automatically to areas enriched in absent blocks with the so-called hollow area finder. It also allows interactive events such as on-the-fly modifications to the core/variable threshold or to the zoom level and hovering over visual elements to display additional information such as functional annotation pop-up windows. All available functionalities are further detailed within Panache's documentation.

Figure 1B illustrates how Panache displays linear pangenome data using a pangenome generated in banana (*Rijzaani et al., 2021*).

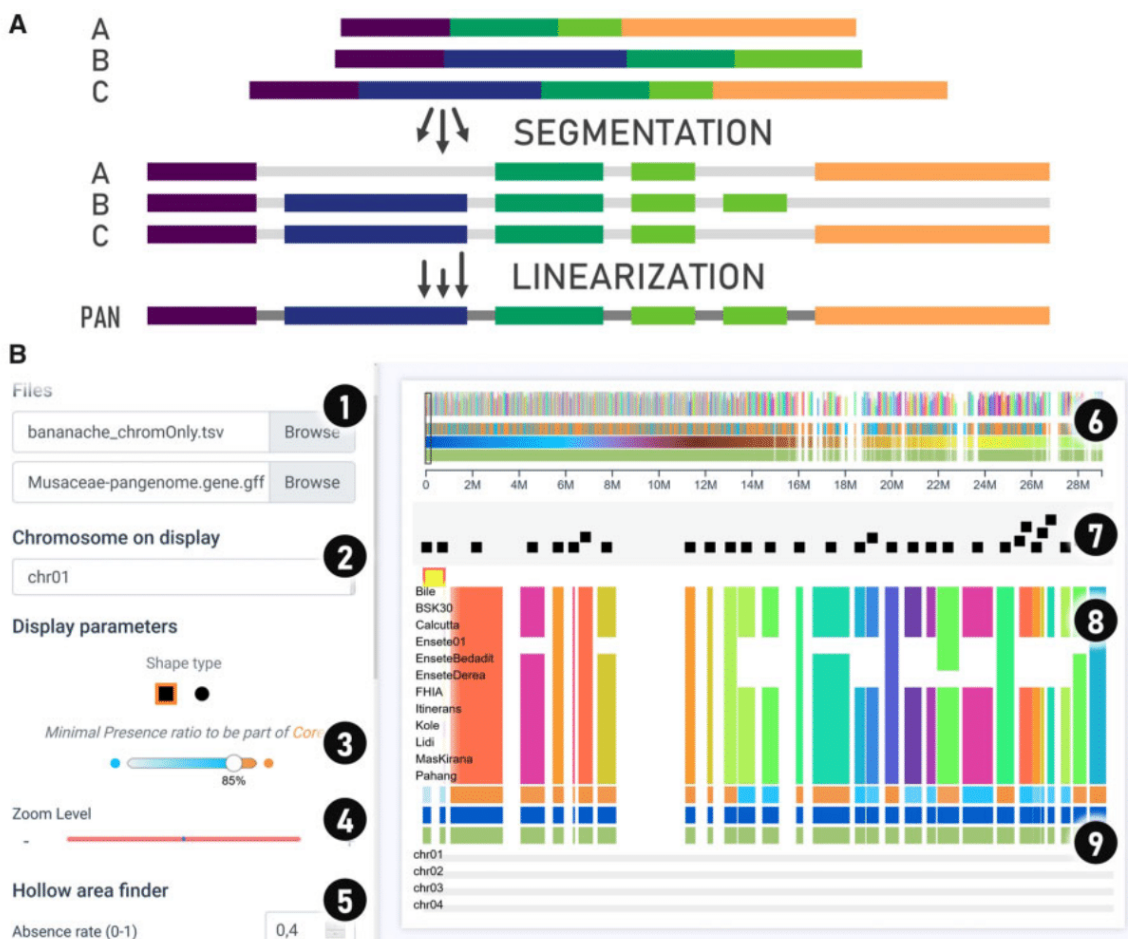


Fig. 1. Panache offers a linear representation of pangenomes, with block information detailed through multiple tracks much like classic genome browsers. (A) Linearized pangenomes represent chains of present/absent pangenomic blocks on a single string. (B) Panache's interface for browsing through the Presence/Absence matrix and navigating through panchromosomes. The interface is divided into multiple parts: (1) file inputs; (2) panchromosome to display and navigation options; (3) customizable threshold for the core and variable genomes; (4) zoom option to modify blocks' sizes; (5) further exploration options including a Hollow Area Finder for automatic detection of areas with consecutive absence and sorting options; (6) miniature overview of a whole panchromosome used for navigation on click; (7) track of gene annotations displayed as centered marks in a swarm plot preventing overlaps, detailed cards of annotation are available on hovering; (8) presence/absence matrix of pangenomic blocks, displaying genomes in line and blocks in column; (9) hoverable tracks of summary information (core/variable status, pangenomic coordinates and blocks' width, amount of repetition and their distribution)

A set of 34 878 genes from 15 genotypes have been grouped into chromosomes and positioned linearly on a panreference. Here, a user can quickly identify lines with missing genes, and which genes belong to the core genome (in orange) or to the variable genome (in blue). Details about the individual gene annotations can be accessed by hovering over the beeswarm-like plot on top of the presence/absence matrix, where genes are represented as non-overlapping marks that can display informative cards of annotation on mouseover.

3 Implementation

Panache is a client-side JavaScript web application built with Vue.js 2 and additional libraries, namely D3.js v5 (enabling linkages between data and SVGs) and Vuex. For easy deployment, we have created a Docker container that can run Panache through nginx but the production files are available for deployment through other means.

Panache takes pre-computed pangenome files as input. The main file is a presence/absence matrix file in a BED-like format. Each line stores information about one pangenomic block (either a gene or a sequence) detailed through multiple columns, starting with linear position data and ending with the presence/absence information within every genome. An optional GFF3 file of annotations on the linear pangenomic coordinates may be loaded in addition to the matrix file. The information of genes' coordinates, exon structure and functional annotations would then be grouped into cards of annotations, available for query on a dedicated track.

As long as the input pangenome file satisfies Panache's criterias, the pangenome construction method is left up to the users. It is possible to use graph-based pangenomes if they are previously linearized with tools such as BioGraph.jl (<https://github.com/nguyetdang/BioGraph.jl>). Example files and details about how to format datasets are provided on the GitHub repository.

4 Discussion

Panache offers an innovative web interface for the linear representation of pre-computed pangenomes, explorable through a web interface, making the exploration and use of pangenomes easier. Rather than focusing on the nucleotide scale or small variations, Panache intends to work at the genome block resolution (e.g. 1–10 kb), facilitating the discovery of presence absence variation (PAV) patterns across a set of sequenced individuals.

As a lightweight application, it can be embedded easily in an independent manner (e.g. i-frame) to complement other interfaces in existing genome information systems—as illustrated with the Banana genome Hub (Droc et al., 2013). It has already been tested with open access datasets (Golicz et al., 2016a; Rijzaani et al., 2021) and proved to be an effective alternative to existing tools by highlighting inherent pangenomic properties. However, as a visualization tool, results are highly dependent on the methods used to create and linearize the pangenomes. Currently, Panache works efficiently with PAV matrices containing dozens up to about three hundred eukaryotic genomes and will be further improved to deal with larger sample size. Further work on pangenome representations, and particularly on structural variations, is also needed to enhance existing representations.

Current plans for Panache include native support of graph files such as GFA (<https://github.com/GFA-spec/GFA-spec>) as an input and faster display technologies like WebGL. Panache is still under active development and new features of interest to the community will be added regularly through its GitHub page.

Acknowledgements

The authors wish to thank Romain Basset, Mel Florance and Alexandre Bousquet for their help on Vue.js and improvements brought to the code, as well as Dr. Eric Ganko and Steve Graham for their useful feedback. The authors acknowledge the ISO 9001 certified IRD itrop HPC (member of the South Green Platform) at IRD Montpellier for providing HPC resources that have contributed to the research results reported within this article. URL: <https://bioinfo.ird.fr/> - <http://www.southgreen.fr>.

Funding

This work was supported by funding from Agropolis Fondation's 'GenomeHarvest' project [ID 1504-006], CIFRE doctoral [2018/1475], Syngenta and the CGIAR Research Program, Roots, Tubers and Bananas.

Conflict of Interest: none declared.

References

- Beyer, W. et al. (2019) Sequence tube maps: making graph genomes intuitive to commuters. *Bioinformatics*, **35**, 5318–5320.
- Computational Pan-Genomics Consortium. (2018) Computational pangenomes: status, promises and challenges. *Brief. Bioinf.*, **19**, 118–135.
- Ding, W. et al. (2018) panX: pan-genome analysis and exploration. *Nucleic Acids Res.*, **46**, e5.
- Droc, G. et al. (2013) The Banana Genome Hub. *Database*, **2013**, bat035.
- Garrison, E. et al. (2018) Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat. Biotechnol.*, **36**, 875–879.
- Golicz, A.A. et al. (2020) Pangenomics comes of age: from bacteria to plant and animal applications. *Trends Genet.*, **36**, 132–145.
- Golicz, A.A. et al. (2016a) The pangenome of an agronomically important crop plant *Brassica oleracea*. *Nat. Commun.*, **7**, 13390.
- Golicz, A.A. et al. (2016b) Towards plant pangenomics. *Plant Biotechnol. J.*, **14**, 1099–1105.
- Hennig, A. et al. (2015) Pan-Tetris: an interactive visualisation for Pan-genomes. *BMC Bioinformatics*, **16**, S3.
- Li, H. et al. (2020) The design and construction of reference pangenome graphs with minigraph. *Genome Biol.*, **21**, 265.
- Maston, G.A. et al. (2006) Transcriptional regulatory elements in the human genome. *Annu. Rev. Genomics Hum. Genet.*, **7**, 29–59.
- Nguyen, N. et al. (2014) Comparative assembly hubs: web-accessible browsers for comparative genomics. *Bioinformatics*, **30**, 3293–3301.
- Nielsen, C. and Wong, B. (2012) Representing the genome. *Nat. Methods*, **9**, 423–423.
- Pedersen, T.L. et al. (2017) PanViz: interactive visualization of the structure of functionally annotated pangenomes. *Bioinformatics*, **33**, 1081–1082.
- Rijzaani, H. et al. (2021) The pangenome of banana highlights differences between genera and genomes. *Plant Genome*, **2021**, e20100.
- Sherman, R.M. et al. (2019) Assembly of a pan-genome from deep sequencing of 910 humans of African descent. *Nat. Genet.*, **51**, 30–35.
- Tranchant-Dubreuil, C. et al. (2019) Plant pangenome: impacts on phenotypes and evolution. *Annu. Plant Rev. Online*, **2**, 453–478.
- Wick, R.R. et al. (2015) Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics*, **31**, 3350–3352.
- Yoghourdian, V. et al. (2020) Scalability of network visualisation from a cognitive load perspective. *IEEE Trans. Vis. Comput. Graph.*, **27**, 1677–1687.
- Yokoyama, T.T. et al. (2019) MoMI-G: modular multi-scale integrated genome graph browser. *BMC Bioinformatics*, **20**, 548.

V. Discussion

A. Outside reach

Panache has already been applied to various datasets: the banana pangenome used within the publication⁴⁵ and a Brassica napus used for the survey (see Panache II.B.2), both built around genes positioned on a panreference [20, 259]; in-house African rice pangenome built from successive blocks of sequence based on a reference [260]. It has also been proposed as an online database to facilitate interrogation and comparison of a graph of wheat cultivar genomes [261], and ongoing discussions could lead to its application on Barley or bacteria. Finally, Panache is considered as one of the components of the South Green Genome Hubs, as implemented in the Banana Genome Hub⁴⁶.

Panache has been presented on multiple occasions during national and international conferences (see posters in Appendix LXXI to Appendix LXXV), including a talk at ISMB/ECCB 2020 BioVis sessions⁴⁷. It also has been cited in 4 publications [124, 261-263] to date since its original publication in December 2021, gaining attention from the pangenome community and illustrating the interest for pangenome visualization tools.

B. Envisioned improvements

While Panache is already a published tool, it can be further improved with the addition of other functionalities, as described in Panache III.B.8.

Among these some are of higher interest, especially the ones requested by users with hands-on experience. For example, the inclusion of the colors associated with biological functions has not been made yet as this information was not present in the first files used. The customization of the categories used with the **core** threshold and of the associated colors, combined with the possibility to export high quality screenshots and filtered datasets would also enable users to make a more personalized use of the tool.

A crucial aspect is the handling of the information on repeated panBlocks and the display of the PAV status of these repeats. This would need implementation of alternative views dedicated to repeats, either as detailed in Panache II.A.2 or with other designs or even within another visualization, either structural or composite (as proposed in SaVanache's chapter).

Finally, performance could still be improved. The static store system (see Panache III.B.7) introduced for the PAV matrix could be extended to other parts of the web application to better handle the biggest files and enable the use of Panache on smaller (therefore more numerous) panBlocks. Faster display technologies like WebGL could bring great improvement but correspond to yet-another-technology to include. Similarly, web workers could enable parallel computation, accelerating the heaviest computing like the Hollow Area Finder or the filter of panBlocks that should be visible for example.

⁴⁵ Available within SouthGreen's Banana Genome Hub: <https://banana-genome-hub.southgreen.fr/content/panache>

⁴⁶ <https://banana-genome-hub.southgreen.fr/content/panache>; as introduced by Droc G, Martin G, Guignon V, Summo M, Sempere G, **Durant E**, Breton C, Cenci A, Baurens FC, Shah T, Aury JM, Ge XJ, Helsop Harrison P, Yahiaoui N, D'Hont A, Rouard M. *The Banana Genome Hub: a community database for genomics in the Musaceae*. Submitted to Horticulture research.

⁴⁷ <https://www.youtube.com/watch?v=IYuMMgQMT9w>

C. Application of the Ten rules

As a first experience in software development, and especially datavis software development, Panache was key in the determination of the rules introduced in the '*Ten rules on Genomic Visualization Tool Development*' chapter.

The need for a new visualization tool (Rule 1) arose from previous benchmarking during my Master degree, where no existing tool was deemed suitable for our plant pangenomes.

The community was involved early on (Rule 2) through multiple discussions within different organisms and a public survey (see Panache II.B.2) in order to identify the needs for this visualization tool.

From these discussions, I identified that a *positioned* tool would be best to work on both pgAtlases and pangenomes, at the scale of panBlocks within a pannable visual representation (Rule 3).

I explored various designs (Rule 4) involving multiple layouts, inspired by genome and outside visualization tools. Inspiration could come from many places, and I also attended generalist visualization conferences⁴⁸ and brought a new eye as I was not used to working with genome browsers.

In order to help the visual analysis (Rule 5), I included multiple '*detail-on-demand*' features, as described in Panache II.A.1 or Panache II.A.3.

The technology used was improved over different iterations (Rule 6), from the use of D3.js to the implementation of Vue.js as a framework, following advice from experienced developers.

To assess performances (both of the visual representation and its implementation), I tried Panache on multiple datasets of various sizes and origins (Rule 7): small and big toy datasets, banana pangenome with 15 genomes, rice pangenome with 83 genomes, *Brassica napus* with 50 genomes. Additional development has already and still can improve Panache's performance with huge files.

To facilitate its deployment (Rule 8), Panache comes with a Docker container and companion scripts that enables the creation of pre-formatted files for instances on web servers.

I maintained a scientific monitoring of pangenome visualization tools which led to the categories described in the State of the Art (Rule 9).

Finally, I advertised my tool on repeated occasions through conferences, lab meetings, publications, or even this PhD dissertation. I also got in touch with the pangenomics community through Twitter and the GitHub issue sections and keep on doing it (Rule 10)!

⁴⁸ Notably OpenVisConf 2018

SaVanache

Panache can be considered as a first step towards an interactive tool for the visualization of pangenomes but has its limitations, being a *positioned* tool (see State of the Art II.B.3) that is neither dedicated to pgAtlases nor pangenomes only. We wanted to extend its capacities to a broader scale that would enable broader analyses of pangenomes as well as the exploration of the **Structural Variations (SVs)** within. Our aim was to build a *composite* tool, with connected visual representations and views for various scales, one of them being the panBlock scale proposed by Panache, which focuses on PAV and supports gene annotations. A user would then be able to navigate between them, following Shneiderman’s “*Overview first, zoom and filter, then details-on-demand*” approach [264].

SaVanache—name given to this project that enables the visualization of SVs to extend Panache—has been designed as such a composite tool, embedded within an integrative interface thought as a **Single Page Application (SPA)**. Divided between four views (*Overall diversity, Structural variations, Presence Absence, Haplotypes*), it also proposes a novel visual paradigm for the representation of SVs within a pangenome.

I. Structural Variations and pangenomes

PgAtlases have been built around the notions of PAVs and CNVs (see State of the Art I.A), but more complex SVs exist, on various scales superior to 50bp.

A. SV nomenclature

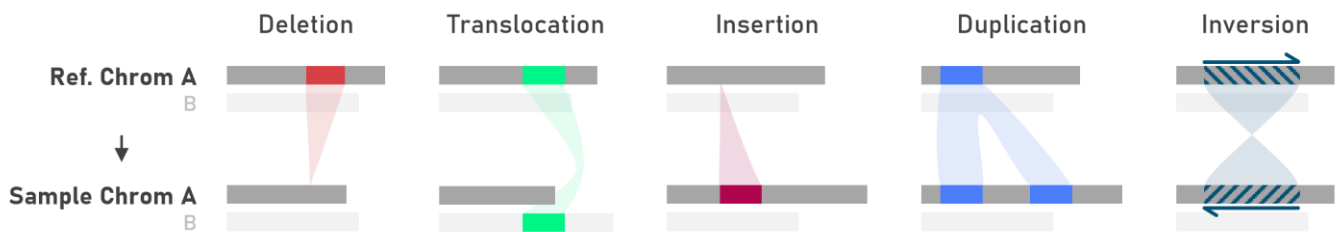


Figure 30: Structural Variations are labeled based on the differences between a genome and a reference; Deletions are an apparent lack of genomic material compared with the reference. Translocations are when a genomic segment is found *elsewhere* in the compared genome. Insertions are reciprocal to deletions: segments present in the compared genome are absent from the reference. Duplications happen when a segment has more occurrences in the compared genome than in the reference. Inversions are similar fragments that have opposite strand directions between the reference and compared genome. These simplest SV patterns can be combined, creating nested Structural Variations, harder to characterize and detect.

SV are observed genomic rearrangements between genomes, and can be categorized into multiple major categories, as illustrated in Figure 30. These categories are depending on the relationship between a subject and a query. They are therefore directed, often with the more recently sequenced genomes being compared to a former one.

An observed translocation does not mean that one genome inherited a fragment from the other and that it has been positioned in a different location. Rather, it describes variations between the two without assuming heredity. An insertion observed when comparing genome A to genome B would therefore be qualified as a deletion when comparing genome B to genome A. The SV names do not reflect an evolution but a static observation of a certain configuration instead.

B. SV visual representations

SVs, by definition, break the linearity of genome sequences, and are therefore difficult to visualize in classic genome browsers, especially when both the variant and reference genomes should be displayed at the same time. As they differ in size and position, showing all types of SVs can be

challenging [265]. Multiple visual representations and visualization tools have been proposed through the years.

Common visual representations (as illustrated in Figure 31) include:

- Linear layout with arcs linking the SVs' breakpoints (*i.e.* the boundaries delimiting every SV) as seen in the newest version of pantograph [217] (see State of the Art II.B.4.k)
- Circular layouts, popularized by Circos [204], highlight links between other chromosomes and can provide an overview
- View of chromosomes with segment colored by origin (either other chromosomes or genomes), as proposed by CINTENY [228]
- Comparative dotplots of the variant and reference sequence, as featured in CORGi [266]
- Graph views comparing the paths taken by two or more genomes, as in Sequence Tube Map [194]
- ... and others as categorized by Nielsen & Wong [265] and later Yokoyama & Kasahara [267]

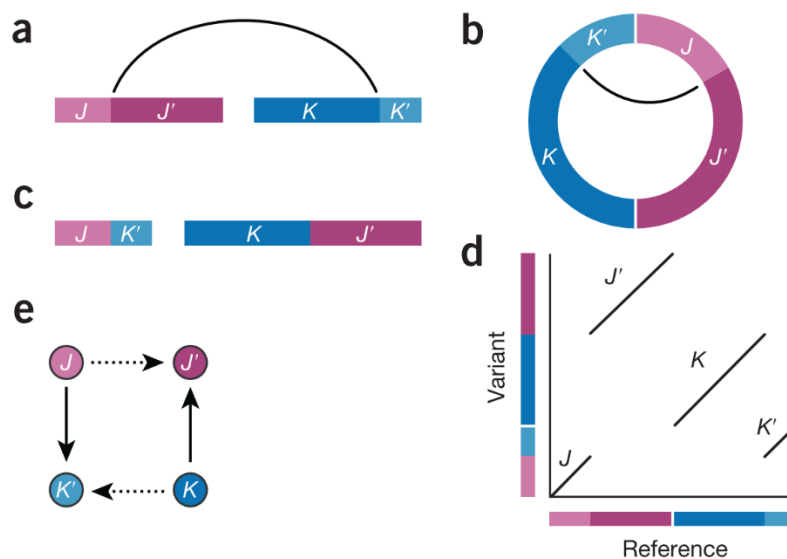


Figure 31: Translocations can be depicted with various layouts; a) arcs between linear sequences. b) arcs within circular layouts. c) linear sequences colored by origin. d) scatterplot of sequence alignments. e) graphs of sequences. Figure from Nielsen & Wong [265].

All these target various scales, from the nucleotide or gene level to the chromosome or even inter-chromosome levels.

Moreover, many tools used for the visualization of SVs focus on conserved blocks of genomic sequence. For example, synteny viewers (e.g., SimpleSynteny [268], Synima [269], SynVisio [270]) display blocks and their relative positions between and within genomes; a user can then infer the positions of SVs by comparing the order (and orientation) of conserved blocks. As the highlight is on conserved blocks, SVs are rarely depicted specifically encoded but can be imagined wherever there are differences. Inversions are an exception as they are often represented as twisted ribbons, creating an hourglass-like shape which can also be color encoded to further reinforce their difference with normal conservation ribbons.

Tools that encode SVs visually, as entities on their own, are rarest. As examples, I found the panGraphViewer [189] (discussed in State of the Art II.B.4.j), and plotsr [193].

As described in State of the Art II.B.4.i, **plotsr** produces static plots of pairwise comparisons between successive genomes assemblies represented as lines. It encodes three types of SVs with different colors: inversions (■), translocations (■), and duplications (■), as shown in Figure 32.

Normal synteny is represented by greyed out (■) ribbons between the horizontal lines representing the assemblies, and additional tracks on top can depict, on a linear axis, densities of genes and SNPs for example.

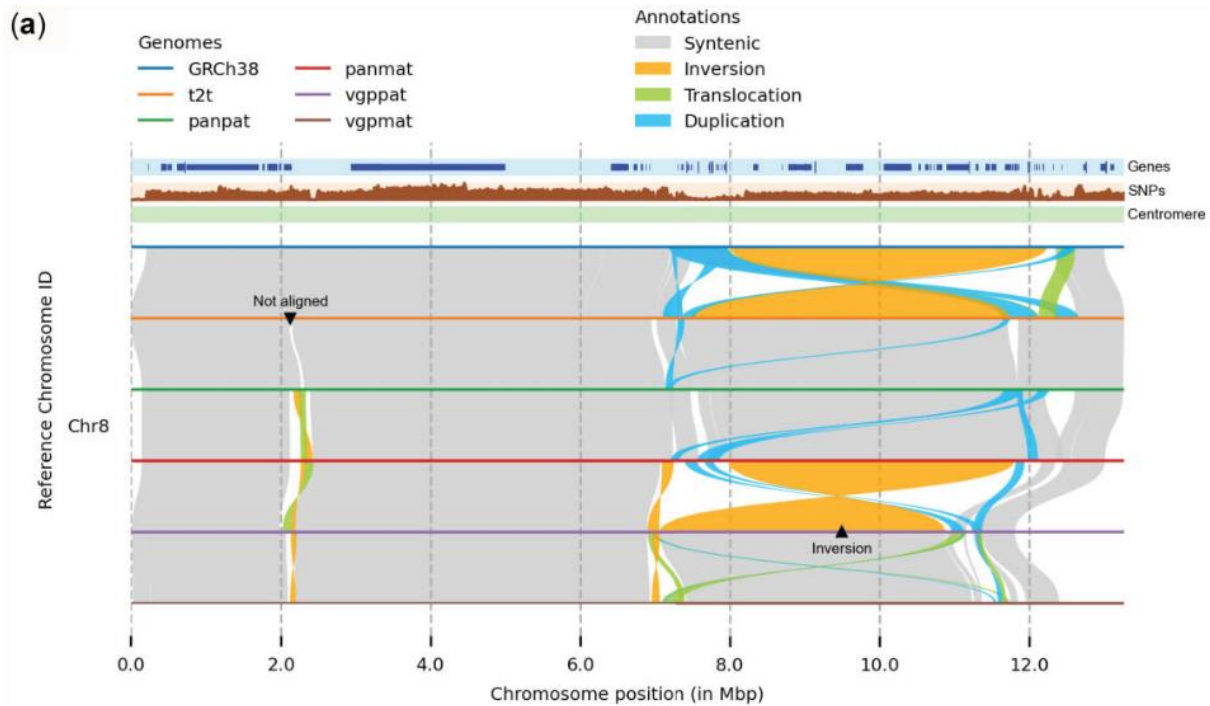


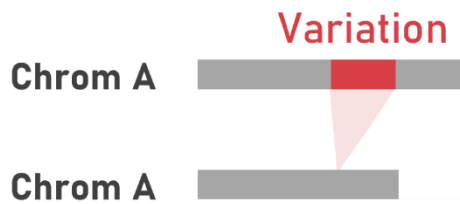
Figure 32: plotsr represents different types of SVs as colored ribbons between lines of genome assemblies; Each assembly is compared to its direct neighbors. Tracks of Genes, SNPs and Centromere position are based on the coordinate system used by the assembly on top. Figure from Goel & Schneeberger [193].

A limitation of existing visual representations of SVs is that they are mostly built for one-versus-one pairwise comparison and could not be properly used for one-versus-many visual representations. A possible technique is to draw multiple pairwise comparisons on screen, with each time the same genome on one end, but this makes an inefficient use of space, with redundant information and generally a lot of scrolling involved. In the case of plotsr, multiple pairwise comparisons are displayed, but the user would have to create another plot to compare an assembly with any of those that are not currently its direct neighbors.

C. SV within genome graphs

Within genome graphs the notion of variant and reference genomes are blurred: insertions, deletions, translocations... all create new connections between sequences within the graph. Every branch (and loop depending on the type of graph) correspond to an SV, but the usual nomenclature cannot apply properly. Since everything is present, there are no insertions or deletions within a graph, only segments that are traversed by some paths and not by others. The usual SV names apply when one genome is compared with another or more, and “visualizing SVs” within a genome graph can get quickly abstract when the linear nature of DNA sequences is completely left out, as illustrated in Figure 33.

Linear



Graph

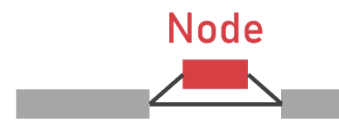


Figure 33: An SV between two genomes will create alternative connections between nodes in a sequence graph; if the variation is considered as a comparison of the bottom genome (variant) versus the top one (reference), it will be characterized as a deletion. Reciprocally, if the reference is at the bottom and the variant at the top, it will then be an insertion. Both scenarios have a unique form as a graph, where there is no deletion and no insertion but only nodes that are connected successively, or bypassed. It cannot be further characterized without the information of at least two paths to compare.

Visualizing graphs can be used to show the arrangements of sequences between genomes, but usually a user would have to follow the paths and compare them mentally to identify SVs.

Tools like the Sequence Tube Maps [194] display a graph directly, showing the actual node successions, while others like SplitThreader [271] use graphs in the background to display linear comparisons of sequences, highlighting the links between breakpoints within the genomes (i.e. the edges connecting nodes from different genomes).

To my knowledge, only the PanGraphViewer [189] tried to visually represent SV annotations within graphs, by distinguishing backbone nodes from variant nodes: one path is being considered as the main one, and every variation from it can be annotated. Nodes are then shaped depending on their status—backbone or variation(s) (as seen in Appendix XXV).

II. SaVanache's design

A. Multiple scales

The main goal of SaVanache is to encapsulate multiple visual representations of pangenomes into one comprehensive and easy-to-use visualization tool. All these representations are as many visualization scales of the pangenome, connected through a sort of semantic zoom: the visual encodings evolve with the scale of interest for the user. As genomic variations are of various sizes with each its influence, from mononucleotide SNPs to chromosomal rearrangements, representing everything at once would be counterproductive for analysis purposes. The goal with these different views is to access various levels of details dynamically, without overwhelming the users with an unnecessary abundance of information.

This idea of a composite visualization tool with a common user interface was first drafted in December 2019. Together with Joffrey Gallais (a UI/UX designer hired by Syngenta) we worked on the different layers of such a tool, resulting in example wireframes and screenshots of the different views, as shown in Appendix LXXVI to Appendix LXXX. I was in charge of imagining the visual representations and filter options for each view, and explaining the concepts to Joffrey, who then laid them out on static mock-ups with pre-determined interactive area that redirected to the next one on click.

This draft version evolved later, with modified visual encodings and a renewed interface, with the goal of being usable by outside research organisms and not just Syngenta. I focused on the

redesign of the first views and their integration within a common interface. We identified four main view scales, detailed as follows.

1. Overall diversity

The first scale would aim at showing an overview of the diversity within a group of assemblies. In-house visualizations showed that diversity as a scatterplot of **Principal Component Analyses (PCA)** built from genomic fragments and their shared occurrences.

We wished to keep this overview to explore an increasing volume of assemblies per species. It would be used to identify the most similar assemblies, and to easily see which pangenome was built with which assemblies. We envision that different pangenomes, or pangenome subsets, might be available: global pangenome for exhaustive comparisons, a pangenome dedicated to domesticated or wild species, pangenomes per geographic origin, major phenotype... Each pangenome subset could be built to focus analyses on certain tasks, for example research with diversity characterization of a species or breeding with the analysis of the genetic diversity of a given pool of individuals. This view would then be useful to quickly identify both assemblies and pangenomes of interest for further analysis.

From this view, a user could select assemblies to focus on and choose the appropriate pangenome for their studies, before proceeding to the rest of the analysis.

2. Structural variations

The second view would focus on large structural rearrangements between genomes, providing an overview of SV breakpoints and connections. This view would provide a finer level of detail on what diversity can be present between genomes and can help users to identify regions mostly free of rearrangements or with multiple variations instead.

Stable regions can be useful as targets for gene introgression with genome editing techniques, while variable regions can create diversity, maybe explaining phenotype differences between subgroups. From this view, users could target a region of interest, see SVs that might happen, and go deeper into details with the next view.

As visual representations of SVs between three genomes or more are rare, I focused on this view and proposed novel visual representations of SVs within a pangenome context as detailed in SaVanache II.D and SaVanache II.E.

3. Presence Absence

A third level dedicated to genes and/or panBlocks within a region would enable the exploration of PAVs and CNVs patterns between genomes, for comparison purposes. This level of visualization corresponds to the one used by Panache, which could be integrated as is.

Selecting a gene or panBlock could then redirect to the final scale, for the exploration of haplotypes.

4. Haplotypes

Finally, the most precise scale would focus on variations at a nucleotide level. It would show variations or groups of variations and how they are distributed between genomes. At this scale SNPs and smallest InDels would appear.

There are already tools enabling the visual comparisons of nucleotide sequence, like MSA browsers or graphs at a nucleotide resolution such as Sequence Tube Maps [194]. For this reason,

I focused on the (re)design of the first two views, in the interest of time and assuming that we could integrate or adapt an already existing tool for the fourth view⁴⁹.

B. SaVanache's UI

The four views and visualization scales must be integrated into a common UI, enabling the navigation between these different scales. Navigation between the scales would correspond to zooming in or out, to explore the genomic diversity at different levels.

I imagined this tool as a **Single Page Application (SPA)**, that is to say a web application built from one document with dynamic modifications of the displayed content without the need to download new web pages. Each visualization scale has dedicated visual representations, which fit into a full screen display with no need to scroll down to access hidden parts.

As visible in Appendix LXXXI, the UI was imagined with a navigation pane on top, indicating which view is currently on display and enabling navigation between already visited views (one would first have to unlock views through the rest of the interface, to store display parameters such as the region targeted). A togglable menu on the left would offer multiple general options (filters, link to documentation, screenshot or subdata exports... and legends for the visual representations on display)

C. View 1 – Overall diversity

1. From existing PCA representation...

I redesigned the Overall Diversity view from in-house visualization, which showed assemblies as dots within a scatterplot, positioned depending on their value on PCA axes. Color could encode metadata (for example groups of phenotypes), and tooltips would appear on hovering with information with the assembly encoded by the hovered dot.

We wanted to extend this representation, by displaying pangenomes and how the assemblies were distributed among them, and by using public algorithms for the creation of the scatterplot, in order to have a tool that could be used by other organisations rather than Syngenta only.

As for Panache, SaVanache would use pre-computed files, with limited computation within the visualization tool. This means that users could use their favorite dimension reduction algorithm (PCA [273, 274], t-SNE [275], UMAP [276]...) for this view as long as the result can be displayed within a two-dimensional scatterplot.

Moreover, we wanted to include information on the available pangenomes and their relations with existing assemblies. The original idea with my supervisors was to encode pangenome as ellipses around group of dots: all dots within would correspond to assemblies contained by the pangenome, as opposed to outside dots that would correspond to assemblies not included. This idea was abandoned during the redesign phase as too many pangenomes would create visual clutter on the scatterplot, and special cases would make the ellipse hard to draw and read (for example if an assembly displayed in the middle of a group does not belong to a pangenome built with these other assemblies).

⁴⁹ 272. van den Brandt, A., et al. *Visual Exploration of Genetic Sequence Variants in Pangenomes*. in *EuroVis 2022*. 2022. Roma: The Eurographics Association. is a most recent example (June 30 2022!), which won the Best Poster award at EuroVis 2022 Rome

2. ...to SaVanache's redesign

I propose a division of the *overall diversity* view into five connected main parts, showed below on Figure 34.

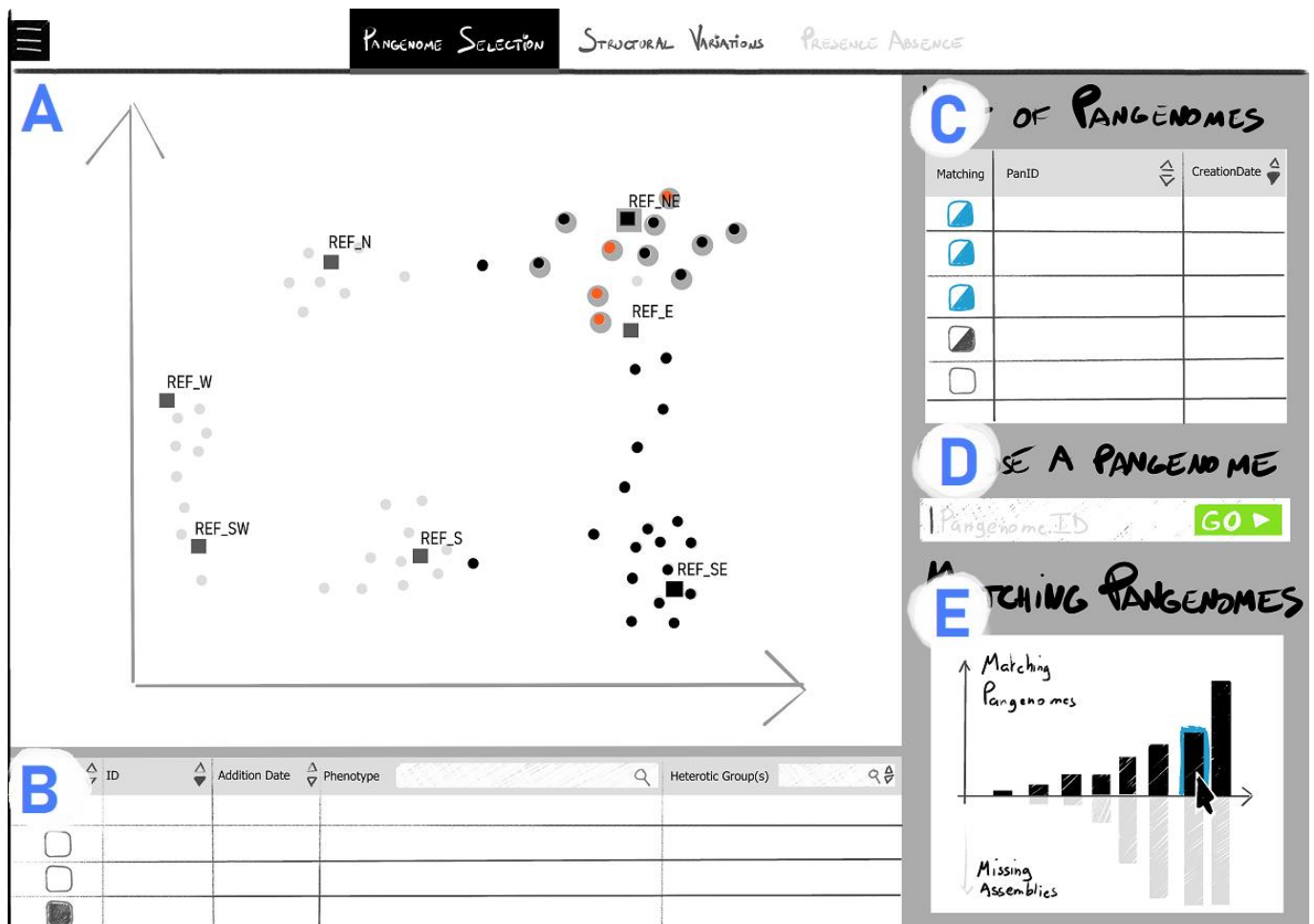


Figure 34: The overall diversity view of SaVanache displays an interactive scatterplot of genome assemblies; A) The scatterplot shows genome assemblies positioned with a two-dimensional space thanks to prior dimension reduction techniques. Canon reference genomes are represented as square shapes with their names always appearing, other genome assemblies are represented as (colored) dots. This scatterplot would support interactivity through tooltip on hovering or lasso selection of assemblies for example. B) Table of the genome assemblies showed on the scatterplot, displaying their related metadata and selection status in a user-defined order. C) List of the pangenomes available with additional information and a glyph acting as a visual cue to the matching score of pangenomes compared with a selection of assemblies. D) Input for the chosen pangenome, enabling redirection to the next view. E) Bar chart of the number of pangenomes matching a selection of assemblies depending on the number of assemblies not captured.

This view is composed of:

- an interactive scatterplot of the genome assemblies, with tooltip displayed on hovering, and multiple color options
- an ordered table of these assemblies, for better parsing and filtering
- a table with the available pangenomes, showing the datasets available for further analysis
- an input button storing the chosen pangenome for the rest of the exploration within SaVanache
- an interactive bar chart of how these pangenomes match a selection of assemblies

For an overview of the actions possible within this view, I propose the following *user flow*⁵⁰. Interested in assemblies with an agronomic trait of interest, a user wants to identify which assemblies have this trait, select them, then choose a pangenome with these selected assemblies for further analyses into subsequent views.

a. Identification of interesting assemblies

To find assemblies of interest, a user could hover the scatterplot. On hovering, dots will be colored depending on specific categorical metadata either pre-stored within the original file or manually added with an optional second file. The dots of all assemblies sharing the same value would also be colored to show related dots. If a hovered dot has more than one value for the metadata used for coloring, then all dots of assemblies sharing at least one of the values will be colored, with the color attributed to their primary value.

For example, imagine that the metadata used for coloring is about geographic origin, based on 5 continents associated with one color each: ■ Africa (**red**), ■ America (**orange**), ■ Asia (**magenta**), ■ Europe (**blue**), ■ Oceania (**green**). Priority is set to the first continent name provided per assembly; we assume that this order is alphabetical in this example.

If an assembly registered as belonging to both “Africa” and “America” is hovered, then all dots of assemblies with “Africa” would be colored in ■ **red** (including the hovered dots), and within the remaining dots those labelled with “America” would be colored in ■ **orange**. Dots without any of these two labels would not be colored. All dots with “Africa” and “America” would be colored in ■ **red**, as “Africa” has the priority. Examples of this hovering system are showed on Figure 35 below.

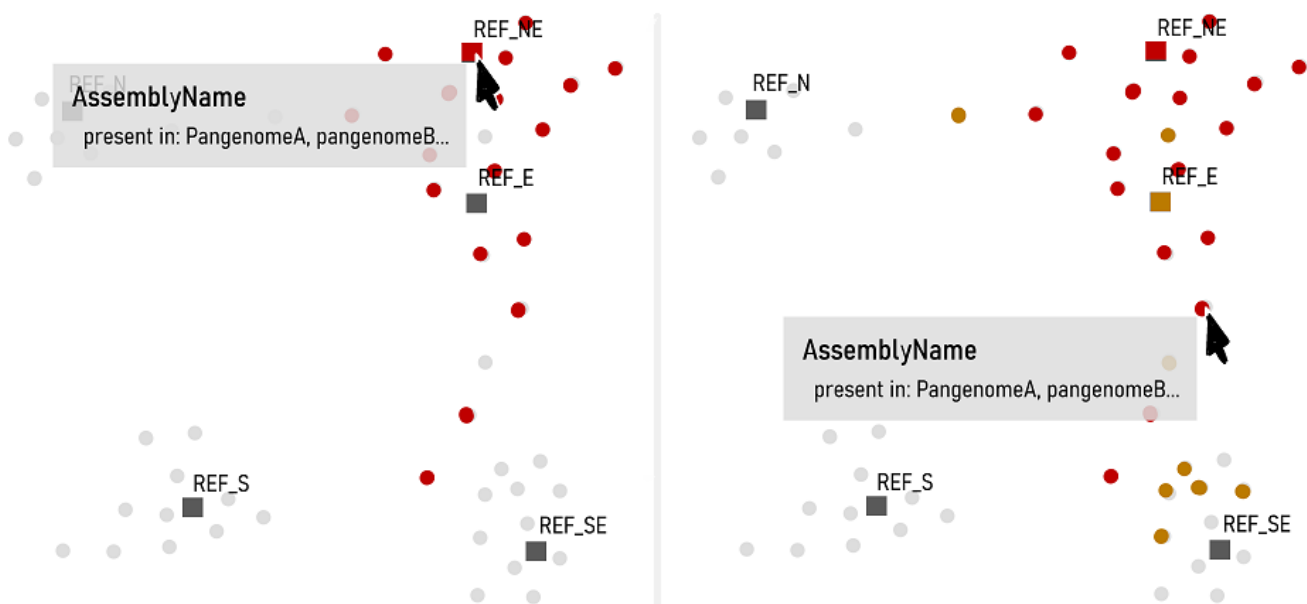


Figure 35: On hovering, dots of assemblies with similar metadata are colored accordingly; Left: the hovered assembly has only one categorical value (red) for the metadata chosen for the coloring system. All assemblies with the same value are colored in red too. Right: The hovered assembly has two categorical values (red and orange). Assembly with the value for red are colored in red, those with the value for orange are colored in orange if they have not been colored in red yet (either because they do not have the right value, or because its priority is lower than that of orange). Moreover, tooltips with details on the hovered assembly could appear on screen.

Besides, a user could use the table below the scatterplot to filter and order assemblies depending on various properties, including the metadata used for the coloring system detailed above.

⁵⁰ A *user flow* is a series of steps taken by a user within an interface to reach a given goal.

b. Selection of identified assemblies

To select assemblies that should appear in the pangenome used within the rest of the analysis, a user could click on a dot, or do a lasso selection of a group of dots directly within the scatterplot. Alternatively, they could select the assemblies within the assembly table, either one by one or by selecting a group having a given value.

On selection, the outline of the corresponding dots turns dark grey (■), to distinguish them from unselected dots.

c. Choice of a pangenome including the selected assemblies

After having selected the assemblies that they are interested in, the user could check which pangenome is the best fit to their needs. The list of pangenomes would be ordered, with the best matching pangenomes on top: pangenomes built with all the selected assemblies first, then pangenomes where some assemblies are not included, and so on. The percentage of compliance (i.e. how many assemblies a pangenome has out of the total number of selected assemblies) would be represented by a dedicated glyph, as illustrated below in Figure 36.

Matching	PanID	CreationDate
■		
◼		
◻		
◼		
◻		
□		

Figure 36: The pangenomes are ordered depending on the percentage of selected assemblies they possess; Glyphs indicating if they have all, most, some, or none of the selected assemblies can be used as visual clues to help a user easily determine the pangenome that would be the best fit for their analysis. Here only three states are showed: full match, partial match, and no match. The pangenome names or ID would be extracted from a pre-computed file.

Moreover, by hovering pangenomes within the pangenome list, dots within the scatterplot could be colored depending on A) their presence status within the hovered pangenome, B) their selection status (selected, not selected), C) their presence status within the previous pangenome if one was chosen already, as illustrated in Figure 37.

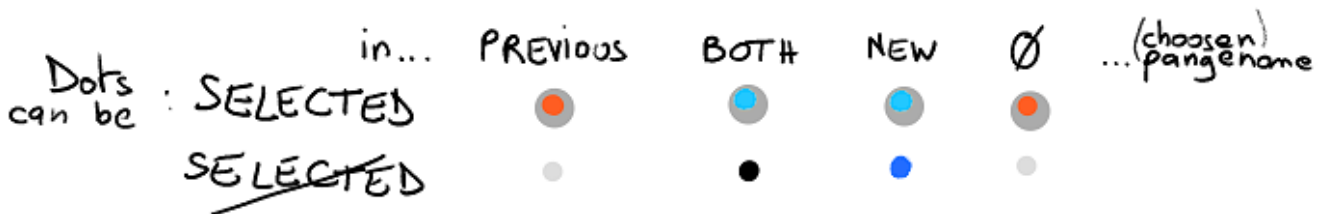


Figure 37: When hovering a pangenome, dots are colored depending on their presence status within; We can identify different categories of assemblies: the *missing* assemblies are selected assemblies absent from the hovered pangenome; the *saved* assemblies are selected assemblies that are present instead and will be included in the analysis if that pangenome

is chosen; then the unselected assemblies can be *trivial*, *constant*, or *new*, depending on their status in both the already chosen pang genome and the hovered one.

In my design, selected assemblies are represented with a dark grey (■) outline. Selected assemblies that are not in the hovered pang genome—the '*missing*' assemblies—are showed in orange (■). Selected assemblies that are in the hovered pang genome—they will be '*saved*' if that pang genome is chosen—are showed in light blue (■). Unselected assemblies that are absent from the hovered pang genome—those of '*trivial*' importance—are showed in light grey (■). Unselected assemblies that are present in the hovered pang genome are showed in black (■) if they were already in a previously chosen pang genome—they are '*constant*' between pang genomes that will be included in further analyses—or in dark blue (■) if they were not—they are '*new*' assemblies going to be included. Once a pang genome has been chosen (by clicking on it within the pang genome list, or by choosing its name within the dedicated input button), assemblies that are present are colored in black (■), and the absent are in light grey (■) or stay in orange (■) if they are selected. This way, the *missing* assemblies will always be highlighted in orange (■), encouraging the user to choose another pang genome that would contain them.

Depending on the succession of chosen and hovered pang genome, a dot can therefore have different colors, as depicted in Figure 38. Hovering a pang genome will highlight the change that would be brought to the subsequent analyses if it were to be chosen. *Missing* assemblies are always going to be easily visible for the user, even when no pang genome is hovered.

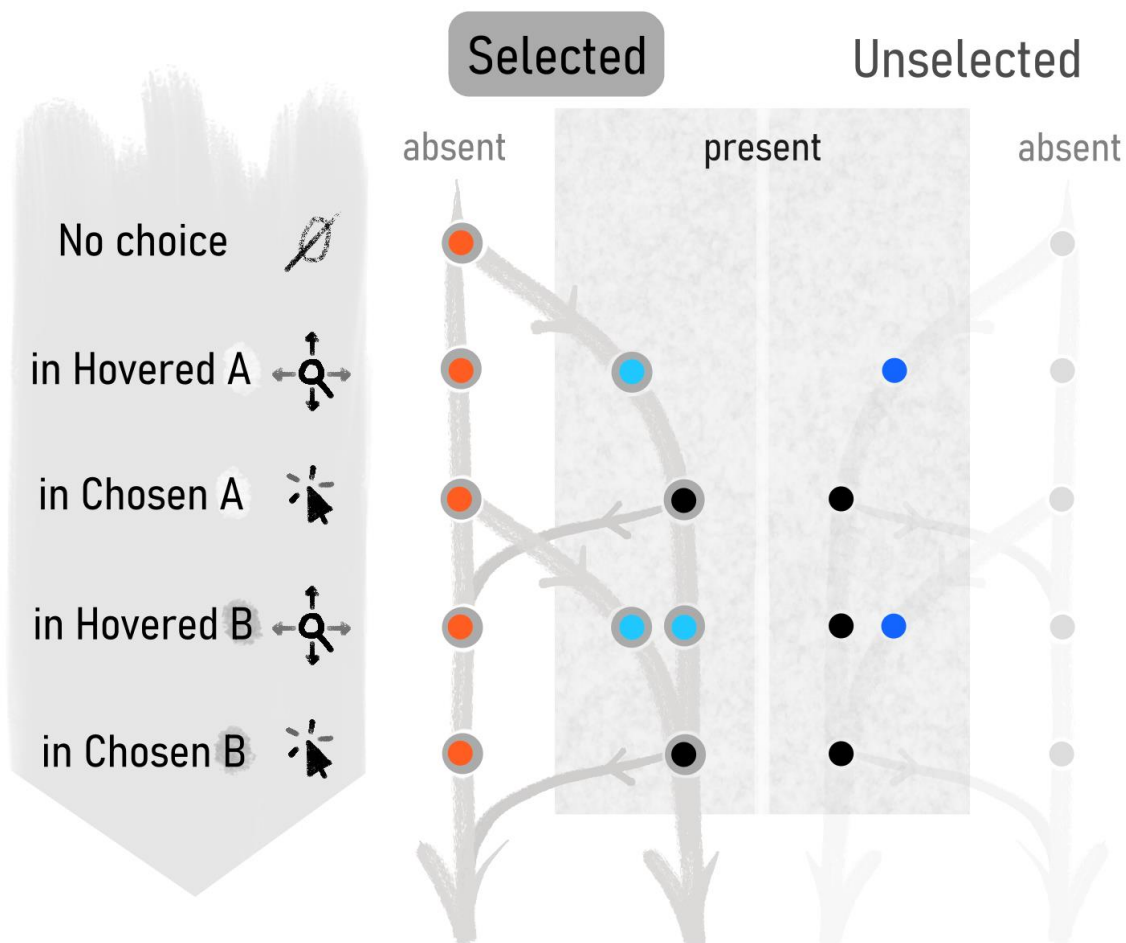


Figure 38: The coloring pattern of a dot for the pang genome-hovering interaction evolves depending on selection and presence status and the chosen pang genome; Each line corresponds to a successive status of the scatterplot. At first no genome is selected, only the *missing* assemblies are highlighted. Hovering a pang genome in the pang genome list highlights the changes that will be made to the assemblies included into subsequent analyses, as well as the status of selected

assemblies. Once a pangenome is selected, the *missing* assemblies are still highlighted and assemblies that are included are in black, to make them visually pop-out compared with the *trivial* assemblies. If a pangenome name is hovered after another pangenome has been chosen, the colors highlight the *missing*, *saved*, *constant* and *new* assemblies. This way, it shows how the set of assemblies included into the subsequent analyses would evolve if that new pangenome were to be chosen instead.

As described, this visual representation uses the visual channel of color on different interactions: hovering a dot on the scatterplot or hovering a pangenome on the pangenome list. Both color encodings should not collide with each other, as the color displayed when hovering a dot is a temporary encoding. If assemblies have been selected, or if a pangenome have been chosen, the dot-hovering interaction would take over the pangenome-hovering/choice interaction and impose its coloring to the dots: dots sharing metadata will be colored and the others will stay in gray, no matter their selection status. Once the dot-hovering interaction is not triggered anymore, the colors return to normal, following their color pattern prior to the dot-hovering event.

The full proposed user flow is presented in Appendix LXXXII to Appendix XCII as successive mock-up screenshots. Note that the color patterns there correspond to an older version of the one presented in this section (*constant* coloring was applied instead of *saved* for selected assemblies already present in the chosen pangenome; *'lost'* assemblies—unselected and present from the chosen pangenome but not the hovered one—were displayed as white dots with dark strokes; *missing* assemblies were not colored when no pangenome was chosen). This coloring pattern was replaced because of its additional complexity.

d. Matching pangenome barchart

The view could also display a barchart showing the number of pangenomes for specific matching intervals, for example the number of pangenomes having 100% of the selected assemblies, 98% to 100% excluded, 95% to 98%, 90% to 95%, etc. A user could click on a bar from that barchart, and all pangenomes corresponding to the selected bar would be highlighted within the pangenome list. This feature has not been developed further as we do not currently have enough pangenomes to make a good use of it.

D. View 2.1 – Visualization of SVs intra pangenome

The second view of the *composite* visualization tool must be focused on SVs. It should give an overview of the major genomic rearrangements happening throughout the pangenome, enabling the identification of stable and highly variable regions. For this view, I first worked on visual representations that could show alternative positions of genomic fragments within a pangenome, providing information on the pangenome itself rather than the individual genomes.

1. Division into panchromosomes

To provide a first visual representation of the level of stability within a pangenome I proposed to use the concept of panchromosome.

Within a sequence graph, nodes that have multiple occurrences can be stored only once, with multiple connections linking to all possible positions found within genomes. In DAGs (see State of the Art I.F.3.c), such node would be duplicated to avoid the creation of loops within the graph. I base my conception of panchromosome on such DAGs. A panchromosome would therefore be an unfolded subgraph of a whole pangenome graph containing all nodes observed within the same chromosome of various genomes, with nodes present in multiple copies if they appear in different order or positions between genomes, as illustrated in Figure 39. Each genome could therefore be retrieved by following a linear path through this panchromosome, without any loop or redirection to nodes placed before within the graph.

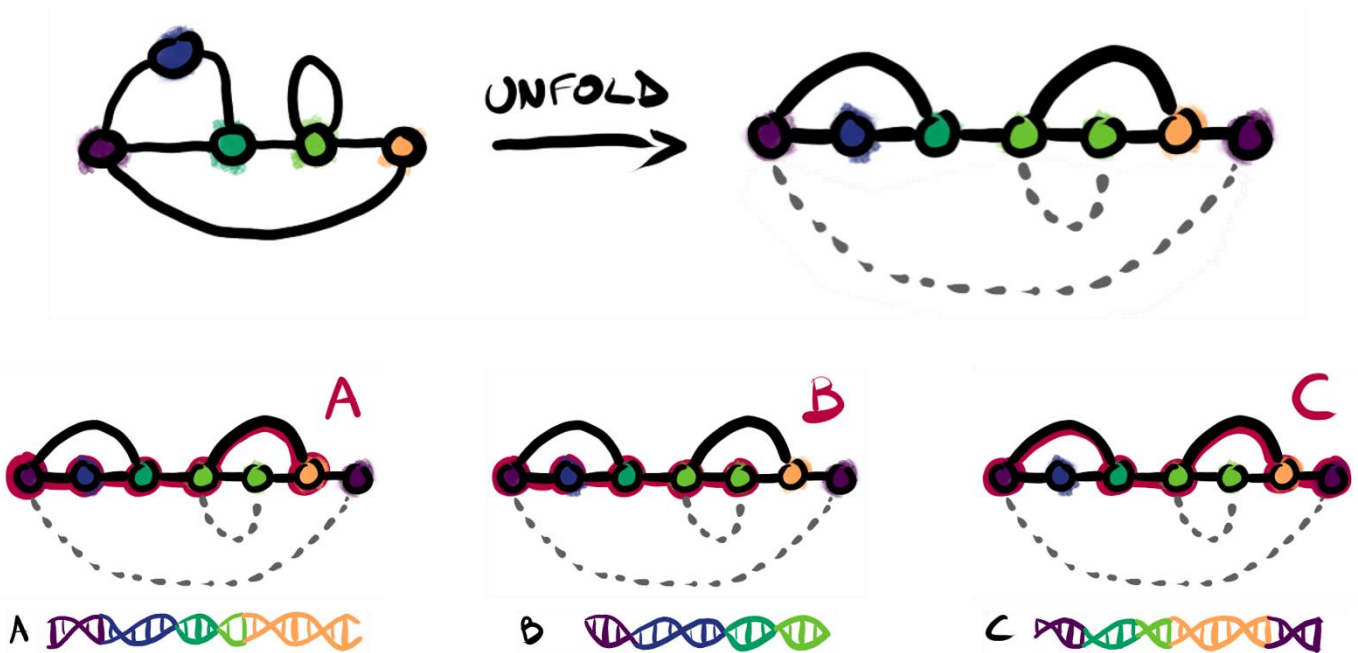


Figure 39: Each observed chromosome sequence can be described as a linear succession of nodes within a panchromosome; On top, a simple, dense sequence graph can be unfolded and turned into a DAG to order nodes linearly. Nodes with copies within the DAG are schematized with dotted links connecting the different cooccurrences. At the bottom, the three genomes (A, B, and C) used to create the panchromosome graph are represented. Each can be displayed as a linear path through the 'DAG-ified' panchromosome, with successions jumping over absent nodes. For example, the genome C starts with the first occurrence of the violet (■) node, jumps over the blue (■) node since it is not present in its sequence, connects to the dark green (■) node, then to the first occurrence of the light green (■) node. It skips the second occurrence of the light green (■) node, which is found only in the genome B, and continues with the beige (■) node, and finally with the second occurrence of the violet (■) node.

From these panchromosomes, an overall profile of variability could be drawn in a shape similar to a violin plot, with the wider portions representing highly variable segments from the panchromosome, as schematized in Figure 40.



Figure 40: Representing panchromosome as violin plots can be thought of as putting a graph in a sock to see its silhouette

The measure of variability along a panchromosome could be based on different properties, I propose to measure it by counting the number of connections (both successions links and cooccurrences of nodes) existing between two successive nodes of the panchromosome, as illustrated in Figure 41. This way, regions with many rearrangements, resulting in numerous jumps through the panchromosome, would appear wider in the violin visual representation, and cooccurrences would also be counted as contributing to this variability.

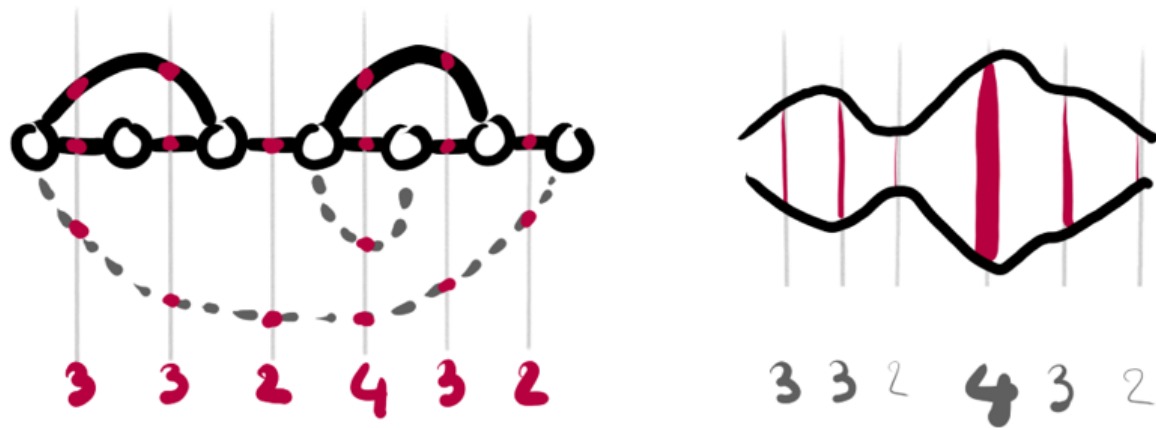


Figure 41: The shape of violin plots for panchromosome can be computed from the connections spanning successive nodes; Left: the variability values are calculated from both the number of edges and connections between cooccurrences existing in the interval between two successive nodes. Right: the width of the resulting violin plot directly depends on these measures.

This measure of the variability would need to be refined, as it currently would add too much weight to distant cooccurrences. For example, a panchromosome with multiple cooccurrences of a node on both ends would have many cooccurrence connections spanning the whole panchromosome, artificially increasing the measure of variability in the middle of the panchromosome. A hybrid approach that would count the succession links between successive nodes, the number of cooccurrences—both intra- and inter- panchromosomes—of each node, and that would take into account the possible inversions could bring better results.

2. PanCircos

To represent SVs throughout a whole pangenome, my first idea was to combine a circular Circos-like layout [204] (see Appendix XCIII) combined with the violin-like visual representation of panchromosome described above. All panchromosomes would have been displayed, with links drawn between cooccurrences. For less visual clutter inter-panchromosome cooccurrences could be drawn inward, taking the available space at the center of the circular layout. Intra-chromosome cooccurrences could be drawn outward in order to not overlap with the inter-chromosome cooccurrences.

For better analysis value, interactivity could be used to highlight cooccurrences from a specific panchromosome or subregion, or to filter out cooccurrences based on their size. A user could therefore choose to display only the smallest cooccurrences for example, hiding all the visual links of the wider cooccurrences, as illustrated in Figure 42.

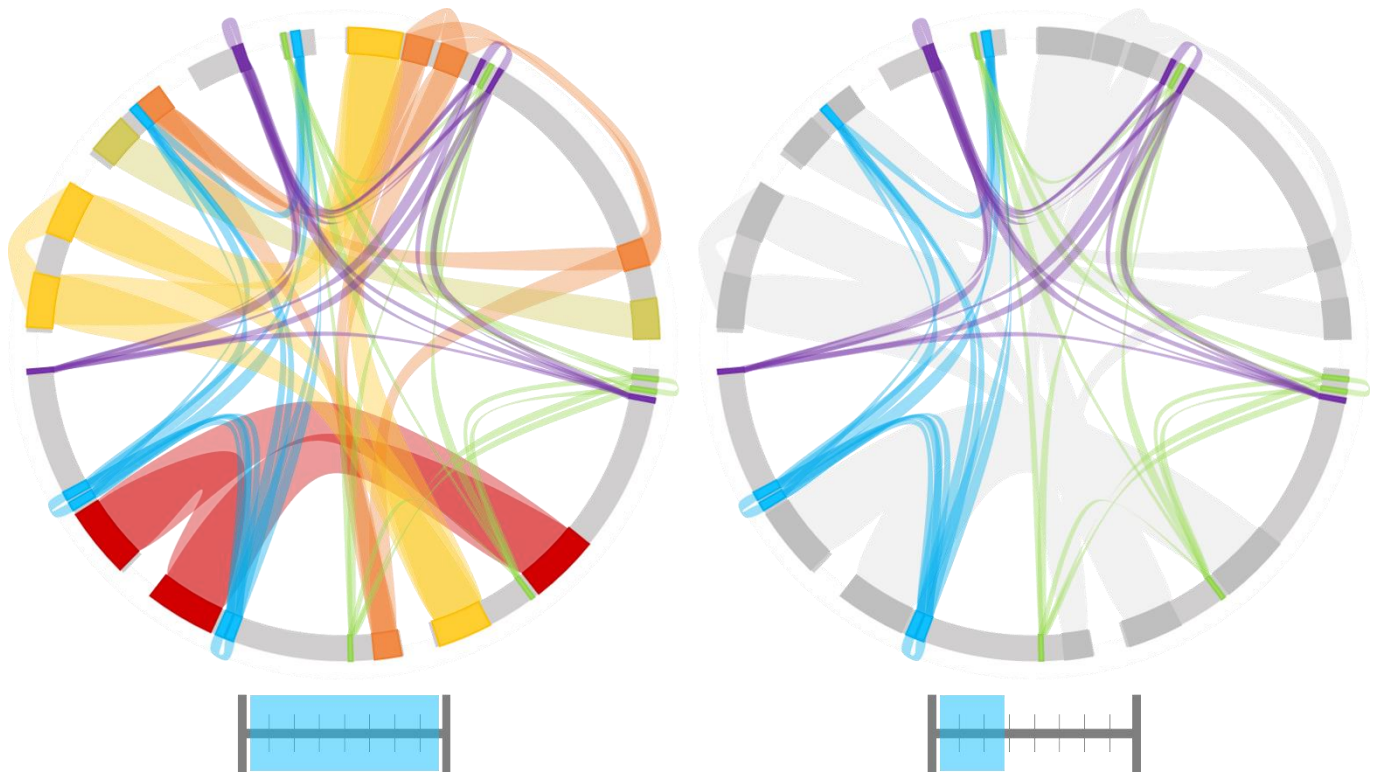


Figure 42: Interactivity on PanCircos would enable dynamic filtering of SVs depending on their sizes; Two states of a same pangenome displayed as a PanCircos, with the selected size range below. The arcs represent panchromosomes, and the links represent chunks of DNA sequence similar between two positions, i.e. SVs corresponding to duplications and translocations. Inner links encode inter-panchromosome relationships while outer links encode intra-panchromosome relationships. Left: all SVs are showed when no size range is selected. Right: Selecting only the small SVs greys out the other on the visual representation.

3. One-versus-all tabular representation of a PanCircos

Two of my supervisors disliked the idea of Circos-like representation, as they can be fancy but hard to read and poorly informative in their experience. I personally think that they are good visual representation for providing on overview of relationships between many elements. Their lack of informational value could be compensated with interactions, though I do agree that they are limited in scope as static visual representations. I therefore tried a more tabular display of the SVs between panchromosomes.

The visual representation would be divided in two parts: the main panchromosome to compare with others at the bottom, and a table with all the panchromosomes (including the main one) on top.

a. The main panchromosome

The main panchromosome would be the one compared with every other. Displayed as a violin plot of the variability along its axis (as detailed in SaVanache II.D.1), a user could select a subregion of that panchromosome by moving two handles acting as borders for that region. All space within these borders would be colored with a linear multi-hued colormap⁵¹, encoding position, as illustrated in Figure 43. This color map would be dynamically updated when the handles are moved, so that the borders are always colored the same.

⁵¹ I used the same one as the colormap for the position track of Panache

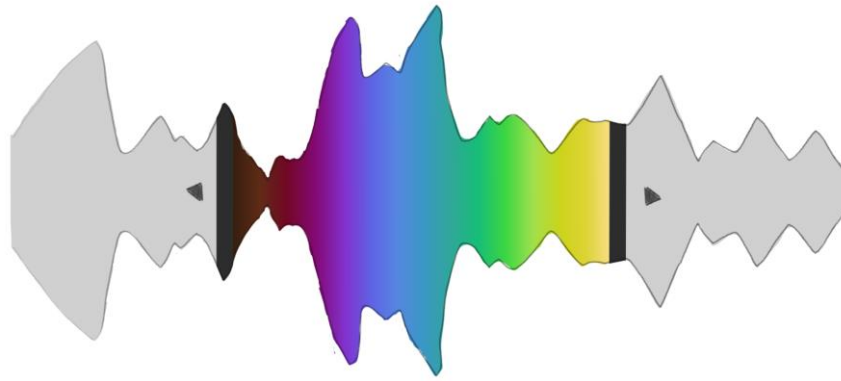


Figure 43: The selected region of the main panchromosome is delimited by two dynamic handles that can be moved left and right; In this violin-like representation, the selected region is colored with a multi-hued linear gradient, as opposed to the unselected region that is in grey. The handles are represented by the two black strokes at the left and right borders of the selected region.

This visual representation would act as a colored legend for the position of genomic segments found in other panchromosomes.

b. The comparative table of panchromosomes

The top part of the screen would be filled by a table or section of a table containing panchromosomes and would be used to visually represent the fragments from the main panchromosome that are found in other panchromosomes as well. This tabular view would therefore be useful to explore the existing cooccurrences within a pangenome.

I proposed to visually represent these panchromosomes—as profile lines instead of violin plots to save space—with the cooccurrences from the main panchromosome being drawn as horizontal lines grouped into swarm plots above and below. Cooccurrences would be accurately positioned horizontally and separated vertically to stay distinguishable from each other. Moreover, to provide the information of position of origin on the main panchromosome, colors would be used following the color pattern of the selected region on the main panchromosome.

I chose to use colors rather than ribbons as usually used within synteny browsers as ribbons can overlap and hide parts of the visualization, especially in one versus many representations, as illustrated in Figure 44 and in Appendix XCIV.

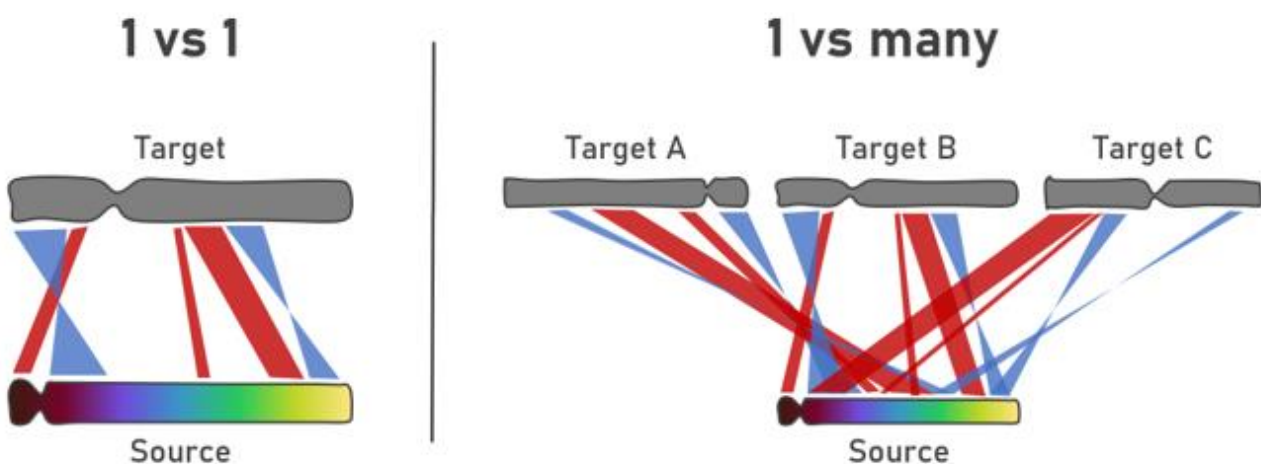


Figure 44: Showing cooccurrences as connecting ribbons between a source and targets scales badly; Synteny viewers usually display common segments with ribbons connecting the equivalent positions, here schematized with the red lines

and blue hourglass-like shapes between a Source panchromosome (below) and a target panchromosome (above). Hourglass-shaped ribbons encode inversions. **Left:** With one-versus-one use cases this visual representation can work, depending on the number of ribbons to display. **Right:** With many targets and/or ribbons it becomes hard to distinguish the ribbons and identify their origins.

Using color for encoding position makes the visual display of ribbon unnecessary, thus leaving less visual clutter, as illustrated in Figure 45. Color brightness alone would be enough to infer the position on the source, hue is a secondary visual channel which improves differentiability between broad regions. Interaction could make the ribbon visible again, for hovered segments or regions for example (see Appendix XCVIII).

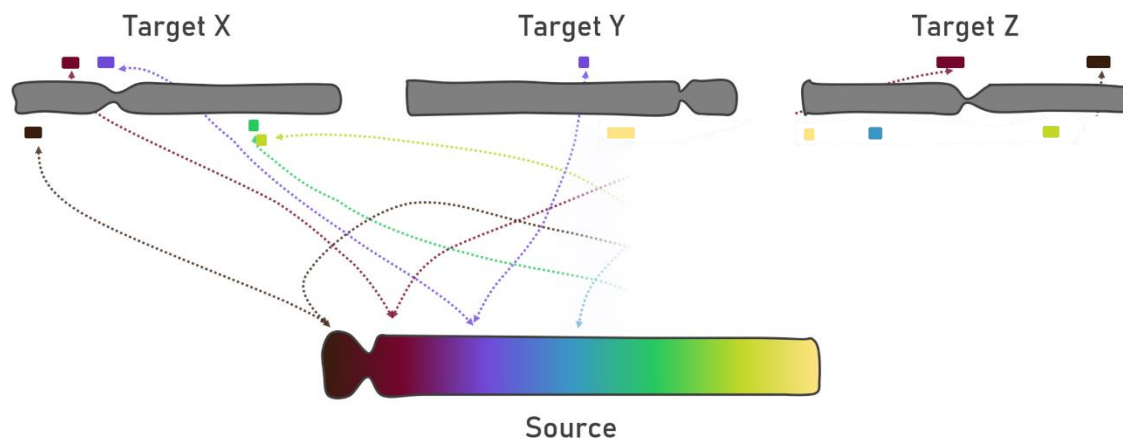


Figure 45: Encoding the position on the source with color makes it possible to hide the ribbons completely; Left: using ribbons to connect the similar segments create multiple overlaps that are hard to read. Right: color is enough to infer the position of origin on the Source and makes the visual representation clearer.

Moreover, I proposed to use the spaces above and below the panchromosome profiles to encode inversions: segments displayed above the panchromosome profile would be in the same direction than their cooccurrence in the main panchromosome, while segments below would be inverted. This distribution was chosen after discussions with colleagues who intuitively thought that the area above was the forward strand (and by extension similar strand), as opposed to the reverse strand below (associated with the inverted strand).

Other uses of the visual channels have been explored, for example encoding similar and inverted strands with hues or double encoding it with both hues and above/below position but I discarded these as they were not making an efficient use of the visual channels available (colors could not be used for the position of origin anymore).

c. Preliminary implementation and abandon

During November-December 2022 I had the opportunity to manage a junior developer on the implementation of this visual representation. I was overseeing the development, controlling quality, providing data, instructions, and explanation to the developer as well as code examples⁵².

This visual representation has not been conserved for the final design of SaVanache, as it was not targeting enough the need of bioinformaticians at Syngenta, focusing to much on the pangenome structure instead of the variations between genomes. The corresponding project is still available in a GitHub repository⁵³ for reference, in an unfinished state. Some questions remain unexplored, as how to best represent the size of cooccurrences—should their glyphs be proportional to the actual size? Should there be a minimal size for on-screen visibility?—and how to represent

⁵² An example interactive implementation of the main panchromosome region selection is available at <https://codepen.io/singingmeerkat/pen/OJjaKyY>

⁵³ <https://github.com/SingingMeerkat/SaVanache>

cooccurrences spanning multiple non-successive nodes on the profile panchromosomes. A tentative user flow illustrating the full visual representation and possible interactions is available in Appendix XCV to Appendix C.

E. View 2.2 – Visualization of SVs inter genomes

The feedback received for the tabular representation of panchromosomes made me redefine the goal of the visual representation of SVs within a pangenome. Users were mainly interested in biologically relevant variations between genomes; the goal therefore was not to visualize SVs within a pangenome but rather to visualize SVs between genomes which were organized within a pangenome. One question which was confusing for example was: “Where can I see the insertions and deletions?” I did not understand it at first as, as explained in SaVanache I.A, pangenomes contain all genomic material from genomes while the notion of deletion and insertion is deeply rooted in pairwise genome comparisons instead. Multiple discussions with colleagues were needed to clarify that what was asked was not to “visualize pangenomes” as originally stated, but to “visualize genomes within a pangenome.” I needed to refocus on the genome paths instead of the pangenome structure.

However, some were also interested in a global view of that structure. My goal was therefore to combine both aspects, with an emphasis on the variations between genomes. I therefore worked on the current design of the *Structural variations* view of SaVanache, illustrated in Figure 46.

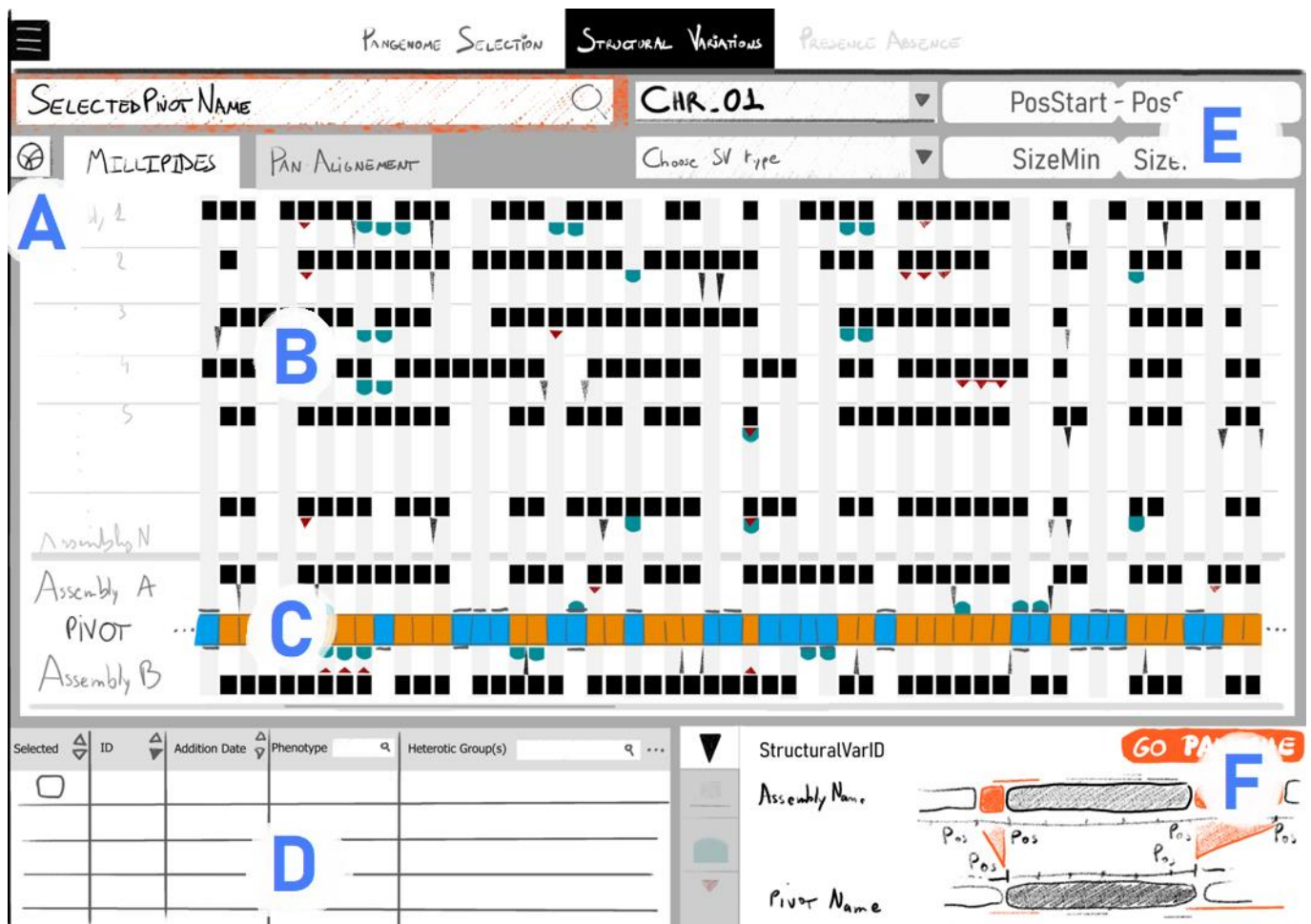


Figure 46: The interface for the visualization of SV is divided into multiple connected subparts; A) Clickable icons and tabs enable a user to change the active visual representation. The left icon could display a PanCircos view, while the two other tabs change the visual representation used in the main view. Other tabs could be added if needed. B) SaVanache’s main visual representation uses a system of glyphs to compare a pivot genome against many other, with a matrix-like

display. C) The pivot genome can be directly compared with two other genomes above and below, updating the local visual representations accordingly. The same color pattern than Panache can be used to identify **core** and **variable** segments. D) The table of assemblies already present in the *Overall diversity* view is kept into the *Structural variations* view and can be used to choose which assemblies should be displayed. E) Input buttons can be used to select the pivot genome, a region of interest, and the types and sizes of SVs to display. F) A panel in the bottom-right displays detail on a selected part of the main visual representations, with information of position, sizes, and the possibility to navigate to the *Presence Absence* view.

1. PanCircos as an optional overview

The concept of PanCircos (see SaVanache II.D.2) was kept as an optional visual representation, present as a togglable window within SaVanache's interface. A user could quickly show or hide a panel displaying the cooccurrences between panchromosomes, showed as violin plots laid out circularly, as illustrated in Figure 47. This panel could therefore be useful for providing an overview of the cooccurrences within the pangenome but would not be the main visual representation used within that view. Again, users could highlight cooccurrences appearing in a certain panchromosome, or filter out cooccurrences based on their sizes.

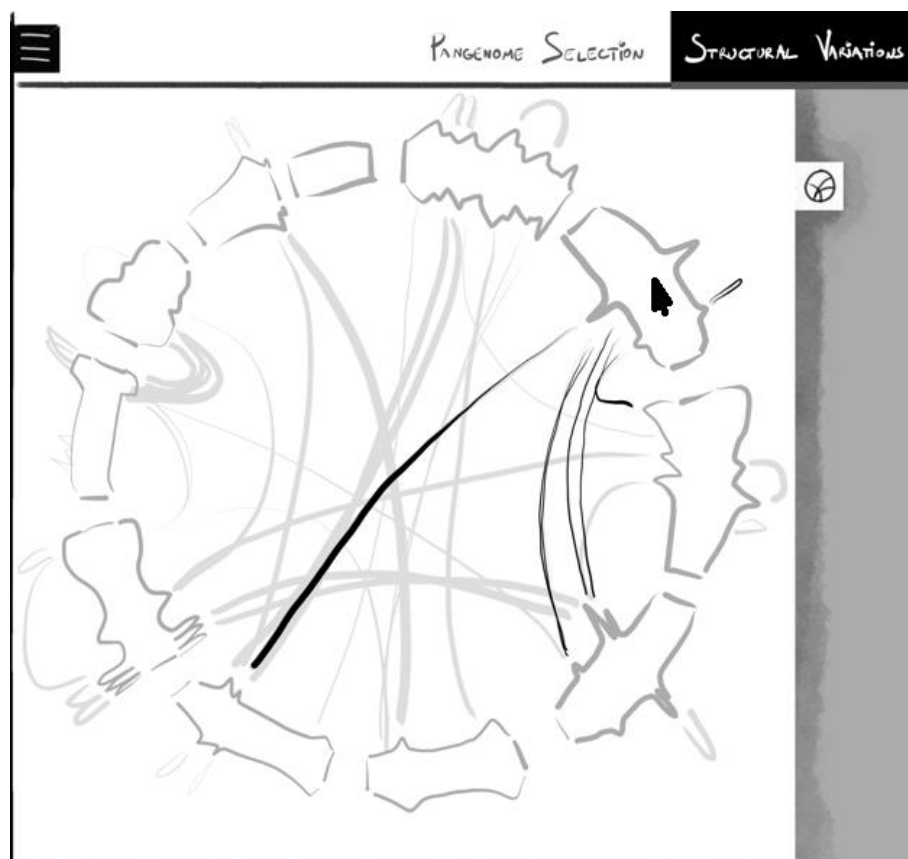


Figure 47: A PanCircos visual representation can be opened by clicking the PanCircos icon within the interface; A little icon representing a circle containing three arcs, as showed in the top-right part of this figure, can be clicked to hide or show the PanCircos overview of cooccurrences throughout the chosen pangenome. Hovering a panchromosome highlights the related cooccurrences on the display, by showing the connections as thick black lines as opposed to the light grey lines of the other cooccurrences.

2. Selection of a 'pivot' genome

The main visual representation had to be focused on genomes and their variations. While redefining the needs for this representation, target users agreed that they were not interested in all-versus-all visualizations but would be more interested in one-versus-all or one-versus-many visualizations. One genome (and most often a region of one genome) would be at the center of the analysis—for example either a widely used reference genome or a genome sequence with a high

quality—and would then be compared to others. Comparisons between other genomes could be of interest too but would not happen at the same time than the comparison with the main genome.

The visual representation could therefore revolve around one main genome, that we call '*pivot*', compared with many others. That genome would serve as a reference system for the whole visualization and would anchor it in an existing biological context. Moreover, that *pivot* could be swapped for another at any time to enable comparisons between other pairs of genomes. Users could therefore choose a region within a known genome, see how it compares with other genomes, and navigate between comparisons at will. It is important to note that the *pivot* could be any linear path through the graph, even paths that have not been observed in linear sequences. For example it could be a panreference path visiting the most common nodes, or the longest, or even the ones followed by the most branches.

3. Glyphs for representing SVs in pangenome graphs

I propose a novel visual representation for the visualization of SV between genomes within pangenome graphs, based on a glyph system. Each type of SV would have a glyph with a dedicated shape and color, laid out within a rectangle corresponding to a node (or group of nodes) from a *pivot* genome within a DAG graph. Using DAGs means that fragments of DNA sequences present in multiple copies or different positions between genomes would be stored as multiple nodes instead of being merged into a unique node connected to many others. Those rectangles would be positioned on a row following the node order from the *pivot*, creating a table of pairwise comparisons where each row shows the existing SVs between a genome and the *pivot*.

a. Proposed SV nomenclature in a pangenome graph

Within a graph, two genomes correspond to two paths following common and different nodes. These paths are similar if the synteny is preserved but diverge with every variation between the two. I call these divergences '*synteny disruptions*' in the context of pangenome graphs, which are close to the notion of '*breakpoints*' used by some SV visualization tools (see SaVanache I.B) in linear contexts.

When using a *pivot* genome (and by extension a *pivot* path) the graph can be read as linearized following the succession order of the nodes within the *pivot*, recreating the original sequence succession. Comparing the linear pivot path to another folded path can reveal special patterns corresponding to usual SVs, as illustrated in Figure 48.

We can identify the same basic SVs than usually described based on the path taken from synteny disruptions:

- **Deletion:** a path skips one or many nodes before reconnecting to the pivot.
- **Insertion:** a path has one or more nodes before continuing the same path as the pivot.
- **Inversion:** a path has a node but in the opposite direction than the pivot.
- **Duplication:** a path has multiple copies of a node from the pivot.
- **Translocation:** a path has one or more copy of a node from the pivot, but not in the same relative node succession.

To these I add two special types:

- **Swap:** upon divergence, the compared path and the pivot each have a succession of at least one different node before reconnecting. Both deletions and insertions are special types of swaps, where one of the paths has no node before the reconnection.
- **Inversion chain:** multiple nodes from the pivot are both inverted and traversed in the opposite order by a path; they would have been described as a single inversion if these nodes were merged. Also called chained inversion.

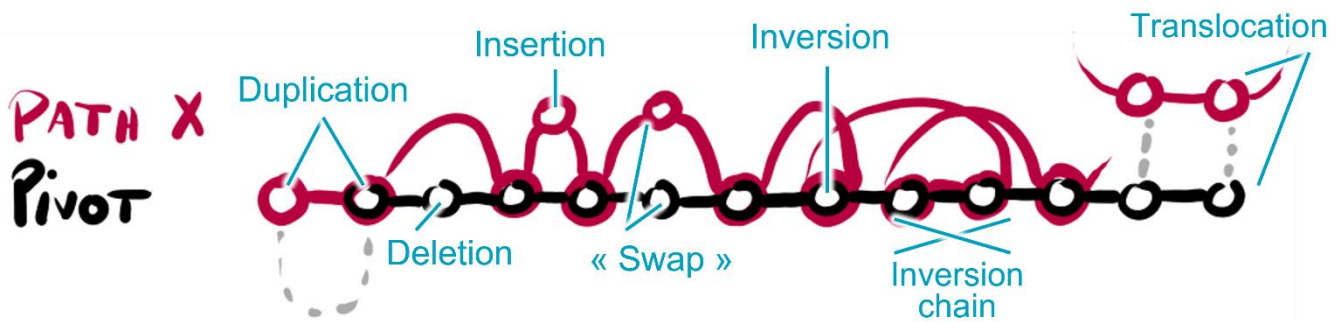


Figure 48: Each comparison to a pivot path reveals several patterns that can be linked with common SV types; Here a portion of a pangenome graph is schematized, with a pivot path represented as a linear succession of black nodes and edges, and a compared path X in magenta, sharing some of its node with the pivot. Dotted strokes represent cooccurrences of nodes through the pangenome graph. From left to right, we can identify: a **duplication** (a node from the pivot is present twice in the path X); a **deletion** (path X skips a node from the pivot); an **insertion** (path X has an additional node before reconnecting to the pivot); a **'swap'** (the pivot and path X have different sequences between two common nodes); an **inversion** (the path X has a node inverted compare with the pivot); an **inversion chain** (or 'chained inversion', multiple nodes that, if merged, would have been labelled as a normal inversion); a **translocation** (nodes from the pivot are found in path X but in a different location, either another chromosome or among a different succession of nodes).

The mostly linear nature of DAG would limit the number of special nested variations, but the existence of complex combinations would depend on the algorithms used to create the pangenome graphs (and to turn them into DAGs if needed).

b. A glyph system for the visual representation of SVs

I propose to visually represent each of these SV types with a glyph or combination of glyphs as described in Figure 49, distributed within two square spaces on top of each other symbolizing the pivot node.

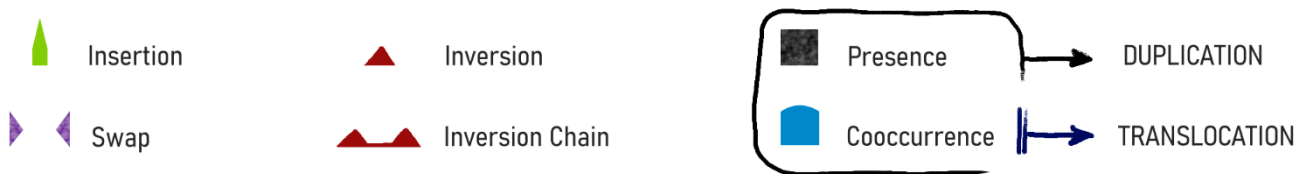


Figure 49: Six different glyphs would be enough to encode all identified SV types; Insertions can be represented as light green pointed shapes. Swaps are depicted with violet triangles facing each other, with emptiness in the middle. Inversions and inversion chains are (connected) red triangles. Presence of a node is encoded by a black square, and absence by no square at all. By elimination, deletions are represented by an absence of square, without any swap glyph either. If a cooccurrence of a node from the pivot is present within the compared path, it is drawn as a blue square with a rounded extremity. Therefore, a combination of a presence glyph and cooccurrence glyph encodes a duplication, while an absent 'presence glyph' associated with a cooccurrence glyph describes a translocation.

The top part encodes the presence or absence of the pivot node within the compared path. The bottom part hosts glyphs characterizing the observed type of SV. Multiple SVs can be depicted at once within the same pair of squares, overlapping if needed. A pairwise comparison of a path and the pivot can therefore be drawn as two rows, one encoding the PAV of pivot nodes and the other below detailing the encountered SVs (apart from *swap* glyphs, that are on the PAV row), as illustrated in Figure 50.

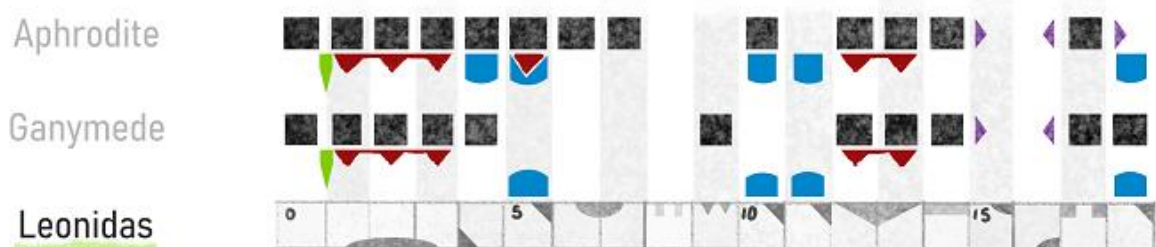
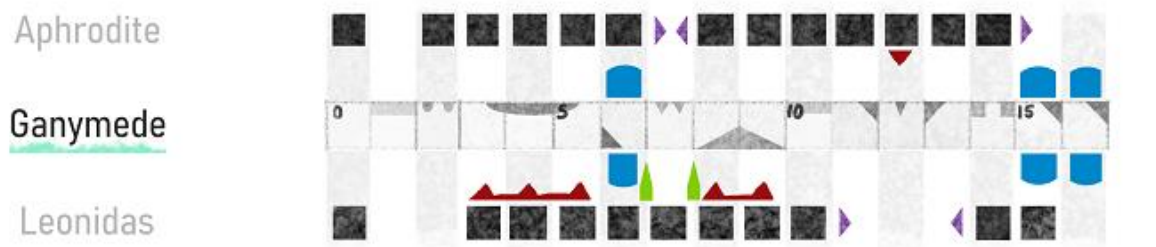
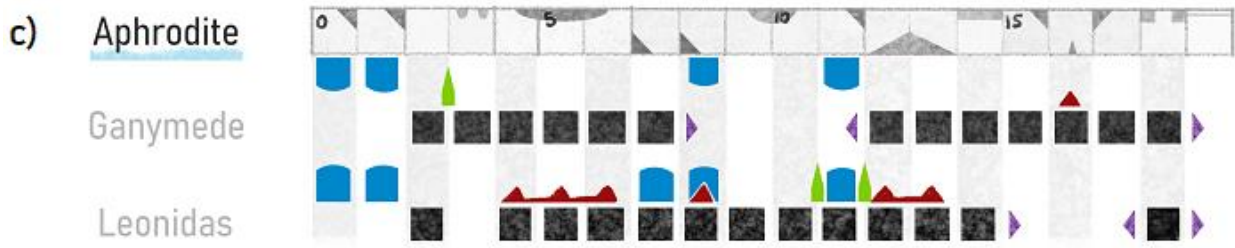
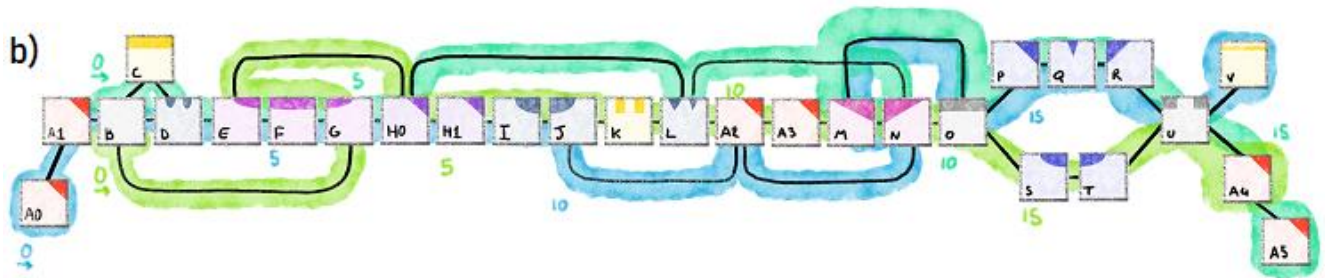
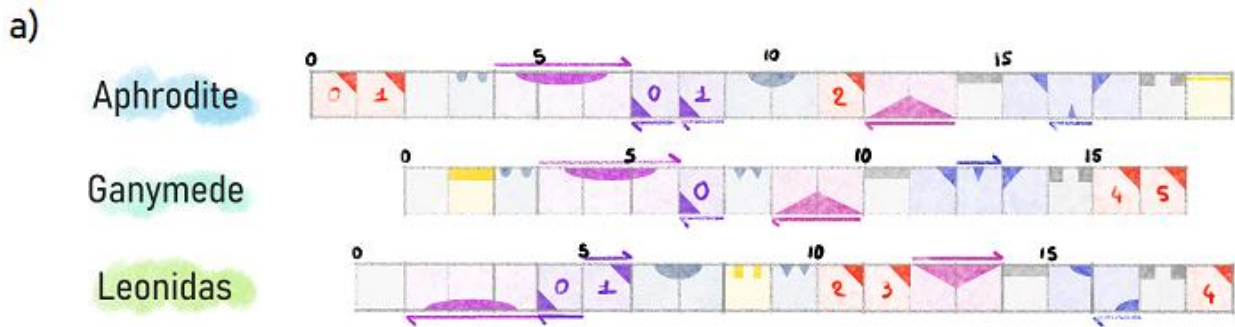


Figure 50: The glyph system depicts SV from a pangenome graph, built from linear genomes; This figure is an overview of the conversion from sequences to graph, then to the glyph system introduced here. **a)** Three genomes (Aphrodite, Ganymede and Leonidas) are represented as successions of blocks of abstracted size. Each block has a mark and color used here to schematize shared and unique blocks between genomes. Two blocks with the same combination of color and mark represent a similar sequence. Numbered blocks are blocks with cooccurrences, i.e. multiple copies within genomes, or differently positioned occurrences between genomes. The colors and marks are drawn here only to highlight how the DAG pangenome graph is related to the three genomes. Arrows depict a block's orientation, by default blocks with marks on top are in the forward direction. Moreover, each genome is associated with a color: Aphrodite with blue (■), Ganymede with turquoise (■), and Leonidas with green (■). **b)** The three genomes can be represented within a pangenome graph. Each block is represented with the same color and marks than before, but only cooccurrences are present multiple times. The original block successions can be retrieved by following the blue, turquoise, or green paths through the graph. Colored numbers indicate at which index a block occurs within the original genomes. For example, the block at index 5 (i.e. the 6th block) is F in Aphrodite, G in Ganymede, and H₁ in Leonidas. **c)** The glyph system is used with three different chosen pivots, Aphrodite first, Ganymede second, and Leonidas third. The pivot's blocks are depicted as greyed-out blocks with the same mark as in the visual representations above, they would not be displayed as such within SaVanache's interface. Each pairwise comparison is represented with two rows, one with the insertions, inversions and cooccurrences glyphs, and the second with the presence and swap glyphs. All glyphs are directed toward the pivot, apart from cooccurrence glyphs if they are directly next to the pivot. Note that inversion and cooccurrence glyphs can overlap, for example on index 5 with Leonidas as a pivot: a red triangle is over a blue shape. **d)** The glyph legend, as detailed earlier.

c. Implementation of the glyph system

This idea of a glyph system for representing SV visually came from two main ideas gathered from discussions with target end users: there was a lack of tools visually encoding SV directly (see SaVanache I.B); block sizes could be abstracted for an overview of SVs, before zooming in another scale. Here each SV has a dedicated glyph or combination of glyphs, and sequence segments are arbitrarily set to the same size visually for readability. A detail view would provide information on the actual block sizes and coordinates in the genomes.

The colors were chosen to be differentiable even for colorblind people, by selecting combination of hues and luminosities that would be still readable (see Figure 51), especially when combined to glyph shapes. Comparisons of the glyph system with four types of colorblindness are displayed in Appendix CI.

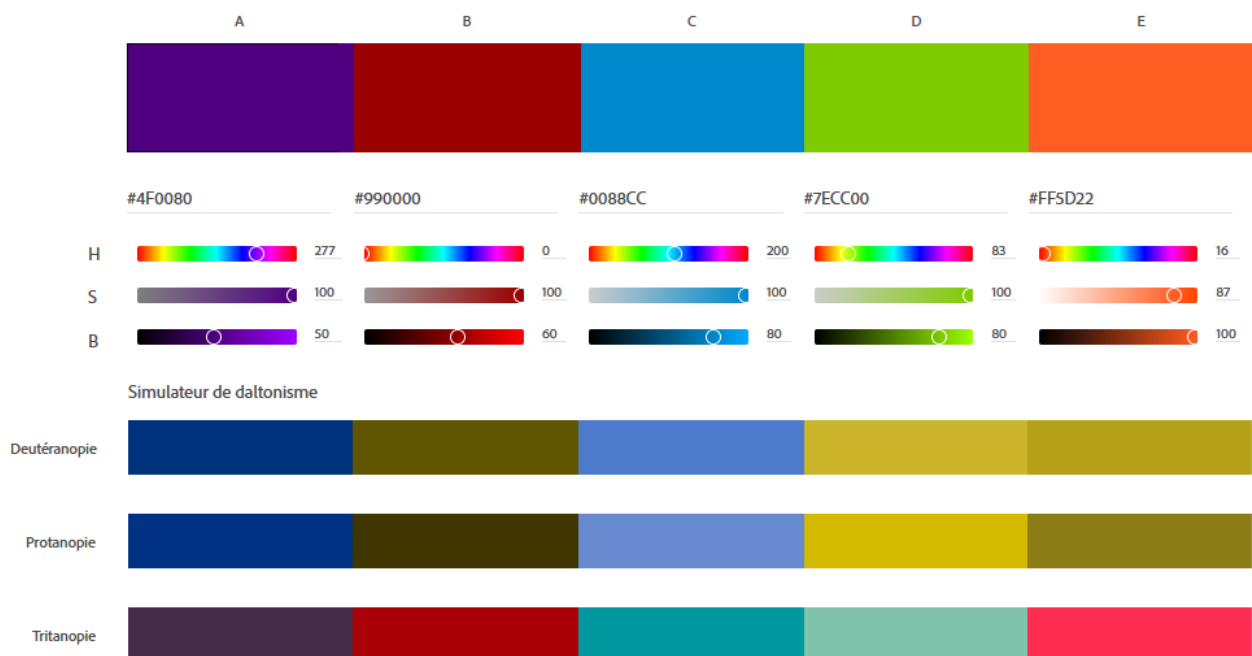


Figure 51: Five colors were chosen for SaVanache UI with the glyph system on display; With varied hues and luminosities, the five colors are differentiable with any kind of colorblindness. Violet (■) for Swaps, red (■) for Inversions, blue (■) for Cooccurrences, (■) green for Insertions. Orange (■) is the main color used for highlights within the interface. It could be

mistaken for green with deuteranopia or red with protanopia, but the different context should be enough to correctly distinguish them. Colors were selected using the Adobe Color Palette tool⁵⁴.

The shapes were chosen to fulfill four criteria, in order of importance:

- Easily distinguishable
- Combinable within one pair of squares
- Ease of understanding / intuitiveness
- Solid shapes

A glyph's shape must encode both the position of an SV compared with the *pivot* and the nature of that SV. It should therefore be easily and quickly identifiable, and **distinguishable** from each other (for example by occupying different niches within the SCI space identified by Huang [245] and detailed in State of the Art III.C.5). Cooccurrences are the only rounded shapes and are therefore highly identifiable, for having low spikiness and high compactness. Presence squares are differentiable as they are bigger than all other shapes and have no diagonal sides—higher spikiness than cooccurrences but especially a different orientation than the triangles. Insertions, inversions, and swap glyphs are all based on triangles, but are differentiable with both their sizes, positions, and orientations (and colors as detailed above). Insertion glyphs are elongated—high spikiness—and positioned between two *pivot* nodes. Inversion glyphs are centered on presence squares and are directed toward a vertical axis. Swap glyphs on the other hand are on the same row as presence squares and are directed towards a horizontal axis. These last two triangles do not differ much in shape, but differ in position, orientation, and color

All glyphs can currently **coexist** within one pair of squares, as illustrated in Figure 52. This was needed as combinations of SVs can exist on one node, and a visual representation for groups of genomes could end with all glyphs in one place, as detailed in SaVanache II.E.5.

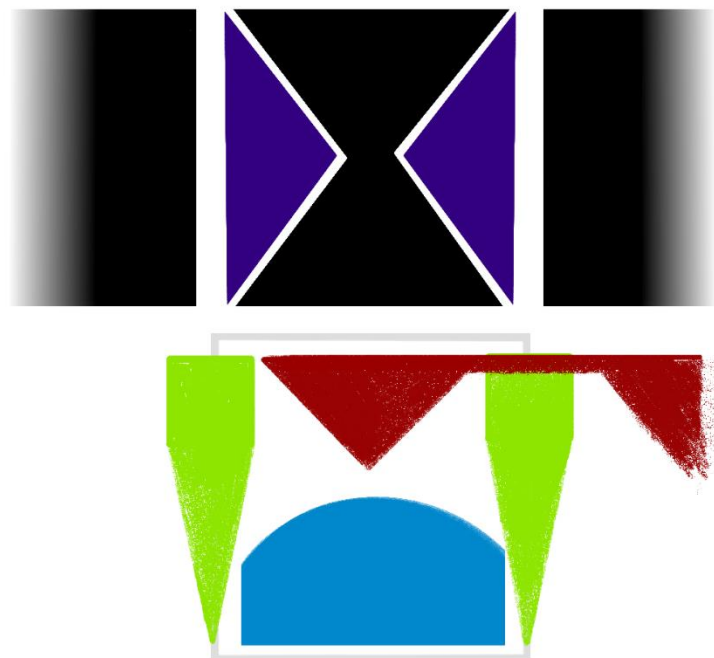


Figure 52: All glyphs can fit within one comparison unit and still be identifiable; Most of the glyphs (insertions, inversions, and cooccurrences) fit below the presence squares. Swaps share the same space as presence squares but cannot be present at the same time as them unless comparisons are grouped into one line. Even when grouped, all shapes are distinguishable.

⁵⁴ <https://color.adobe.com/create/color-accessibility>

Shapes should be easy to interpret, an **intuitive** design would make them more memorable and would reduce the cognitive load for the user (see State of the Art III.C.4), reducing the need to refer to a legend⁵⁵.

Finally, I needed the shape to be **solid** (as opposed to hollow) as I wanted to use the color within for both distinguishability and the visual representation of groups of genomes.

d. Feedback on the glyphs

Feedback was collected from colleagues who had never seen the glyphs proposed. First discussions showed that insertions were easy to recognize as the triangle is a common shape used for this SV. Other shapes were harder to associate with a known SV at first, but it became more intuitive after a dozen of minutes with explanation. The swap glyph could be mistaken for a deletion, as the two facing triangles seem to directly 'connect' to each other.

Proper experiments would be needed to assess the efficiency of the chosen shapes, evaluating both their intuitiveness and learnability, as explanation seemed to make them more understandable. Combined with a legend available within the UI and tutorial(s), an unknown glyph system could quickly become intuitive if it was not already at first sight. Moreover, alternative visual representations for the swap should be explored, to further differentiate them from areas with deletions. Ideas include filling the space between swap glyphs, or to change the shapes to something else that would not give the same sense of connection between the two ends.

Moreover, some colleagues wanted to see comparisons on one row instead of two juxtaposed squares, to display more assemblies at once. An alternative glyph system could be imagined, it would be interesting to measure the efficiency of the two, and maybe integrate the possibility to switch to one or the other. My hypothesis is that it would be harder to find SVs within a mono-row visual representation, as it is easier to detect a glyph in the middle of a white background than in the middle of other marks (for example presence squares against an inversion triangle).

4. Dedicated UI

As showed in Figure 46, this glyph system can be displayed in a tabular matrix, where each row is dedicated to the pairwise comparison of a genome assembly with the *pivot*, and each column corresponds to a node from this *pivot*, ordered following the actual succession of nodes within that path and with a unified width (the sequence length is abstracted and not represented visually).

Below, a table lists all assemblies available within the chosen pangenome, as in the previous *Overall diversity* view (see SaVanache II.C). A user could choose to hide or show any assembly from both this table and the matrix above. The matrix only shows a predefined number of columns at a time, and a slider at its bottom enables users to pan through all the columns that can be associated with the selected region.

The matrix could also be used to move the assemblies or the pivot up and down, or group rows, displaying an alternative glyph system, as described in SaVanache II.E.5. It would also have a togglable side panel providing basic summary statistics, as described in SaVanache II.E.6.

The bottom right part of the display is dedicated to a detail view of SVs, showing realistic proportions for the nodes along with coordinates of the breakpoints and visual representation of the different types of SVs, as described in SaVanache II.E.7.

5. Alternative representation of glyphs for groups

To offer the possibility to compare groups of genome assemblies, I propose an alternative to the glyph system, where the sum of each column from a group of genomes is summarized within one

⁵⁵ This is loosely related to the principle of least astonishment described in State of the Art III.C.4

row. The glyphs are the same, but I propose to use the color intensity of the filling of each shape as a visual channel for the percentage of genomes within a group that have a certain SV glyph. This would be a summary of the SV glyphs rather than of the actual SVs they depict. For example, multiple genomes can have a synteny disruption after the same node, with a swap, but that swap does not have to be of the same length or to connect back to the same *pivot* node in both genomes. The summarized line would only indicate that both genomes have the beginning of a swap at the same position, but their ends might be at different places.

The contour of a glyph would be drawn for each SV glyph found, and the intensity of its filling would range from white to the normal color of that glyph depending on how many genomes share that glyph, as illustrated in Figure 53. Presence square would also be drawn this way, as a pseudo heatmap of glyphs.

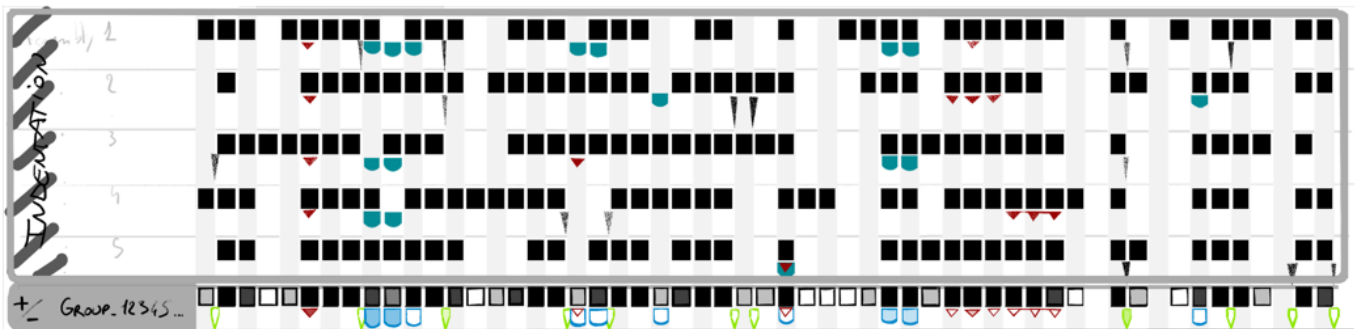


Figure 53: Glyphs can be colored as a heatmap to depict the distribution of SV annotations within a group of assemblies; This example uses an old design for the insertion glyph (black triangles on top, green triangles at the bottom line). As examples, the first insertion in the group row comes only from the assembly 3: it is filled with white. As all assemblies have a presence square for the second column, it is filled with full black in the group row. Then the insertion is present only in the first rows, but not in the last one. The glyph within the group row is filled at $4/5 \times 100 = 80\%$ of intensity (which would correspond to the color obtained by putting the normally filled glyph at 80% of opacity on a white background). The group row therefore acts as a heatmap highlighting where SVs are present, and in what frequency.

6. Summary of SVs within the selected region

The matrix of SV glyphs is completed with a togglable side panel on the right, showing the distribution of each SV glyph between the displayed assemblies on the current region on display. Each column displays bar charts of the percentage of nodes with a certain SV glyph for each row, with a column for the total, as illustrated in Figure 54. This view would enable users to quickly find the genome assemblies with the most SV detected, and if some assemblies are more subject to a certain type of SV. Interactions can be included, like sorting the assemblies based on the count of a certain SV glyph, turning on or off the color of the heatmap for the bar chart, or displaying on hovering a vertical stroke on all rows within a column, acting as a visual cue for easier comparison between rows.



Figure 54: A summary panel can show bar charts of the percentage of nodes with a certain SV glyph within an assembly; Once more, the glyph system in use here is not the most recent, with the first column depicting deletions and the second insertions. This panel comes with sorting options and other interactive options. A whole column's width corresponds to the maximum for all bar charts within, meaning that widths are not comparable between columns: one assembly could have the highest number of insertions and inversions, both bar charts would reach the maximum length, but it does not mean that there would be as many nodes with insertions as nodes with inversions. To compare values between columns, the user could click to toggle on the bar chart colormap common to all columns, with 0% being a dark violet and 100% a light beige (as in the magma color palette). Moreover, written value can be highlighted on hovering if need be.

7. Details on SVs from a selected intersection

Within the glyph matrix, a user could click on the comparison of a given step of the *pivot* path with a genome assembly—simply by clicking on any intersection between a row and column—to display a detail view depicting the sequence segments and SVs involved, as illustrated in Figure 55.

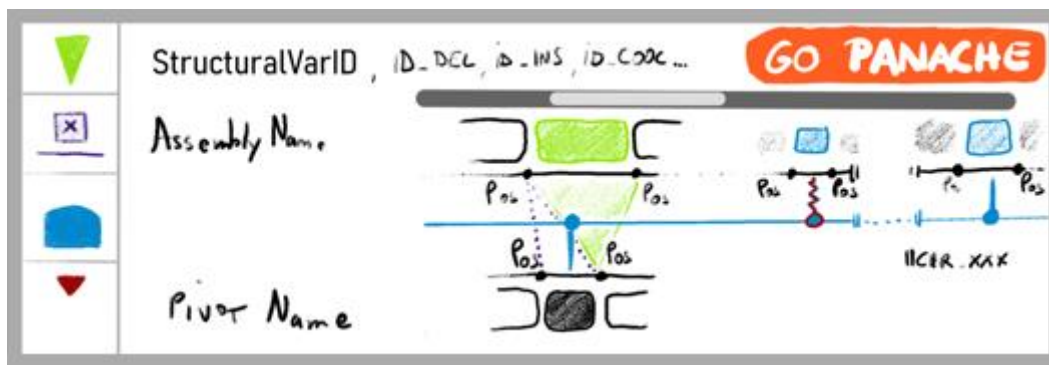


Figure 55: Multiple SVs can be depicted simultaneously in the detail view; The left part of the detail view is composed of toggable tabs, used to highlight one or many type(s) of SVs on the main visual representation on the right. From top to bottom: insertions, deletions, cooccurrences, inversions. Swaps are missing in this draft version. The right part displays the identifiers of SV(s) found at that specific pairwise comparison, along with the assembly's name on top and pivot name at the bottom. A user could pan through this visual representation, which shows blocks of sequence in their genomic context and with proportioned widths. Below, the node targeted by the pivot step is drawn, within the two direct neighbor nodes, and with written coordinates at each breakpoint. On top, nodes from the assembly are drawn too, including cooccurrences (blocks in blue). Every SV can be depicted in between: deletion as dotted triangles, insertions as solid green triangles with the inserted segment in green, cooccurrences as strokes connected to a central line, inversions as zigzags between blocks or connecting to the central line in this case. In the more recent design, a swap would be drawn here instead of a combination of insertion and deletion.

This view correctly depicts the sizes of nodes involved with SVs at the selected location and enables users to highlight some of the existing SVs, or even all types at once. Various combinations can be imagined, as seen in Appendix CII. This view would provide a more detailed view of the SVs happening between two genomes at a specific location, with their coordinates and relative sizes. It could also be used to jump to the *Presence Absence* view, which would be centered on the selected step.

Preliminary drafts included a visual representation of genes as horizontal lines above the assembly or below the pivot blocks, with color on the lines spanning a breakpoint or being included within an SV. After discussions with target users this idea was not kept, for it was deemed unnecessary at that scale and would overcomplicate the view.

III. SaVanache's implementation

Unlike Panache, SaVanache has not been turned into a public tool yet and is under development. Different in scope and scale, we also took a different approach for its development.

A. Building a composite visualization tool

1. Development strategy

Most of the development efforts done on SaVanache were done by web developers, under my supervision.

The first development efforts were focused on the implementation of the first draft of a one-versus-many visualization of SVs, as detailed in SaVanache II.D. This work resulted in a prototype that was later abandoned to better target the users' needs but was key in defining the next steps. Another developer later worked on the implementation of the scatterplot for the *Overall diversity* view of SaVanache. His work was done in parallel to another web developer, working on the overall UI of SaVanache and the implementation of the *Structural variations* view.

2. Chosen technologies

As for Panache, we chose to develop a web application, for portability, ease of use, and the great flexibility offered for the creation of interactive tools.

I kept working with Vue, but version 3 instead of 2 as it came out not too long before development and related libraries were bound to evolve to this later version too.

As for the visualizations, the tabular prototype used vanilla Vue and SVGs. The scatterplot has been built using the library JsCharting for Vue⁵⁶. The implementation of the glyph system and detail view within the *Structural variations* view have been created with positioned 'div' HTML5 web elements as a proof-of-concepts and will be improved with further developments.

3. Overall diversity view

The files used for the *Overall diversity* view are straightforward, simple TSVs storing the positions on the scatterplot along with metadata. An additional TSV is currently used to store the names of assemblies (columns, in a list) contained by each pangenome (rows). Additional TSVs could be uploaded by a user to specify a new categorization of the assemblies to use for the coloring system detailed in SaVanache II.C.2.a.

All basic parts have been created into different Vue components (one for the scatterplot, one for the assembly table, one for the pangenome list), reusable in another project. Due to the novelty of Vue version 3, some of the expected functionalities were not implemented yet in the libraries used for the pre-styled components needed for the UI. A combination of vuetify and vuestic along with in-house modifications were for example needed to implement the hovering behavior on the assembly table too.

The developer chose JsCharting among other plotting libraries, to easily add interaction to the scatterplot representation. Unfortunately, it did not come with the possibility to do a lasso selection of multiple data points, and this feature is missing. The color system used when the pangenome list is hovered is only partially in place and needs further improvement. Moreover, this visualization is overall slow on use, with observed reaction times exceeding two seconds to update the display on hovering interactions. It is unclear whether this slowness comes from the charting library itself or its implementation within a Vue component which has dynamic variables.

I suspect that the UX would be improved greatly with a vanilla Vue and SVG implementation, based on Vue's dynamic update of the webpages, as it is done within Panache. However, this would mean that all interactions (hovering, selection on click or lassos selection...) would have to be hardcoded. Unless another library that would integrate better with Vue components is identified, I would recommend this solution. The number of assemblies could be a problem for the display when they are too numerous, as SVG HTML5 elements would be created for each dot. When more than a thousand assemblies are to be included, alternatives might be needed, preferably WebGL technologies.

4. Integration within a SPA and *Structural variations* view

a. Special format of pangenome graph file

I propose a dedicated JSON structure pre-computed from pangenome graph, to be used within the *Structural variations* view. The pangenome would be divided into two main objects: a collection of nodes, and a collection of paths for each assembly, as presented in Appendix CIII.

⁵⁶ <https://jscharting.com/>

- **The panSkeleton**

The panSkeleton is the structure of a pangenome graphs. It stores only the Nodes and the information of which are related as cooccurrences (i.e.. nodes that are copies of others). The edges are facultative, as the node successions of interest in our case can be stored in the paths. As the nucleotide sequence is not useful for the proposed design, it is not written into these files either.

All Node is stored thanks to an ID and has three internal properties:

- The **length** of the DNA sequence represented by the Node is stored as an integer.
- A dictionary stores the assemblies that **traverse** the Node, by listing their assemblies name, name of the sequence or chromosome under consideration, and the index of the Node within the whole succession.
- An array listing the ID of other Nodes that are cooccurrences.

- **The paths**

The paths are all the succession of Nodes observed within genome assemblies. Each genome assembly stores a list for all its sequences / chromosomes. These lists represent the succession of Nodes taken by that path within the pangenome graph. At each index an object stores the NodeID, the observed start and end positions of the segment on the linear coordinate system of the assembly, and the strand (+1 if forward, -1 if reverse).

b. Annotating SVs from the JSON file

The annotation of the SVs used for the glyph system should be done directly within SaVanache. This way, it would enable on-the-fly modification of the *pivot* path without processing huge pre-computed files storing all the possible pairwise comparisons needed. Expecting users to provide such pre-computed files would be unrealistic, as their creations would take time and consequent computer space.

I propose a naive approach for this annotation directly from within the pangenome file described above, detailed in a decision tree presented in Appendix CIV to Appendix CVI. This algorithm is based on the comparison of Steps taken by the *pivot* path with Steps taken by other paths—a Step being the combination of a Node's and path's details at a given index within the succession, as illustrated in Figure 56.

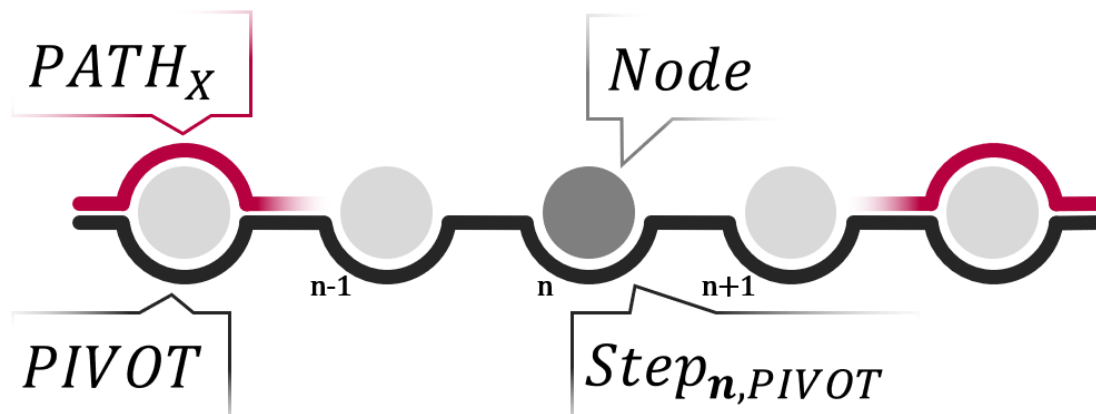


Figure 56: All Nodes from the panSkeleton are described as Steps when included within a path; A Step has the same information as a Node, but also provides detail on the strand, coordinates within the path, and direct neighbors within the succession of a path (as it is easy to retrieve the Step $n+1$ or $n-1$).

Whenever a *synteny disruption* (see SaVanache II.E.3.a) is found after the Step n of the *pivot*, further exploration of the alternate path is done until reaching a Node in common to both assemblies. The characterization of the SV then depends on the behaviors of the diverging paths.

For a naive implementation, insertions and deletions would be limited to cases where the reconnection happens at Step $n+1$ of either the pivot or compared path. An implementation allowing the user to parametrize the algorithm, allowing some degrees of freedom, would be a better solution in a finished visualization tool.

c. *Storage of the SV annotations*

All detected annotations of SVs could then be stored within a sparse matrix, where each index corresponds to a Step within the pivot, and each row to an assembly. SV annotations would be stored as objects at the index where they have been detected, with additional properties used within the visual representation, as described in Appendix CVII to Appendix CIX.

These arrays could be directly converted into the glyph system described above (see SaVanache II.E.3), as it is based on the positioning of glyph at certain positions along the pivot path. Moreover, this structure could be easily parsed, which in turn would make an efficient use of computational resources. The whole comparison could be accessible, and only sub-sections would need to be extracted for the on-screen display (either because only a region from the pivot is selected, or because too many nodes would not fit the display and panning would be needed).

d. *Implementation done*

As of July 2022, SaVanache is under development and has not been made available as a full visualization tool yet. Both the annotation algorithm and visual displays (glyph system, detail view) within the webpage proved to be difficult tasks for the developer, as they are far from usual web development. The panCircos representation will be include in future versions.

Most of the components for the UI are available and functioning, with mock-up visualizations built from HTML 5 div elements and handcrafted files. The available components from the *Overall diversity* view have been integrated within a common interface, with mock-up data for the transition between views.

B. Future developments

As SaVanache is currently in a proof-of-concept state, additional development efforts are still needed.

The *Overall diversity* view should be improved with a new implementation of the scatterplot, which would be more efficient (without noticeable delays between user input and display updates) and would fully integrate the coloring schema introduced in SaVanache II.C.2.c. Addition of interactive selection of multiple assemblies directly within the scatterplot is also expected.

Moreover, as of now the parsing algorithm for the annotation of SVs for the *Structural variations* view, mandatory for the display of the glyph system detailed in SaVanache II.E.3, is not available. Both a mock-up version—useful to check how scalable this approach is and to identify computational bottlenecks—and final version should be implemented. I envision that a webworker technology, enabling parallel computation on multiple threads would be highly beneficial to this algorithm.

Besides, the visual representations still need to be implemented with a scalable technology. Current implementation is in mock-up status and has not been properly implemented yet. As for the scatterplot, SVGs (either from vanilla Vue or handled with D3.js) could be used for a proof-of-concept version, but might not be scalable to highly variable regions, or display of many compared genome assemblies, as each SV glyph would be represented by one HTML5 SVG element. Faster technologies could be needed, and WebGL would be a good way to handle millions of graphic marks seamlessly. Possibly, visualization grammars (Gosling, Vega-lite...) could enable an easy integration of this technology while offering interactive options, but none of the current developers (myself included) are familiar with these technologies.

Finally, additional developments should address the integration of the last two views within the tool: Panache, and the view dedicated to sequence-level variations. All views work with their own set of files and formats; ideally, an inclusive tool would offer transitions between each file and even internal conversions between formats to limit the need for pre-computed files as inputs. Some users raised the need to navigate directly to certain ulterior views, without going through the first views before. The possibility to bypass views should therefore be explored too.

IV. Discussion

In this chapter, I introduced SaVanache as a potential composite visualization tool for pangenomes. It is based on four views, each dedicated to a certain scale of interest: *Overall diversity*, *Structural variations*, *Presence Absence*, and *Haplotypes*. SaVanache would be with MoMIG [186] the only tool offering multi-scale interactive visualization of pangenomes, from overview to sequence levels. I propose an interactive coloring of dots within a scatterplot, based on selected assemblies and chosen pangenomes within the first view. I introduced multiple novel representations of SVs within a pangenome, including a system of glyphs used to visually represent SV annotations for one-versus-many comparisons within a pangenome graph. This representation would enable the visualization of one-versus-many SVs within genomes of pangenome. I then detail these designs and their effective or envisioned implementation within an integrative visualization tool.

A. Design validation and extension

While the novel visual representations described have received positive feedback and have been designed to help biologists and bioinformaticians to explore pangenomic data, it is unclear how efficient these designs are in practice. Validation steps took place as discussions with colleagues from different teams (experienced scientists from the bioinformatics teams of both IRD and Syngenta, biologists from Syngenta, PhD students the IRD...) and would benefit from hands-on experience of the tool and its visual representations. Moreover, the designs have been created for novel workflows that are neither in place yet nor improvements on already existing analysis workflows. This makes comparisons with other approaches difficult.

Possible redesigns of the glyph system could be needed depending on such evaluations. Alternative glyph shapes and colors could be used to better balance familiarity and novelty, if they can be both more intuitive and easily readable regarding the constraints described in SaVanache II.E.3.c. As raised by target end users, narrowing the display of comparisons within one row per comparison instead of two would also be of interest. Moreover, the learnability of the visual representations would also need to be evaluated when a tutorial is provided. Do these novel visual representations become more intuitive after a short time or is there a steep learning curve that would prevent the adoption?

An additional area worth exploring is the representation of uncertainty, of the quality of the underlying data. Currently, every assembly is given the same weight visually, no matter how they have been built; providing representations of metadata (such as the sequencing technology used, or the percentage of identity of each Step compared with their Node) would help users to better distinguish highly reliable regions, enabling more nuanced comparisons of data.

B. Development of an integrated solution

One of the many challenges of SaVanache is that all the visual representations must be connected within a tool, with interactive links between each and a good handling of data states throughout the whole application.

As of now, each view is thought as a separated instance, taking its data from different files and file formats, as identified in Appendix CX. However, some information is redundant between all these views and could be better handled, with on-the-fly computations. For example, the information of the assemblies contained by a pangenome, needed for the *Overall diversity* view, is already present within the JSON files used in the *Structural variations* view, as all paths are listed within. Similarly, Panache uses TSV files or pre-computed JSON files as input, that are not directly linkable to the JSON format used within the *Structural variations* view. Harmonizing the file formats throughout the tool would save memory space and facilitate their parsing.

Furthermore, the current development efforts do not include any database for all the data that would be used, and only has vuex as a state management solution. Depending on the use cases, relying on a server-side database would be a good addition, for example when deploying instances of a visualization tool for genomes of a specific species, as it has been done with multiple genome browsers.

Because of the novelty of the glyph system and the related file formats, there is currently no tool that can convert existing pangenome graph formats into the JSON format needed. Such tools are in development both at Syngenta (in-house effort to connect to other workflows) and IRD, as Nguyet DANG worked during her PhD on the creation of PARROT, a tool converting pangenome graphs into various format. Discussions revealed that the conversion needed for the glyph system could not be achieved within our respective PhD timeframes but could be added in the future.

C. Feedback and general discussion

Designing visual representations for SaVanache proved difficult, because of ill-defined objectives. This visualization tool is built for an analysis workflow that is not currently existing, as biologists and bioinformaticians have not shifted from mono-reference studies to pangenomes yet. This comes with a lot of freedom, which offers a vast creative space, but also means that there are no standards to build upon, and not properly defined data to use. I have been told to “*imagine the representation for this idea, and we will figure out the data later*”, which was really confusing as the design studies usually follow a ‘top down’ approach, from the data domain and abstractions to idioms and algorithms (as explained in Appendix CXI).

Moreover, I had to think of a tool that would be useful for three to four different stakeholders: Syngenta (with breeding approaches), IRD and Bioversity International (research organisms), and potentially other public research labs that could use a pangenome visualization tool.

This variety of end users and use cases made the identification of user flows difficult, with at times incompatible priorities from each. These blurred objectives were what led to the abandonment of the tabular violin-like design, and multiple discussions with all involved parties were needed to retarget a common goal.

Design steps aside, overseeing the development of SaVanache instead of actively participating in the development has been an interesting experience. Having professionals to work on the development was helpful, and I discovered that a lot of efforts had to be put into regular meetings to make sure that development is going in the right direction.

The specificities of datavisualization really showed, as the web developers had difficulties to work on tasks more related to the implementation of a visualization or algorithm. Profusely explaining the biological, datavis and algorithmic concepts took a lot of time, as well as the creation of step-by-step user flows, and the results were not always up to expectations. As datavis leverages skills outside of the usual web developer skillset, it appeared to be a difficult task to work on, especially in a context where real data are not properly formatted. It still greatly helped for the creation of the UI and overall architecture of the tool. As discussed with some of the developers, I believe that specialized developers would be a better fit for the development of the parts related to interactive visualization and/or handling of complex data.

As for SaVanache, some questions still need to be addressed. As the *Haplotypes* view has not been within focus during this PhD, neither its exact design nor an existing tool to reuse have been defined yet. As for nomenclature, I am also unsure of how to call the type of conceptual pangenome graphs used for the *Structural variations* view: DAGs are acyclic by definition, but inversion chains would imply the existence of local loops within a graph.

Besides, even though Panache was thought as the visualization used for the Presence Absence view, the visualization developed for the *Structural variations* view enhances its concepts in many ways. The glyph system also shows PAVs with its presence squares, and the **core** or **variable** status could be encoded directly within the part of the visual representation dedicated to the *pivot*. It currently does not have Panache's sorting options, but it would be fully possible. In a sense, the glyph visual representation is an extension of the first version of Panache. Linking the glyph system directly within Panache makes sense as we wish to reuse the existing development, but the many overlaps between the two views makes their connection appear unnecessary.

One possibility would be to use Panache as an alternative visual representation within the *Structural variations* view. Another would be to enhance Panache to be able to work directly on pangenome graphs, to modify the reference linear coordinate system at will too. This way the connection from the *Structural variations* view would not modify the coordinate system and could correspond to a zoom-in transition. Besides, the questions of a graph's granularity, or resolution, would also have its importance here. The *Structural variations* view, dedicated to bigger rearrangements, could work on low resolution graphs, further refined into high resolution graphs when the view scale switches to Panache, implying smaller blocks. Whether these changes in detail level are to be computed from one exhaustive pangenome graph or extracted from multiple pangenome graphs built with various degrees of resolution remains to be seen. If possible, on the fly modification of a graph's granularity would also be beneficial for the UX.

D. Application of the 10 rules

Working on SaVanache leveraged different skills than Panache, and how the 10 rules from '*Ten rules on Genomic Visualization Tool Development*' have been applied changed accordingly. One of the main differences is that SaVanache is not available as a public tool yet, and all rules related to tool deployment and community (8 to 10) did not apply during the time of my PhD.

As for design, rules 3 to 5 had a great role in the creation of SaVanache. By nature, it is a *composite* visualization tool, which aims at representing multiple genomic scales with different visual representations, as covered in the Rule 3. Shneiderman's mantra from Rule 5 has been applied for the whole tool's concept, but also within a view as illustrated by the glyph system and detail panel from the *Structural variations* view.

Regarding Rule 6, we did use novel technologies, as Vue is a well-known JS framework and there is no doubt about its future maintenance and wide adoption. Its novelty however did come with some problems as not all libraries that we wanted to use had components compatible with Vue 3 yet, and we had to manually combine some of them to achieve our goals.

A rule that took an unexpected turn was the Rule 7. No properly formatted real file was available at the beginning of development, and I had to provide mock-up files to the developer that had to work on the SV annotation algorithm. I first provided files made with an ad-hoc python script, but it quickly became apparent that we could not properly check if the resulting annotations were correct because of the size of the files. I therefore created a handcrafted small dataset, including all the expected patterns, along with the cheat sheet from Figure 50 for reference. The creation of this file was time consuming and included some typos, but it was time well invested as it really helped all parties to better understand how the algorithm was working (or rather not working, here). It also helped to identify unnecessary parts from the file format specification, which has been modified since.

Conclusion

I. Summary

In this dissertation, I first detailed the state of the art of pangenome visualization. I highlighted differences between *pan-gene atlases*, which focus on genes only, and *pangenomes*, which also include intergenic sequences. I presented existing tools usable for the visualization of pgAtlases and pangenomes and identified five existing categories: *unspecific*, *qualifying*, *positioned*, *structural*, and *composite* tools. Since the beginning of this PhD project, new tools embraced pangenomes instead of pgAtlases, shifting from qualifying and positioned tools towards structural tools, along with the advent of graph pangenomes. However, none offers a highly interactive way of exploring pangenomes for visual exploration and analysis.

During my PhD project I identified key elements for the design and development of genomic visualization tools. Useful for newcomers discovering the field of datavis, they have been summarized into ten guidelines, in the shape of a “*Ten simple rules to...*” publication currently in review in the PLOS Computational Biology journal. These ‘rules’ cover multiple parts of a visualization tool’s lifecycle, starting with the design and development phases, finishing with maintenance and community building matters.

I introduced Panache, a browser-like visualization tool for the exploration of PAV of pangenome blocks (either from pgAtlases or pangenomes) within a linear coordinate system. Panache is available as a public tool and has already been applied to public datasets (including banana and wheat), published, and cited in publications.

I finally introduced SaVanache, a design for a composite tool enabling the deep exploration of pangenome through multiple visual representations for multiple view scales: *Overall diversity*, *Structural variations*, *Presence Absence*, and *Haplotypes*. I detailed the design of novel visual representations (including a system of glyphs) and approaches for the visualization of structural variations within a pangenome graph within an integrative UI.

II. An answer

The introduction raised the following question: *What would scalable and meaningful visualization tools for (plant) pangenomes be like?*

The work presented in this dissertation provides an answer, first by identifying components of a good datavis tool for genomics in general, then by highlighting some desirable features of visual representations of pangenomes through Panache and SaVanache.

Visualization tools for pangenomics face multiple challenges, starting with the size and current lack of standards for pangenomics data. A pangenomics visualization tool could be created for pgAtlases, pangenomes, or both, though pangenomes are gaining in popularity since gene annotations can be integrated to panreferences or pangenome graphs alike.

A datavis tool can be used for various goals, with the most preeminent being for exploratory analysis, targeted analysis, or visualization for publication and general outreach. The exact goals should however be thoroughly discussed with all the people involved, from the target users to the stakeholders to correctly identify the needs and available timeframe for the construction of a tool. In pangenomics this will also highlight the category of visualization tools that should be built, depending on the data available and if their positions are of interest. Panache was created as a *positioned* visualization tool so that it could work on both pgAtlases and pangenomes, but it would not yet be usable for exploring SVs in detail. SaVanache has therefore been designed to enable the deep exploration of multi-scale pangenome data.

A high level of interactivity within the visualization tool would make a tool scalable: following Shneiderman’s mantra (see ‘Ten Rules’ Rule 5: Make data complexity intelligible) and carefully designing visual representations for dynamic display rather than static infographics would enable

a good exploration of the underlying data, providing plenty of details when needed and asked for by users. Multiple visual representations can be imagined for one concept, and a visualization tool could include one or more, as done in SaVanache. As pangenomes come in multiple shapes and sizes, providing multiple visual representations would cover different user needs, making the tool even more scalable. Moreover, a good visualization tool should be kept in good health: up to date documentation, maintenance and regular exchange with the target community are essential in keeping a tool relevant.

Visually representing pangenomes meaningfully could not be done without providing some layers of summary information, to help users to make sense of these big data and identify patterns. Particularly, the overall presence / absence status of the genomic items contained is of interest, especially when position dependent. *Composite* visualization tools with multiple connected visual representations for different levels of details, similarly to a semantic zoom, would offer in-depth analysis capacities.


Besides, pangenomes are abstract concepts, useful for shifting away from the single reference genome paradigm, but their real interest lies in the genomes that compose them. A linear coordinate system should still be available, to connect the visualization to existing analyses from the single reference era and to stay anchored into the biological reality of DNA sequences. Linearity also has the advantage of being easily readable, enabling comparisons between sub elements. To take full advantage of pangenomes, systems to change the coordinate system of the layouts should be available too. I propose SaVanache and its multiple views as the first design of such a composite visualization tool for pangenomes.

III. Discussion

Multiple visual representations of a same concept can coexist and often offer multiple complementary viewpoints. Certain visual representations are better fit than others for certain tasks, but there is no absolute best that could be used for everything and please everyone. This is especially true for pangenomes, which cover a wide range of definitions, data, formats, and use cases.

Due to their only recent popularization, pangenomes are not yet well integrated into genomic analysis workflows, and the pool of experienced target users is therefore limited. Panache instances deployed by other research teams already proved the interest of this tool to the scientific community and highlighted missing desirable features to include. Still, more widescale discussions and validation steps during hands-on sessions are necessary for a better assessment of the tools' (Panache and SaVanache alike) strengths and weaknesses for a wider audience. Validation should focus on both the efficiency of proposed visual representations to accomplish certain tasks and on the UI in which these visual representations are displayed. Evaluation of the 'intuitiveness' of the chosen visual representations is especially challenging. Once they are first introduced to target users, a learning factor biases the answers, and it is better to survey different people each time. This proves difficult with pangenomes, as there are few experienced target users that have the knowledge needed to understand the concepts involved.

Identified improvements to the tools described in this dissertation include the update of Panache precomputed linear coordinate system towards a dynamically chosen one. Enabling users to customize the views with colors and PAV categories of their choice, along with the possibility to extract visualizations in high quality images for publications would also expand the usability of the tools to additional use cases. Aside from the tools' designs, efforts should also be made to optimize the current implementations: a better handling of stored data and faster display technologies like WebGL would bring improvements to the UX, reducing the observed delays between user interactions and display updates.



Moreover, as the adoption of novel visual representations can be difficult, tutorials and legends should be developed for a smoother transition from single reference workflows. As a tool can display multiple visual representations, it would also be possible to partly connect novel and classic visualizations (like usual genome browsers) for an easier adoption.

Besides, as the pangenome visualization tools had to be usable by different research organisms, with each its own pipeline for generating files, the file formats proposed for both Panache and SaVanache are *ad hoc* formats, not directly available from pangenome building tools yet. A better connection to existing pangenome creation tools, like minigraph [113] or the pangenome graph builder [118], would enable a wide adoption of the visualization tools.

IV. Food for thoughts

This PhD raised questions related to other fields that would be worth exploring.

Datavis builds upon varied principles but is user-centered in the end. A key marker of a visualization tool's success is its adoption by a community, but too much novelty can be a barrier as the learning process is expensive in both time and energy. Tool intuitiveness could be a good first way to ease that adoption process but studying the exact drivers of resistance to innovation / reluctance towards novelty, especially in the case of datavis, would help building better tools.

Interactions between science and video games are gaining attention, with studies on the effect of gamification (i.e. the introduction of playful principles within scientific software). Video games and board games are known for including detailed and accessible interfaces (dynamic or static, respectively). The application of similar principles to both UI/UX design and processing of highly detailed images within visualization tools would be of interest.

Moreover, Panache and SaVanache have been designed for computer screens, which are common tools in science. The popularization of new technologies like Augmented Reality or Virtual Reality in science would offer a new playground for datavis, offering alternative ways to display and use visualization tools. Whether they would be more efficient than usual two-dimensional displays remains to be seen. Possibilities of 'pocket datavis', available from smartphones, could also be an interesting evolution of the way visual analysis is done. One could for example use their smartphone as a secondary screen to access details from a main visualization available on another display, as a physical division of Shneiderman's mantra.

Datavis as a scientific field also deserves to be better known and understood by scientists. The concepts behind visual communication are essential for many aspects of a scientist's life, from building a slide deck for an oral presentation to designing posters or even creating figures for publication within a scientific article. As expecting everyone to be well-versed in datavis seems preposterous, it would be interesting to see the development of teams of datavis people within research institutes, able to help others to publish their results and spread their research to an uninitiated audience.

Finally, pangenomes are not widely in use yet but genomics and bioinformatics in general evolve fast. The evolution of that field in the coming years might raise new questions and needs worth investigating, especially regarding the availability of multiple pangenomes instead of a single pangenome. There are already studies about pgAtlases, pangenomes, and pantranscriptomes; would pangenomes containing absolutely all the available information be in use or would there be different pangenomes built for different purposes instead?

V. Perspectives

A matter of crucial importance for pangenomics will be the creation of a harmonized ecosystem of files, for fast and efficient analysis. Though pangenome graphs are gaining popularity, there is

no clear standard yet as the main pangenome graph building tools use different file formats. GFA for pangenomes is still maturing, with different coexisting versions (GFAv1, GFAv1+, rGFA, GFAv2) and its current structure is not optimized for efficient parsing. Moreover, it should be compatible with other usual formats from bioinformatics, to be linkable to previous studies and analyses without having to rebuild everything. The JSON structure introduced for SaVanache contains enough information for the associated visual representation of SVs but does not contain important information like the actual DNA sequences or annotations and could therefore not act as a universal file format for pangenomes.

With the development of tools for the creation of pangenomes we can envision for the near future improved data formats and structures, as well as conversion tools. Efforts like the creation of a **Resource Description Framework (RDF)** for pangenome graphs⁵⁷ could pave the way towards such an integrated solution. Moreover, the development of dedicated databases would also enable the storage and fast parsing of files for visualization tool, instead of storing everything client-side as it is currently the case with Panache and SaVanache. More easily available pangenomes would also mean a better adoption of pangenome-related workflows, enabling the better identification of missing features and validation of pangenome visualization tools.

The democratization of pangenomes would also lead to the exploration of new approaches, such as comparisons of pangenomes [277], or even metapangenomics [203], in addition to the already existing pantranscriptomics approaches [68, 278], opening new visualization challenges.

VI. Personal conclusion

Working on Panache and SaVanache led to the ten rules from Chapter 1, which gathered advice that I would have wanted to hear when I first started this PhD. Involving others early on and iterated communication in general is of utmost importance for the construction of a datavis tool. Being at the frontier of multiple fields and people, it is necessary for datavis designers to communicate enough, for both understanding others and being understood by others. For example, identifying the needs for visualizing SVs within pangenomes took multiple discussions with target users. Explaining to developers what should be developed also took a consequent amount of time and preparation—especially while working remotely—that was not expected.

Isolation during the COVID crisis was particularly challenging, especially as I had limited direct contacts with my colleagues and no contact with datavis scientists working on similar subjects. I would advise future PhD students working on similar subjects and their supervisors to contact datavis research labs beforehand.


Moreover, all developers did not have the same types of contracts, and I find working with long term contracts (about 2 months minimum) more comfortable than short-term contracts extended on a day-to-day basis. Longer contracts leave more time to explain the project in detail and to plan the distribution of development, with better defined milestones and well-thought steps.

Data play a huge role in scientific datavis, and as such working on pangenomes has proved difficult. The lack of standard ended in a great freedom for defining the formats and content to use, which raised questions about how these files could be created by the different organisms involved in this PhD, and outsiders. I worked on the creation of these data in parallel, but their lack of wide availability ended in few validation steps for the chosen visual encodings.

This PhD was an opportunity to widen my skillset, as I learned plenty on datavis concepts and techniques, and even web development. I also supervised a total of seven people with a wide range of backgrounds (UI/UX specialist, junior and senior web developers, and a web dev intern), collaborated to online hackathon⁵⁸, wrote and reviewed papers, and shared my work through

⁵⁷ <https://publikationen.bibliothek.kit.edu/1000127608>

⁵⁸ <https://github.com/virtual-biohackathons/covid-19-bh20>



posters and oral presentations on multiple occasions, including international conferences—though most of them were online. Moreover, as part of a CIFRE PhD I had working experience from three different organisms: Syngenta as a private company, the IRD as a public research institute, and Bioversity as an international organization, each with its own goals.

References

1. Stephens, Z.D., et al., *Big Data: Astronomical or Genomical?* PLOS Biology, 2015. **13**(7): p. e1002195.
2. Birney, E. and N. Soranzo, *The end of the start for population sequencing.* Nature, 2015. **526**(7571): p. 52-53.
3. The 3000 rice genomes project, *The 3,000 rice genomes project.* GigaScience, 2014. **3**(1): p. 7.
4. The Anopheles gambiae 1000 Genomes Consortium, *Genetic diversity of the African malaria vector Anopheles gambiae.* Nature, 2017. **552**(7683): p. 96-100.
5. Yang, X., et al., *One reference genome is not enough.* Genome Biology, 2019. **20**(1).
6. Schatz, M.C., et al., *Whole genome de novo assemblies of three divergent strains of rice, Oryza sativa, document novel gene space of aus and indica.* Genome Biology, 2014. **15**(11): p. 506.
7. Tettelin, H., et al., *Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: implications for the microbial "pan-genome".* Proceedings of the National Academy of Sciences of the United States of America, 2005. **102**(39): p. 13950-13955.
8. Wang, T., et al., *The Human Pangenome Project: a global resource to map genomic diversity.* Nature, 2022. **604**(7906): p. 437-446.
9. Computational Pan-Genomics Consortium, *Computational pan-genomics: status, promises and challenges.* Brief Bioinform, 2018. **19**(1): p. 118-135.
10. Bayer, P.E., et al., *Plant pan-genomes are the new reference.* Nature Plants, 2020. **6**(8): p. 914-920.
11. Bezuidt, O.K., et al., *The Geobacillus Pan-Genome: Implications for the Evolution of the Genus.* Frontiers in microbiology, 2016. **7**: p. 723-723.
12. Guimarães, L.C., et al., *Inside the Pan-genome - Methods and Software Overview.* Current genomics, 2015. **16**(4): p. 245-252.
13. Vernikos, G., et al., *Ten years of pan-genome analyses.* Current Opinion in Microbiology, 2015. **23**: p. 148-154.
14. D'Auria, G., et al., *Legionella pneumophila pangenome reveals strain-specific virulence factors.* BMC Genomics, 2010. **11**(1): p. 181.
15. Ernst, C. and S. Rahmann, *PanCake: A Data Structure for Pangenomes,* in *German Conference on Bioinformatics 2013*, T. Beißbarth, et al., Editors. 2013, Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik, p. 35-45.
16. Golicz, A.A., J. Batley, and D. Edwards, *Towards plant pangenomics.* Plant Biotechnol J, 2016. **14**(4): p. 1099-105.
17. Montenegro, J.D., et al., *The pangenome of hexaploid bread wheat.* the Plant Journal, 2017. **90**(5): p. 1007-1013.
18. Tranchant-Dubreuil, C., M. Rouard, and F. Sabot, *Plant Pangenome: Impacts on Phenotypes and Evolution.* Annual Plant Reviews online, 2019(2).
19. Eizenga, J.M., et al., *Pangenome Graphs.* Annual Review of Genomics and Human Genetics, 2020. **21**(1): p. 139-162.
20. Rijzaani, H., et al., *The pangenome of banana highlights differences between genera and genomes.* The Plant Genome, 2021. **n/a**(n/a): p. e20100.
21. Golicz, A.A., et al., *The pangenome of an agronomically important crop plant Brassica oleracea.* Nature Communications, 2016. **7**(1): p. 13390.
22. Tettelin, H. and D. Medini, *The Pangenome - Diversity, Dynamics and Evolution of Genomes.* 1 ed, ed. H. Tettelin and D. Medini. 2020: Springer Cham. XIV, 307.
23. Mabry, M.E., et al., *The Evolutionary History of Wild, Domesticated, and Feral Brassica oleracea (Brassicaceae).* Mol Biol Evol, 2021. **38**(10): p. 4419-4434.
24. Cornille, A., et al., *New insight into the history of domesticated apple: secondary contribution of the European wild apple to the genome of cultivated varieties.* PLoS Genet, 2012. **8**(5): p. e1002703.

25. Groppi, A., et al., *Population genomics of apricots unravels domestication history and adaptive events*. Nature Communications, 2021. **12**(1): p. 3956.
26. Sun, L., et al., *Origin of the Domesticated Horticultural Species and Molecular Bases of Fruit Shape and Size Changes during the Domestication, Taking Tomato as an Example*. Horticultural Plant Journal, 2017. **3**(3): p. 125-132.
27. Shastry, B.S., *SNPs: impact on gene function and phenotype*. Methods Mol Biol, 2009. **578**: p. 3-22.
28. Montgomery, S.B., et al., *The origin, evolution, and functional impact of short insertion-deletion variants identified in 179 human genomes*. Genome Res, 2013. **23**(5): p. 749-61.
29. Robert, F. and J. Pelletier, *Exploring the Impact of Single-Nucleotide Polymorphisms on Translation*. Frontiers in Genetics, 2018. **9**.
30. Lin, M., et al., *Effects of short indels on protein structure and function in human genomes*. Sci Rep, 2017. **7**(1): p. 9313.
31. Ho, S.S., A.E. Urban, and R.E. Mills, *Structural variation in the sequencing era*. Nature reviews. Genetics, 2020. **21**(3): p. 171-189.
32. Sankoff, D., *Rearrangements and chromosomal evolution*. Current Opinion in Genetics & Development, 2003. **13**(6): p. 583-587.
33. Stewart, N.B. and R.L. Rogers, *Chromosomal rearrangements as a source of new gene formation in Drosophila yakuba*. PLoS Genet, 2019. **15**(9): p. e1008314.
34. Xia, Y., et al., *The Origin and Evolution of Chromosomal Reciprocal Translocation in Quasipaa boulengeri (Anura, Dicoglossidae)*. Frontiers in Genetics, 2020. **10**.
35. Freeman, J.L., et al., *Copy number variation: new insights in genome diversity*. Genome Res, 2006. **16**(8): p. 949-61.
36. Springer, N.M., et al., *Maize inbreds exhibit high levels of copy number variation (CNV) and presence/absence variation (PAV) in genome content*. PLoS Genet, 2009. **5**(11): p. e1000734.
37. Díaz, A., et al., *Copy number variation affecting the Photoperiod-B1 and Vernalization-A1 genes is associated with altered flowering time in wheat (Triticum aestivum)*. PLoS One, 2012. **7**(3): p. e33234.
38. Cook, D.E., et al., *Copy number variation of multiple genes at Rhg1 mediates nematode resistance in soybean*. Science, 2012. **338**(6111): p. 1206-9.
39. Tao, Y., et al., *Exploring and Exploiting Pan-genomics for Crop Improvement*. Molecular Plant, 2019. **12**(2): p. 156-169.
40. Hurgobin, B., et al., *Homoeologous exchange is a major cause of gene presence/absence variation in the amphidiploid Brassica napus*. Plant Biotechnology Journal, 2018. **16**(7): p. 1265-1274.
41. Gan, X., et al., *Multiple reference genomes and transcriptomes for Arabidopsis thaliana*. Nature, 2011. **477**(7365): p. 419-423.
42. Song, J.-M., et al., *Eight high-quality genomes reveal pan-genome architecture and ecotype differentiation of Brassica napus*. Nature Plants, 2020. **6**(1): p. 34-45.
43. Monnahan, P.J., et al., *Using multiple reference genomes to identify and resolve annotation inconsistencies*. BMC Genomics, 2020. **21**(1): p. 281.
44. Rouli, L., et al., *The bacterial pangenome as a new tool for analysing pathogenic bacteria*. New Microbes and New Infections, 2015. **7**: p. 72-85.
45. Golicz, A.A., et al., *Pangenomics Comes of Age: From Bacteria to Plant and Animal Applications*. Trends in Genetics, 2019.
46. Sherman, R.M., et al., *Assembly of a pan-genome from deep sequencing of 910 humans of African descent*. Nature Genetics, 2019. **51**(1): p. 30-35.
47. Miga, K.H. and T. Wang, *The Need for a Human Pangenome Reference Sequence*. Annual Review of Genomics and Human Genetics, 2021. **22**(1): p. 81-102.
48. Danilevicz, M.F., et al., *Plant pangenomics: approaches, applications and advancements*. Current Opinion in Plant Biology, 2020. **54**: p. 18-25.
49. Sherman, R.M. and S.L. Salzberg, *Pan-genomics in the human genome era*. Nature Reviews Genetics, 2020. **21**(4): p. 243-254.

50. Jayakodi, M., et al., *Building pan-genome infrastructures for crop plants and their use in association genetics*. DNA Research, 2021. **28**(1).
51. Tay Fernandez, C.G., et al., *Pangenomes as a Resource to Accelerate Breeding of Under-Utilised Crop Species*. International Journal of Molecular Sciences, 2022. **23**(5): p. 2671.
52. Zhao, Q., et al., *Pan-genome analysis highlights the extent of genomic variation in cultivated and wild rice*. Nature Genetics, 2018. **50**(2): p. 278-284.
53. Monat, C., et al., *De Novo Assemblies of Three Oryza glaberrima Accessions Provide First Insights about Pan-Genome of African Rices*. Genome Biol Evol, 2017. **9**(1): p. 1-6.
54. Li, H., et al., *Graph-based pan-genome reveals structural and sequence variations related to agronomic traits and domestication in cucumber*. Nature Communications, 2022. **13**(1): p. 682.
55. Gao, L., et al., *The tomato pan-genome uncovers new genes and a rare allele regulating fruit flavor*. Nature Genetics, 2019. **51**(6): p. 1044-1051.
56. Zhou, Y., et al., *Graph pangenome captures missing heritability and empowers tomato breeding*. Nature, 2022.
57. Li, Y.-h., et al., *De novo assembly of soybean wild relatives for pan-genome analysis of diversity and agronomic traits*. Nature Biotechnology, 2014. **32**(10): p. 1045-1052.
58. Sun, Y., et al., *Pan-Genome Analysis Reveals the Abundant Gene Presence/Absence Variations Among Different Varieties of Melon and Their Influence on Traits*. Frontiers in Plant Science, 2022. **13**.
59. Khan, A.W., et al., *Super-Pangenome by Integrating the Wild Side of a Species for Accelerated Crop Improvement*. Trends in Plant Science, 2020. **25**(2): p. 148-158.
60. Liu, Y., et al., *Pan-Genome of Wild and Cultivated Soybeans*. Cell, 2020. **182**(1): p. 162-176.e13.
61. Tao, Y., et al., *Extensive variation within the pan-genome of cultivated and wild sorghum*. Nature Plants, 2021. **7**(6): p. 766-773.
62. Crysanto, D. and H. Pausch, *Bovine breed-specific augmented reference graphs facilitate accurate sequence read mapping and unbiased variant discovery*. Genome Biology, 2020. **21**(1): p. 184.
63. Leonard, A.S., et al., *Structural variant-based pangenome construction has low sensitivity to variability of haplotype-resolved bovine assemblies*. Nature Communications, 2022. **13**(1): p. 3012.
64. Kim, Y., et al., *Current status of pan-genome analysis for pathogenic bacteria*. Current Opinion in Biotechnology, 2020. **63**: p. 54-62.
65. Medini, D., et al., *The Pangenome: A Data-Driven Discovery in Biology*, in *The Pangenome: Diversity, Dynamics and Evolution of Genomes*, H. Tettelin and D. Medini, Editors. 2020, Springer International Publishing: Cham. p. 3-20.
66. Ahnert, S.E., T.M.A. Fink, and A. Zinovyev, *How much non-coding DNA do eukaryotes require?* Journal of Theoretical Biology, 2008. **252**(4): p. 587-592.
67. Elliott, T.A. and T.R. Gregory, *What's in a genome? The C-value enigma and the evolution of eukaryotic genome content*. 2015. **370**(1678): p. 20140331.
68. Hirsch, C.N., et al., *Insights into the maize pan-genome and pan-transcriptome*. The Plant cell, 2014. **26**(1): p. 121-135.
69. AgBioData Consortium. *Pan-genome viewers (May 2020)*. [online video] 2020 [cited 2022 June 16]; Available from: <https://www.youtube.com/watch?v=ATPzVlrTW0s>.
70. Gautreau, G., et al., *PPanGGOLiN: Depicting microbial diversity via a partitioned pangenome graph*. PLOS Computational Biology, 2020. **16**(3): p. e1007732.
71. Hübner, S., et al., *Sunflower pan-genome analysis shows that hybridization altered gene content and disease resistance*. Nature Plants, 2019. **5**(1): p. 54-62.
72. Gordon, S.P., et al., *Extensive gene content variation in the Brachypodium distachyon pan-genome correlates with population structure*. Nature Communications, 2017. **8**(1): p. 2184.
73. Michael, T.P., *Plant genome size variation: bloating and purging DNA*. Briefings in Functional Genomics, 2014. **13**(4): p. 308-317.

74. Wendel, J.F., et al., *Evolution of plant genome architecture*. Genome Biology, 2016. **17**(1): p. 37.
75. Nagaharu, U., *Genome Analysis in Brassica with Special Reference to the Experimental Formation of B. Napus and Peculiar Mode of Fertilization*. Japanese Journal of Botany, 1935. **7**: p. 389-452.
76. Xue, J.Y., et al., *Maternal Inheritance of U's Triangle and Evolutionary Process of Brassica Mitochondrial Genomes*. Front Plant Sci, 2020. **11**: p. 805.
77. D'Hont, A., et al., *The interspecific genome structure of cultivated banana, Musa spp. revealed by genomic DNA in situ hybridization*. Theoretical and Applied Genetics, 2000. **100**(2): p. 177-183.
78. Mehrotra, S. and V. Goyal, *Repetitive sequences in plant nuclear DNA: types, distribution, evolution and function*. Genomics Proteomics Bioinformatics, 2014. **12**(4): p. 164-71.
79. Feschotte, C., N. Jiang, and S.R. Wessler, *Plant transposable elements: where genetics meets genomics*. Nature Reviews Genetics, 2002. **3**(5): p. 329-341.
80. Sahebi, M., et al., *Contribution of transposable elements in the plant's genome*. Gene, 2018. **665**: p. 155-166.
81. Quesneville, H., *Twenty years of transposable element analysis in the Arabidopsis thaliana genome*. Mobile DNA, 2020. **11**(1): p. 28.
82. Huang, K. and L.H. Rieseberg, *Frequency, Origins, and Evolutionary Role of Chromosomal Inversions in Plants*. Frontiers in Plant Science, 2020. **11**.
83. Darling, A.C.E., et al., *Mauve: Multiple Alignment of Conserved Genomic Sequence With Rearrangements*. Genome Research, 2004. **14**(7): p. 1394-1403.
84. Darling, A.E., B. Mau, and N.T. Perna, *progressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement*. PLOS ONE, 2010. **5**(6): p. e11147.
85. Minkin, I. and P. Medvedev, *Scalable multiple whole-genome alignment and locally collinear block construction with SibeliaZ*. Nature Communications, 2020. **11**(1): p. 6327.
86. Glen, S., *Graph Theory: Definitions for Common Terms*, in *StatisticsHowTo.com: Elementary Statistics for the rest of us!* 2017.
87. Compeau, P.E., P.A. Pevzner, and G. Tesler, *Why are de Bruijn graphs useful for genome assembly?* Nat Biotechnol, 2011. **29**(11): p. 987-91.
88. Carletti, V., et al. *Graph-Based Representations for Supporting Genome Data Analysis and Visualization: Opportunities and Challenges*. in *Graph-Based Representations in Pattern Recognition*. 2019. Cham: Springer International Publishing.
89. Paten, B., et al., *Genome graphs and the evolution of genome inference*. Genome research, 2017. **27**(5): p. 665-676.
90. Garrison, E., et al., *Variation graph toolkit improves read mapping by representing genetic variation in the reference*. Nature Biotechnology, 2018. **36**(9): p. 875-879.
91. Garrison, E.P., *Graphical pangenomics*. 2019, University of Cambridge. p. 199.
92. Muggli, M.D., et al., *Succinct colored de Bruijn graphs*. Bioinformatics, 2017. **33**(20): p. 3181-3187.
93. Muggli, M.D., B. Alipanahi, and C. Boucher, *Building large updatable colored de Bruijn graphs via merging*. Bioinformatics, 2019. **35**(14): p. i51-i60.
94. Holley, G. and P. Melsted, *Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs*. Genome Biology, 2020. **21**(1): p. 249.
95. Li, H., X. Feng, and C. Chu, *The design and construction of reference pangenome graphs*. arXiv e-prints, 2020: p. arXiv:2003.06079.
96. Hu, Z., C. Wei, and Z. Li, *Computational Strategies for Eukaryotic Pangenome Analyses*, in *The Pangenome: Diversity, Dynamics and Evolution of Genomes*, H. Tettelin and D. Medini, Editors. 2020, Springer International Publishing: Cham. p. 293-307.
97. Xiao, J., et al., *A Brief Review of Software Tools for Pangenomics*. Genomics, Proteomics & Bioinformatics, 2015. **13**(1): p. 73-76.
98. Vernikos, G.S., *A Review of Pangenome Tools and Recent Studies*, in *The Pangenome: Diversity, Dynamics and Evolution of Genomes*, H. Tettelin and D. Medini, Editors. 2020, Springer International Publishing: Cham. p. 89-112.

99. Chaudhari, N.M., V.K. Gupta, and C. Dutta, *BPGA- an ultra-fast pan-genome analysis pipeline*. Scientific Reports, 2016. **6**(1): p. 24373.
100. Ozer, E.A., *ClustAGE: a tool for clustering and distribution analysis of bacterial accessory genomic elements*. BMC Bioinformatics, 2018. **19**(1): p. 150.
101. Contreras-Moreira, B. and P. Vinuesa, *GET_HOMOLOGUES, a versatile software package for scalable and robust microbial pangenome analysis*. Appl Environ Microbiol, 2013. **79**(24): p. 7696-701.
102. Contreras-Moreira, B., et al., *Analysis of Plant Pan-Genomes and Transcriptomes with GET_HOMOLOGUES-EST, a Clustering Solution for Sequences of the Same Species*. Front Plant Sci, 2017. **8**: p. 184.
103. Sahl, J.W., et al., *The large-scale blast score ratio (LS-BSR) pipeline: a method to rapidly compare genetic content between bacterial genomes*. PeerJ, 2014. **2**: p. e332.
104. Li, L., C.J. Stoeckert, Jr., and D.S. Roos, *OrthoMCL: identification of ortholog groups for eukaryotic genomes*. Genome Res, 2003. **13**(9): p. 2178-89.
105. Perrin, A. and E.P.C. Rocha, *PanACoTA: a modular tool for massive microbial comparative genomics*. NAR Genom Bioinform, 2021. **3**(1): p. lqaa106.
106. Fouts, D.E., et al., *PanOCT: automated clustering of orthologs using conserved gene neighborhood for pan-genomic analysis of bacterial strains and closely related species*. Nucleic acids research, 2012. **40**(22): p. e172-e172.
107. Zhao, Y., et al., *PGAP: pan-genomes analysis pipeline*. Bioinformatics, 2012. **28**(3): p. 416-418.
108. Page, A.J., et al., *Roary: rapid large-scale prokaryote pan genome analysis, core vs accessory*. Bioinformatics, 2015. **31**(22): p. 3691-3693.
109. Treangen, T.J., et al., *The Harvest suite for rapid core-genome alignment and visualization of thousands of intraspecific microbial genomes*. Genome biology, 2014. **15**(11): p. 524-524.
110. Angiuoli, S.V. and S.L. Salzberg, *Mugsy: fast multiple alignment of closely related whole genomes*. Bioinformatics, 2010. **27**(3): p. 334-342.
111. Jandrasits, C., et al., *seq-seq-pan: building a computational pan-genome data structure on whole genome alignment*. BMC Genomics, 2018. **19**(1).
112. Paten, B., et al., *Cactus graphs for genome comparisons*. J Comput Biol, 2011. **18**(3): p. 469-81.
113. Li, H., X. Feng, and C. Chu, *The design and construction of reference pangenome graphs with minigraph*. Genome Biology, 2020. **21**(1): p. 265.
114. Biederstedt, E., et al., *NovoGraph: Human genome graph construction from multiple long-read de novo assemblies*. F1000Research, 2018. **7**: p. 1391-1391.
115. Warren, A.S., et al., *Panaconda: Application of pan-synteny graph models to genome content analysis*. bioRxiv, 2017: p. 215988.
116. Noll, N., M. Molari, and R.A. Neher, *PanGraph: scalable bacterial pan-genome graph construction*. bioRxiv, 2022: p. 2022.02.24.481757.
117. Sheikhzadeh, S., et al., *PanTools: representation, storage and exploration of pan-genomic data*. Bioinformatics, 2016. **32**(17): p. i487-i493.
118. Garrison, E., et al. *pggb - the pangenome graph builder*. 2020 [cited 2020 September]; Available from: <https://github.com/pangenome/pggb>.
119. Armstrong, J., et al., *Progressive Cactus is a multiple-genome aligner for the thousand-genome era*. Nature, 2020. **587**(7833): p. 246-251.
120. Garrison, E. and A. Guarracino, *Unbiased pangenome graphs*. bioRxiv, 2022: p. 2022.02.14.480413.
121. Marcus, S., H. Lee, and M.C. Schatz, *SplitMEM: a graphical algorithm for pan-genome analysis with suffix skips*. Bioinformatics, 2014. **30**(24): p. 3476-83.
122. Hickey, G., et al., *Genotyping structural variants in pangenome graphs using the vg toolkit*. Genome Biology, 2020. **21**(1): p. 35.
123. Dawson, E.T. *svaha2: linear time, low-memory construction of variation graphs*. 2019 [cited 2019; Available from: <https://github.com/edawson/svaha2>.
124. Guarracino, A., et al., *ODGI: understanding pangenome graphs*. Bioinformatics, 2022.

125. Limasset, A., et al., *Read mapping on de Bruijn graphs*. BMC Bioinformatics, 2016. **17**(1): p. 237.
126. Heydari, M., et al., *BrownieAligner: accurate alignment of Illumina sequencing data to de Bruijn graphs*. BMC Bioinformatics, 2018. **19**(1): p. 311.
127. Liu, B., et al., *deBGA: read alignment with de Bruijn graph-based seed and extension*. Bioinformatics, 2016. **32**(21): p. 3224-3232.
128. Sirén, J., et al., *Haplotype-aware graph indexes*. Bioinformatics, 2019. **36**(2): p. 400-407.
129. Sirén, J., *Indexing Variation Graphs*. 2017 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX), 2017: p. 13-27.
130. Schneeberger, K., et al., *Simultaneous alignment of short reads against multiple genomes*. Genome Biol, 2009. **10**(9): p. R98.
131. Rautiainen, M. and T. Marschall, *GraphAligner: rapid and versatile sequence-to-graph alignment*. Genome Biol, 2020. **21**(1): p. 253.
132. Gonnella, G. and S. Kurtz, *GfaPy: a flexible and extensible software library for handling sequence graphs in Python*. Bioinformatics, 2017. **33**(19): p. 3094-3095.
133. Li, H. *gfatools*. 2020 [cited 2020; Available from: <https://github.com/lh3/gfatools>].
134. Kim, D., et al., *Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype*. Nat Biotechnol, 2019. **37**(8): p. 907-915.
135. Schulz, T., et al., *Detecting high-scoring local alignments in pangenome graphs*. Bioinformatics, 2021. **37**(16): p. 2266-2274.
136. Vaddadi, K., R. Srinivasan, and N. Sivadasan. *Read Mapping on Genome Variation Graphs*. in *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*. 2019. Dagstuhl, Germany: Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik.
137. Niu, J., D. Denisko, and M.M. Hoffman. *The Browser Extensible Data (BED) format*. 2022 [cited 2022 June 6]; Available from: <https://github.com/samtools/hts-specs/blob/master/BEDv1.pdf>.
138. Pearson, W.R. and D.J. Lipman, *Improved tools for biological sequence comparison*. Proceedings of the National Academy of Sciences, 1988. **85**(8): p. 2444-2448.
139. Stein, L. *Generic Feature Format Version 3 (GFF3)*. 2006 [cited 2022 June 6]; Available from: <https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md>.
140. Danecek, P., et al., *The variant call format and VCFtools*. Bioinformatics, 2011. **27**(15): p. 2156-8.
141. Crockford, D. *Introducing JSON*. 2002 [cited 2022 June 6]; Available from: <https://www.json.org/json-en.html>.
142. Hurgobin, B., et al. *Brassica napus GBrowse Viewers and Search*. 2018 [cited 2019; Available from: <http://appliedbioinformatics.com.au/gb2/gbrowse/BnapusPan/>].
143. Hennig, A., J. Bernhardt, and K. Nieselt, *Pan-Tetris: an interactive visualisation for Pan-genomes*. BMC Bioinformatics, 2015. **16**(11): p. S3.
144. Unknown. *MAF - (Multiple Alignment Format)*. 2009 [cited 2022 June 6]; Available from: <http://www.bx.psu.edu/~dcking/man/maf.xhtml>.
145. NCI Genomic Data Commons (GDC). *GDC MAF Format v.1.0.0*. n.d. [cited 2022 June 6]; Available from: https://docs.gdc.cancer.gov/Data/File_Formats/MAF_Format/.
146. Genome Evolution Laboratory. *Mauve Output File Formats*. 2009 [cited 2022 June 6]; Available from: <https://asap.genetics.wisc.edu/software/mauve/mauve-user-guide/mauve-output-file-formats.php>.
147. vgteam. *a handle-graphy reimplementaion of the XG succinct graph index*. 2019 [cited 2022 June 6]; Available from: <https://github.com/vgteam/xg>.
148. Jackman, S., et al. *GFA: Graphical Fragment Assembly (GFA) Format Specification*. 2015 [cited 2019; Available from: <https://github.com/GFA-spec/GFA-spec>].
149. Li, H., *On a reference pan-genome model*. 2019.
150. Garrison, E., *Untangling graphical pangenomics*. 2019.
151. Li, H., *On a reference pan-genome model (Part II)*. 2019.

152. The SAM/BAM Format Specification Working Group. *Sequence Alignment/Map Optional Fields Specification*. 2009 [cited 2019; Available from: <https://samtools.github.io/hts-specs/SAMtags.pdf>].
153. vgteam. *File Formats*. 2019 [cited 2022; Available from: <https://github.com/vgteam/vg/wiki/File-Formats#gam-graph-alignment--map-vgs-bam>].
154. Group, T.S.B.F.S.W. *Sequence Alignment/Map Format Specification*. 2009 [cited 2022; Available from: <http://samtools.github.io/hts-specs/SAMv1.pdf>].
155. Li, H. *The Graph Alignment Format (GAF)*. 2019 [cited 2019; Available from: <https://github.com/lh3/gfatools/blob/master/doc/rGFA.md#the-graph-alignment-format-gaf>].
156. Hickey, G., et al., *HAL: a hierarchical format for storing and analyzing multiple genome alignments*. *Bioinformatics*, 2013. **29**(10): p. 1341-2.
157. Garrison, E. and S. Heumos. *maffer*. 2019 [cited 2019]; Available from: <https://github.com/pangenome/maffer>.
158. Eren, A.M., et al. *An vi'o workflow for microbial pangenomics*. 2016 [cited 2019; Available from: <http://merenlab.org/2016/11/08/pangenomics-v2/>].
159. Pedersen, T.L., et al., *PanViz: interactive visualization of the structure of functionally annotated pangenomes*. *Bioinformatics*, 2017.
160. Ding, W., F. Baumdicker, and R.A. Neher, *panX: pan-genome analysis and exploration*. *Nucleic Acids Research*, 2018. **46**(1): p. e5-e5.
161. Cheng, G., et al., *BGDMdocker: a Docker workflow for analysis and visualization pan-genome and biosynthetic gene clusters of bacterial*. *bioRxiv*, 2017: p. 098392.
162. Zhao, Y., et al., *PanGP: A tool for quickly analyzing bacterial pan-genome profile*. *Bioinformatics*, 2014. **30**(9): p. 1297-1299.
163. Abudahab, K., et al., *PANINI: Pangenome Neighbour Identification for Bacterial Populations*. 2019. **5**(4).
164. Argimón, S., et al., *Microreact: visualizing and sharing data for genomic epidemiology and phylogeography*. *Microb Genom*, 2016. **2**(11): p. e000093.
165. Blom, J., et al., *EDGAR 2.0: an enhanced software platform for comparative gene content analyses*. *Nucleic Acids Research*, 2016. **44**(W1): p. W22-W28.
166. Bastian, M., S. Heymann, and M. Jacomy, *Gephi: An Open Source Software for Exploring and Manipulating Networks*. 2009. 2009.
167. Yuvaraj, I., et al., *PanGeT: Pan-genomics tool*. *Gene*, 2017. **600**: p. 77-84.
168. Pantoja, Y., et al., *PanWeb: A web interface for pan-genomic analysis*. *PLOS ONE*, 2017. **12**(5): p. e0178154.
169. Pedersen, T.L., *PanVizGenerator: Generate PanViz visualisations from your pangenome*. 2016.
170. Clarke, T.H., et al., *PanACEA: a bioinformatics tool for the exploration and visualization of bacterial pan-chromosomes*. *BMC Bioinformatics*, 2018. **19**(1).
171. Zhao, Y., et al., *PGAP-X: extension on pan-genome analysis pipeline*. *BMC genomics*, 2018. **19**(Suppl 1): p. 36-36.
172. Diesh, C.M. *awesome-genome-visualization*. 2021 [cited 2022 May 6]; Available from: <https://cmdcolin.github.io/awesome-genome-visualization>.
173. Eren, A.M., et al., *Anvi'o: an advanced analysis and visualization platform for 'omics data*. *PeerJ*, 2015. **3**: p. e1319.
174. Wick, R.R., et al., *Bandage: interactive visualization of de novo genome assemblies*. *Bioinformatics*, 2015. **31**(20): p. 3350-3352.
175. Sullivan, M.J. and H. van Bakel, *Chromatiblock: scalable whole-genome visualization of structural differences in prokaryotes*. 2019: p. 800920.
176. Whelan, F.J., M. Rusilowicz, and J.O. McInerney, *Coinfinder: detecting significant associations and dissociations in pangenomes*. *Microb Genom*, 2020. **6**(3).
177. Brinton, J., et al., *A haplotype-led approach to increase the precision of wheat breeding*. *Communications Biology*, 2020. **3**(1): p. 712.

178. Shannon, P., et al., *Cytoscape: a software environment for integrated models of biomolecular interaction networks*. *Genome Res*, 2003. **13**(11): p. 2498-504.
179. Cleary, A. and A. Farmer, *Genome Context Viewer: visual exploration of multiple annotated genomes using microsynteny*. *Bioinformatics*, 2018. **34**(9): p. 1562-1564.
180. Herbig, A., et al., *GenomeRing: alignment visualization based on SuperGenome coordinates*. *Bioinformatics (Oxford, England)*, 2012. **28**(12): p. i7-i15.
181. Fischer, C. *gfaestus - Vulkan-accelerated GFA visualization*. 2021 [cited 2022; Available from: <https://github.com/chfi/gfaestus>].
182. Gonnella, G., N. Niehus, and S. Kurtz, *GfaViz: flexible and interactive visualization of GFA sequence graphs*. *Bioinformatics*, 2018. **35**(16): p. 2853-2855.
183. Diesh, C.M. *graphgenomeviewer*. 2022 [cited 2022; Available from: <https://github.com/cmdcolin/graphgenomeviewer/>].
184. Pedersen, T.L., *Hierarchical sets: analyzing pangenome structure through scalable set visualizations*. *Bioinformatics*, 2017. **33**(11): p. 1604-1612.
185. Kuznetsov, M., et al. *The Immersive Graph Genome Explorer: Navigating Genomics in Immersive Virtual Reality*. in *2021 IEEE 9th International Conference on Serious Games and Applications for Health (SeGAH)*. 2021.
186. Yokoyama, T.T., et al., *MoMI-G: modular multi-scale integrated genome graph browser*. *BMC Bioinformatics*, 2019. **20**(1): p. 548.
187. neo4j. *neo4j Graph Data Platform*. n. d. [cited 2022; Available from: <https://neo4j.com/>].
188. Durant, É., et al., *Panache: a Web Browser-Based Viewer for Linearized Pangenomes*. 2021: p. 2021.04.27.441597.
189. Yuan, Y. *PanGraphViewer -- show pangenome graphs in an easy way*. 2021 [cited 2021 August]; Available from: <https://github.com/TF-Chan-Lab/panGraphViewer>.
190. Seaman, J.D. *pantograph: pangenome graph browser for SARS-CoV-2*. 2020 [cited 2020; Available from: <https://graph-genome.github.io/>].
191. Liang, Q. and S. Lonardi, *Reference-agnostic representation and visualization of pangenomes*. *BMC Bioinformatics*, 2021. **22**(1): p. 502.
192. Hadfield, J., et al., *Phandango: an interactive viewer for bacterial population genomics*. *Bioinformatics*, 2017. **34**(2): p. 292-293.
193. Goel, M. and K. Schneeberger, *plotsr: Visualising structural similarities and rearrangements between multiple genomes*. *Bioinformatics*, 2022.
194. Beyer, W., et al., *Sequence tube maps: making graph genomes intuitive to commuters*. *Bioinformatics*, 2019.
195. Kumagai, M., et al., *TASUKE+: a web-based platform for exploring GWAS results and large-scale resequencing data*. *DNA Research*, 2019. **26**(6): p. 445-452.
196. Lex, A., et al., *UpSet: Visualization of Intersecting Sets*. *IEEE Transactions on Visualization and Computer Graphics*, 2014. **20**(12): p. 1983-1992.
197. Peng, Y., et al., *MetaPGN: a pipeline for construction and graphical visualization of annotated pangenome networks*. *GigaScience*, 2018. **7**(11).
198. Beier, S. and N.R. Thomson, *Panakeia - a universal tool for bacterial pangenome analysis*. *BMC Genomics*, 2022. **23**(1): p. 265.
199. Otasek, D., et al., *Cytoscape Automation: empowering workflow-based network analysis*. *Genome Biol*, 2019. **20**(1): p. 185.
200. Conway, J.R., A. Lex, and N. Gehlenborg, *UpSetR: an R package for the visualization of intersecting sets and their properties*. *Bioinformatics*, 2017. **33**(18): p. 2938-2940.
201. Gadhave, K., et al. *UpSet 2: From Prototype to Tool*. in *IEEE Information Visualization Conference - Posters (InfoVis '19)*. 2019.
202. Eren, A.M., et al., *Community-led, integrated, reproducible multi-omics with anvi'o*. *Nature Microbiology*, 2021. **6**(1): p. 3-6.
203. Delmont, T.O. and A.M. Eren, *Linking pangenomes and metagenomes: the Prochlorococcus metapangenome*. *PeerJ*, 2018. **6**: p. e4320.
204. Krzywinski, M., et al., *Circos: An information aesthetic for comparative genomics*. 2009. **19**(9): p. 1639-1645.

205. Croucher, N.J., et al., *Population genomic datasets describing the post-vaccine evolutionary epidemiology of Streptococcus pneumoniae*. Scientific Data, 2015. **2**(1): p. 150058.
206. Makendi, C., et al., *A Phylogenetic and Phenotypic Analysis of Salmonella enterica Serovar Weltevreden, an Emerging Agent of Diarrheal Disease in Tropical Regions*. PLOS Neglected Tropical Diseases, 2016. **10**(2): p. e0004446.
207. Song, J.-M., et al., *BnPIR: Brassica napus pan-genome information resource for 1689 accessions*. Plant Biotechnology Journal, 2021. **19**(3): p. 412-414.
208. Zerbino, D.R. and E. Birney, *Velvet: algorithms for de novo short read assembly using de Bruijn graphs*. Genome research, 2008. **18**(5): p. 821-829.
209. Bankevich, A., et al., *SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing*. Journal of computational biology : a journal of computational molecular cell biology, 2012. **19**(5): p. 455-477.
210. Grabherr, M.G., et al., *Full-length transcriptome assembly from RNA-Seq data without a reference genome*. Nature biotechnology, 2011. **29**(7): p. 644-652.
211. Simpson, J. *ASQG Format*. 2014 [cited 2020; Available from: <https://github.com/jts/sga/wiki/ASQG-Format>].
212. Wick, R., et al. *Bandage-NG*. 2022 [cited 2022 June 8]; Available from: <https://github.com/asl/BandageNG>].
213. Battke, F., S. Symons, and K. Nieselt, *Mayday - integrative analytics for expression data*. BMC Bioinformatics, 2010. **11**(1): p. 121.
214. Garrison, E., S. Heumos, and A. Guarracino. *odgi: optimized dynamic genome/graph implementation*. 2019 [cited 2020; Available from: <https://github.com/vgteam/odgi>].
215. ISCB. *Pantograph - Scalable Interactive Graph Genome... - Andrea Guarracino - BioVis - ISMB 2020 Posters*. [video] 2021 [cited 2022 June]; Available from: https://www.youtube.com/watch?v=-p9aL_9OGmc].
216. Loira, N., A. Zhukova, and D.J. Sherman, *Pantograph: A template-based method for genome-scale metabolic model reconstruction*. Journal of Bioinformatics and Computational Biology, 2014. **13**(02): p. 1550006.
217. Lab Automation Network. *2021 11 11 Webinar "Visual exploration of pangenomes with Pantograph"*. 2022 [cited 2022 January]; Available from: <https://www.youtube.com/watch?v=WoGF0EiInpE>].
218. Fischer, C. *waragraph - a variation graph viewer of sorts*. 2022 [cited 2022 June]; Available from: <https://github.com/chfi/waragraph>].
219. Riehmman, P., M. Hanfler, and B. Froehlich. *Interactive Sankey diagrams*. in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*. 2005.
220. Chen, X., et al., *PGAweb: A Web Server for Bacterial Pan-Genome Analysis*. Front Microbiol, 2018. **9**: p. 1910.
221. Ou, L., et al., *Pan-genome of cultivated pepper (Capsicum) and its use in gene presence-absence variation analyses*. New Phytologist, 2018. **220**(2): p. 360-363.
222. Sun, C., et al., *RPAN: rice pan-genome browser for 3000 rice genomes*. Nucleic Acids Research, 2017. **45**(2): p. 597-605.
223. Persephone Software, *26 Zea mays NAM lines in the pan-genome browser*. 2020.
224. Dongre, N., A. Verma, and I. Singh, *PanGenome Visualization with JBrowse*. 2015: J. Craig Venter Institute,.
225. Vauterin, P., et al., *Panoptes: web-based exploration of large scale genome variation data*. Bioinformatics, 2017. **33**(20): p. 3243-3249.
226. Mikheenko, A. and M. Kolmogorov, *Assembly Graph Browser: interactive visualization of assembly graphs*. Bioinformatics, 2019. **35**(18): p. 3476-3478.
227. Heumos, S. *Interactive Visualization of Genome Variation Graphs*. 2017; Available from: <https://gitlab.codenit.de/computomics/AGV>].
228. Sinha, A.U. and J. Meller, *Cinteny: flexible analysis and visualization of synteny and genome rearrangements in multiple organisms*. BMC Bioinformatics, 2007. **8**(1): p. 82.
229. Pedersen, B.S., H. Tang, and M. Freeling, *Gobe: an interactive, web-based tool for comparative genomic visualization*. Bioinformatics, 2011. **27**(7): p. 1015-1016.

230. Petkau, A., et al., *Interactive microbial genome visualization with GView*. *Bioinformatics*, 2010. **26**(24): p. 3125-3126.
231. Saeed, A.I., et al., *TM4 microarray software suite*. *Methods Enzymol*, 2006. **411**: p. 134-93.
232. Wang, Y.E., et al., *WebMeV: A Cloud Platform for Analyzing and Visualizing Cancer Genomic Data*. *Cancer Res*, 2017. **77**(21): p. e11-e14.
233. Yachdav, G., et al., *MSAViewer: interactive JavaScript visualization of multiple sequence alignments*. *Bioinformatics*, 2016. **32**(22): p. 3501-3503.
234. Tahir Ul Qamar, M., et al., *ppsPCP: a plant presence/absence variants scanner and pan-genome construction pipeline*. *Bioinformatics*, 2019.
235. Kunyavskaya, O. and A.D. Prjibelski, *SGTK: a toolkit for visualization and assessment of scaffold graphs*. *Bioinformatics*, 2019. **35**(13): p. 2303-2305.
236. Shafin, K., et al., *Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes*. *Nature Biotechnology*, 2020. **38**(9): p. 1044-1053.
237. Tuft, E.R., *The visual display of quantitative information*. 2001: Second edition. Cheshire, Conn. : Graphics Press, [2001] ©2001.
238. Inbar, O., N. Tractinsky, and J. Meyer, *Minimalism in information visualization: attitudes towards maximizing the data-ink ratio*, in *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!* 2007, Association for Computing Machinery: London, United Kingdom. p. 185-188.
239. Franconeri, S.L., et al., *The Science of Visual Data Communication: What Works*. 2021. **22**(3): p. 110-161.
240. Koffka, K., *Principles of Gestalt Psychology*. 1935: Harcourt, Brace and Company.
241. Todorovic, D., *Gestalt principles*. 2008. **3**: p. 5345.
242. Wagemans, J., et al., *A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization*. *Psychological bulletin*, 2012. **138**(6): p. 1172-1217.
243. Cairo, A., *How Charts Lie: Getting Smarter about Visual Information*. 2019: W. W. Norton & Company.
244. Hattab, G., T.-M. Rhyne, and D. Heider, *Ten simple rules to colorize biological data visualization*. *PLOS Computational Biology*, 2020. **16**(10): p. e1008259.
245. Huang, L., *Space of preattentive shape features*. *Journal of Vision*, 2020. **20**(4): p. 10-10.
246. Sedlmair, M., M. Meyer, and T. Munzner, *Design Study Methodology: Reflections from the Trenches and the Stacks*. *IEEE Transactions on Visualization and Computer Graphics*, 2012. **18**(12): p. 2431-2440.
247. Meyer, M., M. Sedlmair, and T. Munzner, *The four-level nested model revisited: blocks and guidelines*, in *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors - Novel Evaluation Methods for Visualization*. 2012, Association for Computing Machinery: Seattle, Washington, USA. p. Article 11.
248. Munzner, T., *Visualization Analysis and Design*. AK Peters Visualization Series. 2015: CRC Press.
249. McDermott, J.E., M. Partridge, and Y. Bromberg, *Ten simple rules for drawing scientific comics*. *PLOS Computational Biology*, 2018. **14**(1): p. e1005845.
250. D'Hont, A., et al., *The banana (*Musa acuminata*) genome and the evolution of monocotyledonous plants*. *Nature*, 2012. **488**(7410): p. 213-217.
251. Andry, T., et al., *Interpreting the Effect of Embellishment on Chart Visualizations*, in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, Association for Computing Machinery: Yokohama, Japan. p. Article 613.
252. Bateman, S., et al., *Useful junk? the effects of visual embellishment on comprehension and memorability of charts*, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2010, Association for Computing Machinery: Atlanta, Georgia, USA. p. 2573-2582.
253. Li, H. and N. Moacdieh, *Is "chart junk" useful? An extended examination of visual embellishment*. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2014. **58**(1): p. 1516-1520.

254. Akbaba, D., J. Wilburn, and M. Meyer, *Manifesto for Putting "Chartjunk" in the Trash 2021!*. 2021, visualization design lab: visualization design lab.
255. Few, S., *The Chartjunk Debate - A Close Examination of Recent Findings*. Perceptual Edge - Visual Business Intelligence Newsletter, 2011(April, May and June).
256. Walny, J., et al., *Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff*. IEEE Transactions on Visualization and Computer Graphics, 2020. **26**(1): p. 12-22.
257. Shores, N. and B. Wong, *Data exploration*. Nature Methods, 2012. **9**(1): p. 5-5.
258. Durant, É. *Mémoire de Projet de Fin d'Études : Développement d'interfaces graphiques web pour la gestion de pangénomes*. [MSc2 Thesis] 2018; Available from: <https://drive.google.com/file/d/1HpaoQzJWGwU2XnQX8B0wI1ijytaPYk3F/view?usp=sharing>.
259. Dolatabadian, A., et al., *Characterization of disease resistance genes in the Brassica napus pangenome reveals significant structural variation*. Plant Biotechnology Journal, 2020. **18**(4): p. 969-982.
260. Monat, C., et al., *Comparison of two African rice species through a new pan-genomic approach on massive data*. bioRxiv, 2018: p. 245431.
261. Bayer, P.E., et al., *Wheat Panache: A pangenome graph database representing presence-absence variation across sixteen bread wheat genomes*. The Plant Genome, 2022. **n/a**(n/a): p. e20221.
262. Edwards, D. and J. Batley, *Graph pangenomes find missing heritability*. Nature Genetics, 2022.
263. Hübner, S., *Are we there yet? Driving the road to evolutionary graph-pangenomics*. Current Opinion in Plant Biology, 2022. **66**: p. 102195.
264. Shneiderman, B. *The eyes have it: a task by data type taxonomy for information visualizations*. in *Proceedings 1996 IEEE Symposium on Visual Languages*. 1996.
265. Nielsen, C. and B. Wong, *Representing genomic structural variation*. Nature Methods, 2012. **9**(7): p. 631-631.
266. Stephens, Z., et al., *Detection and visualization of complex structural variants from long reads*. BMC Bioinformatics, 2018. **19**(20): p. 508.
267. Yokoyama, T.T. and M. Kasahara, *Visualization tools for human structural variations identified by whole-genome sequencing*. Journal of Human Genetics, 2020. **65**(1): p. 49-60.
268. Veltri, D., M.M. Wight, and J.A. Crouch, *SimpleSynteny: a web-based tool for visualization of microsynteny across multiple species*. Nucleic Acids Res, 2016. **44**(W1): p. W41-5.
269. Farrer, R.A., *Synima: a Synteny imaging tool for annotated genome assemblies*. BMC Bioinformatics, 2017. **18**(1): p. 507.
270. Venkat, B. and C. Gutwin. *Interactive Exploration of Genomic Conservation*. in *46th Graphics Interface Conference on Proceedings of Graphics Interface 2020 (GI'20)*. 2020. University of Toronto: Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine,
271. Nattestad, M., et al., *SplitThreader: Exploration and analysis of rearrangements in cancer genomes*. 2016: p. 087981.
272. van den Brandt, A., et al. *Visual Exploration of Genetic Sequence Variants in Pangenomes*. in *EuroVis 2022*. 2022. Roma: The Eurographics Association.
273. Pearson, K., *LIII. On lines and planes of closest fit to systems of points in space*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1901. **2**(11): p. 559-572.
274. Jolliffe, I.T. and J. Cadima, *Principal component analysis: a review and recent developments*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2016. **374**(2065): p. 20150202.
275. van der Maaten, L. and G. Hinton, *Visualizing Data using t-SNE*. Journal of Machine Learning Research, 2008. **9**(86): p. 2579-2605.
276. McInnes, L., et al., *UMAP: Uniform Manifold Approximation and Projection*. Journal of open source software, 2018. **3**(29): p. 861.


- 
277. Hyun, J.C., J.M. Monk, and B.O. Palsson, *Comparative pangenomics: analysis of 12 microbial pathogen pangenomes reveals conserved global structures of genetic and functional diversity*. BMC Genomics, 2022. **23**(1): p. 7.
278. Jin, M., et al., *Maize pan-transcriptome provides novel insights into genome complexity and quantitative trait variation*. Scientific Reports, 2016. **6**(1): p. 18936.
279. Anscombe, F.J., *Graphs in Statistical Analysis*. The American Statistician, 1973. **27**(1): p. 17-21.
280. Smith, N.J., et al., *matplotlib/viscm v0.9*. 2019, Zenodo.

Table of Appendices

Appendix I: GFAv2 stores graph elements labelled as S, E, O, U, G and F lines.....	142
Appendix II: Example of graph generated with Cytoscape for MetaPGN.....	143
Appendix III: Gephi used to represent a pangenome graph built with PPanGGOLiN	144
Appendix IV: Example of graph built with neo4j for PanTools.....	145
Appendix V: The final screenshot of anvi'o's pangenomic workflow shows an ordered circular heatmap with metadata	146
Appendix VI: Association network and heatmap representation from Coinfinder are static plots built from gene clusters.....	147
Appendix VII: Gingr can switch its visual representations depending on the zoom level; it does a semantic zoom	148
Appendix VIII: The icicle plots from Hierarchical Sets show common elements between closely related genomes only.....	148
Appendix IX: Microreact provides linked views, connectde visual representations of a same dataset.....	149
Appendix X: PanGeT's flower plots still have interactivity, with clickable hyperlinks for data download	150
Appendix XI: Pan-Tetris can group genes from different clusters when they are complementary.....	151
Appendix XII: PanViz's circular visual representation can be interactively switched to alternative representations	152
Appendix XIII: Phandango provides PAV heatmaps with metadata and phylogeny. Figure taken from the online example of Phandango used with [205]'s Streptococcus data.	153
Appendix XIV: Chromatiblock displays conserved blocks ordered depending on a chosen reference.....	153
Appendix XV: The Genome Context Viewer's main visual representation displays syntenic genes as 'beads-on-a-string'.....	154
Appendix XVI: PGV's browser displays genomic blocks of alignment based on a reference-agnostic consensus ordering.....	154
Appendix XVII: TASUKE+ can be used for the display of SNPs density within a pangenome	155
Appendix XVIII: Crop-Haplotypes displays shared and unique haplotype blocks found in 15 wheat assemblies	156
Appendix XIX: GenomeRing uses a circular visual representation for displaying shared pangenome blocks between species.....	157
Appendix XX: gfaestus displays genome graphs quite like Bandage does.....	158
Appendix XXI: GfaViz is a genome graph visualization tool dedicated to GFA files.....	158
Appendix XXII: The graphgenomeviewer displays genome graphs in a simple interface.....	159
Appendix XXIII: The IGGE is a VR tool that displays genome graph in a 3D environment.....	160
Appendix XXIV: Panaconda draws its pan-syteny graphs with Gephi.....	160
Appendix XXV: panGraphViewer has an interface for displaying genome graphs with a unusual visual representation.....	161

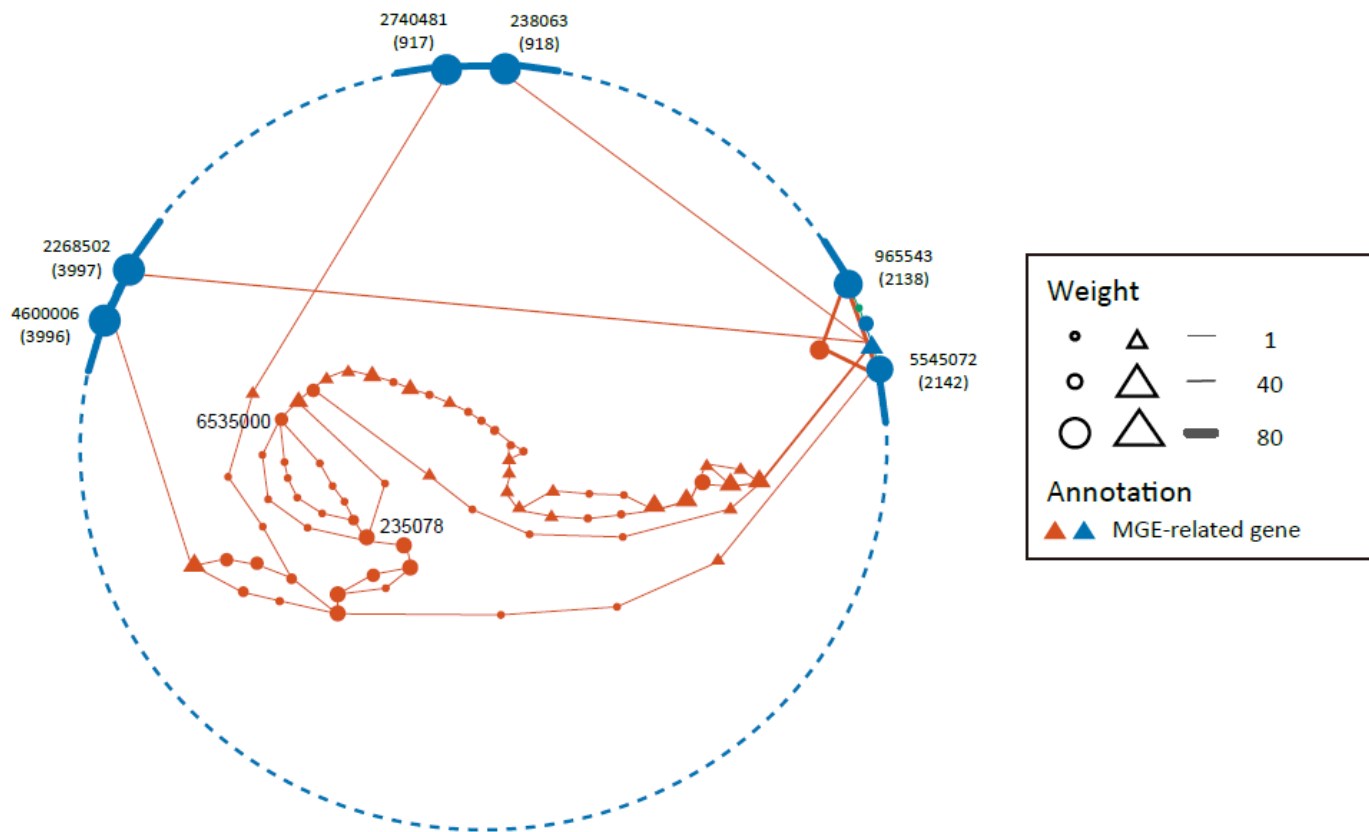
Appendix XXVI: Computomics' version of pantograph displays the successions of present and absent pangenomic blocks within genomes.....	161
Appendix XXVII: Sequence Tube Maps represent sequences as lines going through different pangenomic blocks	162
Appendix XXVIII: vg view displays portions of a genome graph, with the graph backbone and the aligned paths	163
Appendix XXIX: PGAP-X views can be customized with different color palettes	164
Appendix XXX:The Augmented Graph Viewer's poster won best BioVis poster award in 2017	165
Appendix XXXI: Linx uses a circular representation with multiple visual encodings detailing SVs.....	166
Appendix XXXII: Corteva's TagDots and PANDA are tools creating static visualizations of pangenomes at different scales	167
Appendix XXXIII: Anscombe's quartet features four visually different datasets that share the same summary statistics	168
Appendix XXXIV: Boustrophedon switches the direction of writing with every line	168
Appendix XXXV: Two identical colors can be perceived as different depending on their context.....	169
Appendix XXXVI: Pan-Tetris has different behaviors depending on the source and format of data.....	170
Appendix XXXVII: Visual representations of pgAtlases could have included heatmaps of clustered gene families	171
Appendix XXXVIII: I explored semi-circular representations along the same lines as anvi'o.....	171
Appendix XXXIX: The PAV matrix is represented as a histogram within the miniature of Panache.....	172
Appendix XL: The position track uses a alternative color palette to the usual rainbow	172
Appendix XLI: The menu panel was thought to display filters option and legends	173
Appendix XLII: The color legends for the tracks were supposed to be dynamically updated	174
Appendix XLIII: The geometric zoom level can be modified directly within the menu panel	174
Appendix XLIV: The linearization of the semi-circular display resulted in the PAV heatmap.....	175
Appendix XLV: The PAV matrix was originally planned to be displayed vertically	176
Appendix XLVI:The first version of the horizontal PAV matrix included multiple buttons for customization....	177
Appendix XLVII: Cooccurences were originally visualized as a track and circles representing the distribution of the repetitions within the pangenome.....	178
Appendix XLVIII: The detail of the distribution of cooccurrences within the pangenome is better encoded with rectangles	178
Appendix XLIX: The first draft of a view dedicated to a selected repeat would have appeared on click	179
Appendix L: An alternative visual representation of the repeats showed their position within panchromosomes	179
Appendix LI: A view dedicated to repeats could show the PAV status of other repeats and their context as 'windows-with-blinds'	180
Appendix LII: A draft version of a detail view for repeated panBlocks included the visualization of the PAV status of all occurrences.....	181

Appendix LIII: Another option for repeats was to show them as togglable tooltip overlapping the main PAV matrix	182
Appendix LIV: Hollow areas can overlap within the PAV matrix	183
Appendix LV: Face to face discussions were critical during the design of the visual representations	184
Appendix LVI: The ‘needs and expectations’ survey had a branching structure	185
Appendix LVII: Survey results 1/11 – Profile of the respondents	216
Appendix LVIII: Survey results 2/11 – Pangenomics background	216
Appendix LIX: Survey results 3/11 – Envisioned application field of pangenomics	217
Appendix LX: Survey results 4/11 – Feedback on Panache’s prototype	217
Appendix LXI: Survey results 5/11 – Most desirable features for lookup	218
Appendix LXII: Survey results 6/11 – If ‘Sorting’ or ‘Filtering’ were chosen in the previous question, details were asked	218
Appendix LXIII: Survey results 7/11 – Needed visual aids	219
Appendix LXIV: Survey results 8/11 – Data exchanges	219
Appendix LXV: Survey results 9/11 – Navigation within the visual representation and UI	220
Appendix LXVI: Survey results 10/11 – Customizations	220
Appendix LXVII: Survey results 11/11 – Possible formats	221
Appendix LXVIII: Respondents clicked on a lot of non-interactive parts of the display	221
Appendix LXIX: The panBlocks that should be represented must be within a certain window around the coordinates on display	222
Appendix LXX: The exact limit for the coordinates of panBlocks that should be drawn depends on the maximum width of blocks	223
Appendix LXXI: Panache’s poster at the Plant Genome Evolution conference 2019	224
Appendix LXXII: Panache’s poster at PAG 2020	225
Appendix LXXIII: Panache’s poster at VIZBI 2021	226
Appendix LXXIV: Panache's poster at JOBIM 2021	227
Appendix LXXV: Panache’s poster at ISMB/ECCB 2021 BioVis session	228
Appendix LXXVI: UI design 1/5 – Choice of the species to work on	229
Appendix LXXVII: UI design 2/5 – Overall diversity, visual representation of the diversity and available datasets	229
Appendix LXXVIII: UI design 3/5 – Visual representation and exploration of SVs at the whole genome scale ..	230
Appendix LXXIX: UI design 4/5 – Gene or panBlock PAV, targeted on a region of interest	230
Appendix LXXX: UI design 5/5 – Sequence-level comparison, MSA-like representation	231
Appendix LXXXI: The planned UI would have a navigation pane on top to change the views	231
Appendix LXXXII: Scatterplot user flow 1/11 - Welcome page	232
Appendix LXXXIII: Scatterplot user flow 2/11 – Hovering an assembly with one clustering value	232
Appendix LXXXIV: Scatterplot user flow 3/11 – Hovering an assembly with two clustering values	233

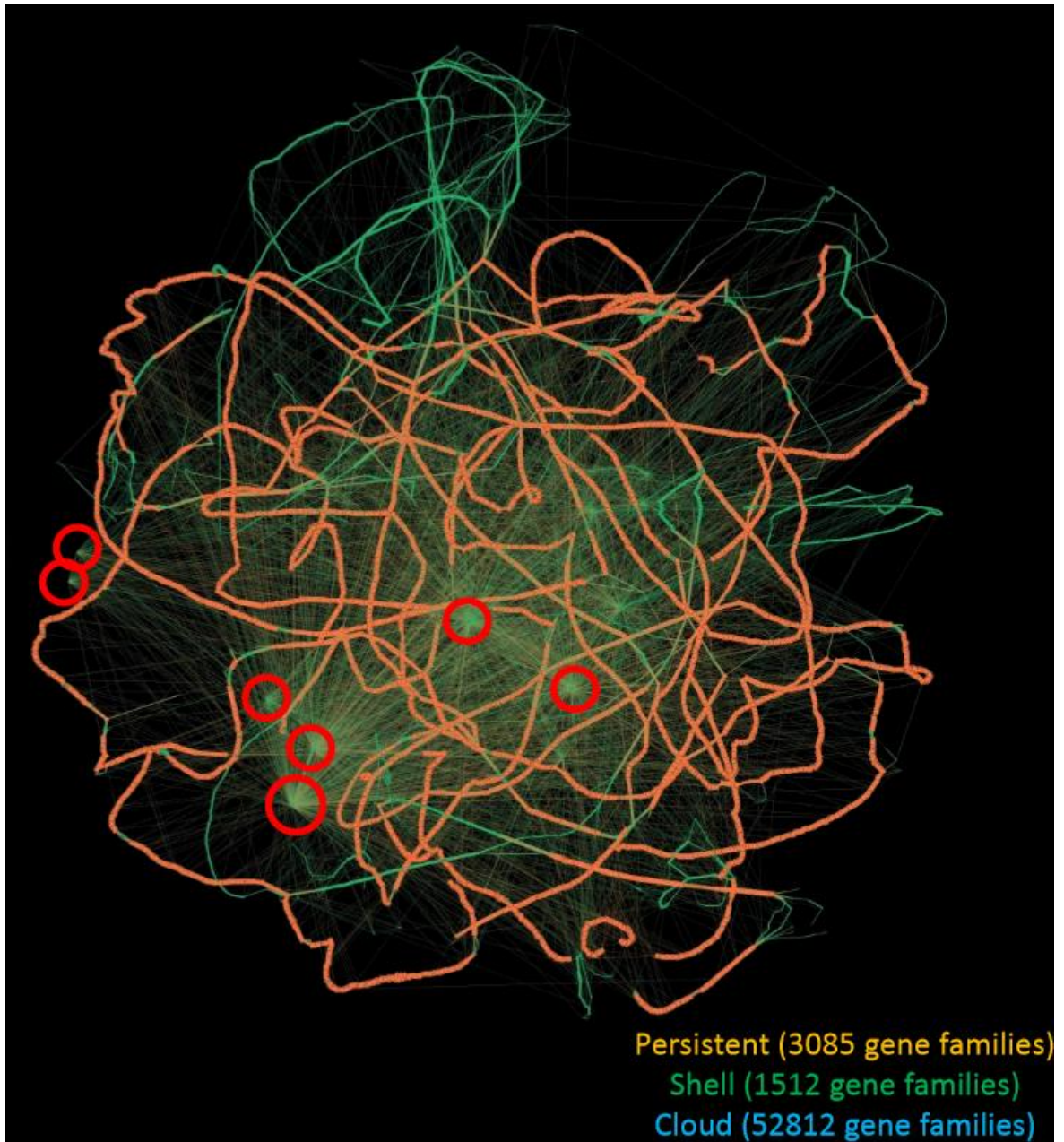
Appendix LXXXV: Scatterplot user flow 4/11 – Assemblies have been selected by lasso selection, matching pangenomes are listed	233
Appendix LXXXVI: Scatterplot user flow 5/11 – The best matching pangenome is hovered.....	234
Appendix LXXXVII: Scatterplot user flow 6/11 – The best matching pangenome is chosen.....	234
Appendix LXXXVIII: Scatterplot user flow 7/11 – The second best matching pangenome is hovered	235
Appendix LXXXIX: Scatterplot user flow 8/11 – The second best matching pangenome is chosen.....	235
Appendix XC: Scatterplot user flow 9/11 – Multiple partially matching pangenomes are highlighted from the bar chart.....	236
Appendix XCI: Scatterplot user flow 10/11 – A phenotype filter tag is added to the assembly table	236
Appendix XCII: Scatterplot user flow 11/11 – A second phenotype filter tag is added, with the first one still active	237
Appendix XCIII: Circos layouts could be applied to panchromosomes.....	238
Appendix XCIV: The Comparative Genome Viewer visually represents all-versus-all alignments of chromosomes from two genome assemblies.....	239
Appendix XCV: Tabular panchromosomes user flow 1/6 – Only cooccurrences from a small region are selected	240
Appendix XCVI: Tabular panchromosomes user flow 2/6 – Moving the handles on the main panchromosome widens the selected region	240
Appendix XCVII: Tabular panchromosomes user flow 3/6 – Profiles could be displayed rather than violin-like silhouettes	241
Appendix XCVIII: Tabular panchromosomes user flow 4/6 – Hovering a region would highlight the related cooccurrences	241
Appendix XCIX: Tabular panchromosomes user flow 5/6 – Hovering a segment could display its ribbon only	242
Appendix C: Tabular panchromosomes user flow 6/6 – Clicking on a panchromosome profile would change the main panchromosome.....	242
Appendix CI: The glyph system remains understandable under different colorblind visions.....	243
Appendix CII: The detail view can display many combinations of SVs surrounding a step from the pivot	244
Appendix CIII: Pangenome graph can be divided into collection of nodes and paths within a JSON file	245
Appendix CIV: SV Annot Decision Tree 1/3 – The algorithm starts by following every Node of a path.....	246
Appendix CV: SV Annot Decision Tree 2/3 – ...it will then look for <i>Synteny disruptions</i>	246
Appendix CVI: SV Annot Decision Tree 3/3 – ...and explore the diverging paths for characterization.....	247
Appendix CVII: SV Annot Storage 1/3 – All annotations can be stored within a sparse matrix.....	247
Appendix CVIII: SV Annot Storage 2/3 – ...that would contain SV objects only at specific indices.....	248
Appendix CIX: SV Annot Storage 3/3 – ...with each their own properties	248
Appendix CX: All views within SaVanache would be connecting different files and formats	249
Appendix CXI: Usually design studies follow a top-down approach, going from the data to the abstraction	250



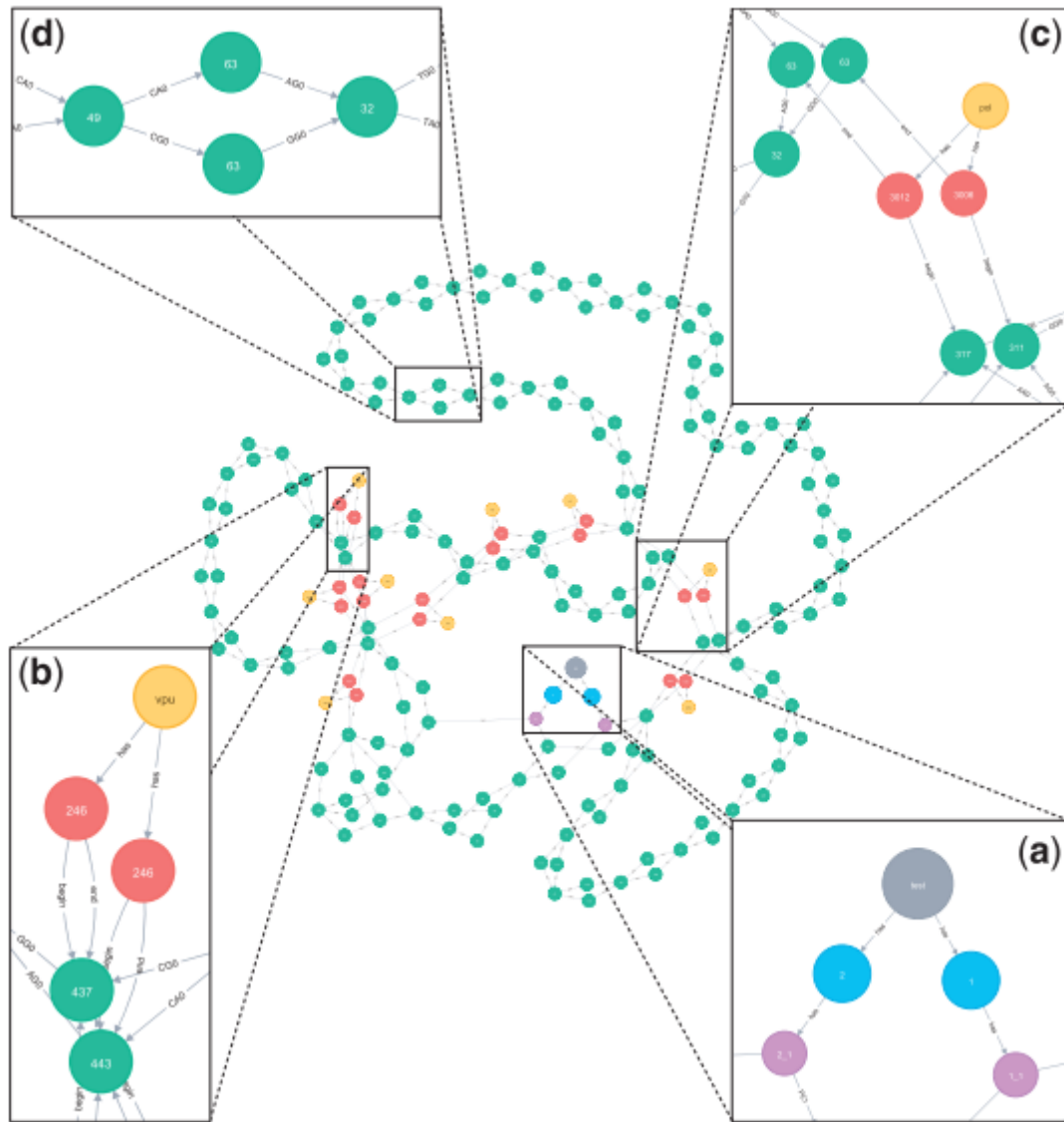
Appendix



Appendix II: Example of graph generated with Cytoscape for MetaPGN; Figure from [197]



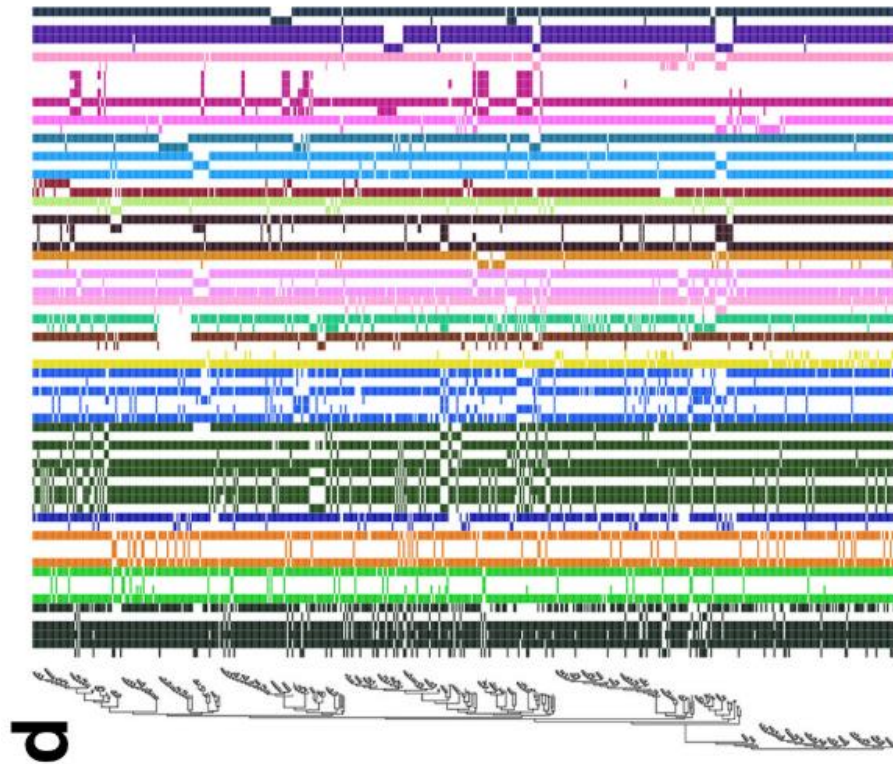
Appendix III: Gephi used to represent a pangenome graph built with PPanGGOLiN; [70] from 2407 *Acinetobacter baumannii* strains. Figure from Guillaume Gautreau.



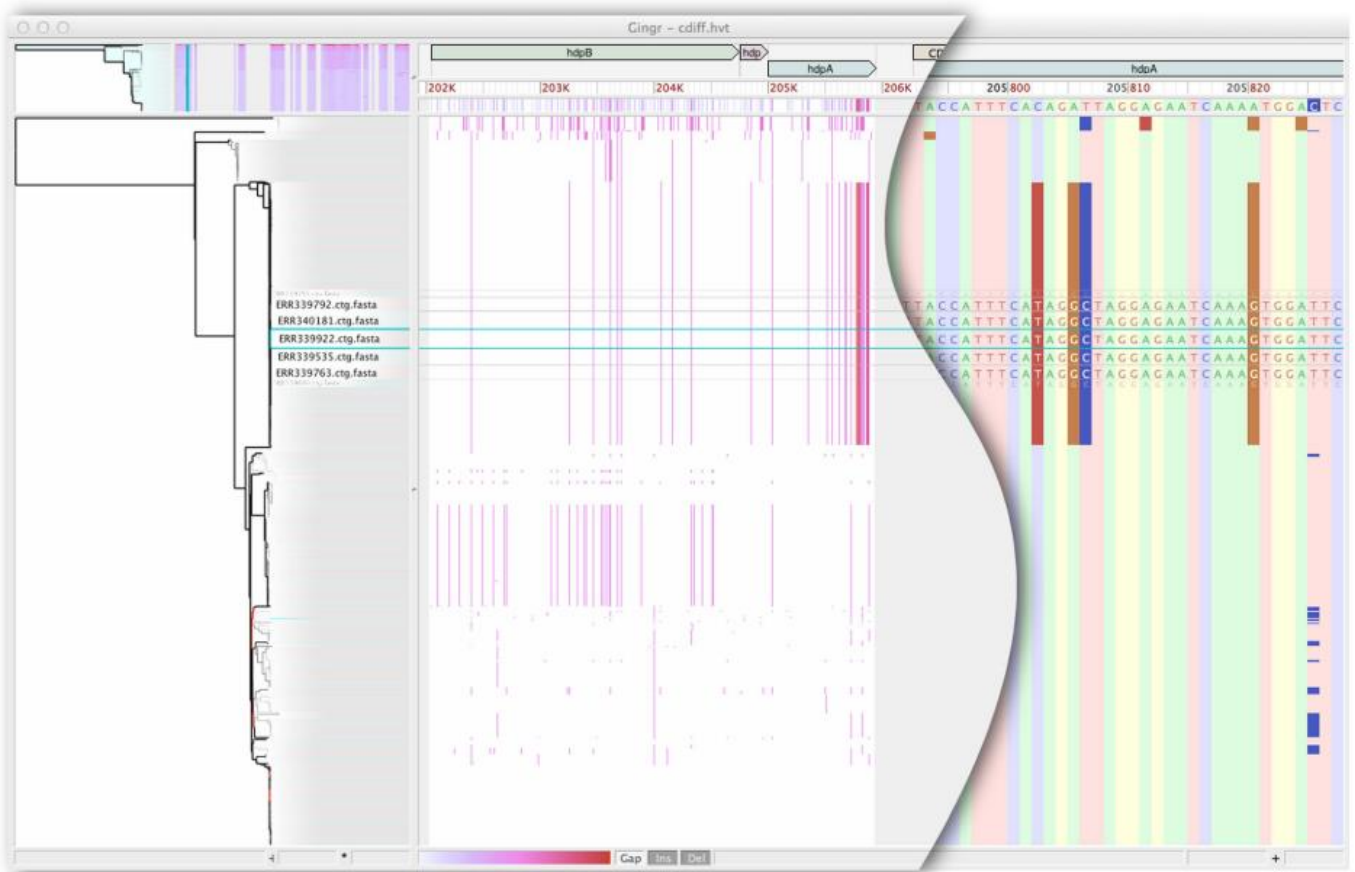
Appendix IV: Example of graph built with neo4j for PanTools Figure from [117]



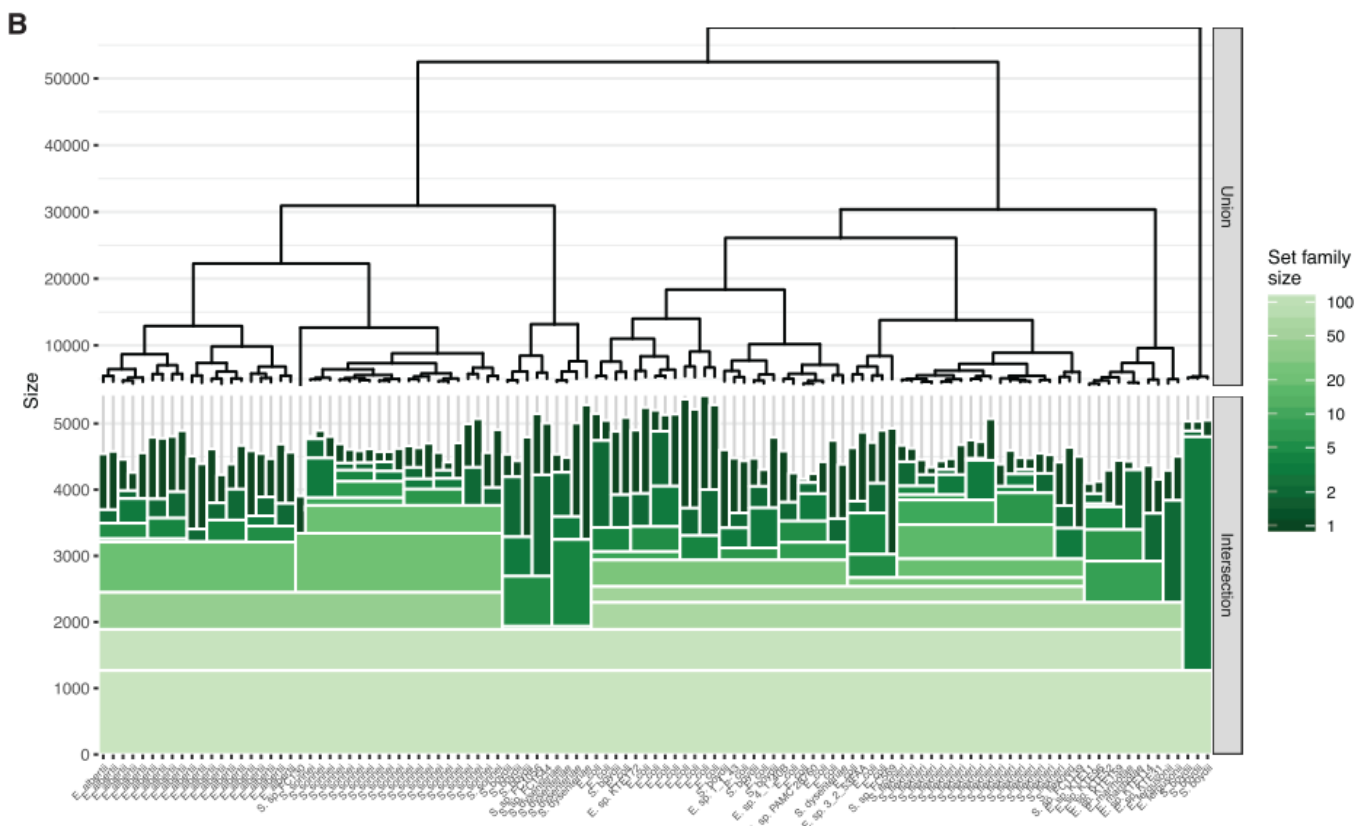
Appendix V: The final screenshot of anvio's pangenomic workflow shows an ordered circular heatmap with metadata; Figure from [158]



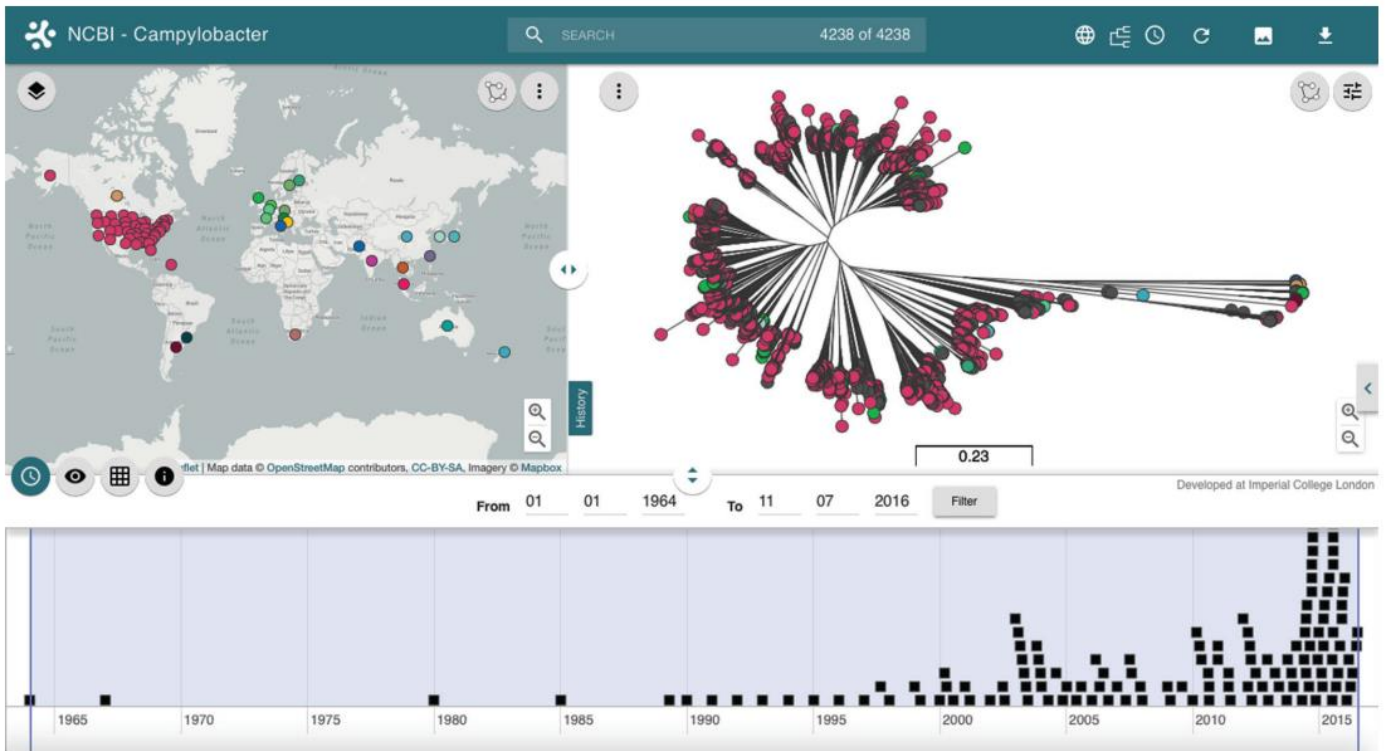
Appendix VI: Association network and heatmap representation from Coinfinder are static plots built from gene clusters; Figure from [176]



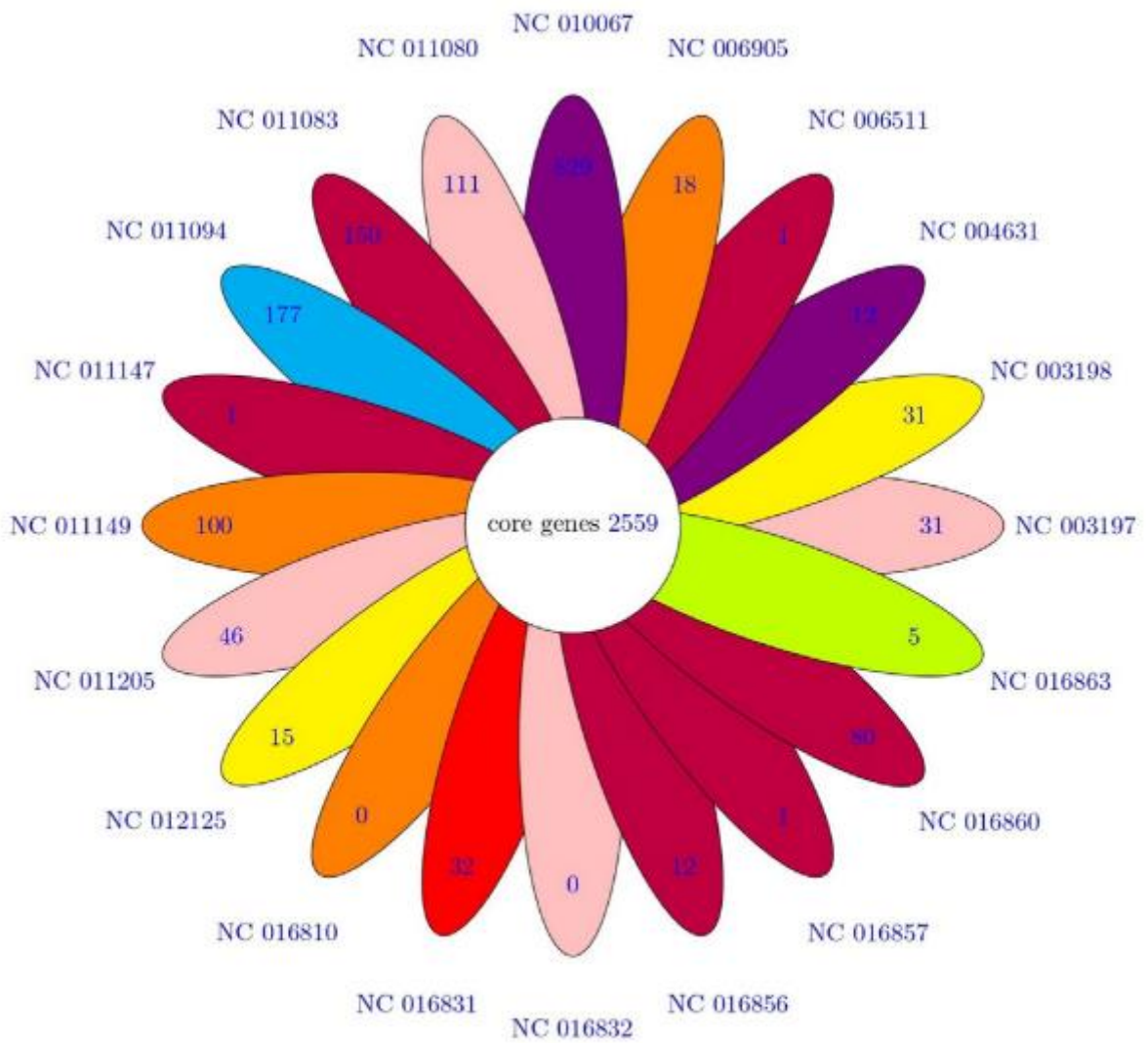
Appendix VII: Gingr can switch its visual representations depending on the zoom level; it does a semantic zoom; Figure from [109]



Appendix VIII: The icicle plots from Hierarchical Sets show common elements between closely related genomes only; Figure from [184]



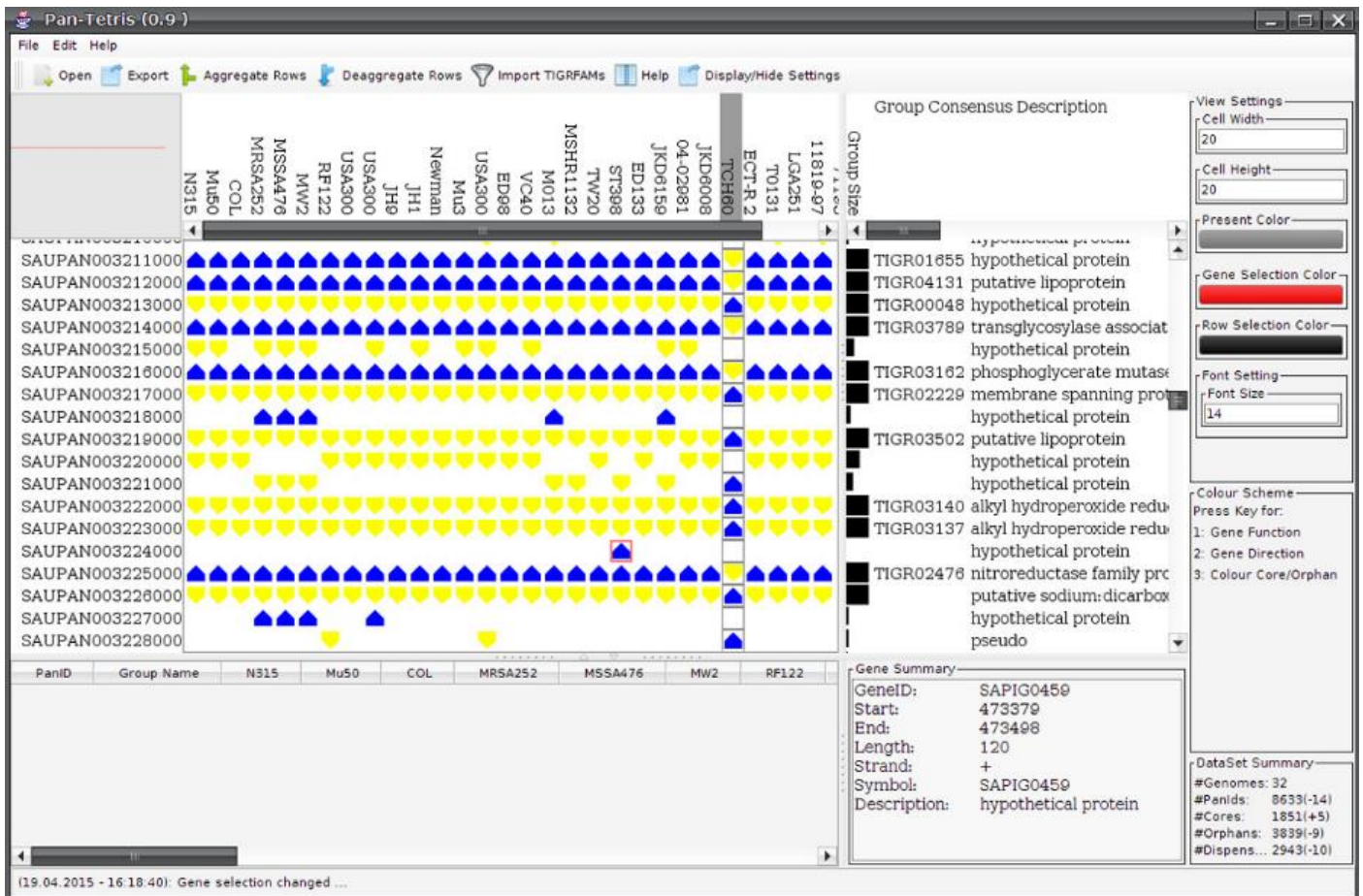
Appendix IX: Microreact provides linked views, connectde visual representations of a same dataset; Figure from [164]



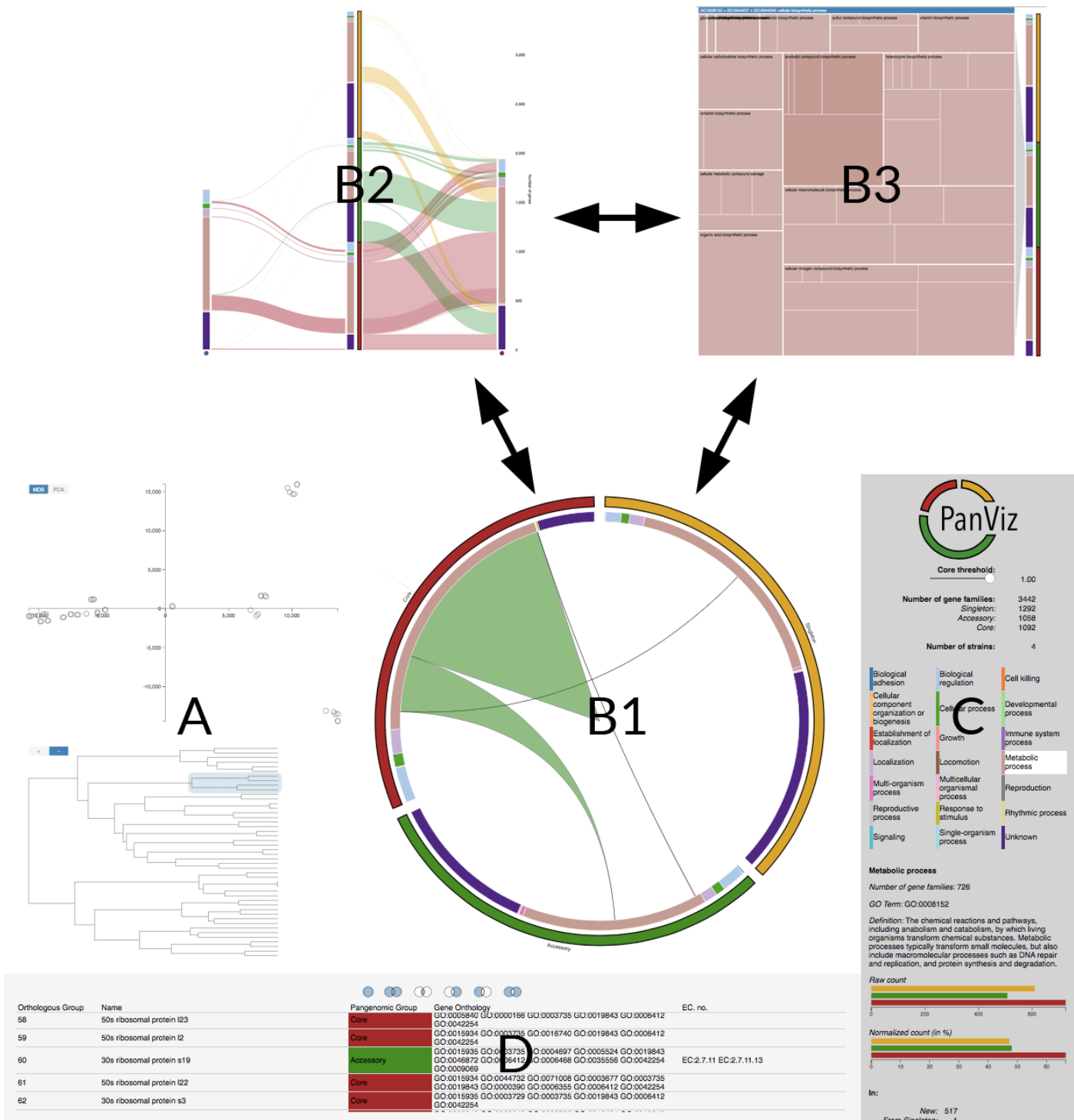
[Click here for Dispensable gene](#)

Number of genes in each genome

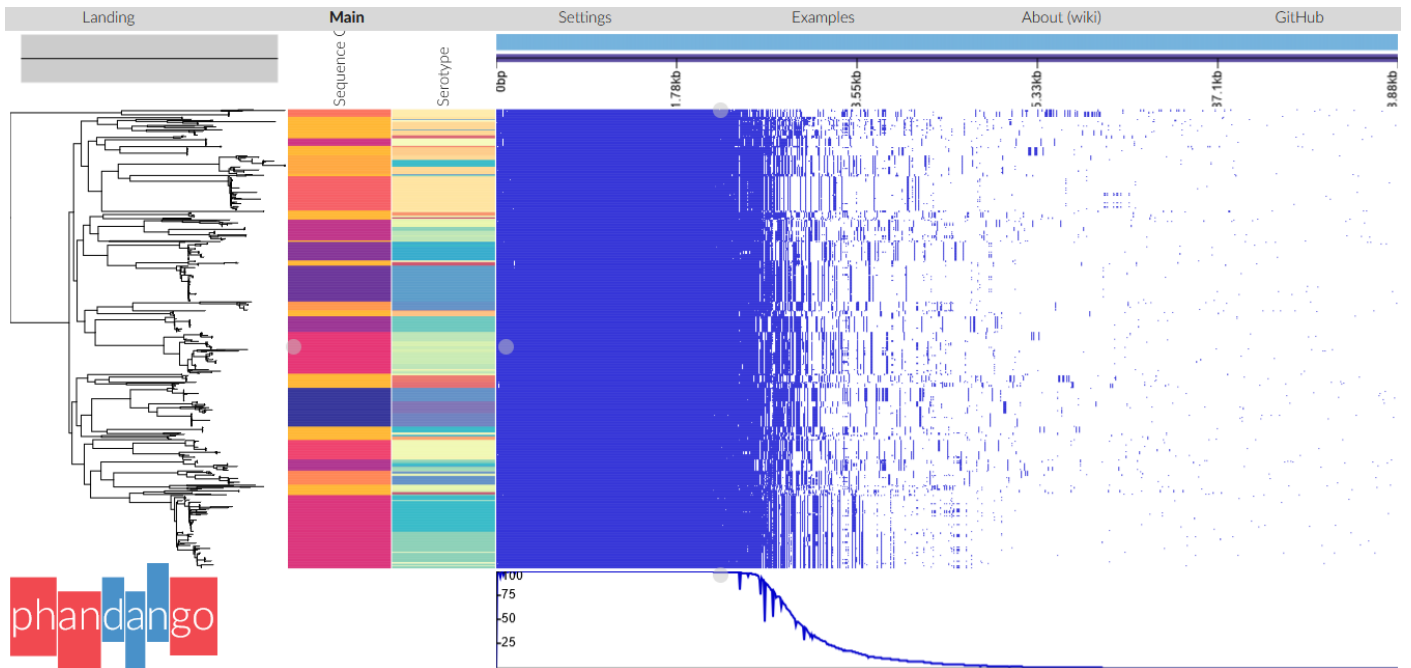
Appendix X: PanGeT's flower plots still have interactivity, with clickable hyperlinks for data download; Figure from [167]



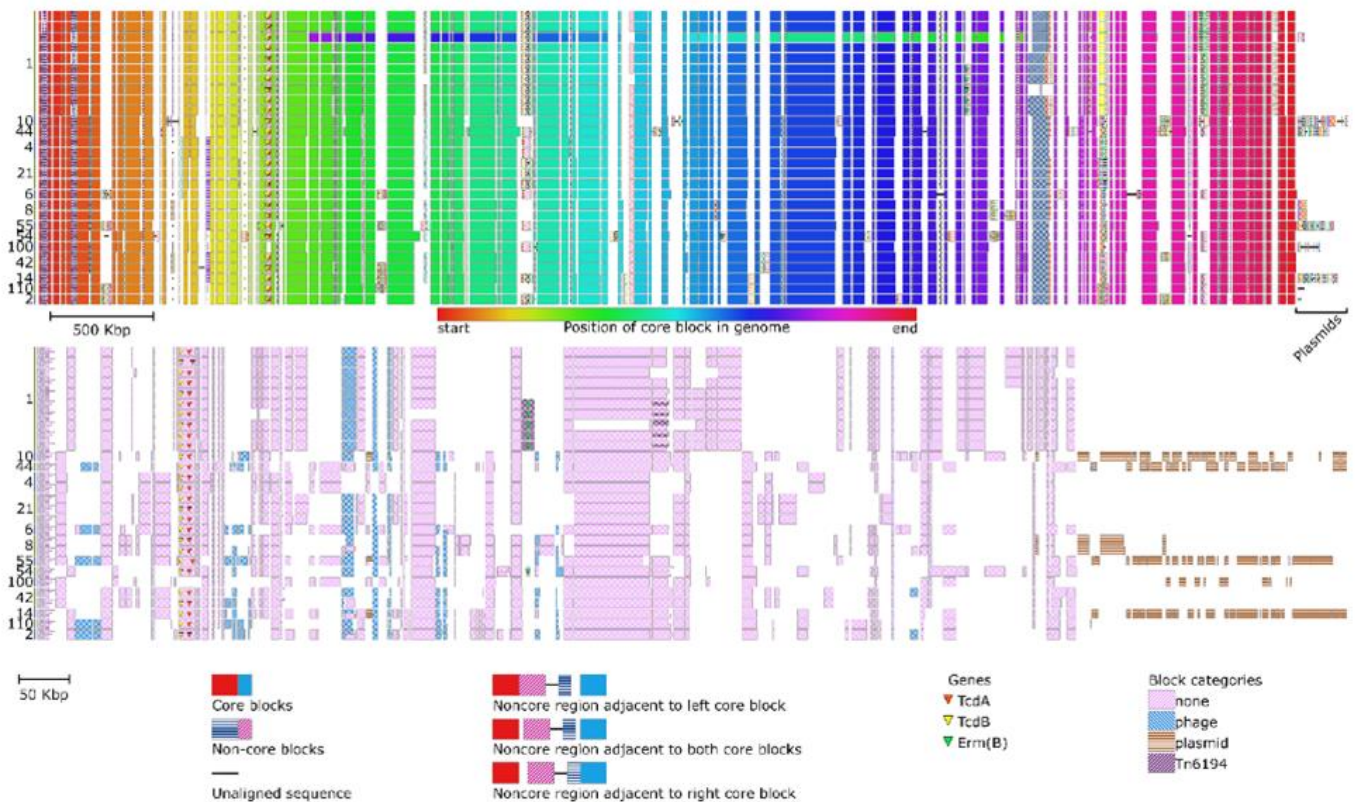
Appendix XI: Pan-Tetris can group genes from different clusters when they are complementary; Moreover, gene strands can be double encoded with both glyph orientation and color. Figure from [143]



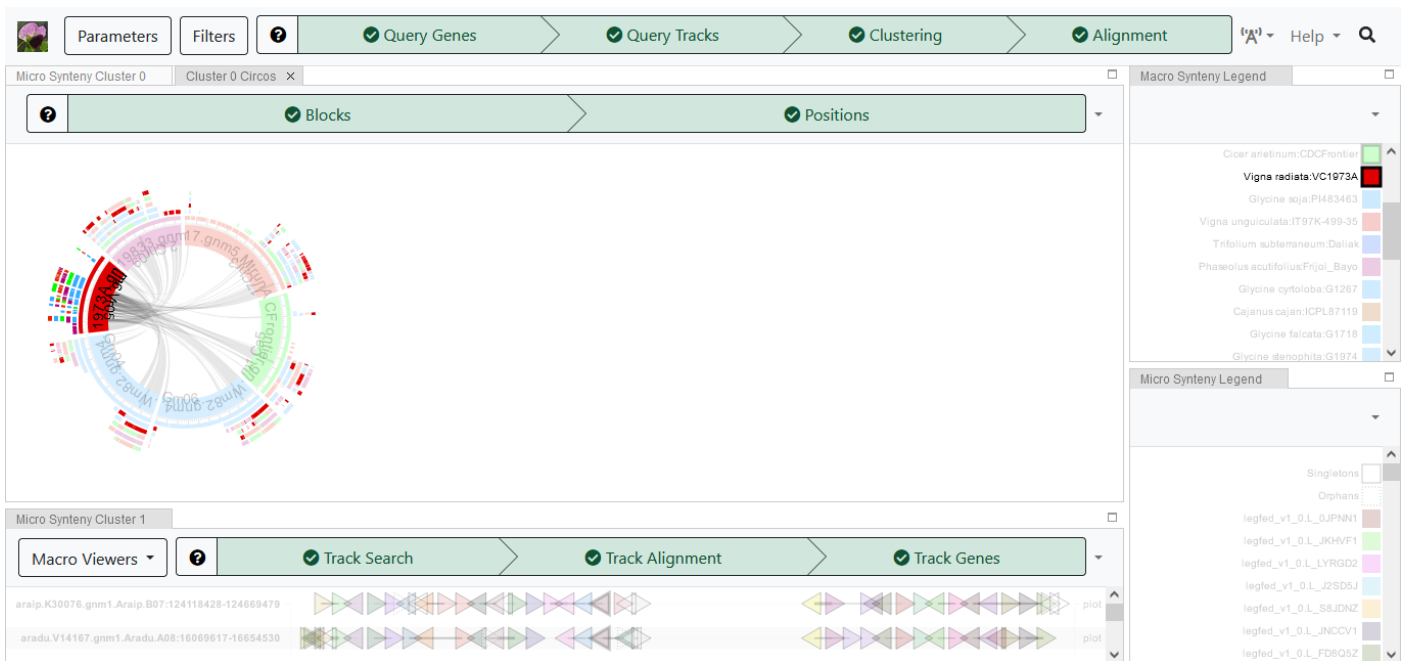
Appendix XII: PanViz's circular visual representation can be interactively switched to alternative representations; Figure from [159]



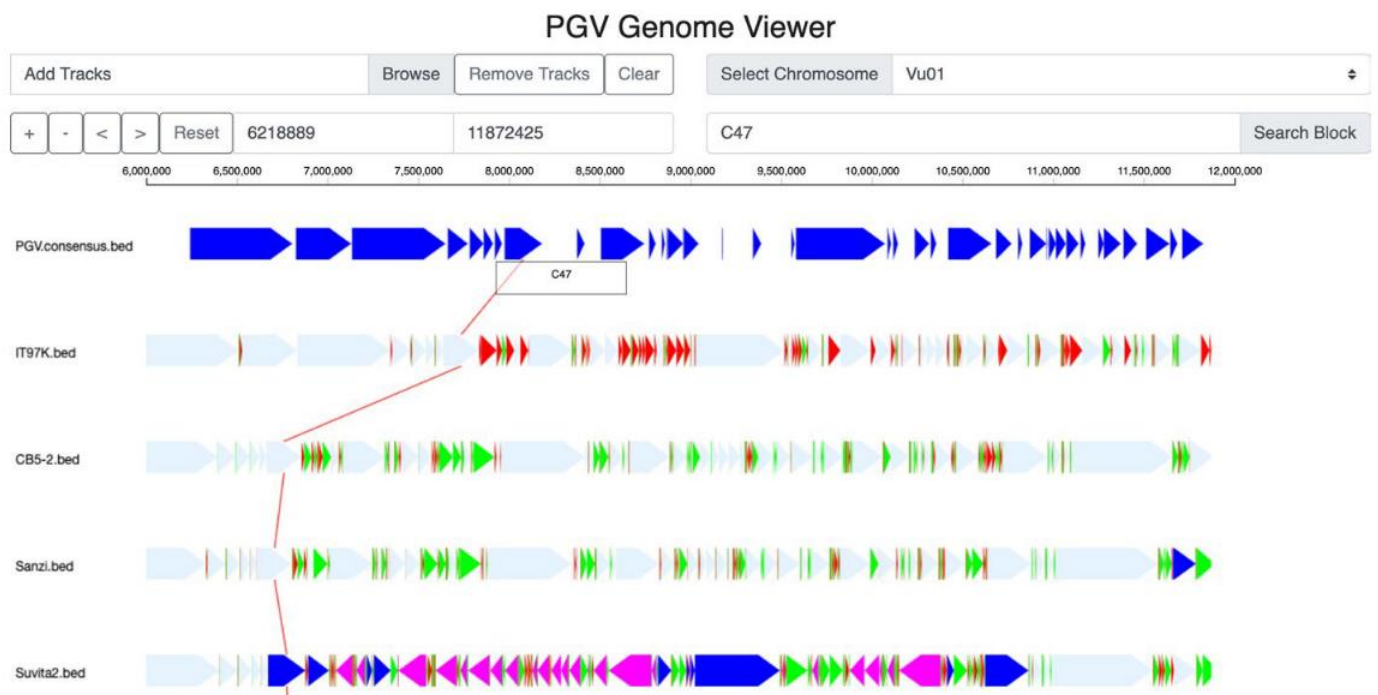
Appendix XIII: Phandango provides PAV heatmaps with metadata and phylogeny. Figure taken from the online example of Phandango used with [205]'s *Streptococcus* data.



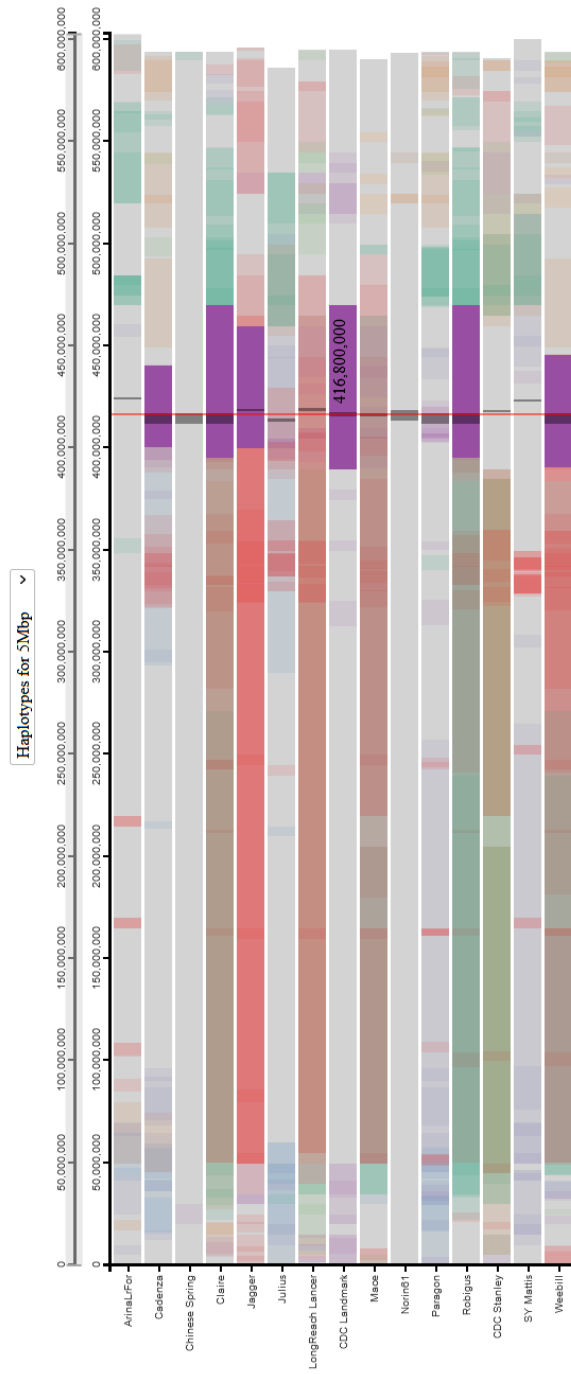
Appendix XIV: Chromatiblock displays conserved blocks ordered depending on a chosen reference; Additional blocks are squeezed in between. Figure from [175].



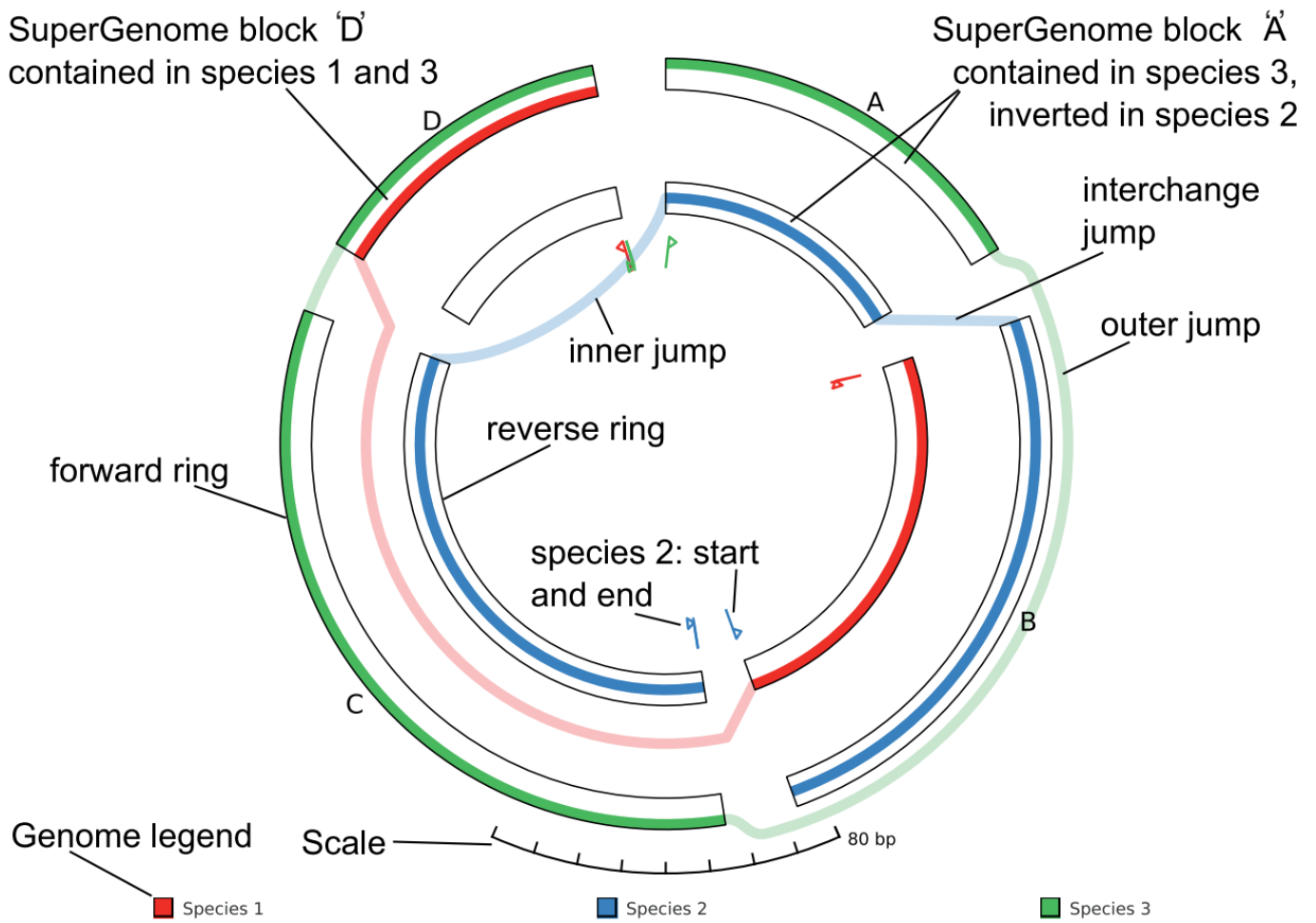
Appendix XV: The Genome Context Viewer's main visual representation displays syntenic genes as 'beads-on-a-string'; It also offers a Circos-like macro-synteny view. Figure from an online demo built with a set of genes from the tryptophan-tRNA ligase family.



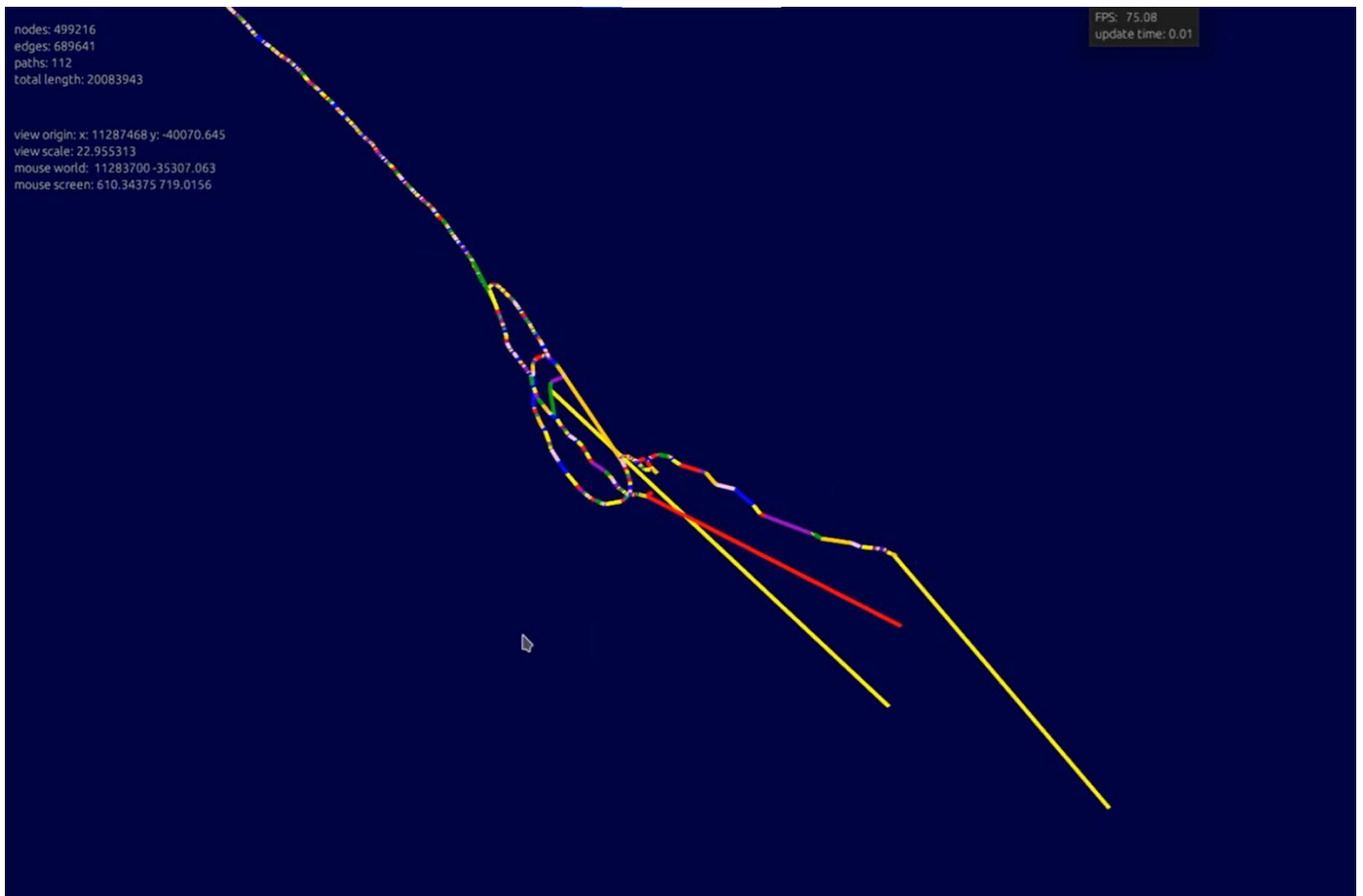
Appendix XVI: PGV's browser displays genomic blocks of alignment based on a reference-agnostic consensus ordering; Figure from [191]



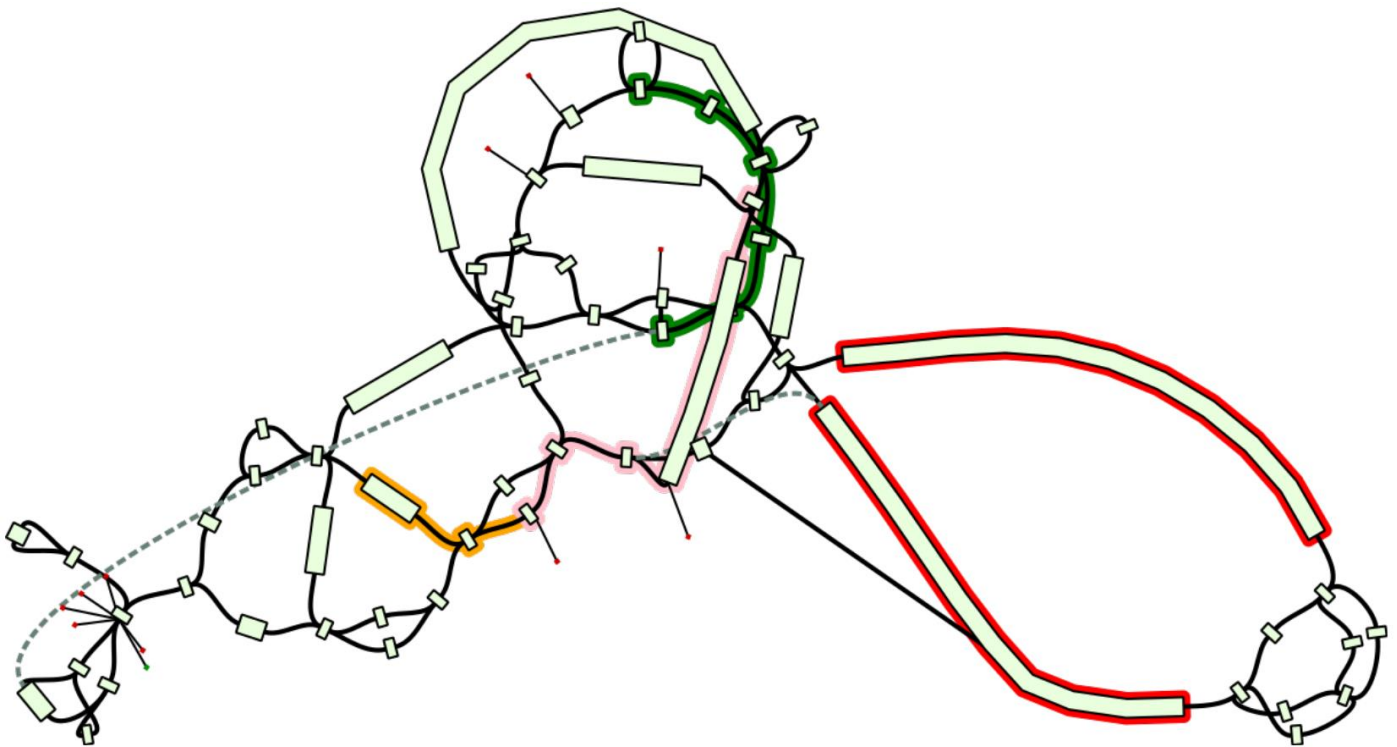
Appendix XVIII: Crop-Haplotypes displays shared and unique haplotype blocks found in 15 wheat assemblies; Screenshot from <http://crop-haplotypes.com/>



Appendix XIX: GenomeRing uses a circular visual representation for displaying shared pangenome blocks between species; Figure from [180]



Appendix XX: gfaestus displays genome graphs quite like Bandage does; Figure from the demo video available on GitHub



Appendix XXI: GfaViz is a genome graph visualization tool dedicated to GFA files; Figure made from an example GfaViz dataset.

Settings

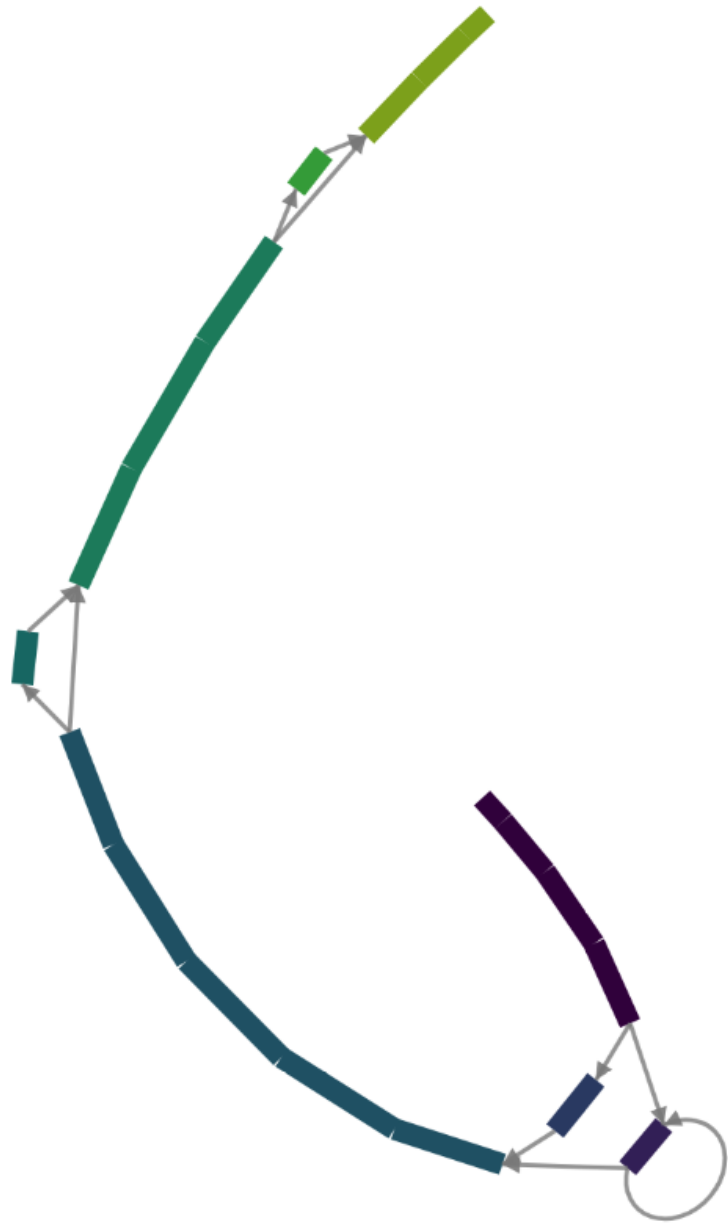
Color

Viridis ▾

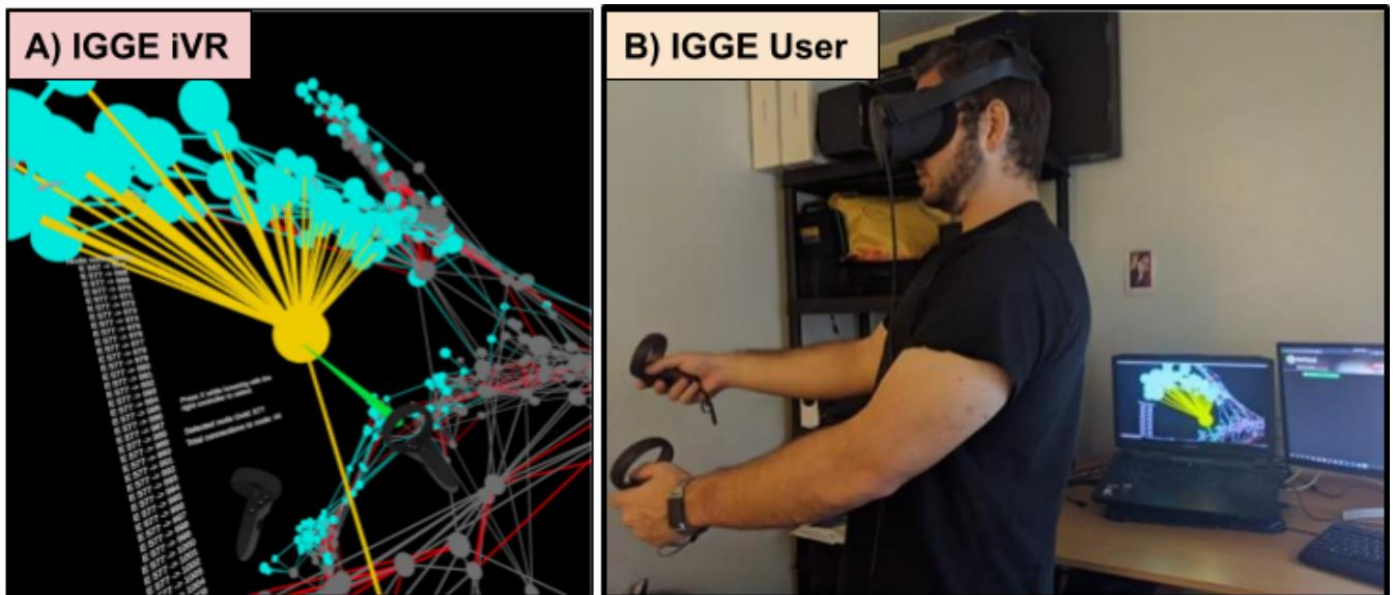
Draw paths

Draw labels

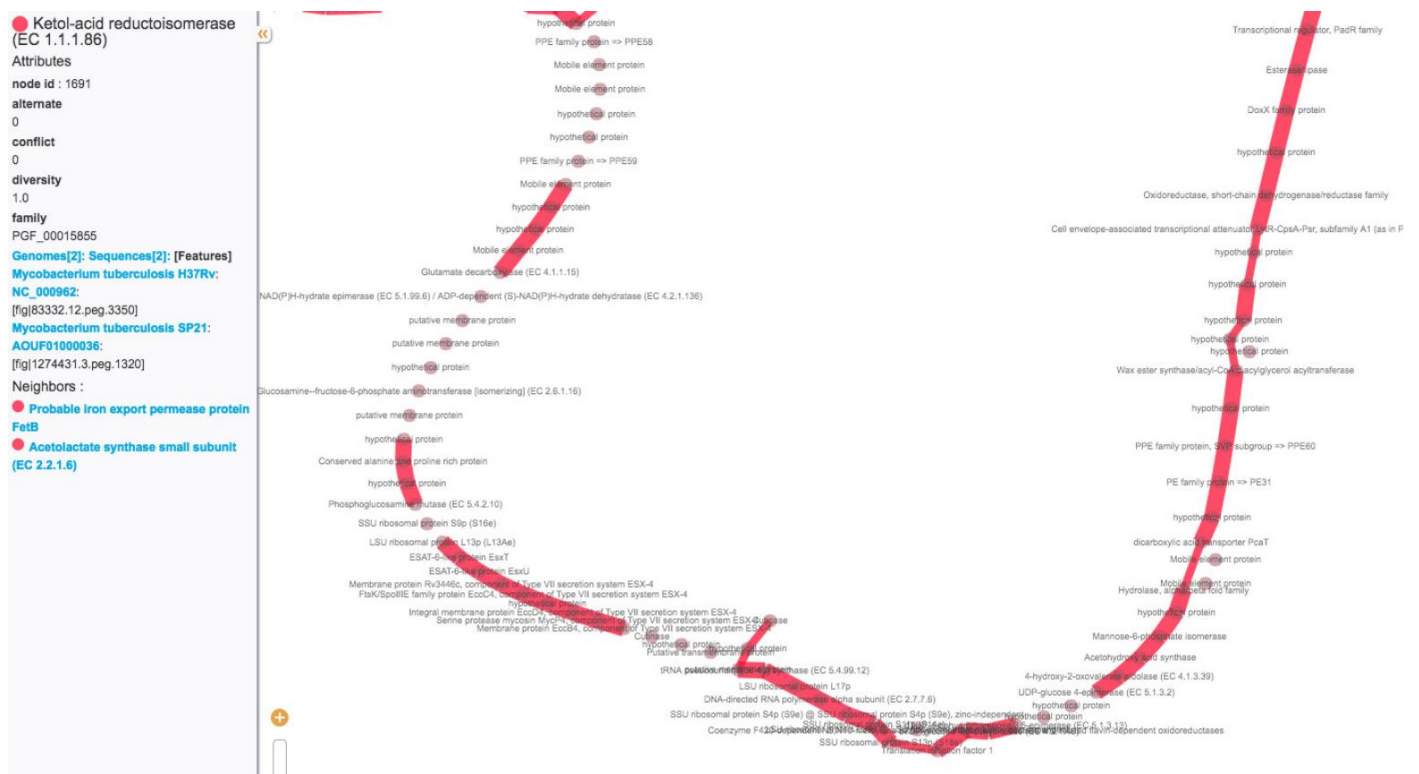
Redraw



Appendix XXII: The graphgenomeviewer displays genome graphs in a simple interface; Figure made from the interactive demo available at <https://cmdcolin.github.io/graphgenomeviewer/>



Appendix XXIII: The IGGE is a VR tool that displays genome graph in a 3D environment; Figure from [185]



Appendix XXIV: Panaconda draws its pan-synteny graphs with Gephi; Figure from [115]

panGraph Graph Utilities Config Username: demo Logout

Input

(r)GFA VCF

Uploaded (r)GFA file (*) Select uploaded (r)GFA Parse GFA

Plot pangenome (sub)graph Extract nodes Plot nodes falling in a gene region

Plot parameters (*) Select backbone Select chr Enter start (optional) Enter end (optional) Plot

NodeId: s300; Resource: Sample3_Chrom1; Len: 565; Info: DUP_10_574_REF_s2_s2; Seq: TTTGGGTTT...

© 2021 The TF Chan Lab, the Chinese University of Hong Kong - All Rights Reserved

PanGraphViewer is a tool helping to create a graph-based pangenome using a given VCF file with or without a FASTA file. It can also accept a reference graphical fragment assembly (rGFA) file from other software, such as minigraph. PanGraphViewer shows genome graph in a graphical user interface (GUI) or a webpage.

During graph display, users can customise the layout of a graph. Users can also check genes overlapping with specific genomic regions (nodes). Multiple nodes selection enable users to browse the selected nodes only or the rest nodes only.

PanGraphViewer also provides functions to check the sequence of selected nodes and enable users to save/download the corresponding sequences when right click the mouse.

PanGraphViewer provides an easy way to display a pangenome graph from rGFA or VCF files.

Appendix XXV: panGraphViewer has an interface for displaying genome graphs with a unusual visual representation; The nodes shape can be customized depending on the underlying structural variations depicted. Figure from panGraphViewer's documentation at <https://github.com/TF-Chan-Lab/panGraphViewer/tree/main/doc>

Pantograph computomics® Machine learning-based data analysis

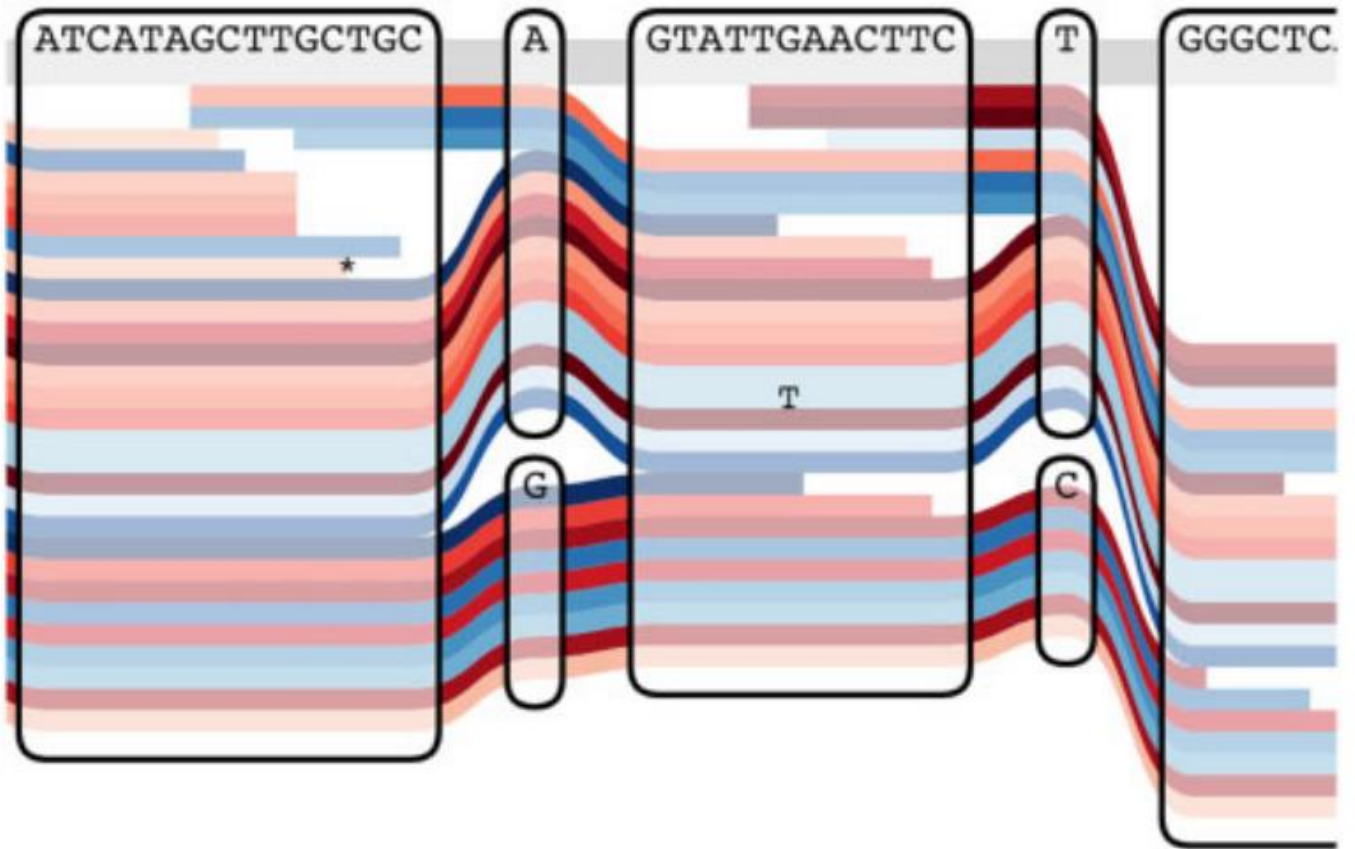
1 bp 4.0M

Position: 330,040 - 335,660

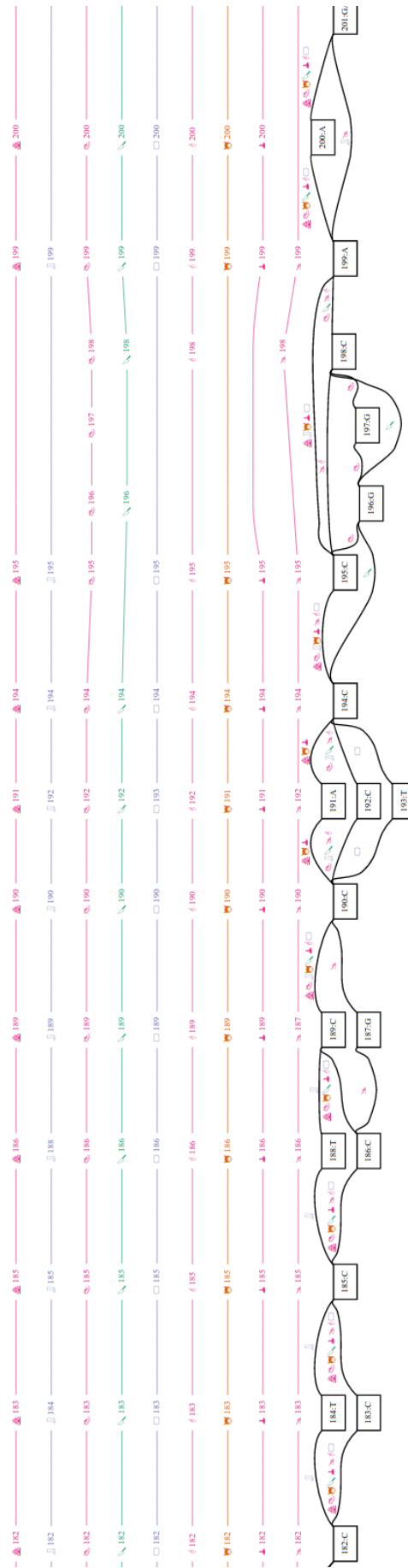
Dataset: Hpylori Bin Width: 10 Sorting: id Color by Meta...: none JUMP IN PANGENOME

NC_000915.1_26695
 NC_017382.1_51
 NC_019560.1_AK1av1k117
 NC_019561.1_p1HPAKL117
 NC_019562.1_p2HPAKL117
 NZ_CP011485.1_ausabr705
 NC_017365.1_F30
 NC_017369.1_pHPPF30
 NC_017371.1_Gamb1a94-24
 NC_017364.1_pHPCW94-24
 NC_017733.1_HUP-B14
 NC_017734.1_pHPB14
 NC_017372.1_IndriA7
 NC_000921.1_799
 NZ_CP011486.1_k26A1
 NZ_CP011482.1_L17
 CP001217.1_P12
 CP001218.1_pHPP12
 NZ_CP011487.1_PNG84A
 NC_017379.1_Puno135
 NC_017361.1_SouthAfrica7
 NC_017373.1_pHPSAF7

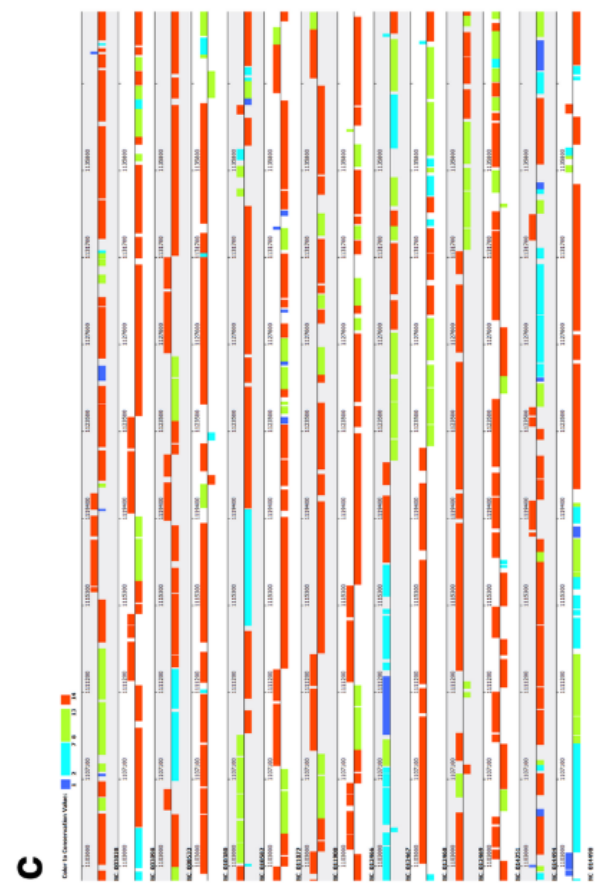
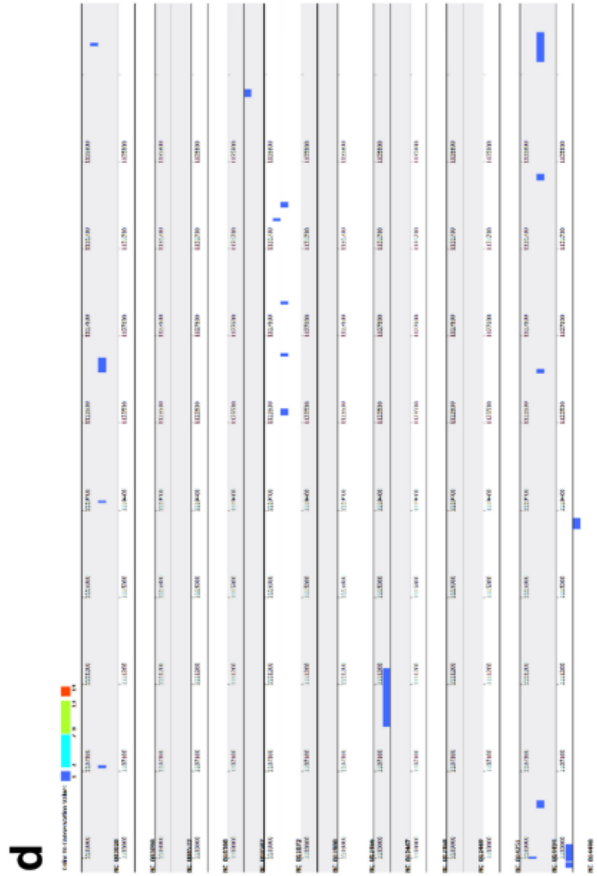
Appendix XXVI: Computomics' version of pantograph displays the successions of present and absent pangenomic blocks within genomes; Arcs flank these blocks where structural variations are observed. Figure from <https://pantograph.computomics.com/>



Appendix XXVII: Sequence Tube Maps represent sequences as lines going through different pangenomic blocks; Here two SNPs are present. Figure from [194]



Appendix XXVIII: vg view displays portions of a genome graph, with the graph backbone and the aligned paths; Figure from <https://github.com/vgteam/vg/wiki/Visualization>



Appendix XXIX: PGAP-X views can be customized with different color palettes; Here the left view distinguishes core (orange), high conserved variable (green), low conserved variable (cyan), and unique (blue) genes. The right view shows only these unique genes. Figure adapted from [171]

Interactive Pangenome Visualization Using Variant Graphs

Simon Heumos^{1,2}, Björn Geigle², Jörg Hagmann², Theodore R Gibbons², Sebastian J Schultheiss², Verena JW Schünemann³, Daniel Huson¹

¹Algorithms in Bioinformatics, University of Tübingen, Germany

²Computomics GmbH, Germany

³Department of Archaeological Sciences, University of Tübingen, Germany



Contact: agv@computomics.com



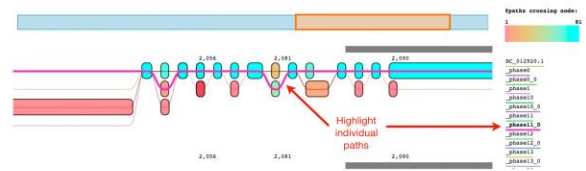
Background and Motivation

Variant graphs offer flexible ways to analyze, visualize, and explore the growing number of pangenomic data sets, without requiring the selection of any individual sequence as the global reference against which all others are anchored and compared.

The sequenceTubeMap software [1] uses D3.js [2] to quickly generate tube maps, which take advantage of the generally linear nature of most sequence graphs to create familiar and intuitive representations of sequence graphs, while still being able to display complex structural variations of arbitrary size. The current version lacks many important features of standard genome browsers, however, such as the ability to display subsections of very large graphs.

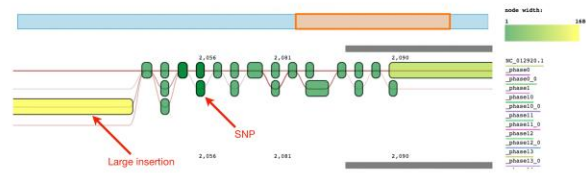
We have combined sequenceTubeMap with the vg (variation graphs) software suite, and extended both to create the Augmented Graph Viewer (AGV). AGV can be used to efficiently analyze much larger data sets than would be reasonable to render as a single image. It also provides many useful and familiar features of traditional genome browsers.

Visualize Large Populations



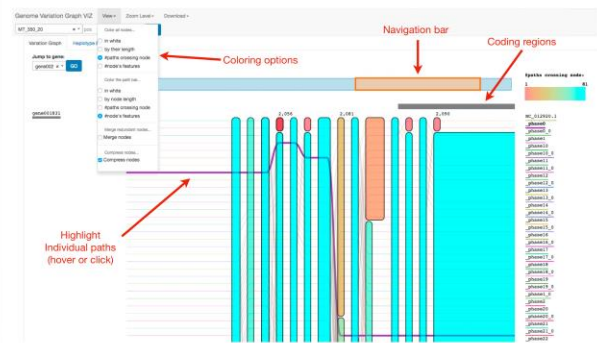
Nodes can be vertically collapsed, making it easier to interpret global complexity and identify distinct sub-populations, while still allowing the paths of individual genomes to be quickly and easily identified.

Simultaneously View Large and Small Variants



AGV computes useful statistics while generating the initial genome graph, which can then be used to color the nodes or the paths.

User Interface Overview

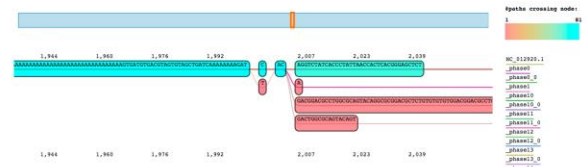


Nodes represent shared or unique sequences of variable length, through which genome paths pass to reveal complete genome sequences.

The menu provides options that allow the user to customize the visualization:

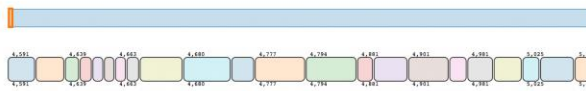
- Node coloring or path bar coloring by node sequence length, allele frequency and annotation features
- Navigation bar allows convenient navigation of large sequences and detailed inspection of local regions
- Merge consecutive nodes that have been split by e.g. annotation features
- Compress nodes, i.e. overlay all genome paths
- Zoom levels, from nucleotide level to one hundredth of the node's sequence length
- Quickly switch between multiple data sets

Zoom to Nucleotide Resolution



The AGV viewer can be zoomed in to seamlessly inspect individual SNPs, InDels, and other small-scale variations.

Infer Haplotype Blocks



AGV also precomputes haplotype blocks during the initial creation of the genome graph [4]. The haplotype blocks viewer is accessible as a separate tab in the UI window.

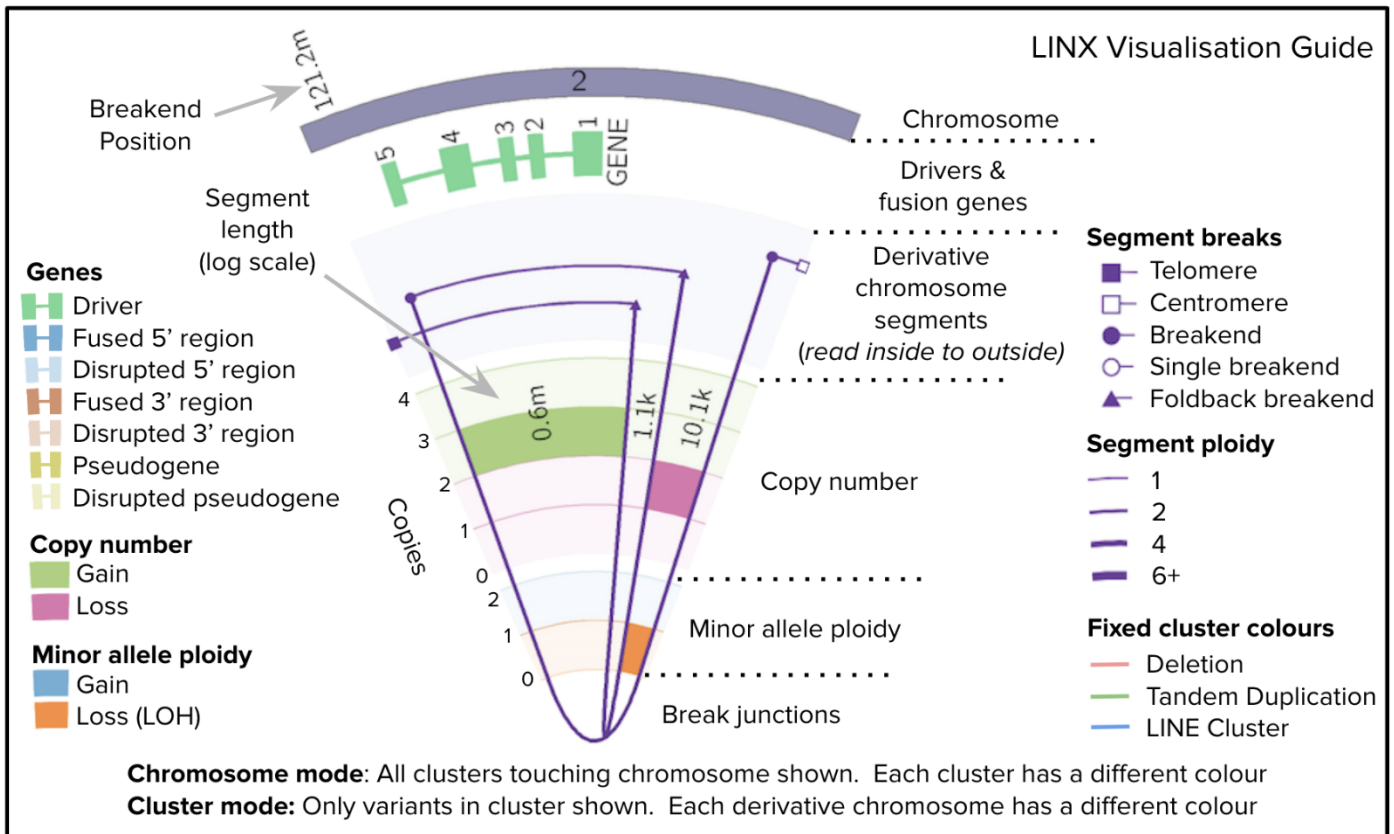
Client-Server Architecture

The lightweight AGV client renders the requested subgraph, along with a bit of the flanking regions, and requests new subgraphs from the server as needed. All other computation is handled by the AGV server, including:

- Creation and storage of the global variant graph
- Extraction of annotated subgraphs
- Computation of haplotype blocks [4] and subgraph statistics

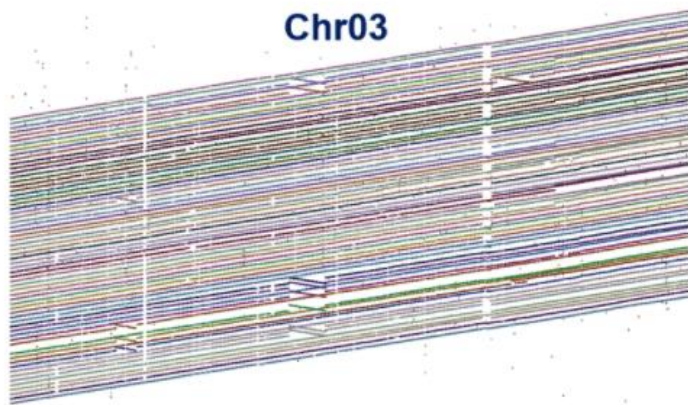
References

- [1] github.com/wolfib/sequenceTubeMap
- [2] github.com/d3/d3
- [3] github.com/vgteam/vg
- [4] Wall JD, Pritchard JK (2003) Haplotype blocks and linkage disequilibrium in the human genome. *Nat Rev Genet*, 4(8):587– 597.

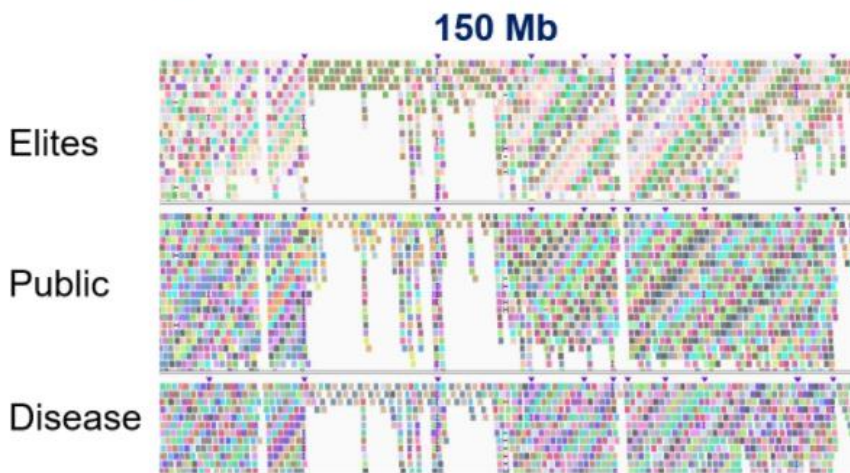


Appendix XXXI: Linx uses a circular representation with multiple visual encodings detailing SVs

Practical approaches for comparing assemblies at many levels



TagDots:
SV at chromosome scale



PANDA:
Sequence variation from
locus to gene

**68 maize lines compared
to a "reference"**

Appendix XXXII: Corteva's TagDots and PANDA are tools creating static visualizations of pangenomes at different scales; Figure from a talk given by Kevin Fengler in September 2020

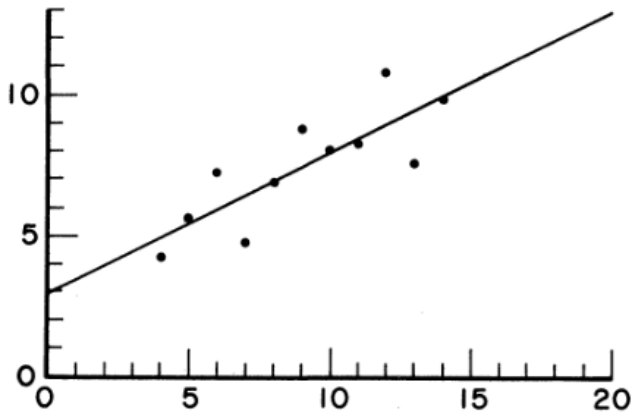


Figure 1

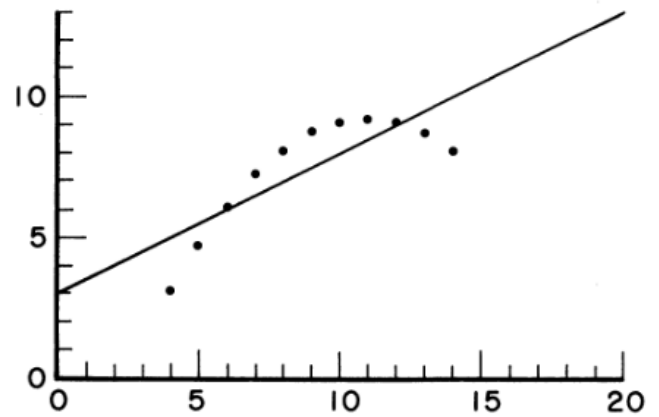


Figure 2

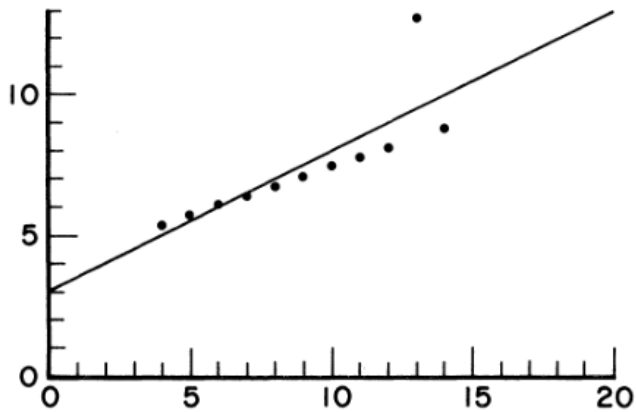


Figure 3

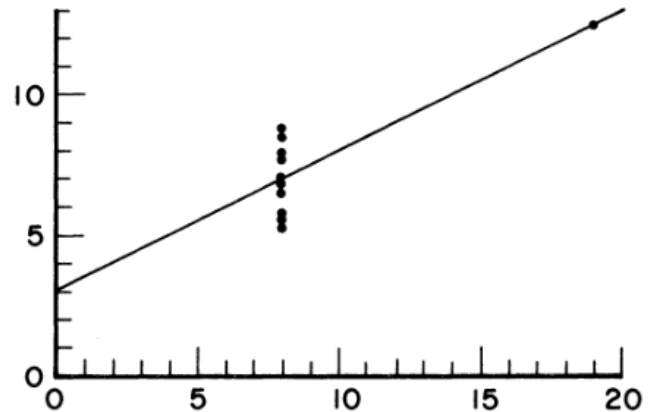
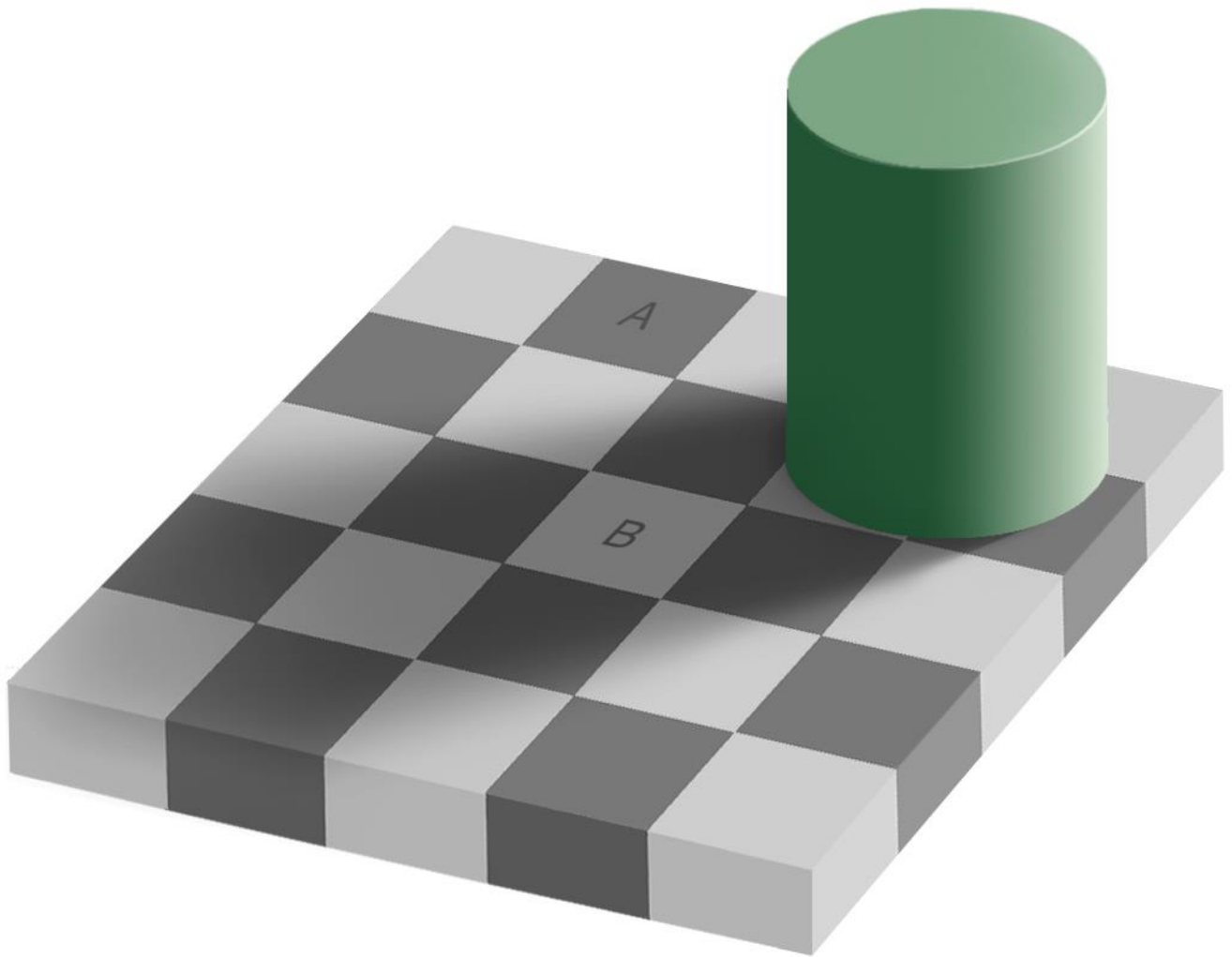


Figure 4

Appendix XXXIII: Anscombe's quartet features four visually different datasets that share the same summary statistics; Figure from [279]

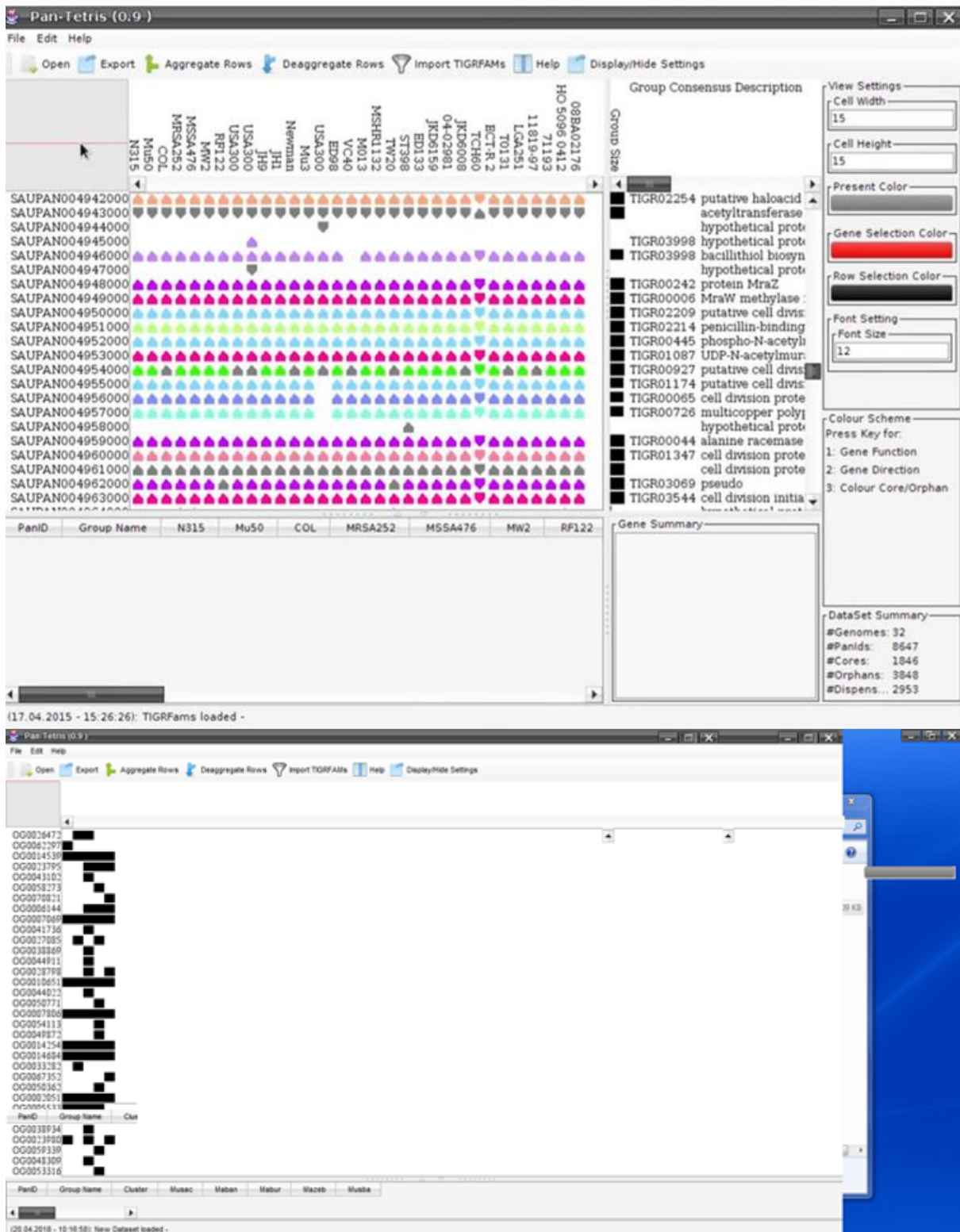
**THIS IS AN EXAMPLE OF TEXT THAT
 HT NI TUO WTTIHW NEN SAH
 BOUSTROPHEDON STYLE, WHERE
 WTTIHW YIETANETIA ERA SENI
 LEFT TO RIGHT AND RIGHT TO LEFT.**

Appendix XXXIV: Boustrophedon switches the direction of writing with every line; There are different types of boustrophedon, here the lines are mirrored horizontally, as used in Ancient Greece. Another type of boustrophedon reversed every other line both vertically and horizontally. Figure from <https://en.wikipedia.org/wiki/Boustrophedon>



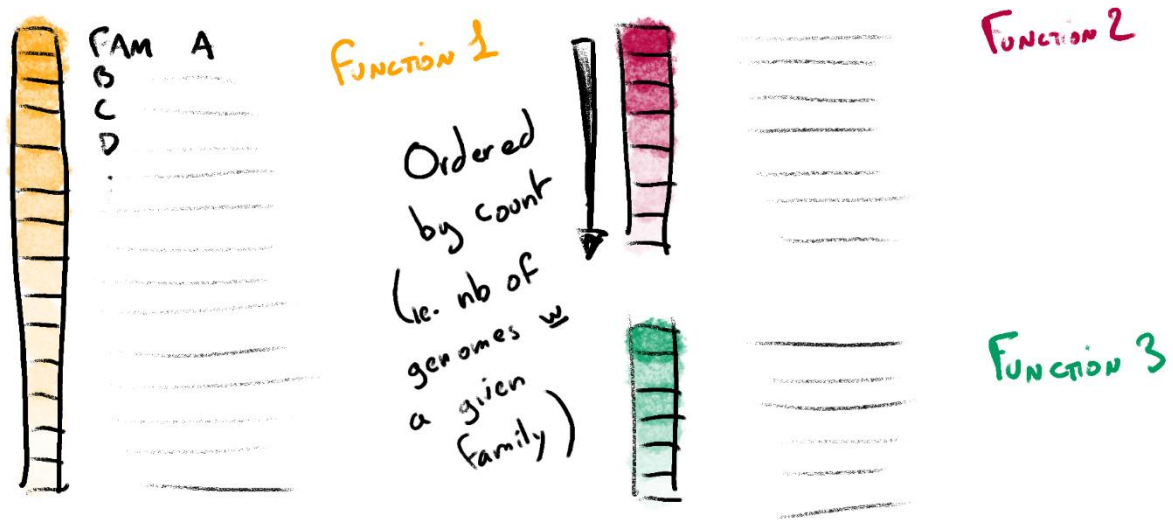
Appendix XXXV: Two identical colors can be perceived as different depending on their context; In this famous example by Adelson, squares A and B from the checker seem to be of different colors but have in fact the exact same!

II. Appendices from Panache's chapter

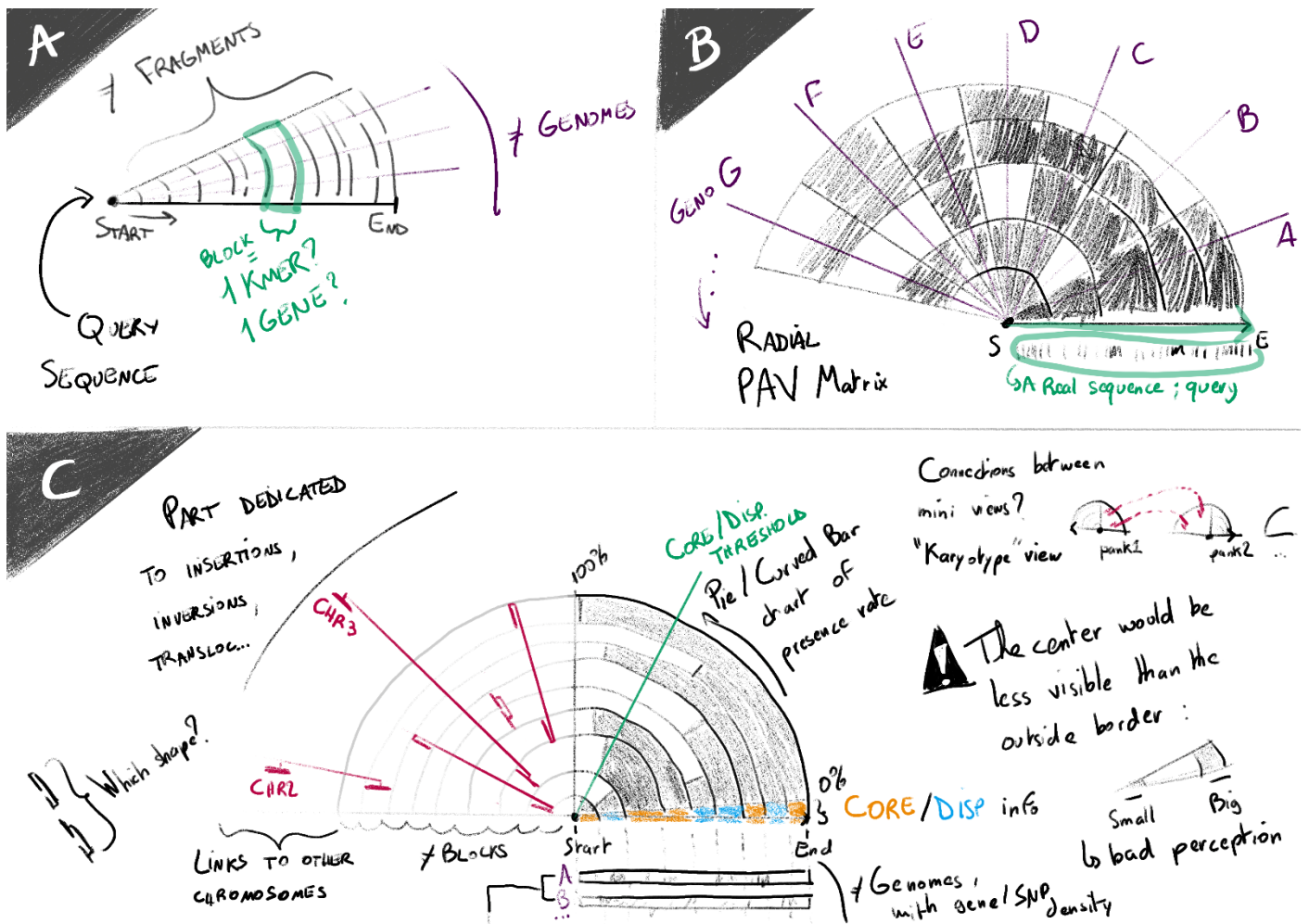


Appendix XXXVI: Pan-Tetris has different behaviors depending on the source and format of data; Data built from the unpublished tool PanGee (top) are displayed with more colors than user provided PAV matrices (bottom). Benchmark with our own data also revealed bugs from the UI.

Heatmaps of Gene Families - per Function



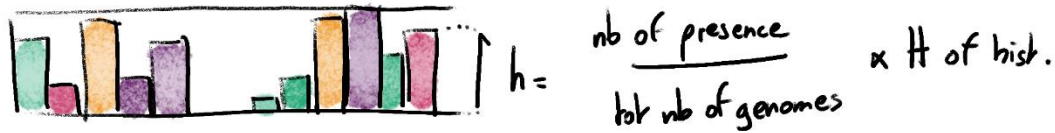
Appendix XXXVII: Visual representations of pgAtlases could have included heatmaps of clustered gene families; Based on PanViz's idea [159] and the original banana dataset, gene families could be clustered by gene families and ordered depending on the number of genomes in which they would appear. This idea was rejected as it was not space efficient and did not consider position; it would therefore not be applicable to pangomes.



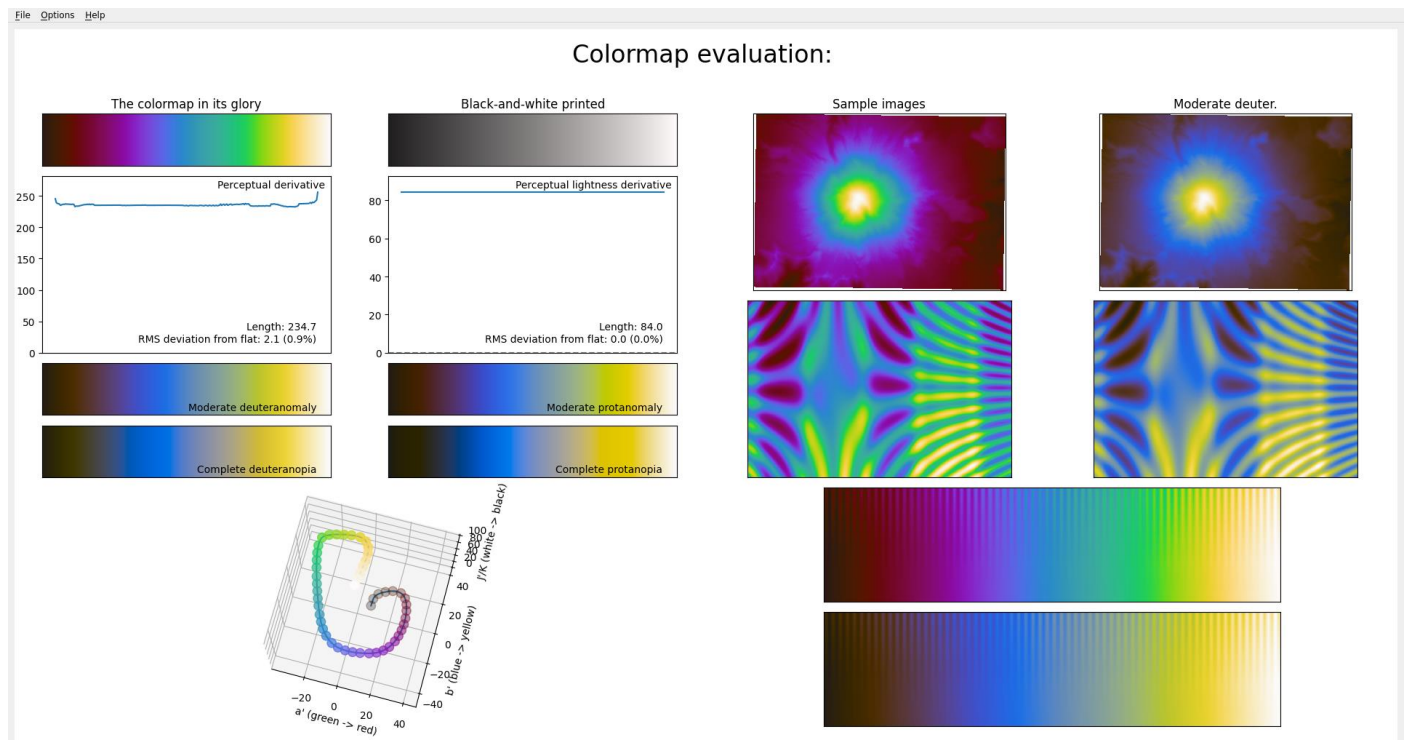
Appendix XXXVIII: I explored semi-circular representations along the same lines as anvio; A) Each panBlock was originally positioned on a queried linear coordinate system, originating from the center towards the outer part of the semi-circle. Genomes would have been distributed on radiuses. B) All genomes could have been added until forming a full circle, with a PAV matrix of the panBlocks for each. C) Alternatively,

summarized representations could have showed circular bar charts illustrating the PAV status of panBlocks, with a section dedicated to the display of observed repeats throughout the pangenome (left). This visual representation has been discarded as the curvature would have been misleading, showing different areas for equivalent elements. For example, in C the first panBlock (inner ring) almost has the same PAV status than the last panBlock (outer ring) but has a significantly smaller size, making it less visible.

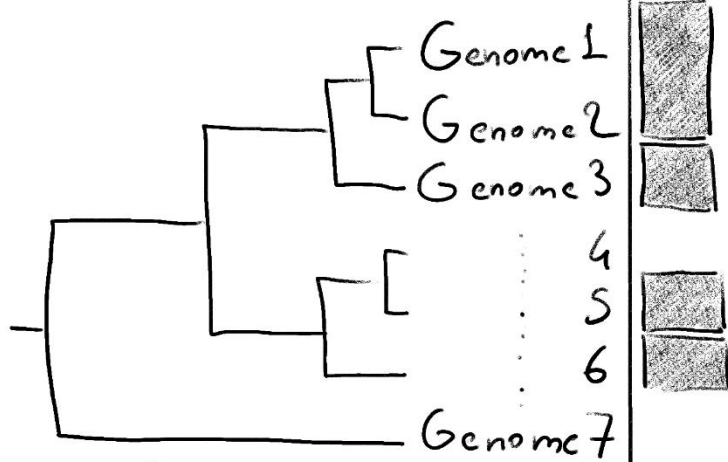
Addition of a hist of PAV:



Appendix XXXIX: The PAV matrix is represented as a histogram within the miniature of Panache; The exact detail of the PAV would not have been readable at such a small scale, the histogram enables the comparison of lengths to have a quick idea of the PAV rate of a panBlock.

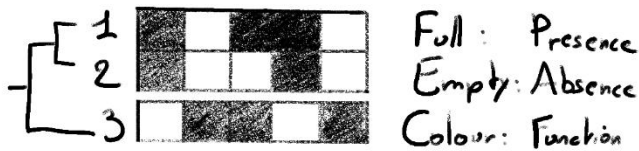


Appendix XL: The position track uses a alternative color palette to the usual rainbow; Varying through around six identifiable hues (brown, violet, blue, green, yellow, and beige), the proposed sequential color palette linearly in luminance and can be used to identify broad regions of origin along a linear axis. I used viscm [280] to create this colormap.



PAN MATRIX
as a clustered
heatmap?

1 column is associated with 1 panBlock



None All Number of genomes with the panBlock

Start End PanBlock position on the panChromosome

None Many Number of panBlocks with a similar sequence

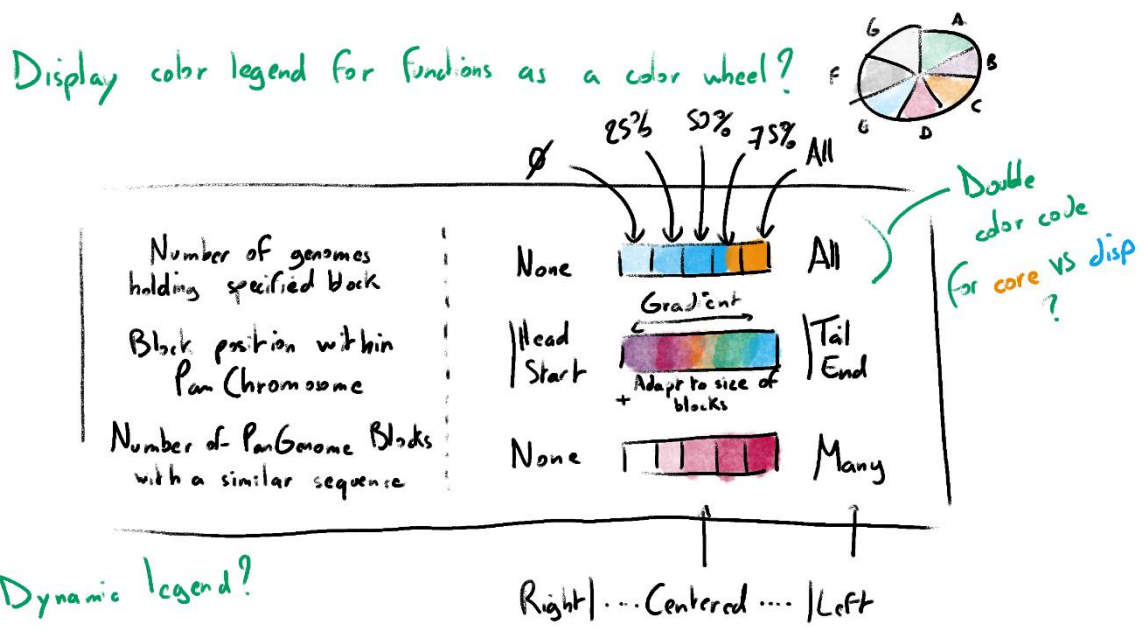
Distribution of repeated panBlocks within PanChromosomes



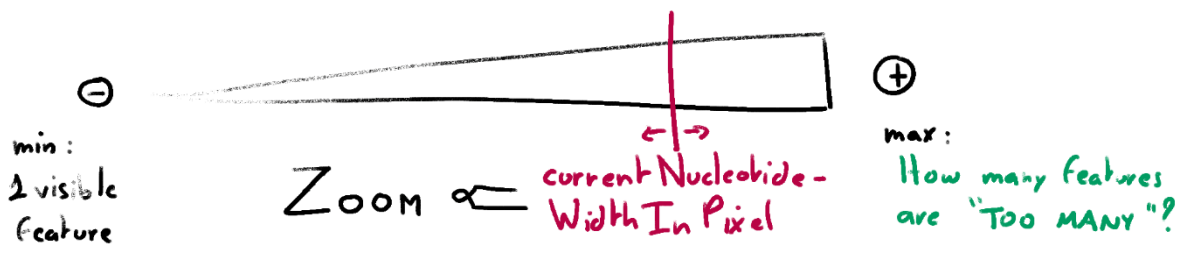
Where to put chrom names?

Better if the legend is close BUT Empty space?

Appendix XLI: The menu panel was thought to display filters option and legends; From top to bottom: the core threshold for modifying the minimal presence ratio for panBlocks to be considered as part of the core genome. A phylogenetic tree of the genomes displayed in the PAV. Static legends for the PAV matrix and the tracks below.



Appendix XLII: The color legends for the tracks were supposed to be dynamically updated



Appendix XLIII: The geometric zoom level can be modified directly within the menu panel; it was initially divided between two linear scales as to offer the possibility to view everything at once and switch to more detailed levels of zoom.

Overall principle:

"SINGLE VIEW",
K or function-specific

ordered
based on
panK
sequence

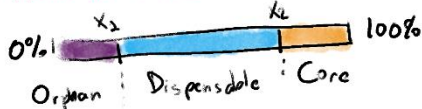


panK, as a heatmap
encoding the count of
genomes w a given block
Which block resolution?
↳ fixed? zoomable? adaptable?

COLORS:

- luminosity \propto count
(low count \rightarrow high lumin.)
- hue \Rightarrow category
↳ core / disp. / orphan...

↳ Have a SLIDER for thresholds



Monochrome version?

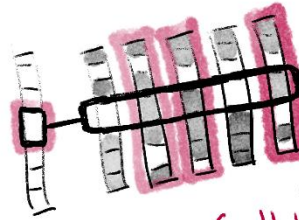


OR \rightarrow Hue change
only for CORE?

AVOID RADIAL

↳ distorts perception

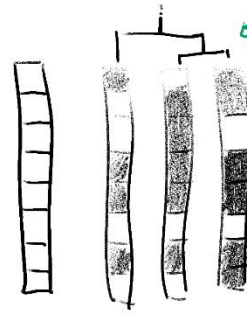
With HOVERING
↳ for highlight



↳ "click" would fix block
detail info in a dedicated
box.

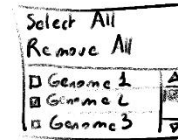
↳ Enable multiple selections?

↳ Color couldn't be
used for highlight!



Maybe have
a tree/proximity
heatmap?
↳ cf 'cluster
heatmap

↳ Implies ADD/REMOVE button



SEARCH q
↳ Search for
genome
easily
scroll bar

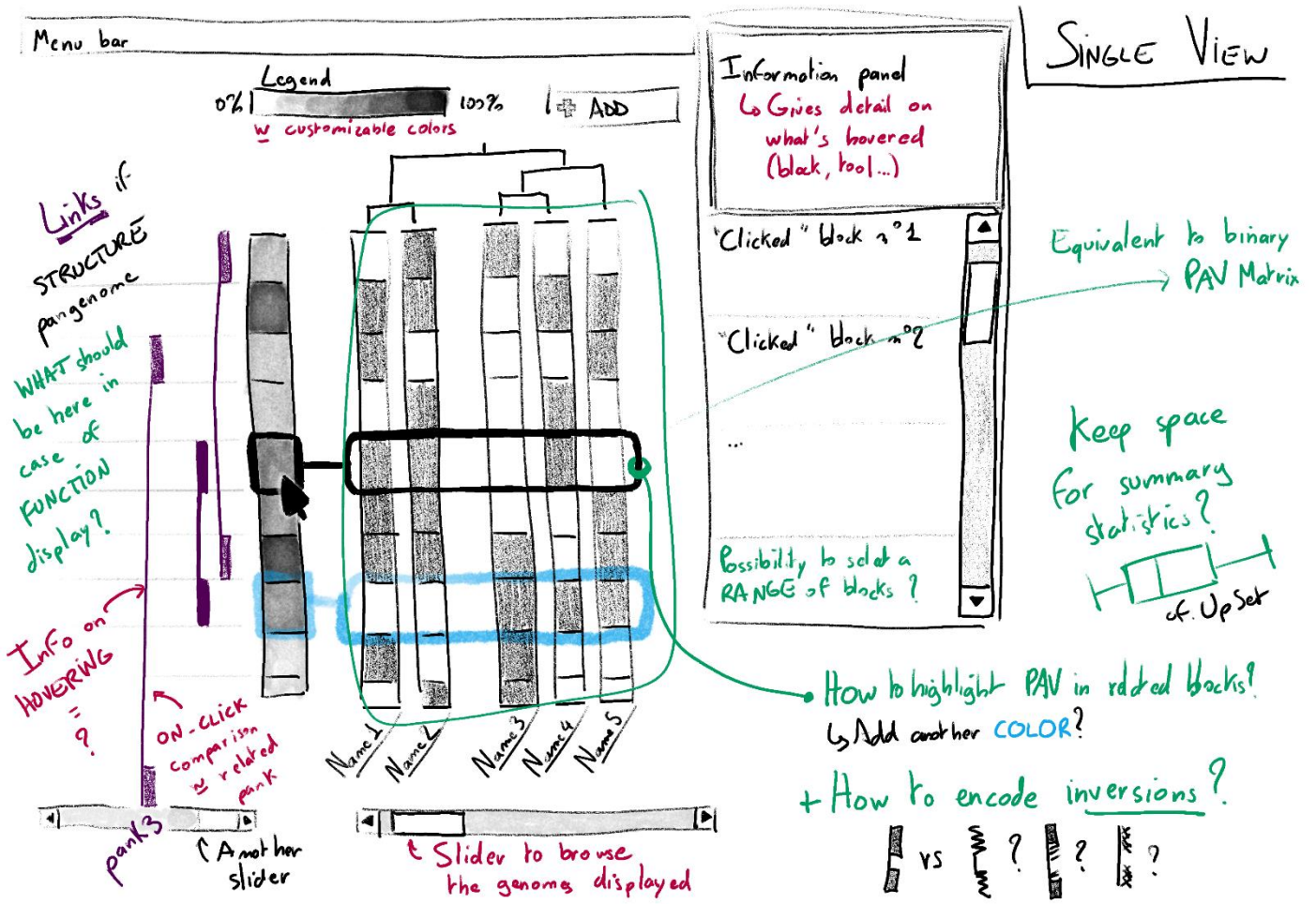
How would it be?

↳ Fatter border?

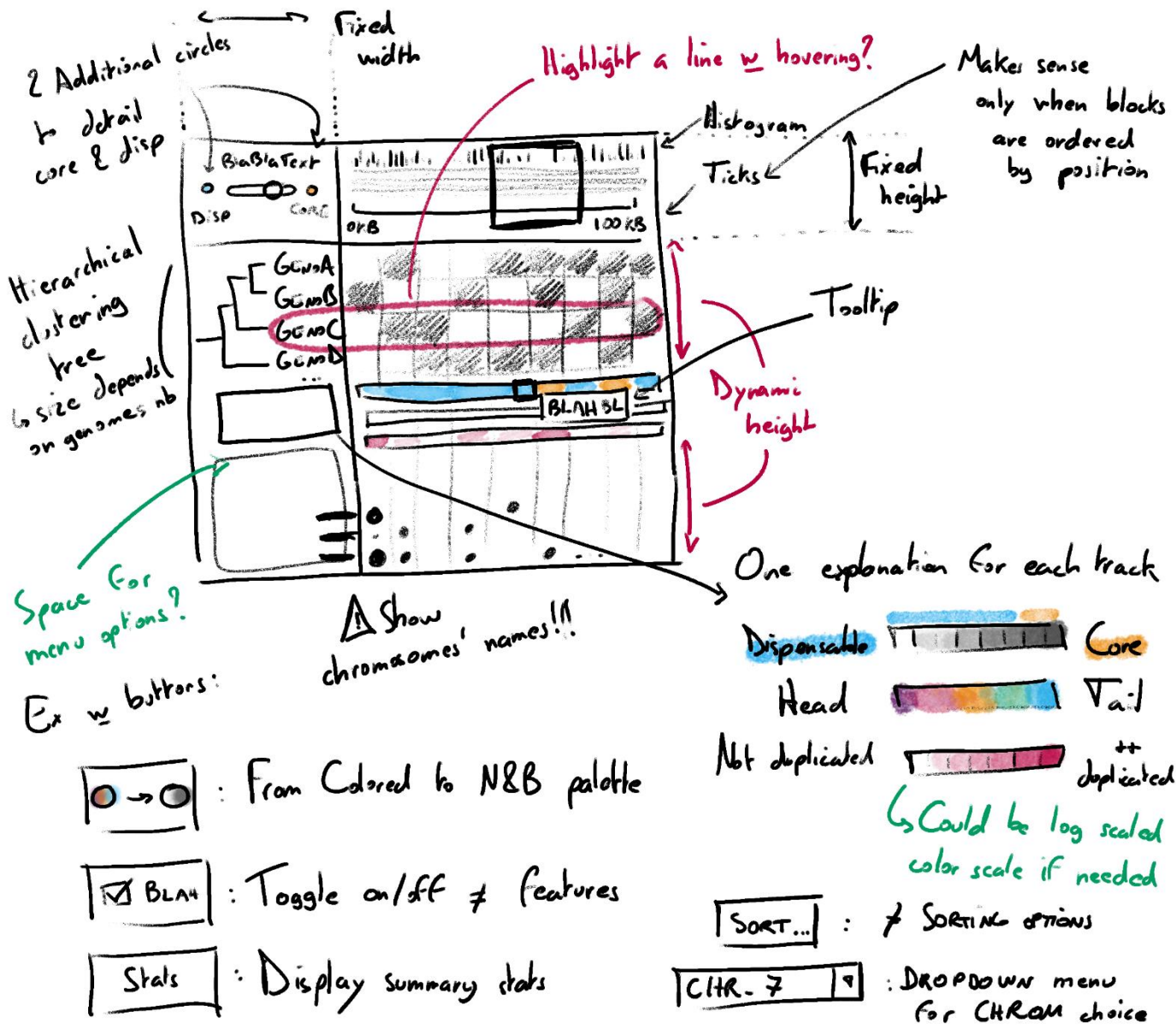
↳ Highlighted shadow?

↳ Depends on space
between genomes.
↳ "o" mark under the
matching genomes?

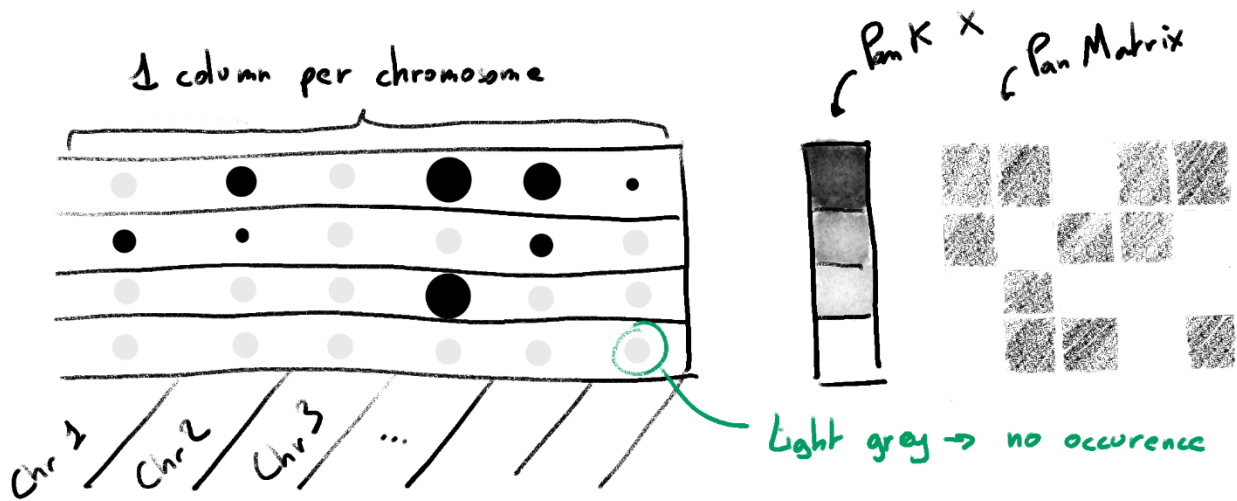
Appendix XLIV: The linearization of the semi-circular display resulted in the PAV heatmap; Its first iteration was a linearization of the semi-circular visual representation described earlier, with the genomes-as-radiuses being turned into genomes-as-columns.



Appendix XLV: The PAV matrix was originally planned to be displayed vertically; The first drafts already included sliders for panning, a phylogenetic tree, and a visual representation of cooccurrences.



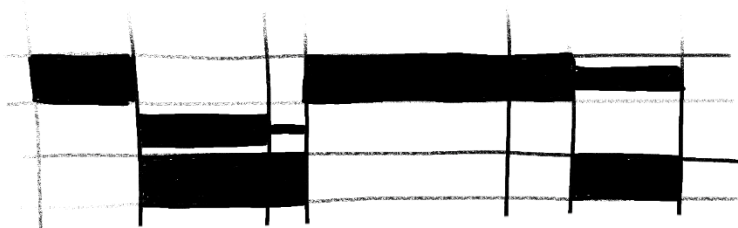
Appendix XLVI: The first version of the horizontal PAV matrix included multiple buttons for customization



- • • ● ●● Radius proportional to
- count of similar block in panchromosome
 - tot nb of occurrences of that block
 - block dimensions on screen

Appendix XLVII: Cooccurrences were originally visualized as a track and circles representing the distribution of the repetitions within the pangenome; The layout was still vertical then.

Drawing circles within blocks won't be good, prefer rectangles with variable size instead?



Pb: width could misguide, especially since comparisons should be made by columns. + differentiation between blocks

For comparisons between columns, use width variability instead:

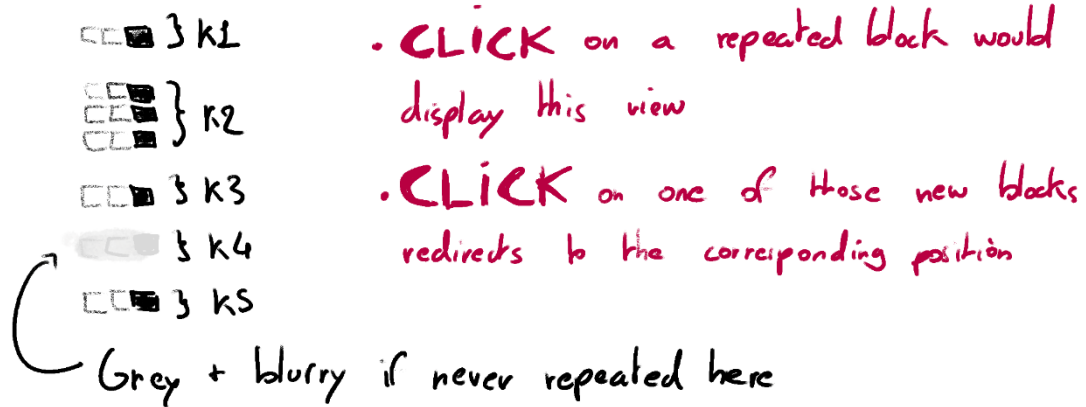


↓ Smaller height to better distinguish the lines?

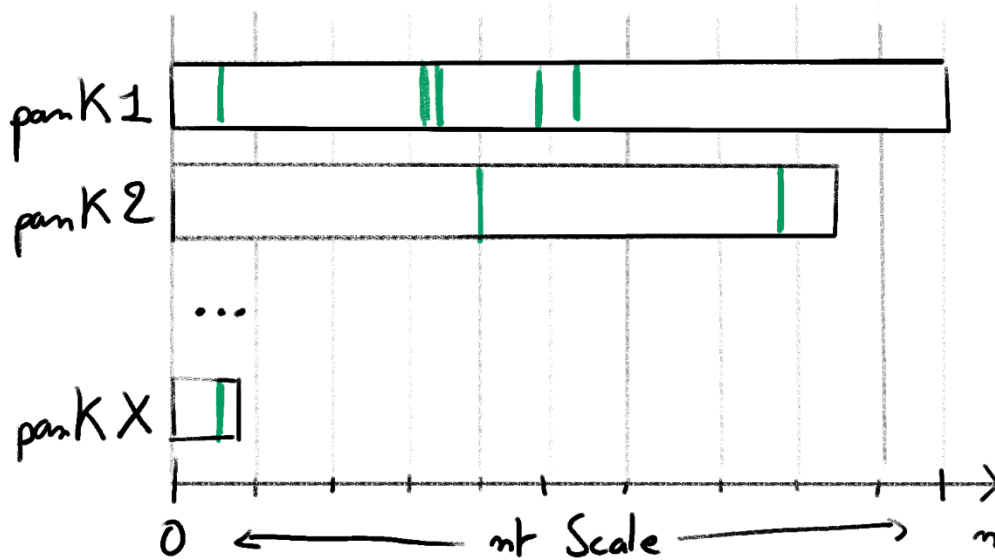


Appendix XLVIII: The detail of the distribution of cooccurrences within the pangenome is better encoded with rectangles


What dedicated representation for repeated blocks?

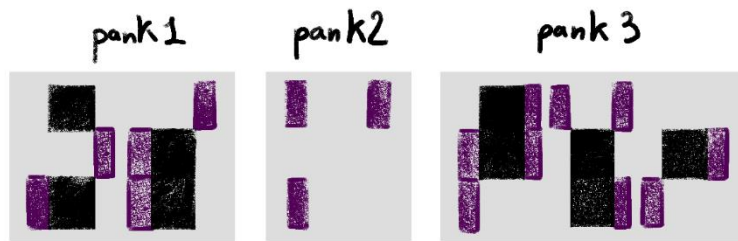


Appendix XLIX: The first draft of a view dedicated to a selected repeat would have appeared on click



Appendix L: An alternative visual representation of the repeats showed their position within panchromosomes; This representation constituted a detail view, which would have been available on click on a repeated panBlock. All green stroked represent the position of a repeat on the linear coordinate system chosen for the panchromosome.

What if neighboring blocks are represented,
as "blinds" (cf windows)? 

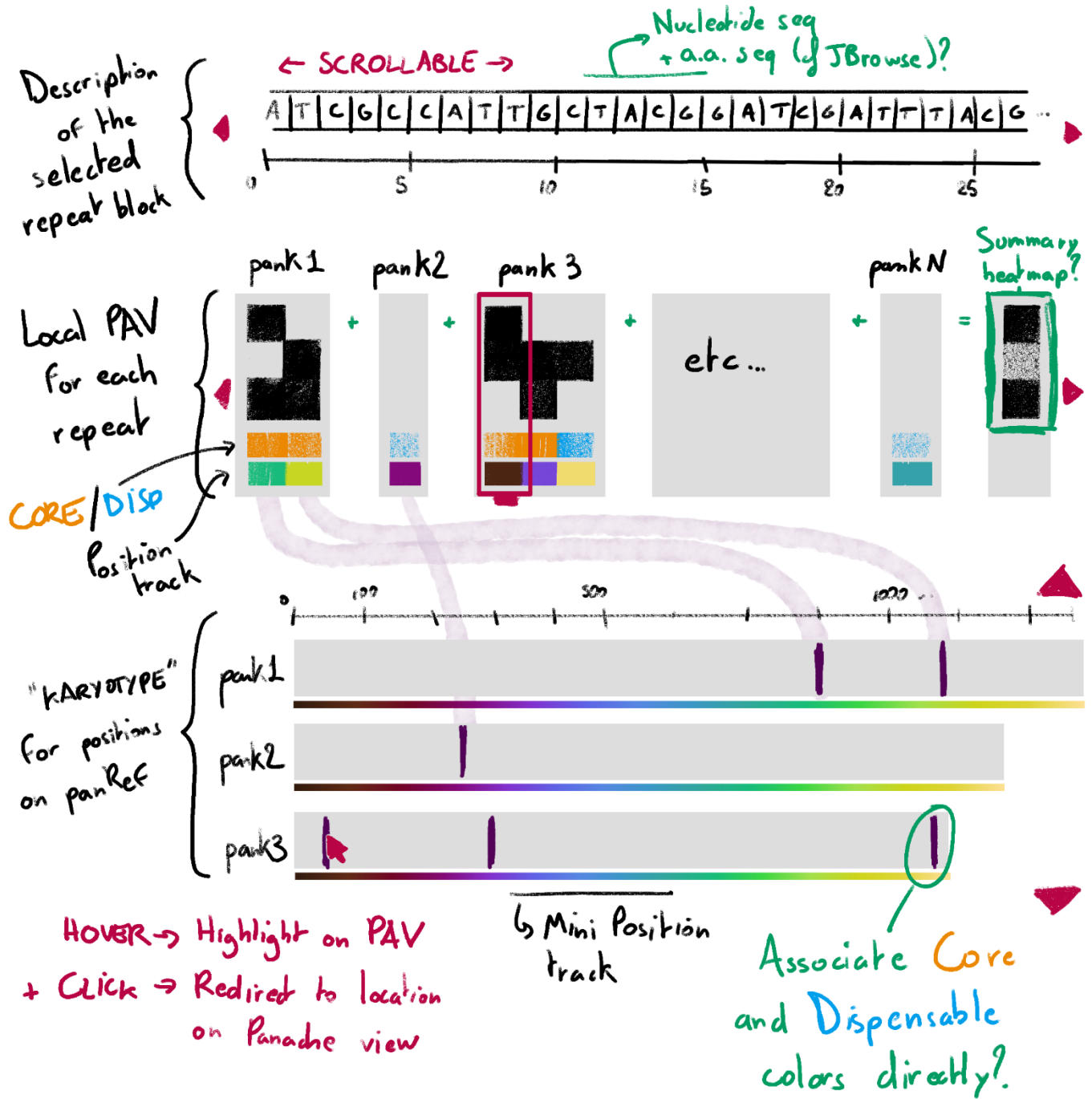


Only color blocks that also are repeat? :



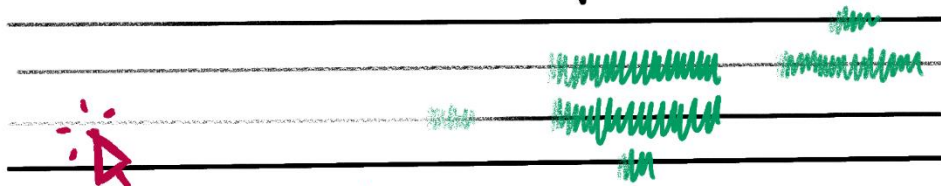
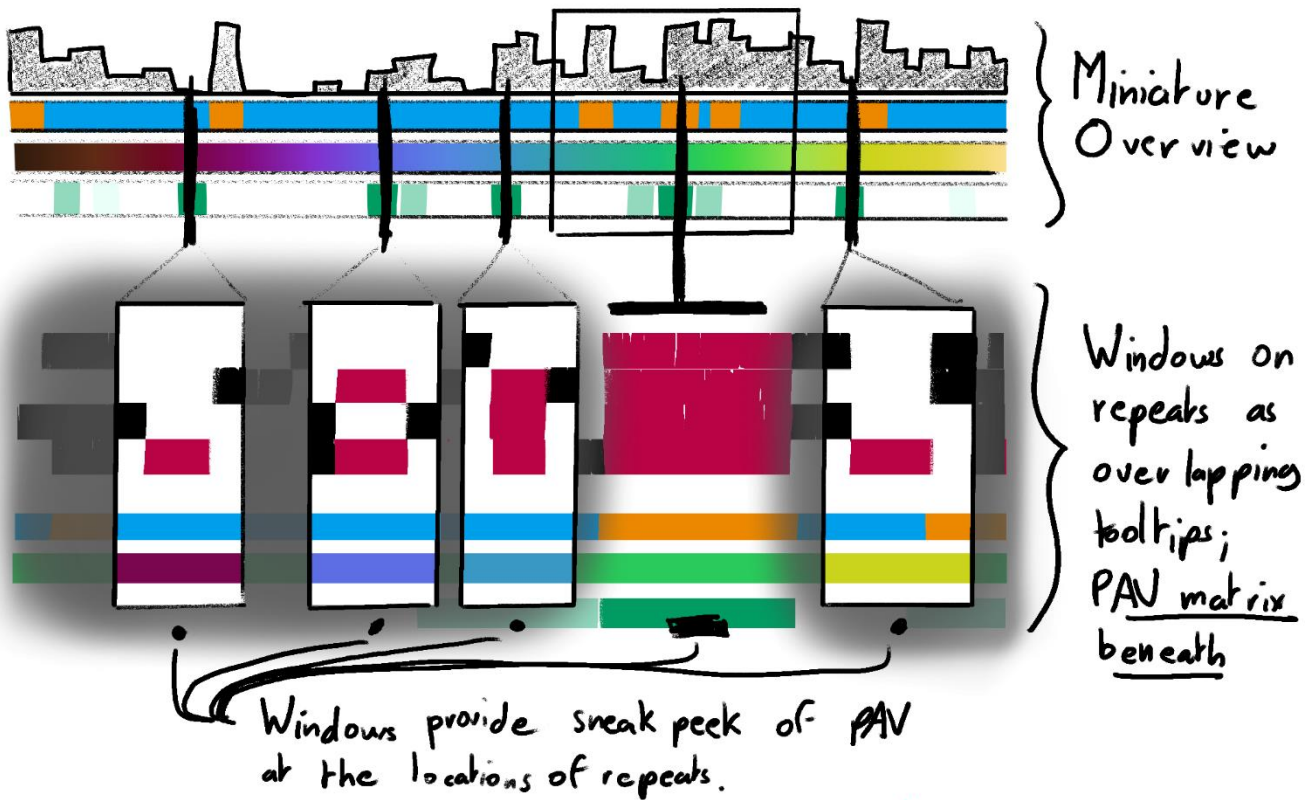
Appendix LI: A view dedicated to repeats could show the PAV status of other repeats and their context as 'windows-with-blinds'

There could be an active "REPEAT" view: **Q ZOOMABLE**



Appendix LII: A draft version of a detail view for repeated panBlocks included the visualization of the PAV status of all occurrences; This detail view was divided in three parts, from top to bottom: the detail of the nucleotide sequence of the selected panBlock, a Panache-like representation of the PAV matrix for each repeat, the position of the repeats within the different panchromosomes, as described earlier. Interactivity would be used to link the PAV matrices and the position of their respective panBlocks within the 'karyotype' view.

Multiple windows on regions with repeats?



On click change of panck ; with repeat display still active

Appendix LIII: Another option for repeats was to show them as togglable tooltip overlapping the main PAV matrix; On click on a repeat, tooltip would appear showing the position of repeats on the current panchromosome. Contrary to the previous draft, this view could not show repeats on another panchromosome, and the user would have to actively change the panchromosome on display to further explore the PAV status of these repeats.



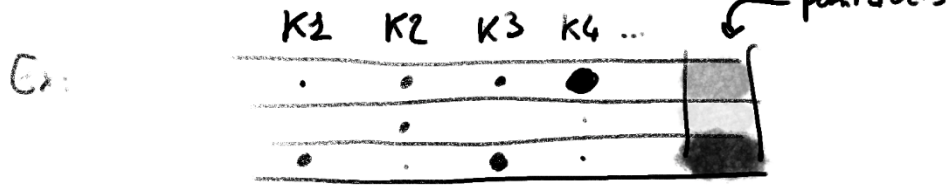
Appendix LIV: Hollow areas can overlap within the PAV matrix; The blocks spanned by these two areas are delimited by bracket on top of the PAV matrix, with different heights so than the no area is hidden.



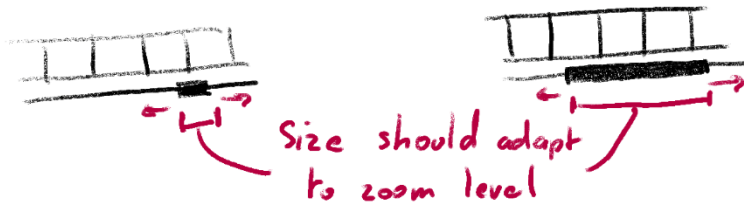
MATHIEU
& VALENTIN

DRAFT FEEDBACK

- A • Abandon lines that links similar blocks
 - ↳ Give an idea of proportions x position instead?



- B • Have a slider to browse the pan k



- C • Have a color pattern to highlight close areas when block are reordered



- D • Think about a view dedicated to conserved blocks



Appendix LV: Face to face discussions were critical during the design of the visual representations; In this discussion, we identified that A) showing links between every repeat could be abandoned; B) a slider could be used to pan through the whole PAV matrix; C) position of the panBlocks within the linear coordinate system should be visible; D) Additional visual representation were needed to visualize the presence status of repeats.

Needs and expectations for a pangenome visualization tool

There are 35 questions in this survey.

Introduction

Hello and thank you for participating in this survey.

Your answers will greatly help us in developing a pangenome visualization tool that would be useful and user-friendly, and your feedback is really appreciated.

The main goal of this survey is to determine what our development priorities should be, depending on what you, current or 'soon-to-be' member of the pangenomics community, need.

This survey takes around 15 minutes to complete, therefore please make sure that you have some spare time available.

Mandatory questions are preceded with a * . Moreover, you may find underlined elements. Some are **hyperlinks** (**<https://en.wikipedia.org/wiki/Hyperlink>**) that will redirect you to related web pages when clicking on it, while the others will show **tooltips** on hovering, providing additional information.

We will ask questions about your scientific profile, your interest in pangenomics, and especially potential functionalities that you will have to rank according to your preferences. In order to provide you with a better understanding of what could be done, you will also be offered to try a prototype of a visualization tool for pangenomes.

All data are collected for the sole scope of this survey and the development of the resulting visualization tool. If you choose to leave us your email address, we will not share it nor use it for purposes other than this study.

Scientific profile

First of all we would like to get to know you a bit more.

This will help us determine if there are needs specific to a given work field, and will also allow us to provide you with a personalized survey.

Some questions might be asked or not depending on your answers.

In which professional sector are you working ? *

🗨️ Check all that apply

Please choose **all** that apply:

Public

Private

Other:

What are your current work fields ? *

📌 Check all that apply

Please choose **all** that apply:

- Bioinformatics
- Evolutionary biology
- Molecular biology
- Population genetics
- Varietal Selection

Other:

Which kind of organism(s) are you studying ?

📌 Check all that apply

Please choose **all** that apply:

- Animals
- Bacteria
- Human
- Plants

None of the above:

Have you already used pangenomes ? *

📌 Choose one of the following answers

Please choose **only one** of the following:

- Yes
- No

Would you like to have explanation about pangenomes and the related concepts ? *

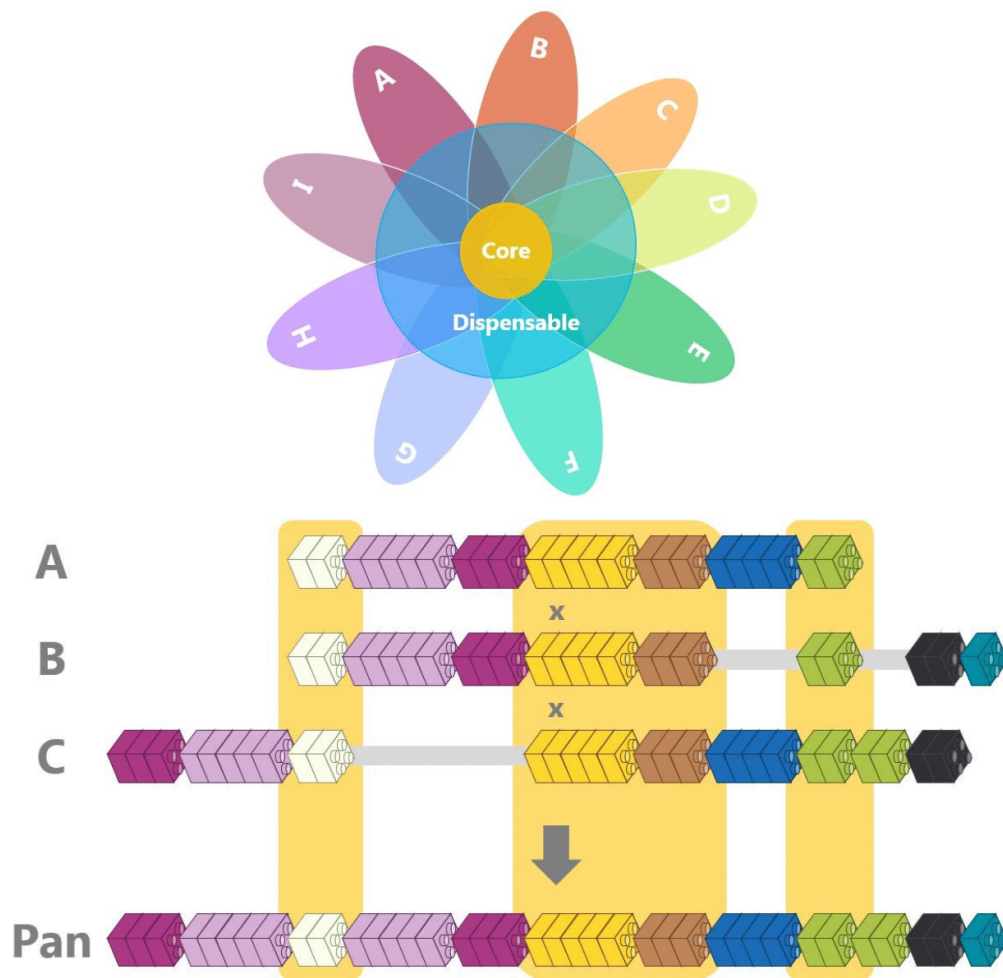
🗳️ Choose one of the following answers

Please choose **only one** of the following:

- No, thanks
- Yes, I would like a short reminder
- Yes, I would like a detailed explanation

Pangenomics Definitions

Pangenomes are complex entities that might not have exactly the same definition depending on the studies. The first definition introduces them as the full repertoire of genes within available genomes of a same species, distinguishing genes that are present in every genome (forming the “core genome”), and others that are absent in at least one of them (thus called the “dispensable genome”). Though this definition is still in use in some studies, others want to extend it to the rest of the genomic material, even without being genes.



One of the main properties of a pangenome is the classification of its elements depending on how well they are shared amongst genomes. The two main classes are elements of the **core genome**, which is said to be enriched in essential components (growth genes...), compared to those of a **dispensable genome**, as introduced earlier. Some scientists want to decline those again into sub categories depending on the level of sharing between genomes. For instance it is possible to find the terms “Core”, “Softcore”, “Shell” and “Cloud”, ranging from the always-shared items to those present in only one or two genomes (Gordon et al., 2017 (<https://www.nature.com/articles/s41467-017-02292-8>)).

For instance, this property is particularly interesting when it comes to genes that can know **Presence Absence Variations (PAVs)**, or **Copy Number Variations (CNVs)**. Indeed, **it so happens that within a single species there are individuals that might have genes, or multiple copy of a gene, while others do not have it at all!**

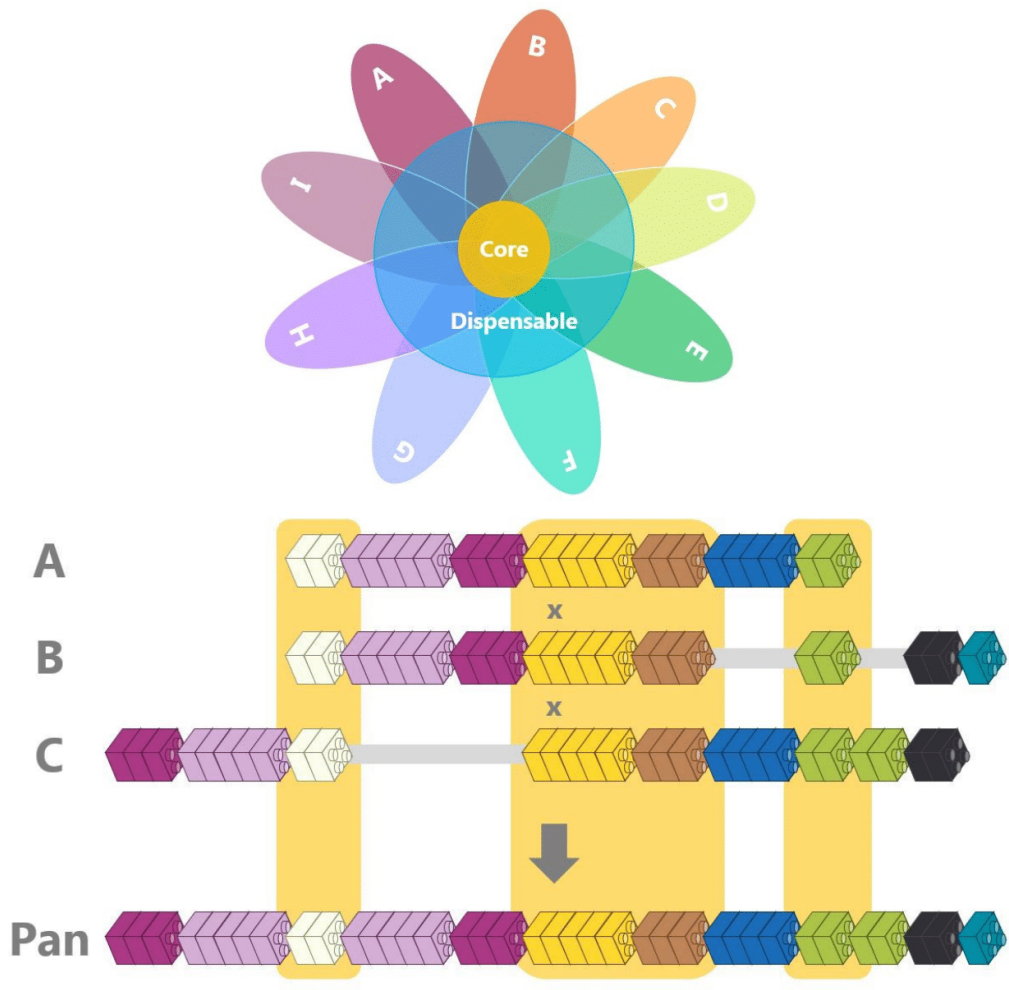
Only answer this question if the following conditions are met:

Answer was 'Yes, I would like a short reminder' at question '7 [QReminders]' (Would you like to have explanation about pangenomes and the related concepts ?)

Pangenomics is an integrative approach which aims to the assessment of every possible genomic variation within a group of closely related individuals. Although it has already often been applied to bacteria as it is its first application field (Tettelin et al., 2005 (<https://www.ncbi.nlm.nih.gov/pubmed/16172379>)), its use with larger organisms is quite recent.

Pangenomes are complex entities that **might not have exactly the same definition depending on the studies**. The first definition introduces them as the full repertoire of genes within available genomes of a same species, distinguishing genes that are present in every genome (forming the “core genome”), and others that are absent in at least one of them (thus called the “dispensable genome”). Though this definition is still in use in some studies, others want to extend it to the rest of the genomic material, even without being genes.

One simple definition that we could give is that “A pangenome is an inventory of genomic items that are shared or not within a group of related individuals”, such items being genes, gene families, blocks of nucleotides or others.



One of the main properties of a pangenome is the classification of its elements depending on how well they are shared amongst genomes. The two main classes are elements of the **core genome**, which is said to be enriched in essential components (growth genes...), compared to those of a **dispensable genome**, as introduced earlier. Some scientists want to decline those again into sub categories depending on the level of sharing between genomes. For instance it is possible to find the terms “Core”, “Softcore”, “Shell” and “Cloud”, ranging from the always-shared items to those present in only one or two genomes (Gordon et al., 2017 (<https://www.nature.com/articles>

/s41467-017-02292-8)).

For instance, this property is particularly interesting when it comes to genes that can know **Presence Absence Variations (PAVs)**, or **Copy Number Variations (CNVs)**. Indeed, **it so happens that within a single species there are individuals that might have genes, or multiple copy of a gene, while others do not have it at all!**

This can have powerful effect on the observed phenotypes of different individuals. An example of the importance of such variation exists within Indian rice, where **the presence of the SUB1 gene provides resistance to submergence** up to 20 days or more ([Mackill et al., 2012](https://www.sciencedirect.com/science/article/pii/B9780123942760000068) (<https://www.sciencedirect.com/science/article/pii/B9780123942760000068>)).

Pangenomics tends to be increasingly used, especially with the price reduction induced by the development of the latest sequencing technologies that led to the acquisition of numerous data and multiple reference genomes for a single species. This emerging field is still under active work, to improve the related tools, analysis workflow and data storage, but multiple pangenomics studies are already published, and there are many more to come.

Only answer this question if the following conditions are met:

Answer was 'Yes, I would like a detailed explanation' at question '7 [QReminders]' (Would you like to have explanation about pangenomes and the related concepts ?)

Pangenomics Background

In your opinion, what would be the best applications of pangenomes from a group of individuals ? *

🗳️ Check all that apply

🗳️ Please select at most 4 answers

Please choose **all** that apply:

- Comparison of the genomic content of individuals
- Detection of SNPs or genes of interest
- Exploration of the genetic diversity of a population
- Exploration of the genomic content within a given region
- Identification of stable and unstable genomic region
- Reads alignment
- Studying polymorphisms of transposable elements
- Studying the impact of Structural Variations
- Studying the presence/absence status of a given gene
- No particular application

Other:

Have you already tried pangenomes visualization tools, and if so, which ones ? *

Only answer this question if the following conditions are met:

Answer was 'Yes' at question '6 [QCurrentUseOfPan]' (Have you already used pangenomes ?)

🗨 Check all that apply

Please choose **all** that apply:

- Anvi'o
- Augmented Graph Viewer
- MoMI-G
- odgi
- PanTetris
- PanViz
- PanX
- RPan
- None
- Other:

Panache Screencast

We have already developed the prototype below:



- A column represents information about a pangenomic block (eg. gene, nucleotidic sequence, TE)
 - Those blocks are similar to the Locally Collinear Blocks (LCBs) from multiple sequence alignments (<http://darlinglab.org/mauve/user-guide/introduction.html>)
- The idea is to provide a linear representation of a pangenome, in a way similar to a pangenome browser
- For now, there are three tracks of information: **Core /Dispensable** status; Position; Number of repetitions
- Additional information can be encoded within the Presence/Absence Matrix (eg. colored Gene Ontology)

As it is web based, **it is totally possible to try it right now without any installation**, with fake data files created for the occasion but also with real pangenomic data of Brassica napus, obtained from Golicz et al. (<http://brassicagenome.net/databases.php>).

Doing so might help you better understand what a pangenome visualization could be (another possible representation would be for instance a genome graph).

This trial is facultative, therefore **you are completely free to give it a try or to continue directly with the rest of the survey!**

Would you like to give this prototype a try now? *

🗳️ Choose one of the following answers

Please choose **only one** of the following:

Yes

No

Caution : If you choose to skip this trial, you won't be able to try it later in this survey.

Panache Test

Link to the visualization of *Brassica napus* pangenomic data (panchromosomes up to number 10, from both A and C sub-genomes), obtained and adapted from data from Golicz et al., available [here \(http://brassicagenome.net/databases.php\)](http://brassicagenome.net/databases.php). The visualization may take a dozen of seconds to load.

You may also want to scroll through the different genomes. To do this it is possible to use the hidden slider on the righthmost part of the window - just hover your mouse pointer there and it will appear.

[Brassica napus pangenome \(https://meerketeer.ird.fr/PanacheNapus\)](https://meerketeer.ird.fr/PanacheNapus)

Only answer this question if the following conditions are met:

((QAccessPrototype.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/5/qid/10) == "A1"))

Link to the visualization of a fake pangenome data file, with no biological meaning, showing all the implemented functionalities:

[Fake pangenome data \(https://meerketeer.ird.fr/PanacheFake\)](https://meerketeer.ird.fr/PanacheFake)

Only answer this question if the following conditions are met:

Answer was 'Yes' at question '13 [QAccessPrototype]' (Would you like to give this prototype a try now?)

On a scale from 1 ("not at all") to 5 ("extremely"), how much are you interested in such a visualization ? *

Only answer this question if the following conditions are met:

Answer was 'Yes' at question '13 [QAccessPrototype]' (Would you like to give this prototype a try now?)

Please choose **only one** of the following:

- 1
- 2
- 3
- 4
- 5

Amongst the following missing functionalities, which ones do you want to see the most? *

Only answer this question if the following conditions are met:

Answer was 'Yes' at question '13 [QAccessPrototype]' (Would you like to give this prototype a try now?)

🗨 Check all that apply

Please choose **all** that apply:

- Changing colors
- Clicking on blocks to find more information
- Comparing repeated blocks
- Finding a 'help' screen
- Finding functional annotation
- Reordering genomes
- Rotating the main view
- Seeing multiple regions at once
- Trying it with my own data
- Zooming in to see the nucleotides

Other:

Transition to Functionalities

We will now ask you to choose and rank different potential functionalities that you may like to have within a pangenome visualization tool. Please note that they are unrelated to answers you may have given while trying our prototype.



The functionalities will be divided in 5 categories : **Research, Analysis advice, Interoperability, Navigation, Customization.**

Moreover, if you think of an interesting functionality while ranking, you are welcome to write it in the text box under the ranking question.

Functionalities - RESEARCH

About data selection depending on different criteria

Please rank, within the following, the functionalities your are interested in, in your order of preference:

❗ Please select from 1 to 7 answers.

Please number each box in order of preference from 1 to 7

Please choose at least 1 items.

Filter data

Find samples owning a combination of genes within a list

Save/Load search criterias

Search based on keywords (eg. gene function, IDs)

Selection of a region per genomic position

Selection of a region per nucleotidic sequence

Sort data according to different characteristics

Double-click or drag-and-drop items of your liking in the left list to move them to the right - your highest ranked item should be at the top, and the lowest at the bottom

If those options made you think of other interesting features that do not appear here, please write them down in here so that we know what you are looking for:

Please write your answer here:

Functionalities - RESEARCH - Filters

You seem interested in filters. In which of the following filtering criteria would you be particularly interested? *

Only answer this question if the following conditions are met:

((QRESEARCH_1.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A1") or (QRESEARCH_2.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A1") or (QRESEARCH_3.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A1") or (QRESEARCH_4.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A1") or (QRESEARCH_5.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A1") or (QRESEARCH_6.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A1") or (QRESEARCH_7.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A1"))

🗨 Check all that apply

🗨 Please select at most 4 answers

Please choose **all** that apply:

- Genomic items depending on a Presence and/or Absence threshold
- Metabolic pathway
- Ontology (Gene Ontology, Mapman ontology...)
- Phenotype (one or many)
- Regions of a given size with specific content
- Selection of a sub-genome of a polyploid species
- Other:

Functionalities - RESEARCH - Sorting

Sorting your data seems important to you. In which of the following sorting criteria would you be particularly interested?

*

Only answer this question if the following conditions are met:

((QRESEARCH_1.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A2") or (QRESEARCH_2.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A2") or (QRESEARCH_3.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A2") or (QRESEARCH_4.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A2") or (QRESEARCH_5.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A2") or (QRESEARCH_6.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A2") or (QRESEARCH_7.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/9/qid/19) == "A2"))

🗨 Check all that apply

🗨 Please select at most 4 answers

Please choose **all** that apply:

- Classification by Gene Ontology
- Manual
- Number of repetitions
- Phylogeny
- Presence Absence patterns similarity
- Presence Absence ratio of the genomes
- Presence Absence ratio of the genomic items
- Other:

Functionalities - ANALYSIS ADVICE

About tips and information to display in order to help the analysis

Please rank, within the following, the functionalities that interest you in your order of preference:

🗨 Please select from 1 to 6 answers.

Please number each box in order of preference from 1 to 6

Please choose at least 1 items.

Accessing signature sequence(s) of a selected group of genomes

Automatic screening and focus on region(s) of interest

Fast visual identification of core/dispensable regions

Fast visual identification of repeated regions

Seeing the genomic context of a region

Statistics and numbers to display

Double-click or drag-and-drop items of your liking in the left list to move them to the right - your highest ranked item should be at the top, and the lowest at the bottom

If those options made you think of other interesting features that do not appear here, please write them down in here so that we know what you are looking for:

Please write your answer here:

Functionalities - ANALYSIS ADVICE - Statistics

You value the display of statistics and numbers about your data. In which of the following ones would you be particularly interested? *

Only answer this question if the following conditions are met:

((QADVICE_1.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/10/qid/22) == "A2") or (QADVICE_2.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/10/qid/22) == "A2") or (QADVICE_3.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/10/qid/22) == "A2") or (QADVICE_4.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/10/qid/22) == "A2") or (QADVICE_5.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/10/qid/22) == "A2") or (QADVICE_6.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/10/qid/22) == "A2"))

🗨 Check all that apply

🗨 Please select at most 4 answers

Please choose **all** that apply:

- % of homology within genomic items
- % of presence of a given function
- Core/Dispensable status of categories of ontologies
- Core/Dispensable ratio within a displayed region
- Number of GC per region
- Number of genes per genomic item
- Number of proteins per genomic item

Other:

Functionalities - INTEROPERABILITY

About metadata and links with other information system

Please rank, within the following, the functionalities your are interested in, in your order of preference:

🗨 Please select from 1 to 5 answers.

Please number each box in order of preference from 1 to 5

Please choose at least 1 items.

Addition of a layer of functional annotation

Addition of a phylogenetic tree to cluster genomes

Export of publishable images

Export of selected data (depending on filters...)

Links to other tools to get more detailed information

Double-click or drag-and-drop items of your liking in the left list to move them to the right - your highest ranked item should be at the top, and the lowest at the bottom

If those options made you think of other interesting features that do not appear here, please write them down in here so that we know what you are looking for:

Please write your answer here:

Functionalities - NAVIGATION

About dynamic functionalities for visual ergonomoy and the exploration of displayed data

Please rank, within the following, the functionalities your are interested in, in your order of preference:

❗ Please select from 1 to 8 answers.

Please number each box in order of preference from 1 to 8

Please choose at least 1 items.

All information in one window

Changes of scale (karyotype, blocks, sequences...)

Changing reference

Display of multiple parts of the pangenome at the same time

Focus on every repetition of a genomic item

Mini-map to link different scales together

Possibility to merge or split blocks/genomic items

Reordering, toggling, clustering genomes to display

Double-click or drag-and-drop items of your liking in the left list to move them to the right - your highest ranked item should be at the top, and the lowest at the bottom

If those options made you think of other interesting features that do not appear here, please write them down in here so that we know what you are looking for:

Please write your answer here:

Functionalities - NAVIGATION - Different scales

You value changes of scale or levels of details. In which of the following ones would you be particularly interested? *

Only answer this question if the following conditions are met:

((QNAVIGATION_1.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1") or (QNAVIGATION_2.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1") or (QNAVIGATION_3.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1") or (QNAVIGATION_4.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1") or (QNAVIGATION_5.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1") or (QNAVIGATION_6.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1") or (QNAVIGATION_7.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1") or (QNAVIGATION_8.NAOK (/limesurvey/index.php/admin/questions/sa/view/surveyid/741912/gid/12/qid/24) == "A1"))

- 🚫 Check all that apply
 - 🚫 Please select at most 2 answers
- Please choose **all** that apply:

- Nucleotidic scale, allowing to see SNPs
- Pangenomics blocks of PAV - small variations appearing
- Pangenomics blocks of PAV - small variations hidden
- "Karyotype"-level view of Structural Variations
- Other:

Functionalities - CUSTOMIZATION

About how to display data

Please rank, within the following, the functionalities your are interested in, in your order of preference:

❗ Please select from 1 to 8 answers.

Please number each box in order of preference from 1 to 8

Please choose at least 1 items.

Adjust the sizes of displayed elements

Choose the level of detail to display for a given scale

Choose tracks/information layers to display

Create/change colour palettes for displayed elements

Creation of more categories than core VS dispensable

Custom core/dispensable threshold

Genome graph representation

Rotate the views

Double-click or drag-and-drop items of your liking in the left list to move them to the right - your highest ranked item should be at the top, and the lowest at the bottom

If those options made you think of other interesting features that do not appear here, please write them down in here so that we know what you are looking for:

Please write your answer here:

File Formats

You have already worked with pangenomes. What pangenome file formats do/did you like to use ? *

Only answer this question if the following conditions are met:

Answer was 'Yes' at question '6 [QCurrentUseOfPan]' (Have you already used pangenomes ?)

🗨 Check all that apply

Please choose **all** that apply:

- FASTA
- Graphical Fragment Assembly - GFA
- Multiple Alignment Format - MAF
- Presence absence matrix (or a derivative)
- Tab Separated Value, or Comma Separated Value - TSV, CSV
- vg - variation graph (and other related formats : xg...)

Other:

You can have supplementary information about each format by hovering your mouse on it.

What file format are you at ease working with ? *

Only answer this question if the following conditions are met:

Answer was 'No' at question '6 [QCurrentUseOfPan]' (Have you already used pangenomes ?)

🗖 Check all that apply

Please choose **all** that apply:

- Tab Separated Value, or Comma Separated Value - TSV, CSV
- Excel files
- Graphical Fragment Assembly - GFA
- FASTQ
- FASTA
- Variant Call Format - VCF
- Browser Extensible Data - BED
- General Feature Format - GFF
- Presence absence matrix
- I have no idea
- Other:

You can have supplementary information about each format by hovering your mouse on it.

Thank you

Thank you very much for completing our survey this far, your answers will greatly help us delivering a tool fit to the needs and expectations of the pangenome community !

If you wish to be informed of the results of the survey and our incoming work, you are welcome to leave us your email adress.

We will make sure to give you some updates.

Please write your answer here:

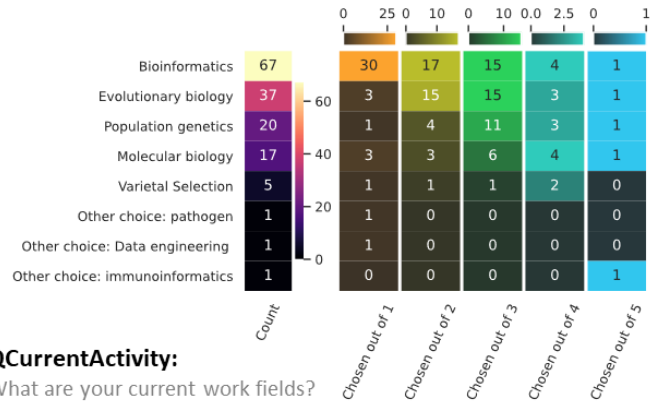
You have answered all the questions and fully completed this survey.

If you would like to ask direct questions or provide us with your feedback, please write us a mail at **eloi.durant [at] ird.fr**

01/04/2020 – 01:04

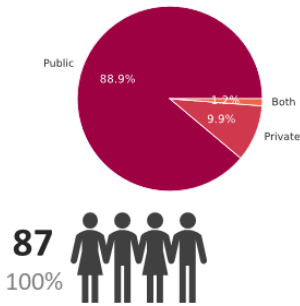
Submit your survey.

Thank you for completing this survey.



QCurrentActivity:

What are your current work fields?

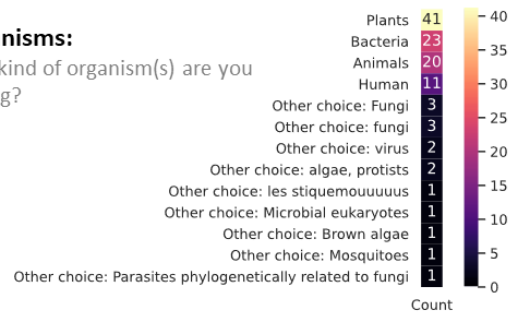


QPublicPrivate:

In which professional sector are you working?

QOrganisms:

Which kind of organism(s) are you studying?

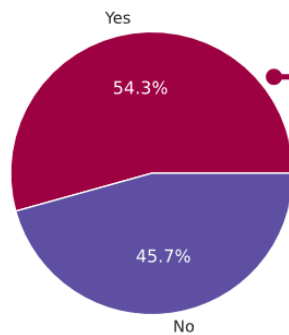


Appendix LVII: Survey results 1/11 – Profile of the respondents

; Out of 587 views, only 87 respondents answered at least one question of the survey. The number of respondents decreased over time, as illustrated by the icon at the bottom left, updated for each following slides. For questions with one or more possible answers, as in QCurrentActivity, the counts are displayed in different columns depending on how many answers the respondents provided. For example, here only one person gave 5 answers at once for that question, detailed in the 'Chosen out of 5' column.

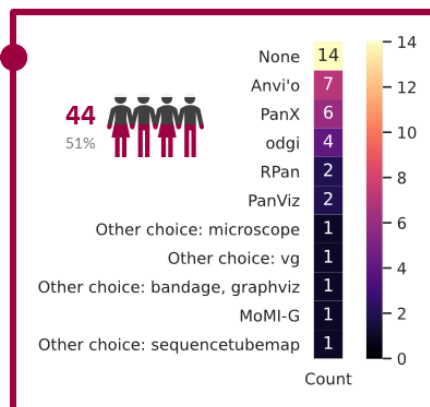
QCurrentUseOfPan:

Have you already used pangenomes?



QVisToolTried:

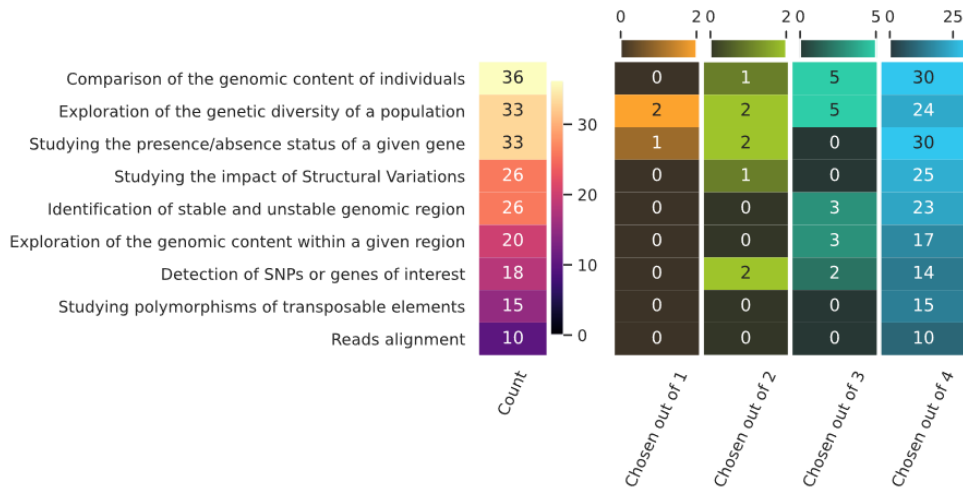
Have you already tried pangenome visualization tools?



Appendix LVIII: Survey results 2/11 – Pangenomics background; Users with previous experience of pangenomes could fill in the visualization tools they had already used

QDesiredUseOfPan:

In your opinion, what would be the best applications of pangenomes from a group of individuals?

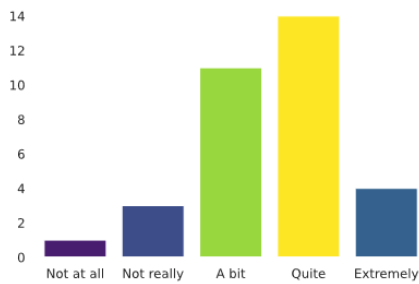


61
70%

Appendix LIX: Survey results 3/11 – Envisioned application field of pangenomics

QPanacheInterest:

How much are you interested in such a visualization?

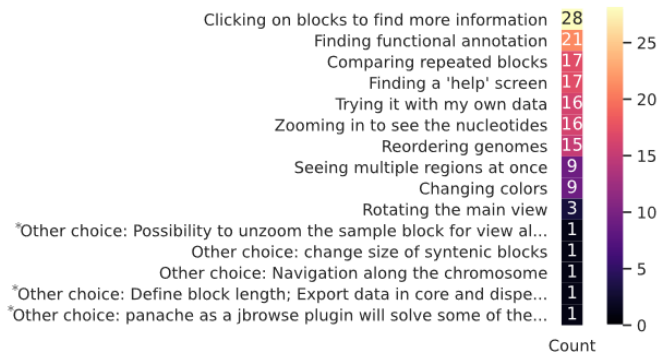


33
38%

Appendix LX: Survey results 4/11 – Feedback on Panache's prototype

QMissingFunction:

Amongst the following missing functionalities, which ones do you want to see the most?

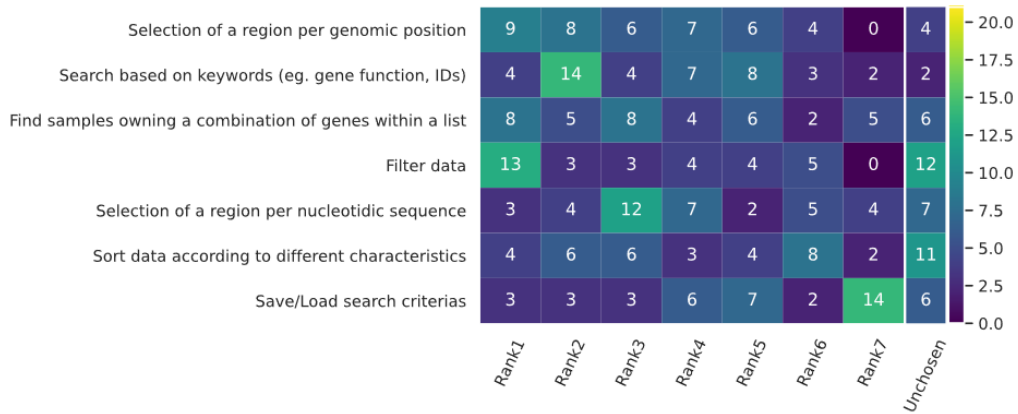


*Here are the 'other' propositions, full length:

- Possibility to unzoom the sample block for view all samples
- Define block length; Export data in core and dispensable compartments; Hide/Show genome to check pan-genome status; Graph to show size of core and dispensable compartments when the minimal presence ratio is changed
- Panache as a jbrowse plugin will solve some of the requested features

QRESEARCH:

Please rank, within the following, the functionalities you are interested in:



44

51%



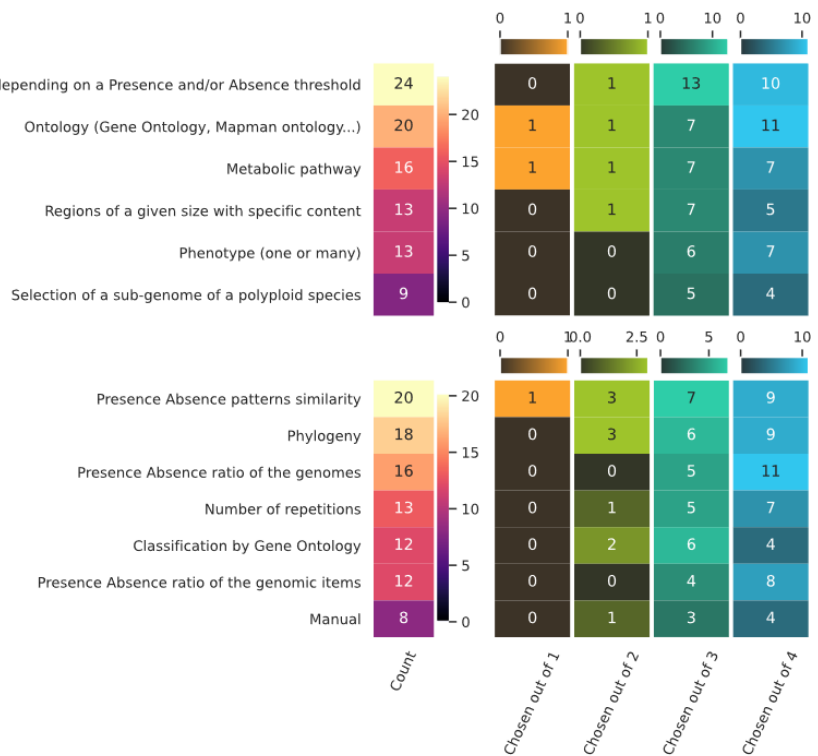
Appendix LXI: Survey results 5/11 – Most desirable features for lookup; Functionalities are sorted in decreasing order depending on a score based on both the number of times an answer has been chosen and in which position this answer appeared in the set of answers given by a respondent. Funnily enough 'Filter data' was both the functionality that has been chosen first most of the time and the functionality that has been chosen the least.

QFiltering:

In which of the following filtering criteria would you be particularly interested?

QSorting:

In which of the following sorting criteria would you be particularly interested?



31

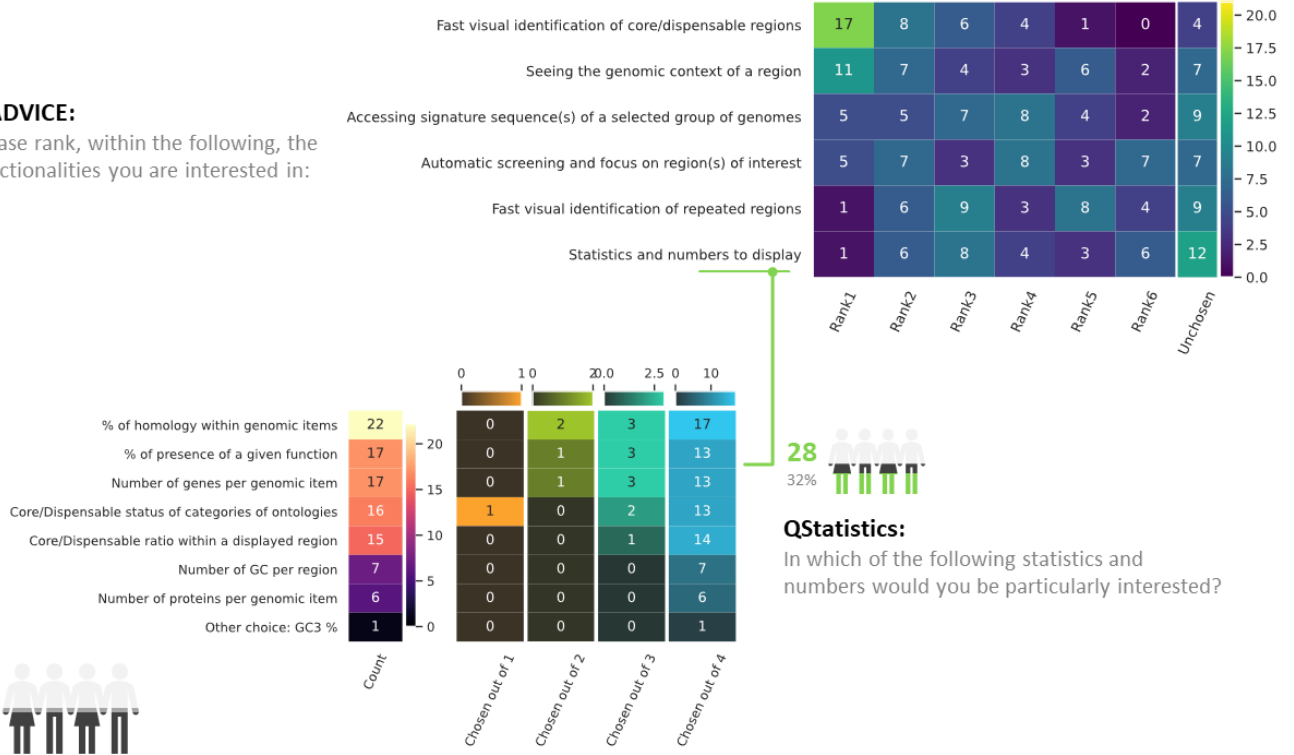
36%



Appendix LXII: Survey results 6/11 – If 'Sorting' or 'Filtering' were chosen in the previous question, details were asked

QADVICE:

Please rank, within the following, the functionalities you are interested in:



Appendix LXIII: Survey results 7/11 – Needed visual aids

QINTEROPERABILITY:

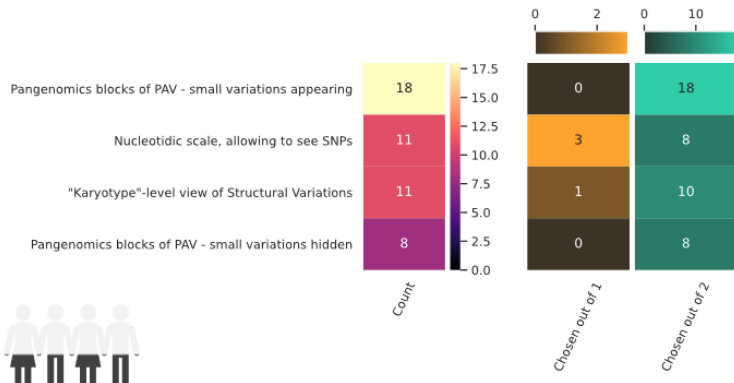
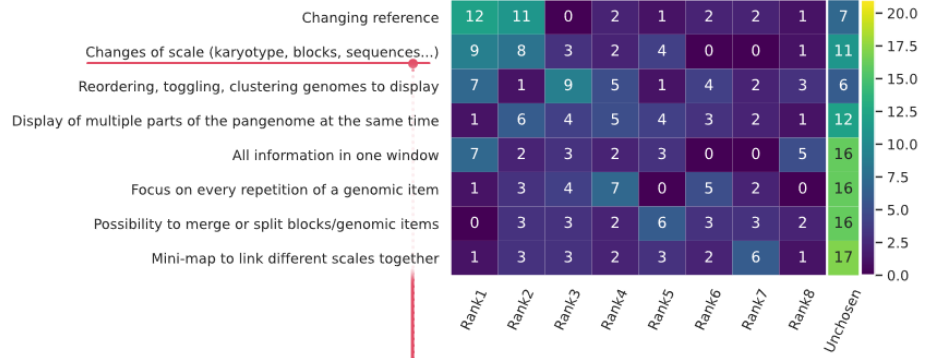
Please rank, within the following, the functionalities you are interested in:



Appendix LXIV: Survey results 8/11 – Data exchanges

QNAVIGATION:

Please rank, within the following, the functionalities you are interested in:



38
44%

QScals:

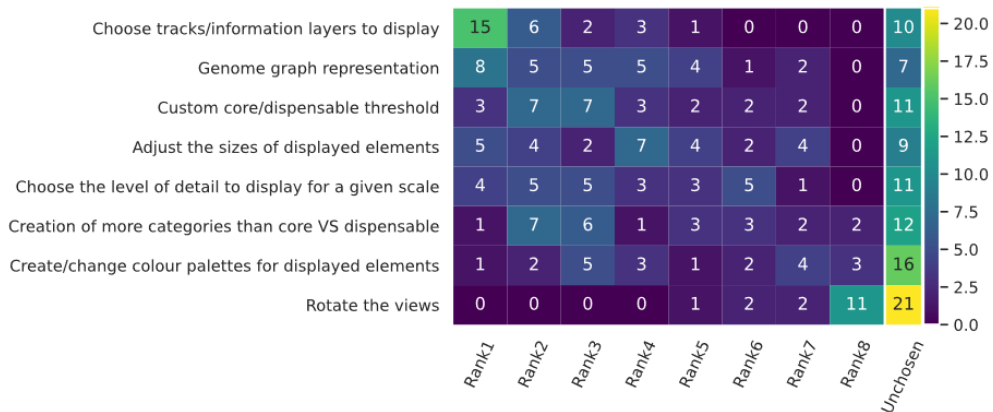
In which of the following scales would you be particularly interested?

26
30%

Appendix LXV: Survey results 9/11 – Navigation within the visual representation and UI

QCUSTOMIZATION:

Please rank, within the following, the functionalities you are interested in:



QMissedFunctionalities:

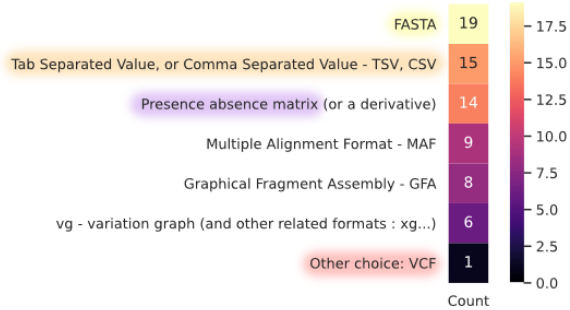
« I would like to select only a group of individuals to see the core and dispensable part within this group of individuals »
« Perhaps it could be interesting to order data according to the pylogeny tree of the individuals and to filter on this criteria »

37
43%

Appendix LXVI: Survey results 10/11 – Customizations

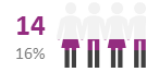
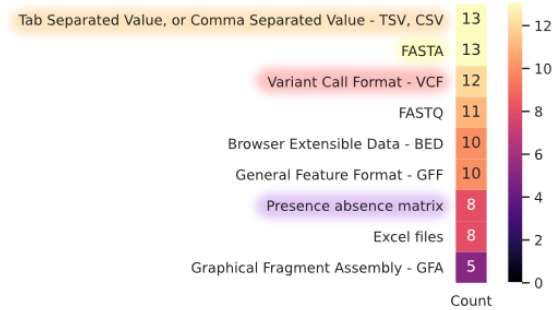
QPanFormat:

You have already worked with pangenomes. What pangenomes file formats do/did you like to use?

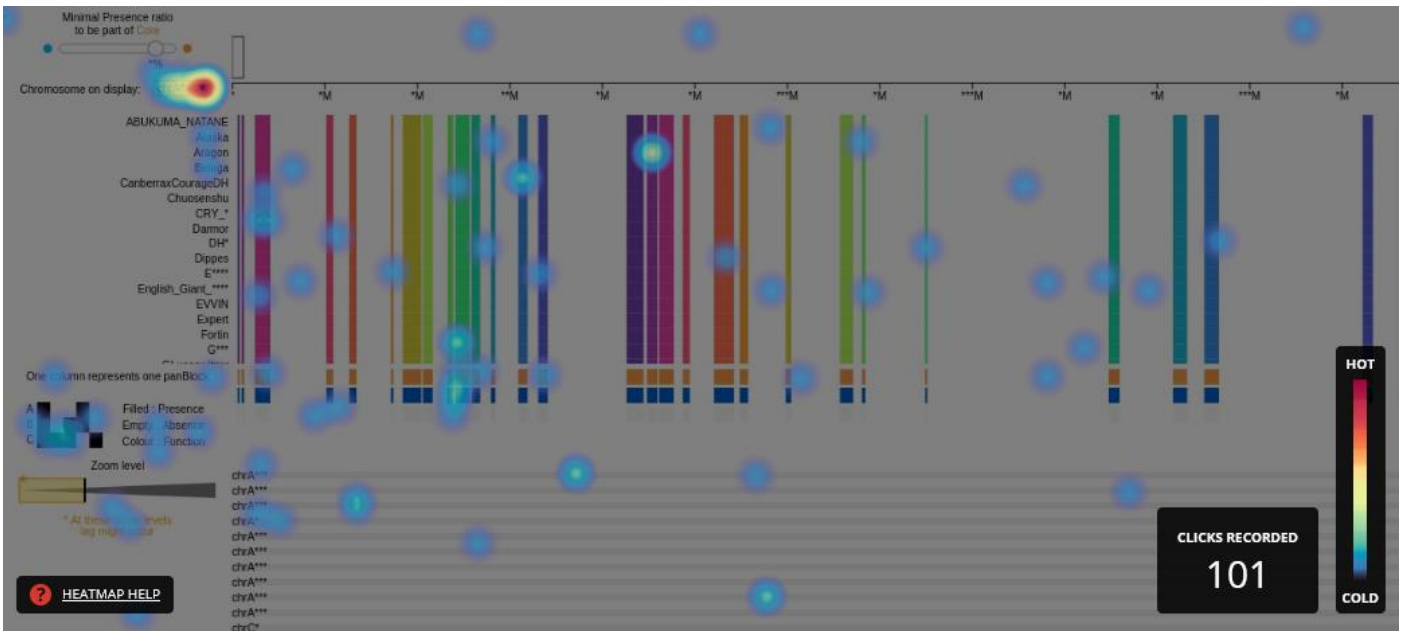


QEverydayFormat:

What file formats are you at ease working with?

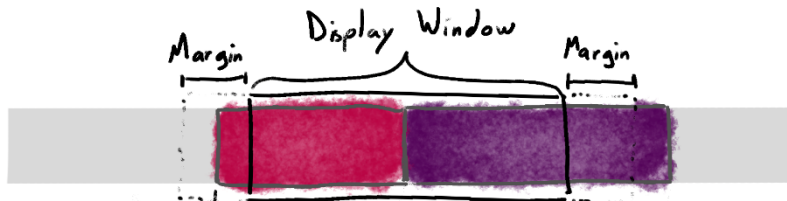


Appendix LXVII: Survey results 11/11 – Possible formats; I asked or the formats familiar to respondents, depending on their knowledge of pangenomics. Formats common between the two questions are highlighted with color.



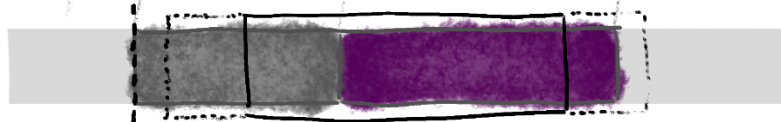
Appendix LXVIII: Respondents clicked on a lot of non-interactive parts of the display; Hotjar is a tool that can track users' interactions on a web page. I used it on Panache's prototype: while many clicked on the dropdown menu to change the panchromosome on display, some also clicked directly on the PAV matrix and the legend. I suppose they were expecting to access more information on these parts of the visualization. Few clicked on the core threshold slider or the zoom slider.

⊙ If 2 blocks take all the display space:



↪ It's okay as long as this border stays within the filtering range

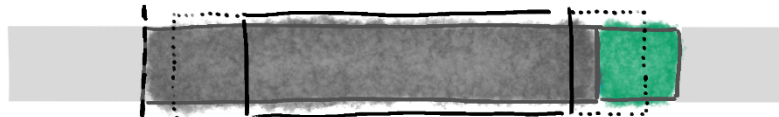
⊙ If previous blocks are slightly moved to the left:



↪ Border is outside the selection range !!
↳ Block not displayed even though it should be

BLOCKS WHOSE WIDTH > MARGIN WOULD ALWAYS LEAVE AN EMPTY SPACE THERE

⊙ If a block is bigger than display window + margin:



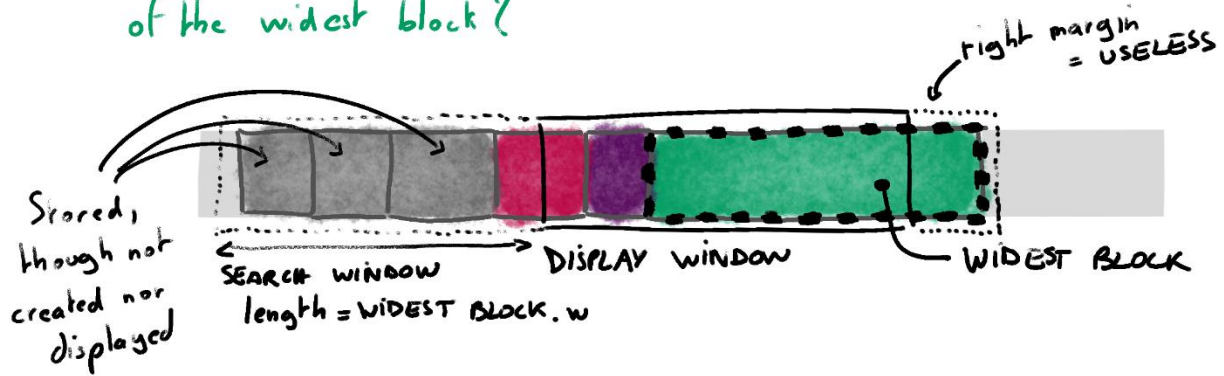
↪ Border is outside → nothing visible

- Have margins dependant on something else than mean block size?
- Always include blocks larger than the margin?
- Always include the block closest to left border?

Appendix LXIX: The panBlocks that should be represented must be within a certain window around the coordinates on display; PanBlocks are filtered depending on their StartPosition and StopPosition properties. All panBlocks contained within the display window and a margin before will be represented with SVGs, while others will not create any web element, making navigation within the PAV matrix more responsive.

PBL: When display window is on the right, the left part has too many elements to parse

↳ Limit search of displayable blocks to a margin the size of the widest block?



This way, any block, even the widest, could be displayed!



⚠ If there is an empty space wider than SEARCH + DISPLAY windows, D3's data() would bug again!...

Appendix LXX: The exact limit for the coordinates of panBlocks that should be drawn depends on the maximum width of blocks; A margin the size of the widest panBlock is added to the main window when looking for panBlocks to draw. This way, even the widest blocks would still be found and correctly drawn on screen.

Panache : a visualization tool for the exploration of plant pangenomes

Durant E.^{1,2,4}, Sabot F.^{1,4}, Rouard M.^{3,4}

¹ UMR DIADE, IRD, University of Montpellier, France, ² Syngenta, Toulouse France ³ Bioversity International, Montpellier France, ⁴ South Green Bioinformatics Platform, Montpellier France

Introduction

High-throughput sequencing technologies enabled the production of multiple reference genome sequences for a single species. Comparisons of such sequences showed that there are structural variations between individuals from the same species such as Copy Number Variations (CNV) and Presence Absence Variations (PAV) that can have a significant impact on phenotypic variation in plants and could be suitable for breeding improved crop varieties^{1,2}. Thus, a single reference genome is insufficient to capture all variations.

Pangenomics is an integrative approach which aims to the assessment of such genomic variations and more within a group of closely related individuals³. Its definition can focus on the whole repertoire of genes within a group or can include blocks of genomic sequences shared between species².

We introduce here a new visualization tool, based on a linear representation: the PANgenome Analyzer with CHromosomal Exploration (PANACHE). It is a web-based application which enables its users to explore a pangenomic reference divided in multiple panchromosomes. For now, it allows a quick identification of the genomic blocks belonging to either the core or dispensable genomes with the representation of the corresponding Presence/Absence Matrix, and navigation within and between panchromosomes.

Implementation

Panache is a web visualization tool written in JavaScript and is mainly based on the D3.js library. As there is no dedicated standard for the data format yet, we use a TSV file that extends BED format to allow the insertion of new information such as a presence absence matrix and functional categories (Table 1).

Chrom	Start	Stop	Similar Blocks	Function	A	B	C
chr1	56	78	chr1:56:78; chr2:74:96	7	0	0	1
chr1	210	356	-	4	1	0	1
chr2	30	43	chr2:30:43; chr4:120:133	3	1	1	1
chr2	74	96	chr1:56:78; chr2:74:96	7	0	1	1
chr3	756	823	-	0	1	1	1
chr4	120	133	chr2:30:43; chr4:120:133	3	0	1	1

Table 1. Data overview

A 8 tabular column that stores start and stop positions of the chromosome blocks, computed block similarity (optional), functional category (optional) and presence/absence matrix (1/0)

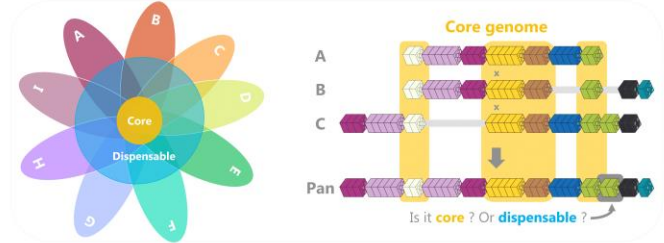
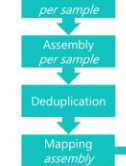


Figure 1. Pangenome composition (left) The core genome is the common set of sequences shared by all individuals (A-I) and the dispensable genome is composed of sequences present in some individuals only. **(right)** Genomic blocks colored by similarities. The pangenome can be considered in two different ways: function-based (as the sum of all genes within a given set of individuals) or structure-based (as the complete set of non-redundant sequences of approximately 100 base pairs (bp) or more within a given group of individuals).

Figure 2. Data processing to create pangenomes.

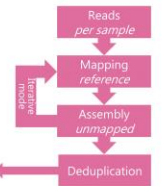
Assemble-then-map



i) **Assemble-then-map**: sequences from each individual are assembled separately, then mapped all-versus-all and to a reference in order to reduce redundancy.

ii) **Map-then-assemble**: all the sequences are mapped upon a reference sequence then re-assembled per individual using unmapped data. An alternative way is to re-assemble through an iterative mode, where samples are mapped successively on a panreference.

Map-then-assemble



User Interface

Figure 3. Panache overview (left) Description of the main interface. **(right)** Panache applied to real data on a set of African rice mapped on the external reference from Asian rice *O. sativa*. We then use a novel DepthOfCoverage approach to identify missing genes⁴. (Colors only highlight different blocks)

Display options: Fill: Present, Empty, Absence, Color, Function. Zoom level: 1 column = 1 panBlock.

Core vs Dispensable Position Similarity: This panBlock has other occurrences in panchromosomes 2, 3, 4, 7 and 9.

PAV shared in a subset of African rice genomes region (chr2) for further investigation

Source: [Gbrowse - ongenomesdb.cirad.fr](https://genomebrowser.org/genomesdb.cirad.fr)

References

- Tao, Y. et al. Exploring and Exploiting Pan-genomics for Crop Improvement. *Molecular Plant* 12, 156–169 (2019).
- Tranchot, C., Rouard M., Sabot F. Plant Pangenome: impacts on phenotypes and evolution. *Annual Plant Review* (2019)
- Marschall, T. et al. Computational pan-genomics: status, promises and challenges. *Brief Bioinform* 19, 118–135 (2018).
- Monat, C. et al. Comparison of two African rice species through a new pan-genomic approach on massive data. *bioRxiv* (2018).

Conclusion

Panache will include additional features to support the filtering and sorting of samples based on user preferred criteria such as taxonomy, genetic similarity, traits or PAV/CNV percentage. Some optimization will be conducted for very large datasets. New formats for data storage will be investigated with regards to graph-based strategies.

Panache : a visualization tool for the exploration of plant pangenomes

Durant E.^{1,2,3,4}, Conte M.², Sabot F.^{1,4}, Rouard M.^{3,4}

¹ UMR DIADE, IRD, University of Montpellier, France; ² Syngenta, Toulouse France; ³ Biodiversity International, Montpellier, France; ⁴ South Green Bioinformatics Platform, Montpellier, France

Introduction

High-throughput sequencing technologies enabled the production of multiple reference genome sequences for a single species. Comparisons of such sequences showed that there are structural variations between individuals from the same species such as Copy Number Variations (CNV) and Presence Absence Variations (PAV) that can have a significant impact on phenotypic variation in plants and could be suitable for breeding improved crop varieties^{1,2}. Thus, a single reference genome is insufficient to capture all variations.

Pangenomics is an integrative approach which aims to the assessment of such genomic variations and more within a group of closely related individuals³. Its definition can focus on the whole repertoire of genes within a group or can include blocks of genomic sequences shared between species².

We introduce here a new visualization tool, based on a linear representation: the PANgenome Analyzer with CHromosomal Exploration (PANACHE). It is a web-based application which enables its users to explore a pangenomic reference divided in multiple panchromosomes. For now, it allows a quick identification of genomic blocks belonging to either the core or dispensable genomes with the representation of the corresponding Presence/Absence Matrix, and navigation within and between panchromosomes.

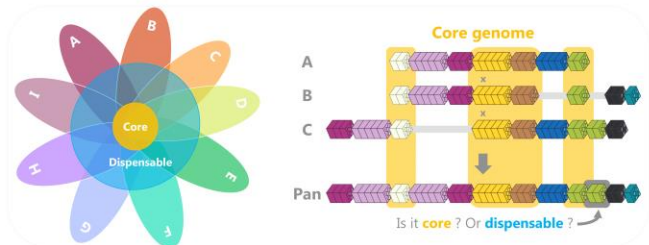


Figure 1. Pangenome composition (left) The core genome is the common set of sequences shared by all individuals (A-I) and the dispensable genome is composed of sequences present in some individuals only. **(right)** Genomic blocks colored by similarities. The pangenome can be considered in two different ways: function-based (as the sum of all genes within a given set of individuals) or structure-based (as the complete set of non-redundant sequences of approximately 100 base pairs (bp) or more within a given group of individuals).

Implementation

Panache is a web visualization tool written in JavaScript and is mainly based on the D3.js library. As there is no dedicated standard for the data format yet, we use a TSV file that extends BED format to allow the insertion of new information such as a presence absence matrix and functional categories (Table 1).

Chrom	Start	Stop	Similar Blocks	Function	A	B	C
chr1	56	78	chr1:56:78; chr2:74:96	7	0	0	1
chr1	210	356	.	4	1	0	1
chr2	30	43	chr2:30:43; chr4:120:133	3	1	1	1
chr2	74	96	chr1:56:78; chr2:74:96	7	0	1	1
chr3	756	823	.	0	1	1	1
chr4	120	133	chr2:30:43; chr4:120:133	3	0	1	1

Table 1. Data overview

A 8 tabular column that stores start and stop positions of the chromosome blocks, computed block similarity (optional), functional category (optional) and presence/absence matrix (1/0)

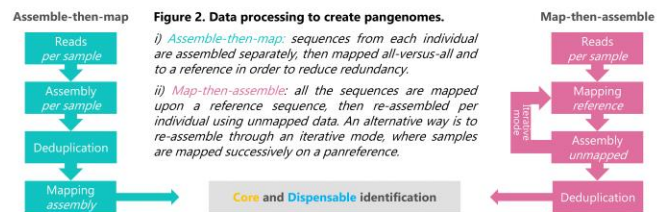


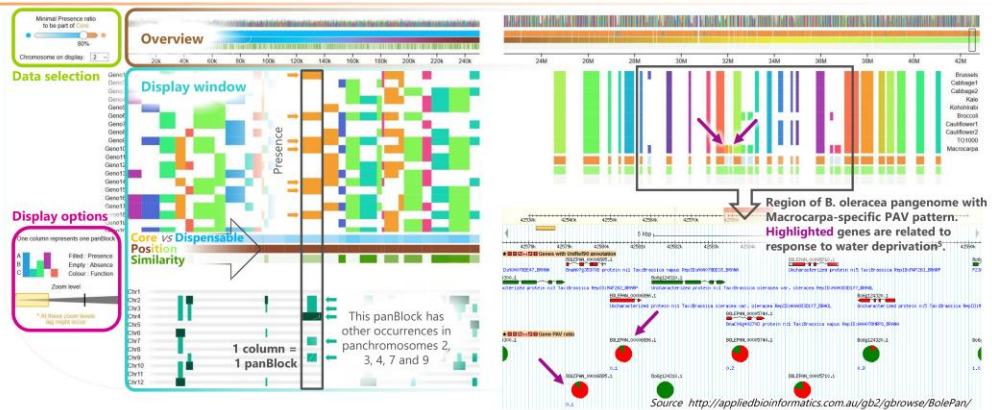
Figure 2. Data processing to create pangenomes.

i) **Assemble-then-map**: sequences from each individual are assembled separately, then mapped all-versus-all and to a reference in order to reduce redundancy.

ii) **Map-then-assemble**: all the sequences are mapped upon a reference sequence then re-assembled per individual using unmapped data. An alternative way is to re-assemble through an iterative mode, where samples are mapped successively on a panreference.

User Interface

Figure 3. Panache overview (left) Description of the main interface. **(right)** Panache applied to real data on a set of *Brassica oleracea* from a study by A. Golicz et al.⁵ Ten different genomes were used to create the *Brassica oleracea* pangenome, with positioned genes. The study showed multiple genes specific to *Macrocarpa*.



GitHub
SingingMeerkat/Panache

Interested in pangenome visualization? Please fill in our current survey at <https://tinyurl.com/PAGSurvey-Panache>



References

- Tao, Y. et al. Exploring and Exploiting Pan-genomics for Crop Improvement. Molecular Plant 12, 156–169 (2019).
- Tranchant C, Rouard M, Sabot F. Plant Pangenome: impacts on phenotypes and evolution. Annual Plant Review (2019).
- Marschall, T. et al. Computational pan-genomics: status, promises and challenges. Brief Bioinform 19, 118–135 (2018).
- Monat, C. et al. Comparison of two African rice species through a new pan-genomic approach on massive data. bioRxiv (2018).
- Golicz, A. et al. The pangenome of an agronomically important crop plant *Brassica oleracea*. Nature Communications 7(1): 13390 (2016).

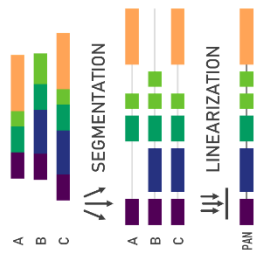
Conclusion

Panache will include additional features to support the filtering and sorting of samples based on user preferred criteria such as taxonomy, genetic similarity, traits or PAV/CNV percentage. Some optimization will be conducted for very large datasets. New formats for data storage will be investigated with regards to graph-based strategies.

Escape the pang genome maze with Panache's thread

Eloi Durant^{1,2,3,4}, Francois Sabot^{1,4}, Matthieu Conte² and Mathieu Rouard^{3,4}
 1. INRAE, UR140, Montpellier, France; 2. Syngenta Seeds SAS, Saint-Sauveur, France; 3. University of Montpellier, Parc Scierie Agropolis, Montpellier, France; 4. French Institute of Bioinformatics (IFB)—South Green Bioinformatics Platform, Bioversity, CIRAD, INRAE, IRD, Montpellier, France

Pangenomes have been growing in popularity for the past years but visualizing them still is a challenge partly because multiple definitions coexist, ranging from the full set of genes within a species to all possible successions of sequences within a clade. Newest representations are based on graphs, linking nodes of sequences together according to paths taken by different genomes.



Such graphs, while useful for handling pangenome data, are far from being easily readable, sometimes creating a *hairball* when there are too many links. They can also create confusing loops when confronted to structural variations, hard to follow for a human eye.

Turning them into linear representations would greatly improve both their legibility and explorability.

We therefore introduce Panache, our Pangenome Analyzer with Chromosomal Exploration, a browser-based interface created for the exploration of linear representations of pangenomes. It displays *pangenomic blocks* (either genes or sequences) as ordered on a single string in a genome browser-like fashion. Presence absence information and others can be displayed as tracks following this coordinate system.

Panache's set of functionality will be expanded with the time being for further explorability. Its linear representation may widen the adoption of pangenomes by providing a more user-friendly entity.

¹<https://eagereyes.org/techniques/graphs-hairball>

Available on [Github.com](https://github.com/SouthGreenPlatform/panache) / SouthGreenPlatform.com/panache

PAV matrix

- Shows present and absent blocks (columns) for each genome (rows)

Miniature

- Overview of full pangenome
- Navigation on click

Annotation cards

- Displays annotations hovering

Chromosome on display

chr01

Display parameters

- Shape type
- Minimal Presence ratio to be part of Core: 85%
- Zoom Level

Core threshold

- Custom categorization of your core and variable genomes

Hollow Area Finder

- Quickly jumps to areas of consecutive absence

Info tracks

- Sums up information per block
- Can show distribution of repeated blocks

Other features:

- This block appears in 11 selected genome(s)
- Hollow area finder: Absence rate (p-1), Number of consecutive blocks, There are 15 regions matching these criteria
- Core threshold: Minimal Presence ratio to be part of Core (0.4 to 0.8)

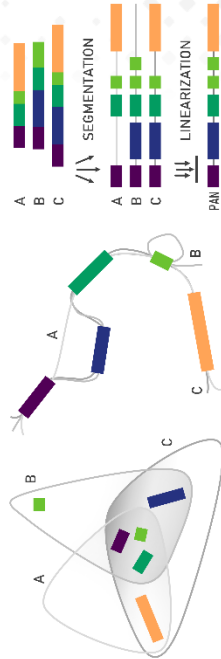
Appendix LXXIII: Panache's poster at VIZBI 2021

Panache and the Linearized Pangenomes: a Visualization Story

Éloi Durant^{1,2,3,4}, François Sabot^{1,4}, Matthieu Conte² and Mathieu Rouard^{3,4}

¹DIADÉ, Univ. Montpellier, IRD, CIRAD, Montpellier, France; ²Syngenta Seeds SAS, Saint-Sauveur, France; ³Bioversity International, Parc Scientifique Agropolis II, Montpellier, France; ⁴French Institute of Bioinformatics (IFB)—South Green Bioinformatics Platform, Bioversity, CIRAD, INRAE, IRD, Montpellier, France

Pangenomes are inventories of genomic material found in related genomes. Applied first to bacteria, they were defined as sets of genes that may be shared between strains, containing a **core genome** (genes present in every strain) as opposed to the **dispensable genome** (Tettelin et al., 2005). To fit eukaryotic organisms this definition later evolved, now including all chunks of sequences and their succession. Their representations evolved accordingly, from sets to graph but without any tool for their effective visualization applied to eukaryotes. Our efforts toward a user-friendly visualization tool led to the representation of **linearized pangenomes**.



Venn diagrams are usual representations for sets. They lack information of position, and do not scale well with the number of sets. Alternatives like **UpSet** diagrams face similar scaling problems.

Sequence graphs are recent representations, with every genome being a **path** through the sequence nodes of the graph. New genomes create branches and nodes, ending in visual clutter.

Using a **linear coordinate system** can improve the readability and explorability. Completed with a presence / absence matrix, succession in the genomes can be inferred.

We therefore introduce Panache, our Pangenome Analyzer with Chromosomal Explanation, a browser-based interface created for the exploration of linear representations of pangenomes. It displays **pangenomic blocks** (either genes or sequences) as ordered on a single string in a genome browser-like fashion, with a set of tracks displaying summary information.

A demo version with datasets and documentation is already available on GitHub! For more on how to build a linearized pangenome, check out our poster **IT5**

Tettelin, H. et al., *Genome analysis of multiple pathogenic isolates of *Syngentococcus agalactiae*: implications for the interrelated 'pan-genome'*, Proceedings of the National Academy of Sciences of the United States of America, 2005.

Available on [Github.com](https://github.com/SouthGreenPlatform/panache) / SouthGreenPlatform.com/panache

Annotation cards

- Displays annotations from gff on hovering

Miniature

- Overview of full panchromosome
- Navigation on click

PAV matrix

- Shows present and absent blocks (columns) for each genome (rows)

Chromosome on display

chr01

Display parameters

- Shape type
- Minimal Presence ratio to be part of Core: 80%
- Zoom Level

Hollow area finder

Absence rate (0-1): 0.4

Core threshold

- Custom categorization of your core and dispensable genomes

Hollow Area Finder

- Quickly jumps to areas of consecutive absence

Info tracks

- Sums up information per block
- Can show distribution of repeated blocks

Info tracks

- 1 2 3 4

This block appears in 11 selected genome(s)

Hollow area finder

Absence rate (0-1): 0.4

Number of consecutive blocks: 2

How many 10 regions matching these criteria:

Reflex: 14

Alter: 117519

Top regions:

413230

Alliance

syngenta

IRD Institut de Recherche pour le Développement I.R.D. C.E.

Bioversity International

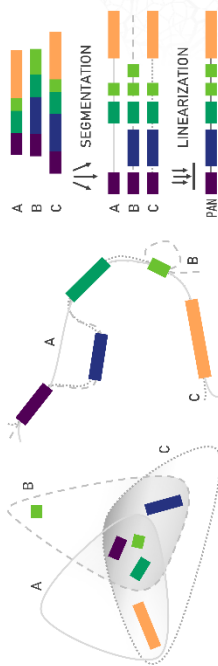
CIAT International Center for Tropical Agriculture

South Green bioinformatics platform

A Look at Trails through the Pangenome Visualization Jungle

Éloi Durant^{1,2,3,4}, François Sabot^{1,4}, Matthieu Conte² and Mathieu Rouard^{3,4}
 1)ADEL, Univ. Montpellier, IRD, CIRAD, Montpellier, France; 2)Syngenta Seeds SAS, Saint-Sauveur, France; 3)Bioversity International, Montpellier, France; 4)INRAE, Montpellier, France; 5)INRAE, Montpellier, France; 6)South Green Bioinformatics Platform, Bioversity, CIRAD, INRAE, IRD, Montpellier, France

Pangenomes are inventories of genomic material found in related genomes. Applied first to bacteria, they were defined as sets of genes that might be shared between strains, containing a **core genome** (genes present in every strain) as opposed to the **dispensable genome** (Tatain et al., 2005). To fit eukaryotic organisms this definition later evolved now including all chunks of sequences and their succession. Their representations evolved accordingly, from sets to graph but without any user-friendly tool for their effective visualization applied to eukaryotes. Our efforts toward a user-friendly visualization tool led to the representation of linearized pangenomes.



Venn diagrams are usual representations for sets. They lack information on position, and do not scale well with the number of sets. Alternatives like **UpSet** diagrams face similar scaling problems with dozens of genomes/sets.

Sequence graphs are recent representations, with every genome being a path through the sequence nodes of the graph. New genomes create branches, ending in visual clutter. They can be visualized with tools like **Bandage** or **Olviz**.

Using a **linear coordinate system** can improve the readability and explorability. Existing implementations include **UCSC's snake_tracks** and **PGAP-X**. A presence / absence matrix can be added to infer succession in the genomes.

Linearity, while common for the visualization of genomes and their alignments, is not widely used for pangenomics. Closest examples include visualizations of positioned genes on a linear coordinate system—without information on the intergenic space—or visualizations displaying all structural variations from the genomes—with the drawback of being hard to read and space consuming.

We therefore introduce Panache, our **PANgenome Analyzer with Chromosomal Exploration**, a browser-based interface created for the exploration of linear representations of pangenomes. It displays **pangenomic blocks** (either genes or sequences) as ordered on a single string in a genome browser-like fashion, with a set of tracks displaying summary information as illustrated on the right → → → → →

A demo version with datasets and documentation is already available on GitHub!
 Tattalin, H. et al., *Genome Analysis of Multiple Genomes and Related Phylogenetic Implications for the microbial 'pan-genome'* Proceedings of the National Academy of Sciences of the United States of America, 2008.

Available on [Github.com](https://github.com/SouthGreenPlatform/panache)
[/SouthGreenPlatform.com/panache](https://SouthGreenPlatform.com/panache)

PAV matrix
 Shows present and absent blocks (columns) for each genome (rows)

Miniature
 Overview of full panchromosome
 Navigation on click

Annotation cards
 Displays annotations from gff on hovering

Core threshold
 Custom categorization of your core and dispensable genomes

Files
 bananaethe_chromonlyviz
 Musaceae_pangenome.gene

Chromosome on display
 chr01

Display parameters
 Shape type
 Minimal Presence ratio to be part of Core: 0.4
 Zoom Level: 0.4

Hollow Area Finder
 Quickly jumps to areas of consecutive absence

Sorting Options
 Sort the tracks
 None, Alphanumeric, Reverse alphanumeric, Local presence/absence pattern, Gene presence status, Phylogenetic tree

Info tracks
 Sums up information per block
 Can show distribution of repeated blocks

NEW
 This block appears in 11 selected genome(s)

South Green bioinformatics platform

Alliance

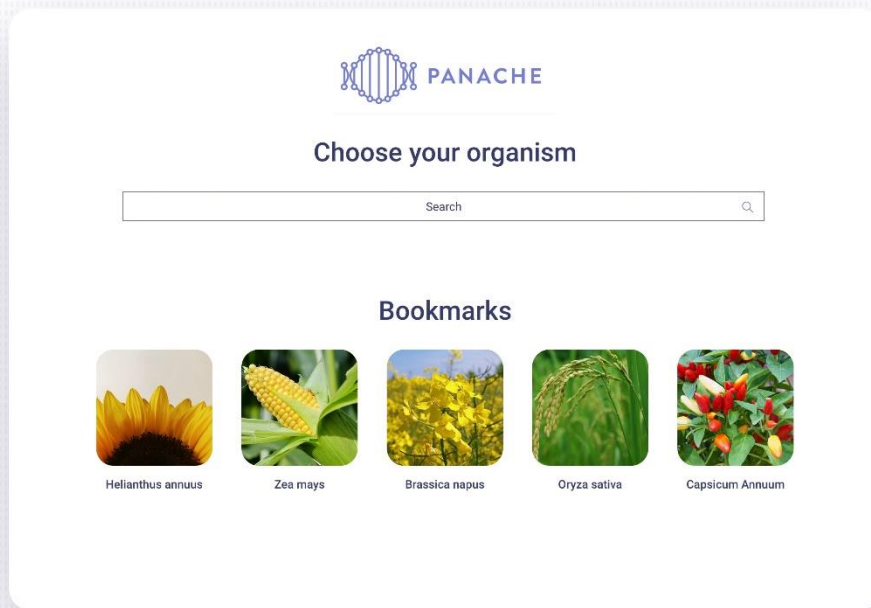
CIAT
 International Centre for Tropical Agriculture

Bioversity International

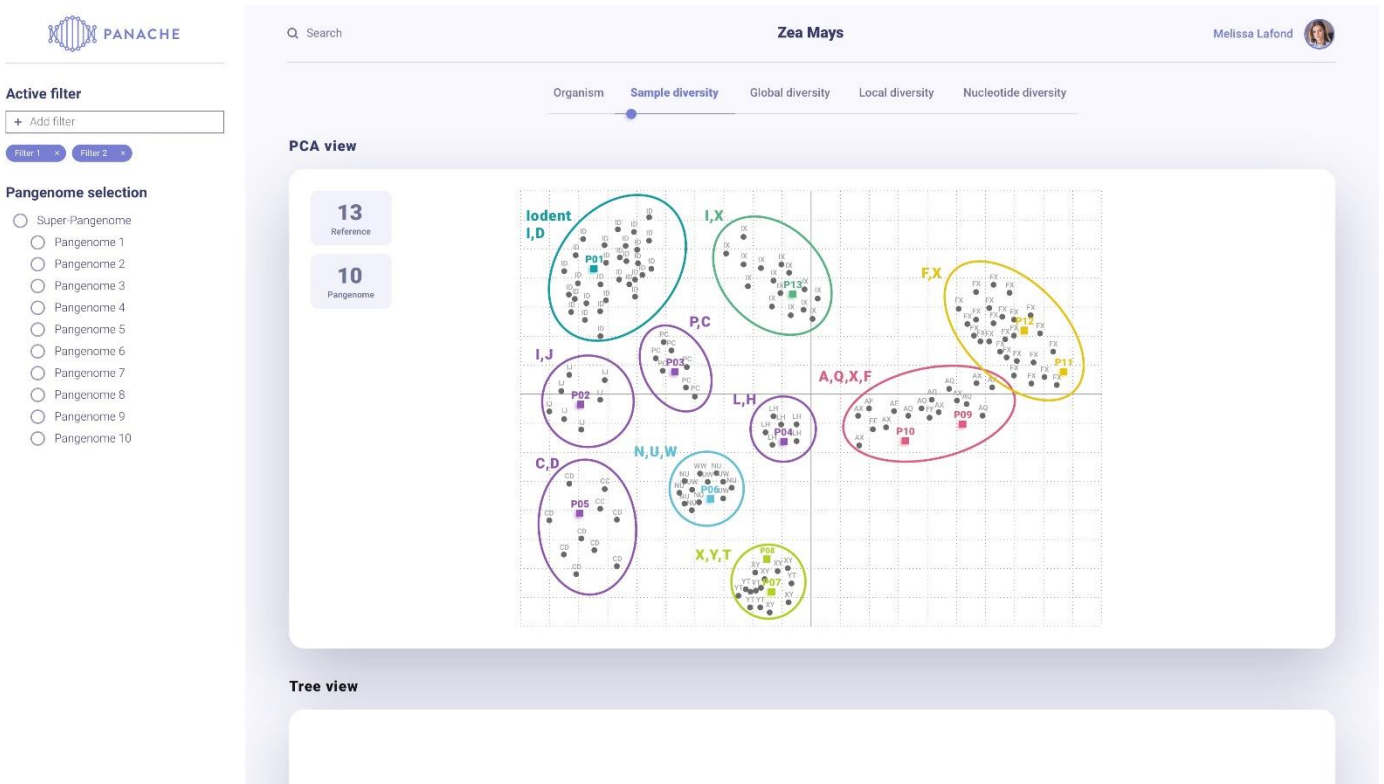
syngenta

IRD
 Institut de Recherche pour le Développement
 F. R. G. C.

III. Appendices from SaVanache's chapter



Appendix LXXVI: UI design 1/5 – Choice of the species to work on; Credit to Joffrey Gallais.



Appendix LXXVII: UI design 2/5 – Overall diversity, visual representation of the diversity and available datasets; Alternative views were envisioned, the scatterplot would have been the main one, with alternative phylogenetic tree and table representations to better compare assemblies. Credit to Joffrey Gallais.

PANACHE

Active filter

+ Add filter

Filter 1 x Filter 2 x

Region of interest

Chromosome
Select a chromosome

Start - Stop -

Sample Selection

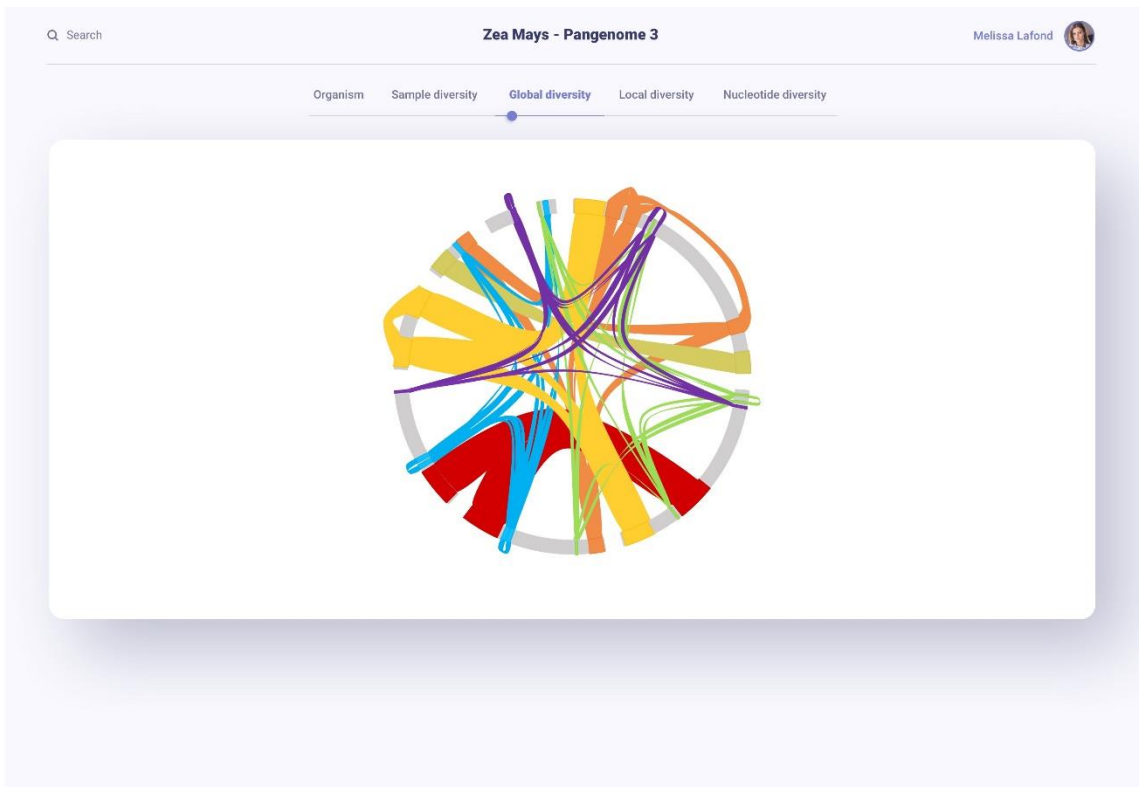
Sample
Select a sample

Structural Variation Size

Min - Max -

Dispensable - Core

Threshold -



Appendix LXXVIII: UI design 3/5 – Visual representation and exploration of SVs at the whole genome scale; Credit to Joffrey Gallais.

PANACHE

Active filter

+ Add filter

Filter 1 x Filter 2 x

Region of interest

Reference
Select a reference

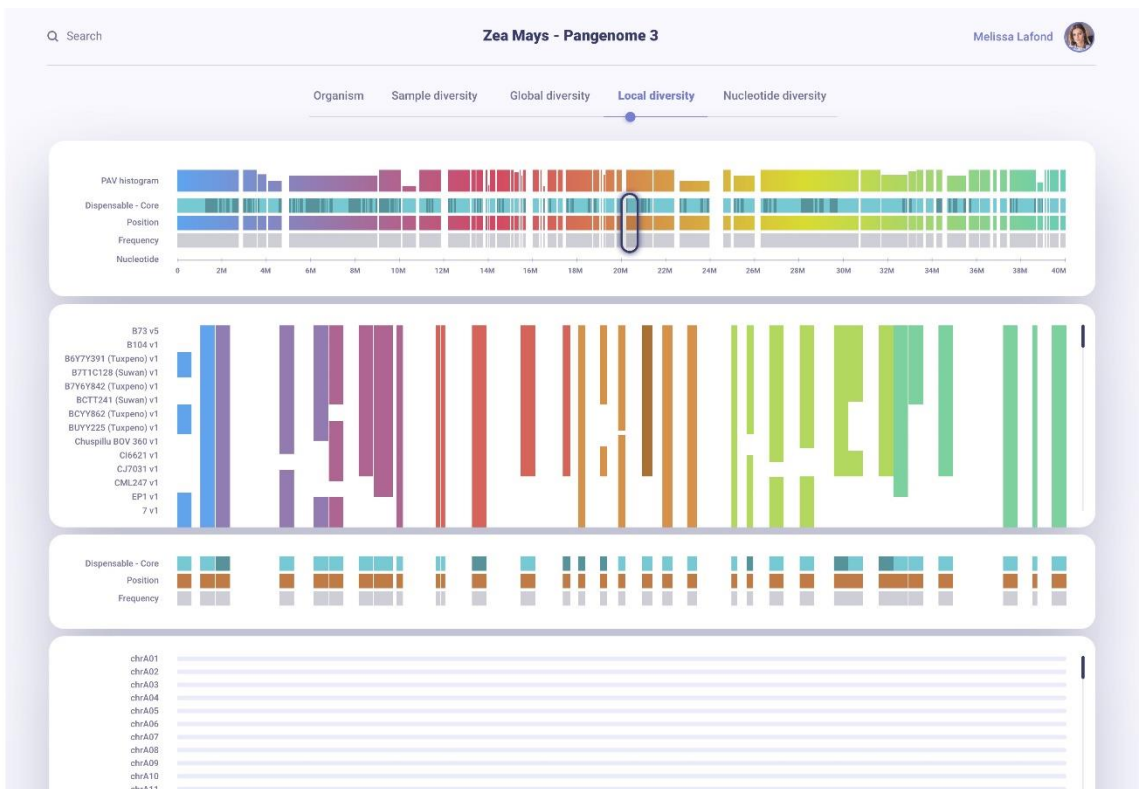
Chromosome
Select a chromosome

Start - Stop -

Dispensable - Core

Threshold -

Zoom levels



Appendix LXXIX: UI design 4/5 – Gene or panBlock PAV, targeted on a region of interest; Credit to Joffrey Gallais.

Organism Sample diversity Global diversity Local diversity **Nucleotide diversity**

Active filter

+ Add filter

Filter 1 Filter 2

Region of interest

Reference
Select a reference

Chromosome
Select a chromosome

Start Stop

Zoom levels

Filter material

Attribute
-

Name
-

Attribute type

PIC

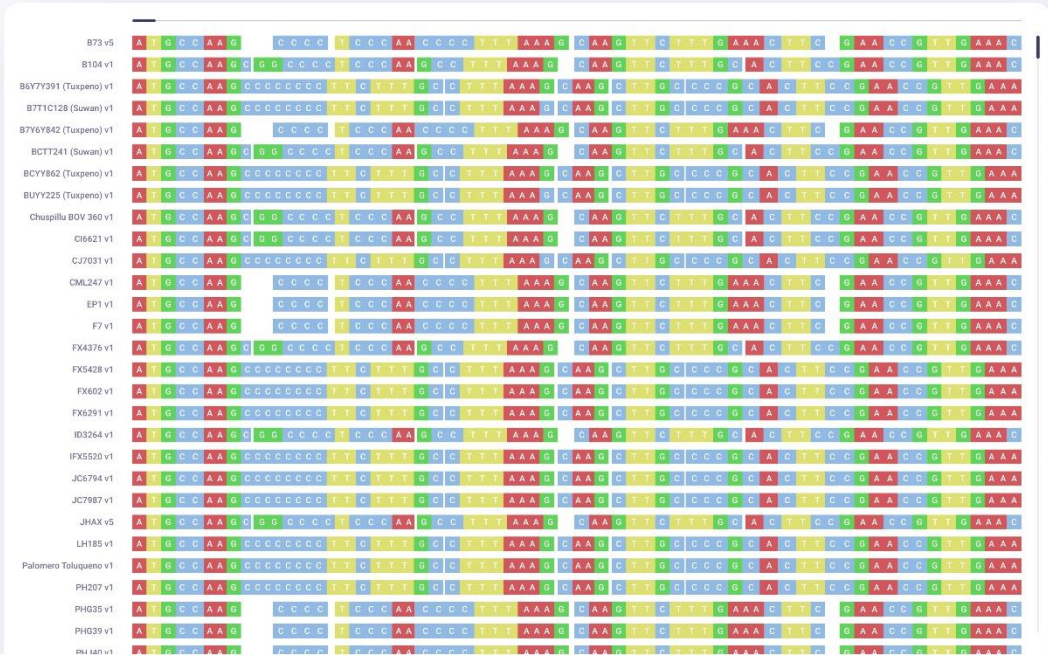
MAF

Heterozygous

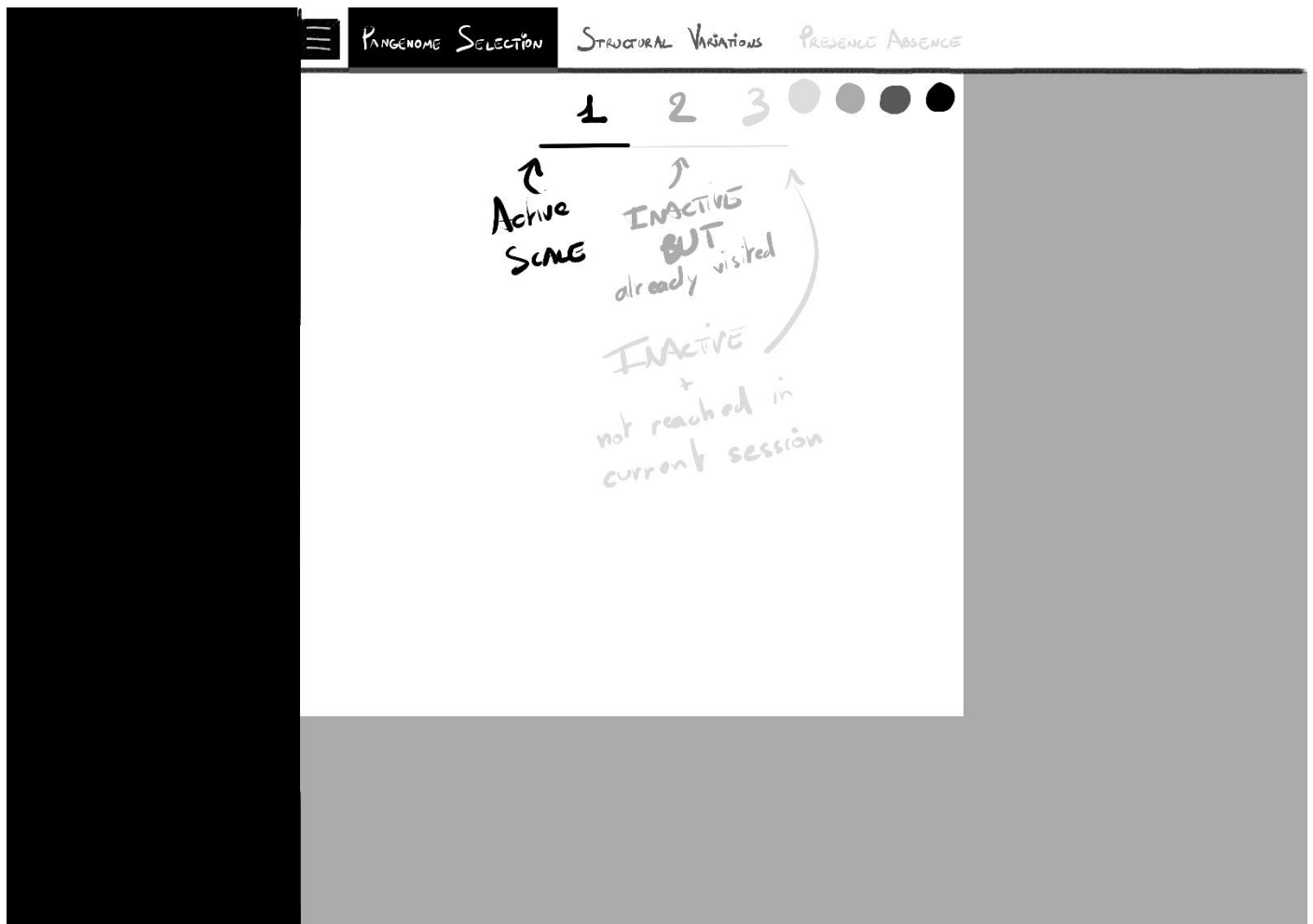
Filter variant

% NA

Start Impact



Appendix LXXX: UI design 5/5 – Sequence-level comparison, MSA-like representation; Credit to Joffrey Gallais.



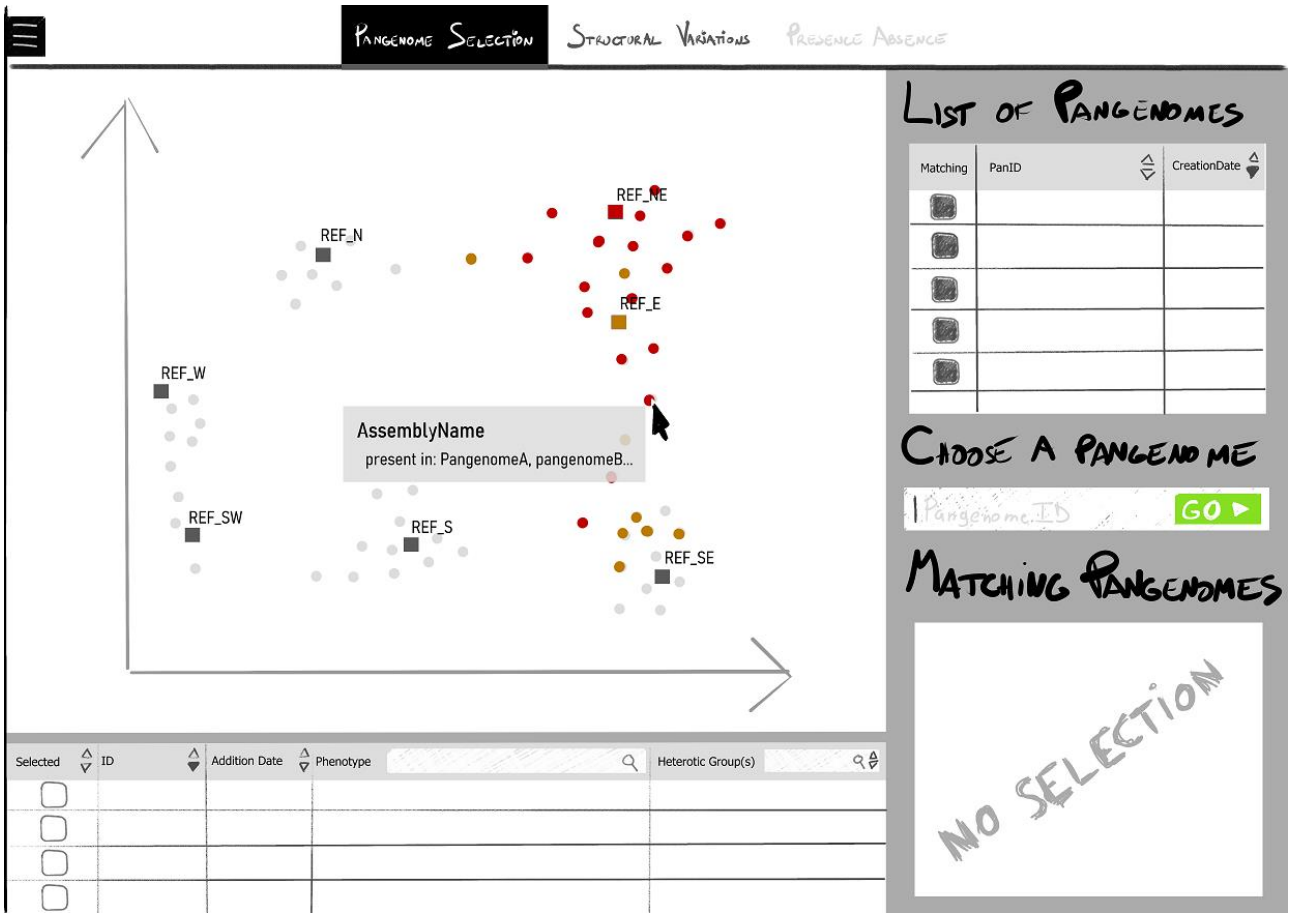
Appendix LXXXI: The planned UI would have a navigation pane on top to change the views; It also features a toggleable menu on the left for options.



Appendix LXXXII: Scatterplot user flow 1/11 - Welcome page



Appendix LXXXIII: Scatterplot user flow 2/11 - Hovering an assembly with one clustering value



Appendix LXXXIV: Scatterplot user flow 3/11 – Hovering an assembly with two clustering values



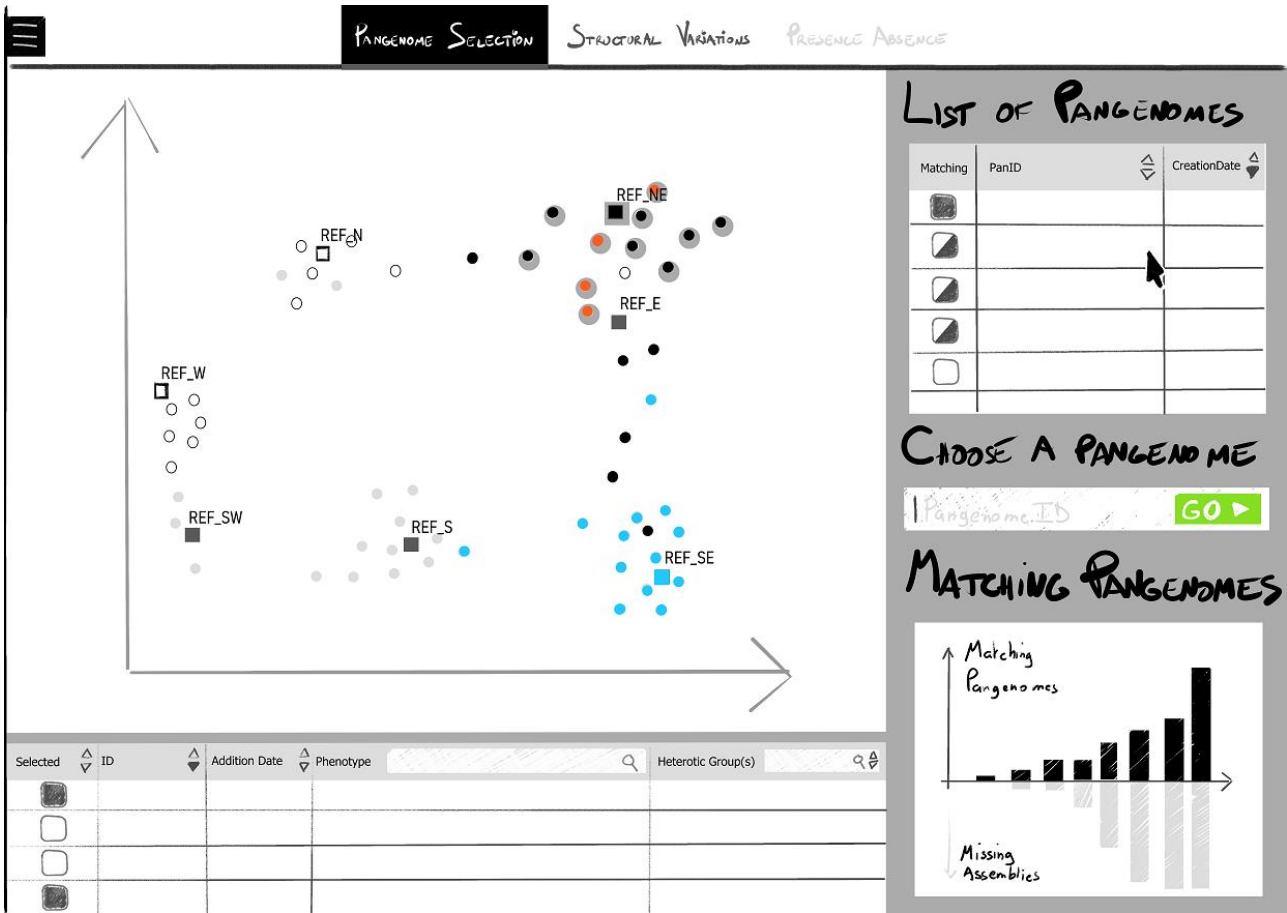
Appendix LXXXV: Scatterplot user flow 4/11 – Assemblies have been selected by lasso selection, matching pangenomes are listed



Appendix LXXXVI: Scatterplot user flow 5/11 – The best matching pangenome is hovered



Appendix LXXXVII: Scatterplot user flow 6/11 – The best matching pangenome is chosen



Appendix LXXXVIII: Scatterplot user flow 7/11 – The second best matching pangenome is hovered



Appendix LXXXIX: Scatterplot user flow 8/11 – The second best matching pangenome is chosen

PANGENOME SELECTION STRUCTURAL VARIATIONS PRESENCE ABSENCE

LIST OF PANGENOMES

Matching	PanID	CreationDate
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		

CHOOSE A PANGENOME

1 Pangenome.ID GO ▶

MATCHING PANGENOMES

Selected ID Addition Date Phenotype Heterotic Group(s)

Appendix XC: Scatterplot user flow 9/11 – Multiple partially matching pangenomes are highlighted from the bar chart

PANGENOME SELECTION STRUCTURAL VARIATIONS PRESENCE ABSENCE

LIST OF PANGENOMES

Matching	PanID	CreationDate
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		

CHOOSE A PANGENOME

1 Pangenome.ID GO ▶

MATCHING PANGENOMES

Selected ID Addition Date Phenotype Heterotic Group(s)

Appendix XCI: Scatterplot user flow 10/11 – A phenotype filter tag is added to the assembly table

PANGENOME SELECTION
STRUCTURAL VARIATIONS
PRESENCE ABSENCE

LIST OF PANGENOMES

Matching	PanID	CreationDate
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		

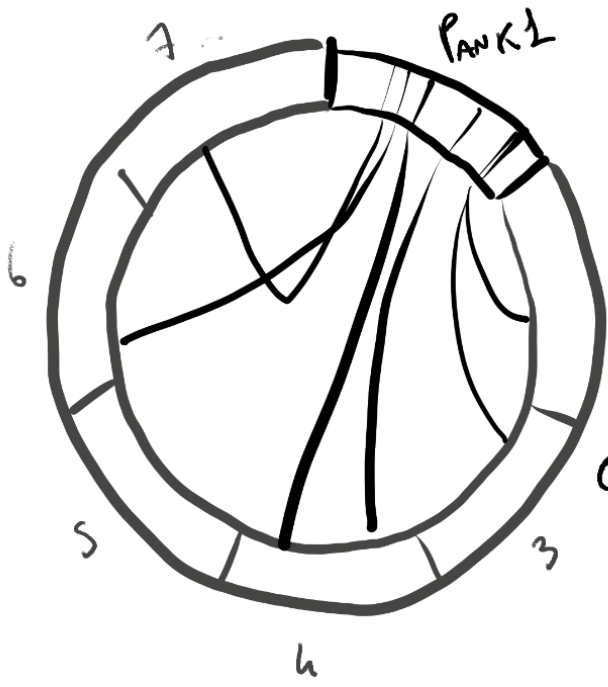
CHOOSE A PANGENOME

1 Pangenome.ID GO ▶

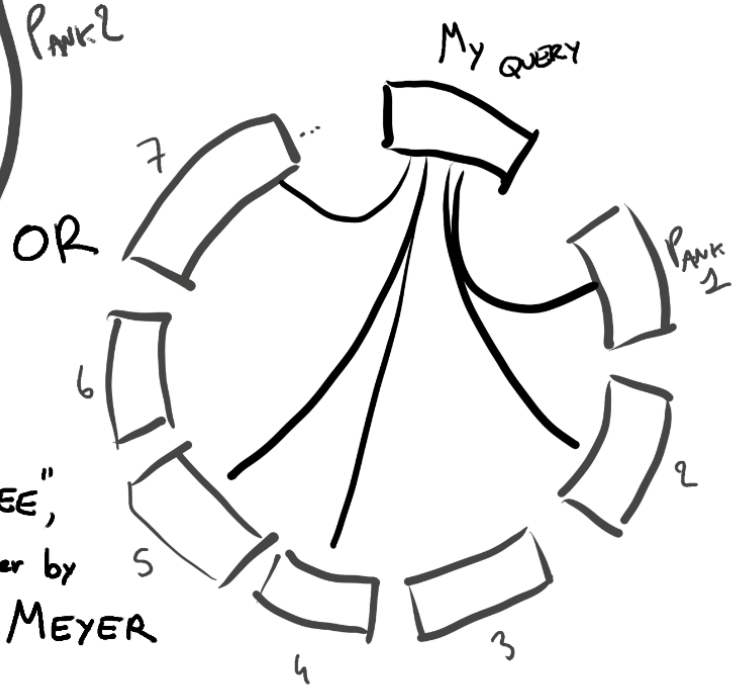
MATCHING PANGENOMES

Selected	ID	Addition Date	Phenotype	Heterotic Group(s)
<input checked="" type="checkbox"/>				
<input checked="" type="checkbox"/>				
<input checked="" type="checkbox"/>				
<input checked="" type="checkbox"/>				

Appendix XCII: Scatterplot user flow 11/11 – A second phenotype filter is added, with the first one still active

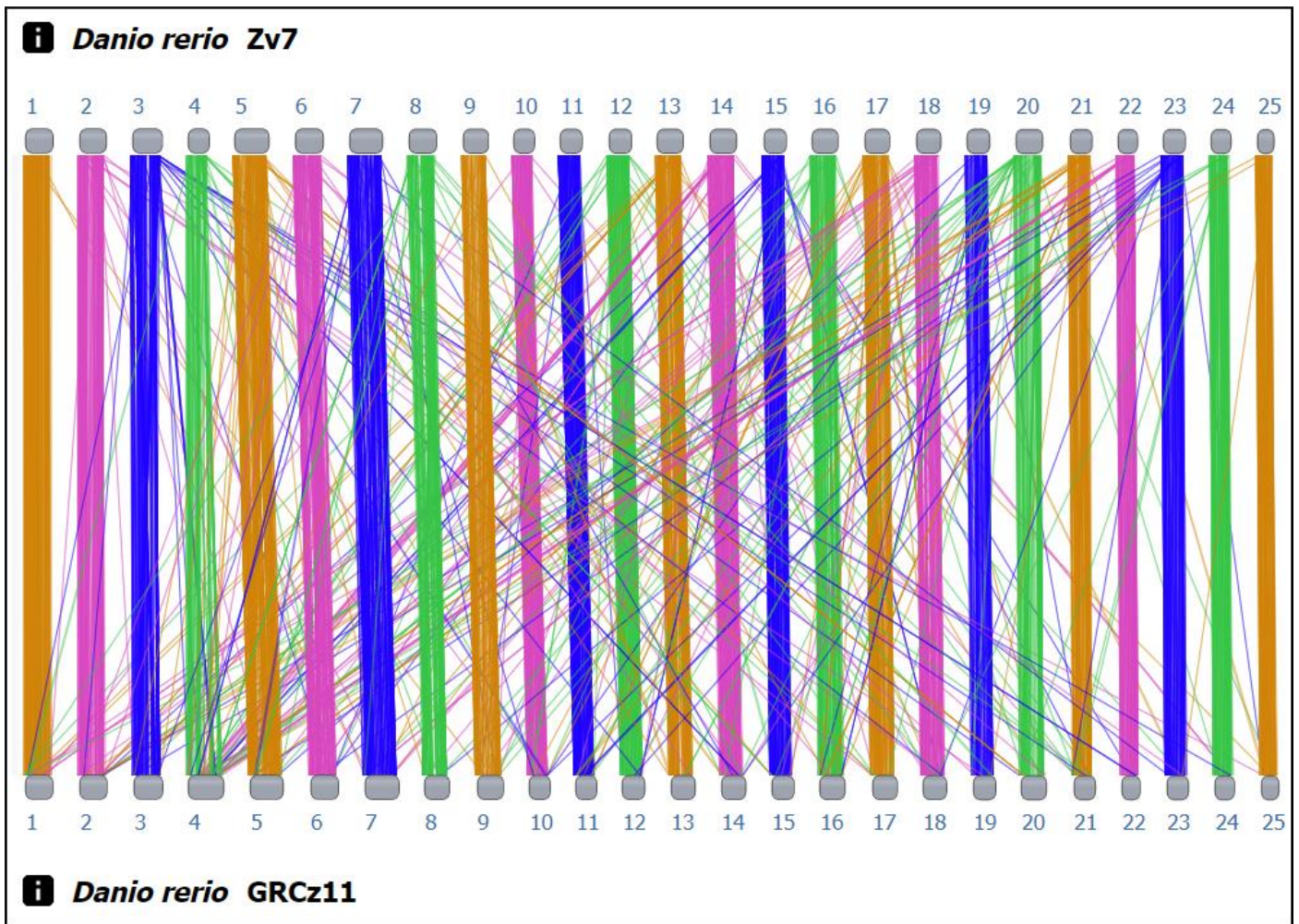


Circos for inversions
between pank?

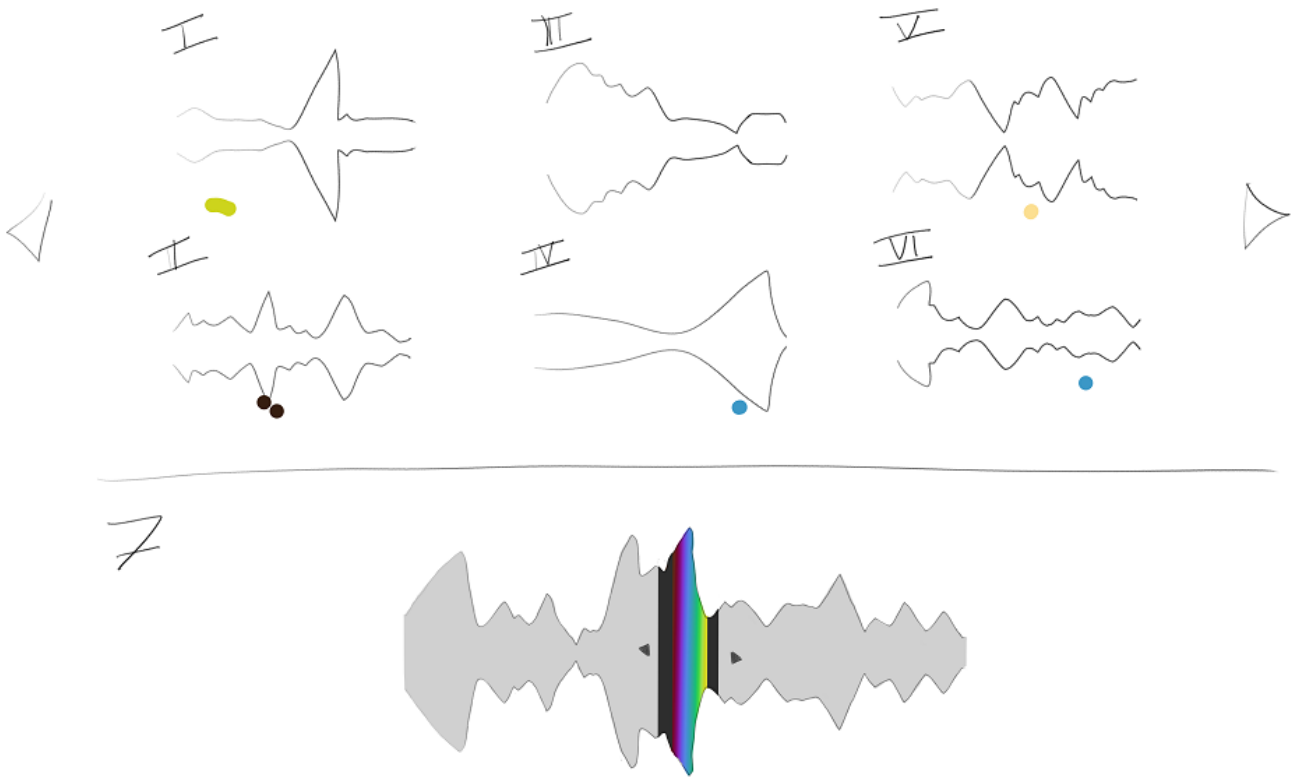


CF "MizBEE",
synteny browser by
MARIAH MEYER

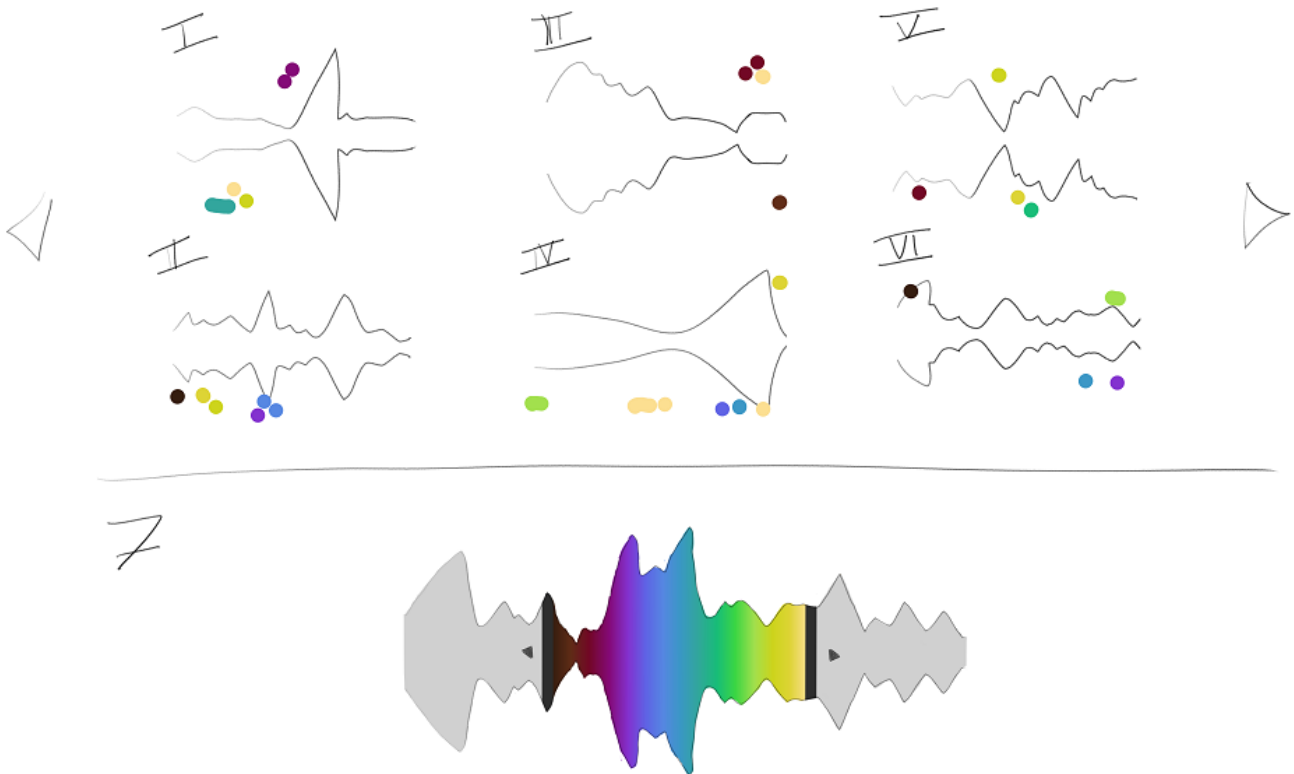
Appendix XCIII: Circos layouts could be applied to panchromosomes



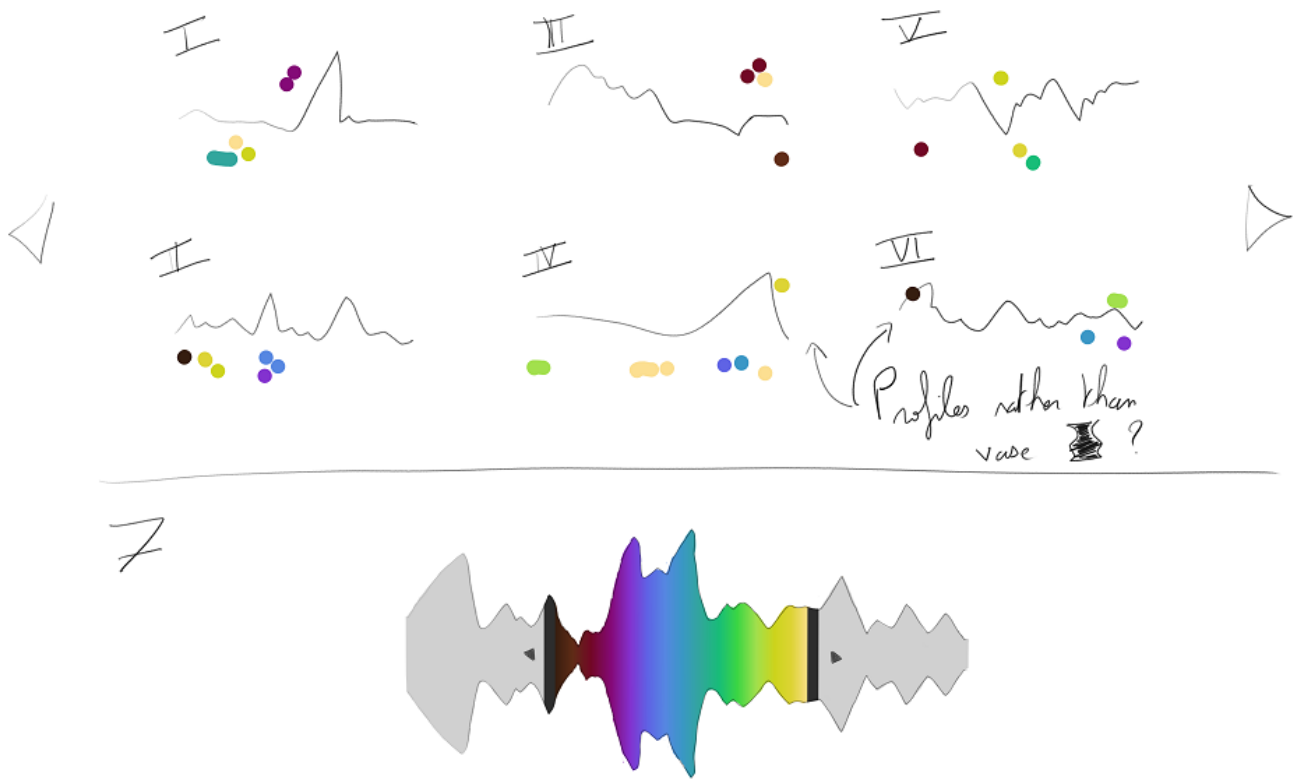
Appendix XCIV: The Comparative Genome Viewer visually represents all-versus-all alignments of chromosomes from two genome assemblies; The NCBI released a beta version released on July 6, 2022, and it features interaction which enables users to see one-versus-all alignments instead. One could also click on a connection to access additional information on a block alignment. Its overview remains hard to read and is representative of the difficulty of representing structural rearrangements within between multiple entities.



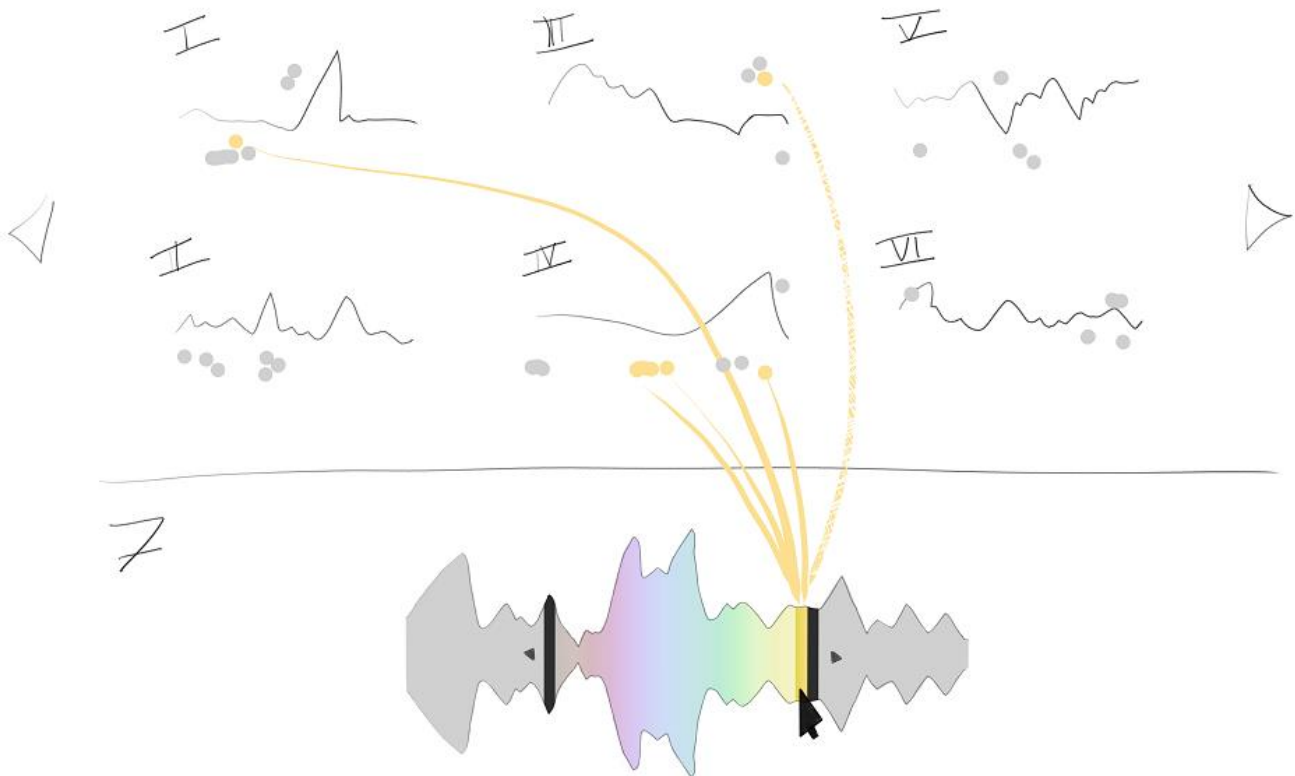
Appendix XCV: Tabular panchromosomes user flow 1/6 – Only cooccurrences from a small region are selected



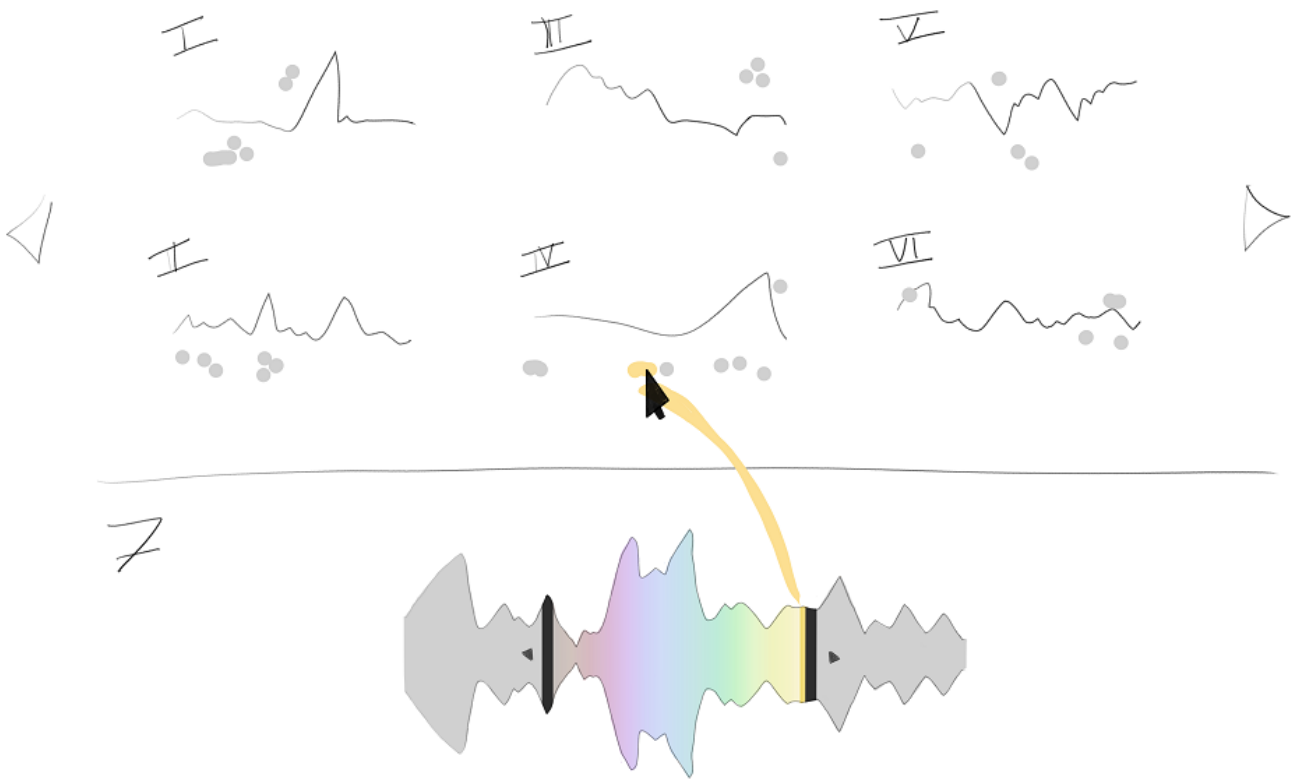
Appendix XCVI: Tabular panchromosomes user flow 2/6 – Moving the handles on the main panchromosome widens the selected region



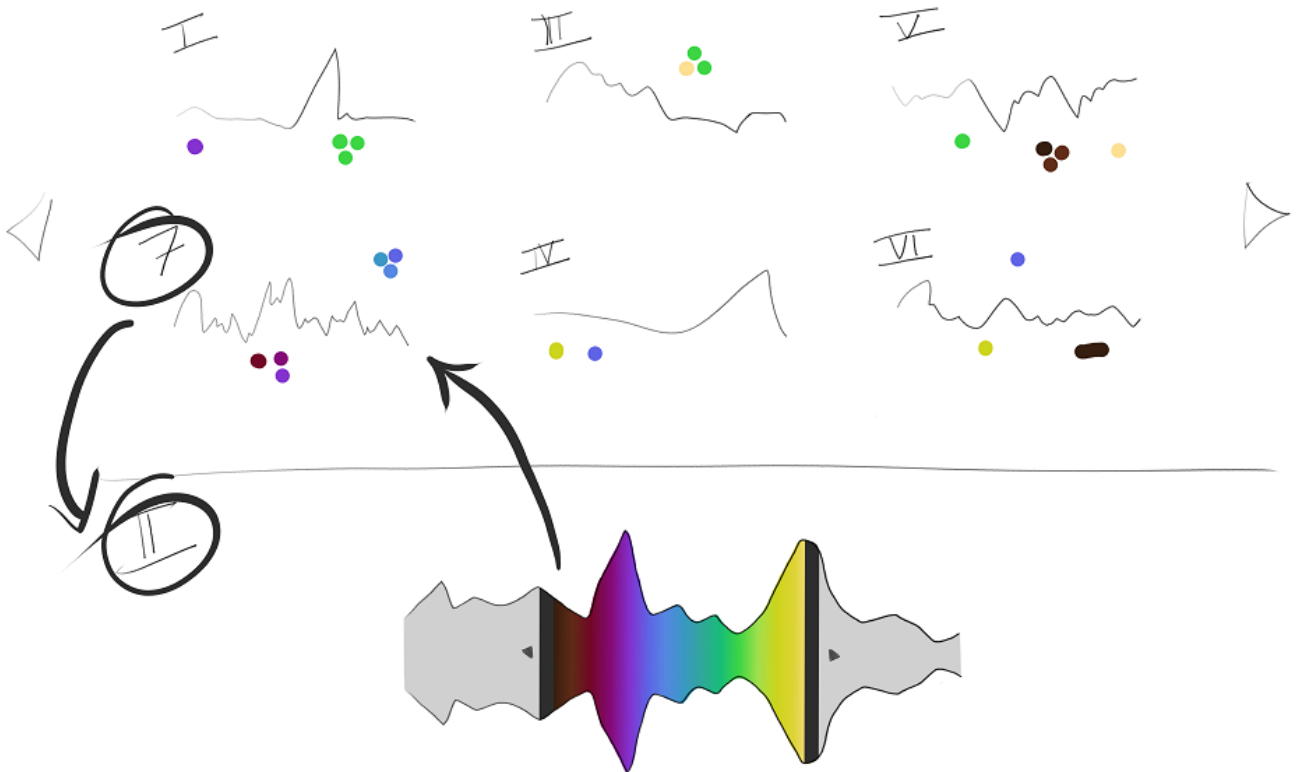
Appendix XCVII: Tabular panchromosomes user flow 3/6 – Profiles could be displayed rather than violin-like silhouettes



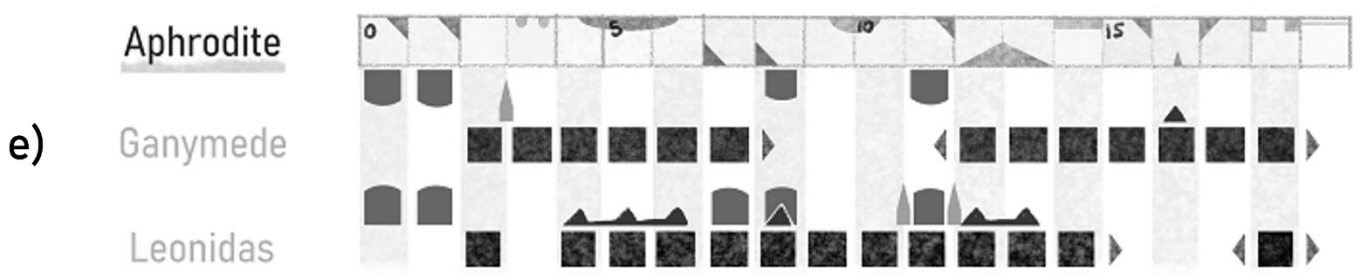
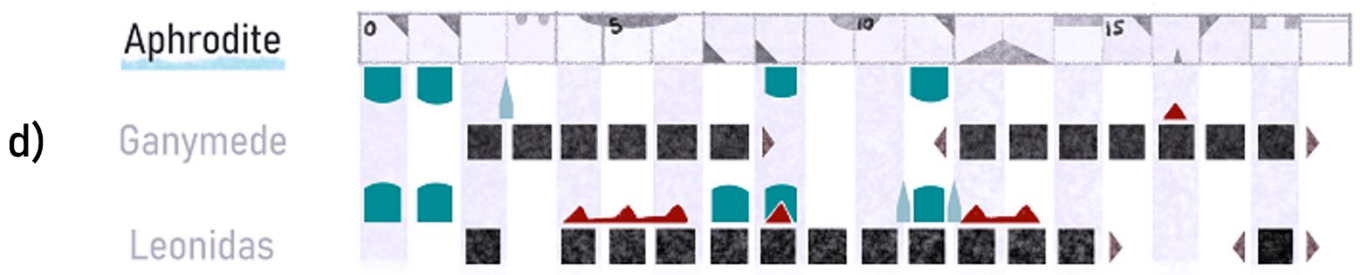
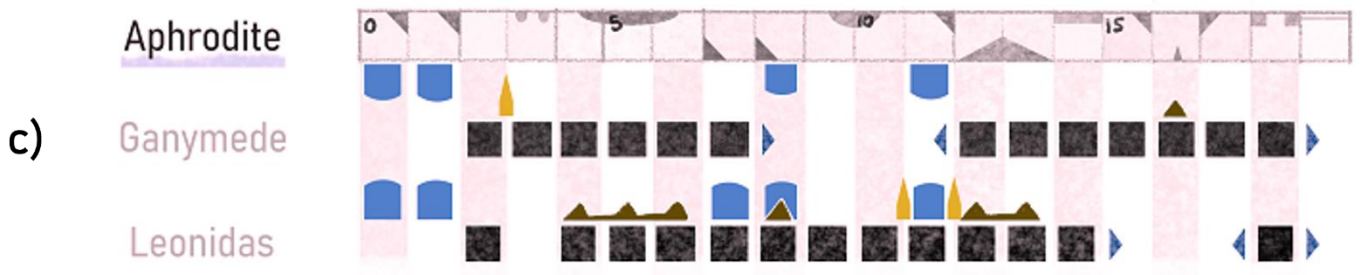
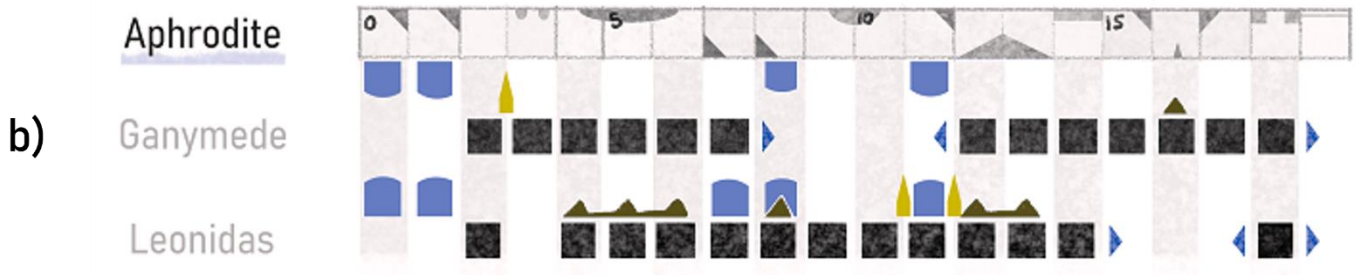
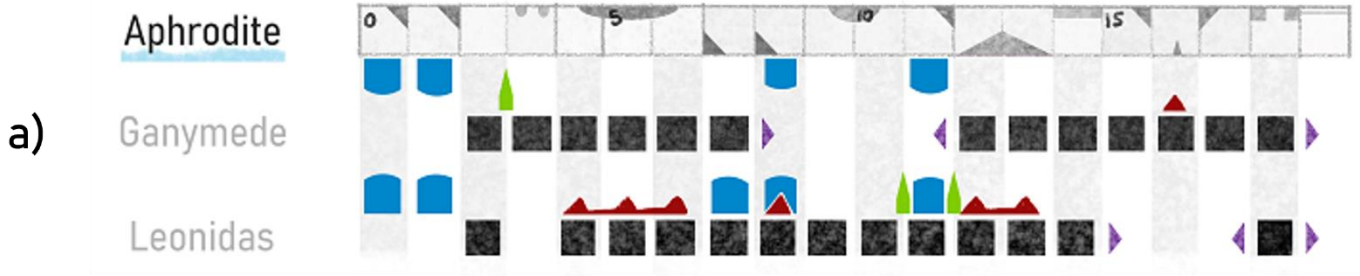
Appendix XCVIII: Tabular panchromosomes user flow 4/6 – Hovering a region would highlight the related cooccurrences; With this interaction connecting ribbons specific to the region hovered would be displayed



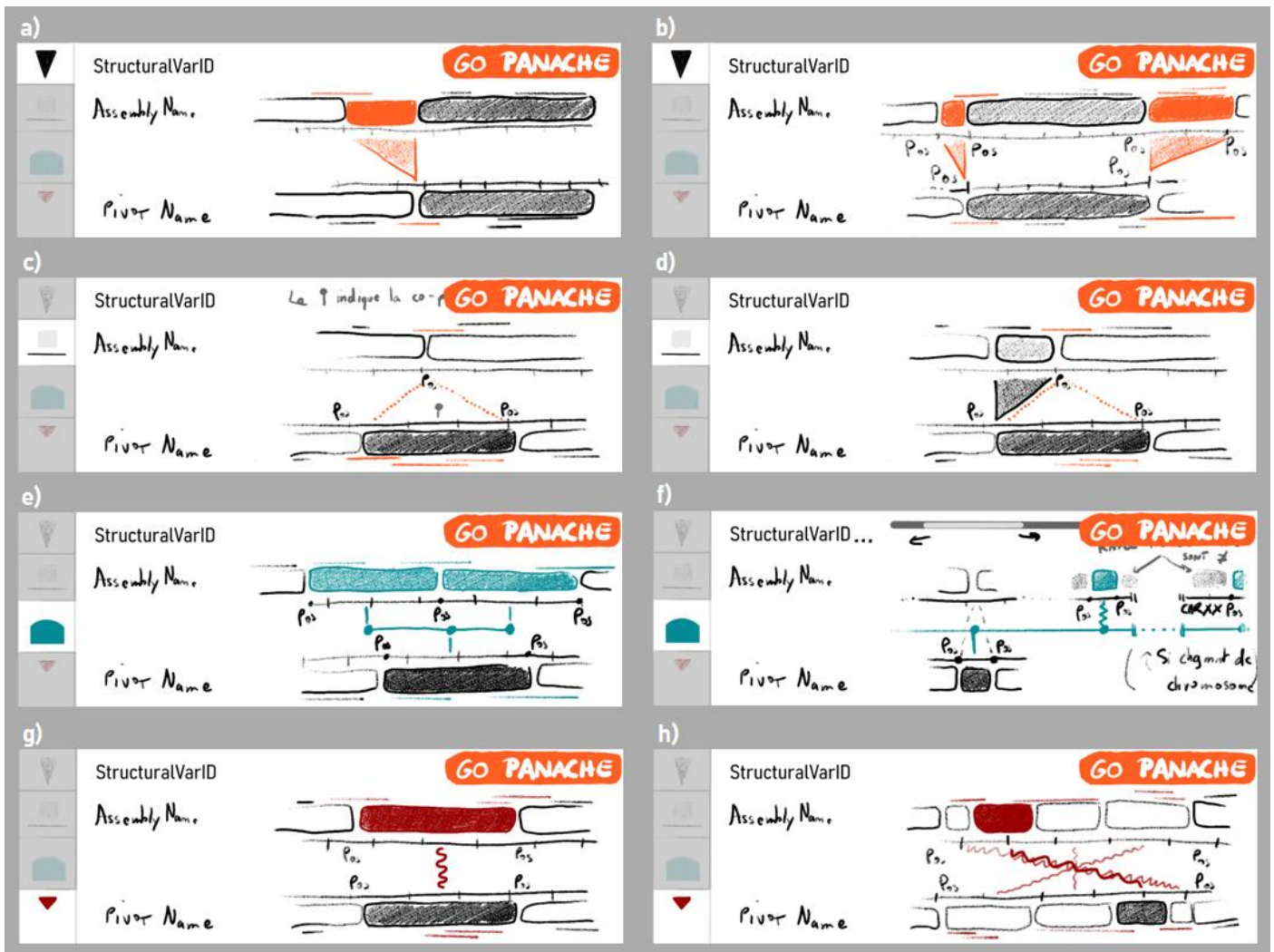
Appendix XCIX: Tabular panchromosomes user flow 5/6 – Hovering a segment could display its ribbon only



Appendix C: Tabular panchromosomes user flow 6/6 – Clicking on a panchromosome profile would change the main panchromosome; It would therefore update the whole view to show the related SVs



Appendix CI: The glyph system remains understandable under different colorblind visions; a) Normal vision. b) Protanopia. c) Deuteranopia. d) Tritanopia. e) Achromatopsia. Colorblindness simulated with the online tool Coblis – Color Blindness Simulator.



Appendix CII: The detail view can display many combinations of SVs surrounding a step from the pivot; a) A simple insertion, highlighted. b) Two flanking insertions, highlighted. c) A highlighted deletion, with a nonhighlighted cooccurrence. d) A highlighted deletion, with a nonhighlighted insertion; this conformation would be replaced by the representation of a swap in the most recent design. e) A tandem duplication, highlighted. f) A translocation, highlighted, with nonhighlighted deletion and inversion. g) An inversion, highlighted. h) A chain inversion, highlighted.

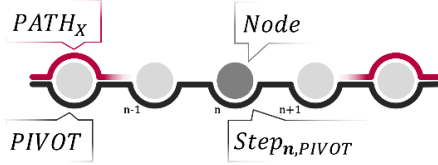
```

{
  "panSkeleton": {
    // A Node / panBlock, stored by ID, as a Key for a dictionary object
    nodeID: {
      "length": nodeLength,
      "traversals": {
        // We assume a panNode can be traversed only once by a genome assembly
        // Cooccurrences would create new nodes, as in a DAG
        assemblyName: {
          "sequenceOrigin": chromName
          "index": indexOfPosInPathU,
        },
        ... // and possibly more assemblies
      },
      // Every other copies of this node are listed here
      // Empty array if no cooccurrence
      "cooccurrences": [
        nodeID,
        ...
      ],
    },
    ... // and other Nodes
  },
  "paths": {
    // An assembly
    assemblyName: {
      // Each chromosome or sequence is associated with a list of the Steps...
      // ...taken along the path of that assembly through the pangenome graph
      chromName: [
        // The first node is at index 0
        {
          "panBlock": nodeID,
          "startPosition": startCoordOnThisAssemblyAndChrom,
          "endPosition": endCoordOnThisAssemblyAndChrom,
          "strand": -1 // or +1, depending
        },
        ... // Second node, etc until the end of the path
      ],
      ... // and paths from other sequences / chromosomes
    },
    ... // and other assemblies
  }
}

```

Appendix CIII: Pangenome graph can be divided into collection of nodes and paths within a JSON file; Here, the pseudo-JSON shows the overall structure expected by the *Structural Variations* view of SaVanache. Variable property names and values (written in white), comments (written in grey) and ellipses would be replaced in a real file. **nodeID**: the unique identifier (*string*) used to store a node of the pangenome graph; **nodeLenth**: the length in nucleotides (*integer*) of the DNA sequence represented by a node; **assemblyName**: the name (*string*) of a genome assembly; **chromName**: the name (*integer*) of the sequence of origin of a given path through the graph; **indexOfPosInPath**: index (*integer*) of a Node within the path; **startCoordOnThisAssemblyAndChrom**: start coordinates (*integer*) of that node's sequence in the original linear coordinate system of the assembly; **endCoordOnThisAssemblyAndChrom**: end coordinates (*integer*) of that node's sequence in the original linear coordinate system of the assembly.

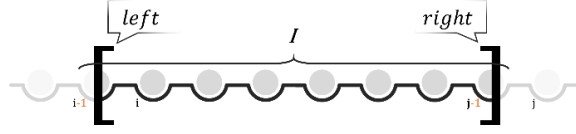
SaVanache – Parsing decision Tree



For each Node targetted by a Step n:

$$Node_{n,path} = Step_{n,path}(panBlock)$$

1. Select the *Interval* of nodes from the *PIVOT*

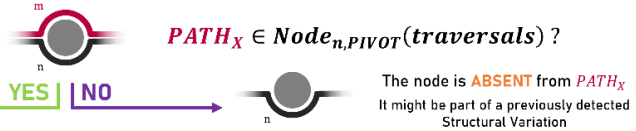


$$I = [Step_{i-1,PIVOT}, \dots, Step_{j-1,PIVOT}] \text{ for } (i,j) \in \mathbb{N} \text{ such as:}$$

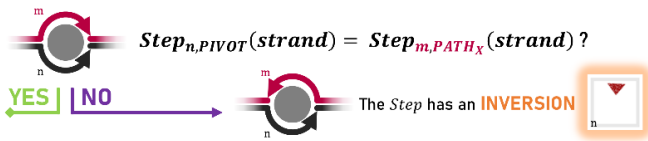
- $\{ i \text{ the smallest integer for which } Step_{i,PIVOT}(startPosition) > left$
- $\} j \text{ the highest integer for which } Step_{j,PIVOT}(startPosition) > right$

2. For each path compared with the *PIVOT*...

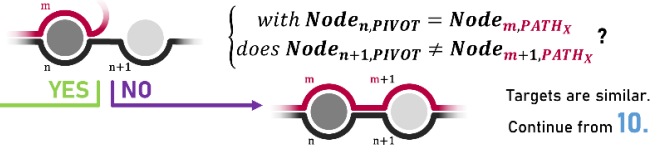
3. Is the parsed Node present in both paths?



4. Is the strand similar in both steps?



5. Is there a *synteny disruption*?



Appendix CIV: SV Annot Decision Tree 1/3 – The algorithm starts by following every Node of a path...

SaVanache – Parsing decision Tree

2. 3. 5.

$$\text{Let } w = \min \left(\begin{array}{l} x \in \mathbb{N} \\ \text{such as } x > m \\ \text{and } PIVOT \in Node_{m+x,PATH_X}(traversals) \end{array} \right)$$

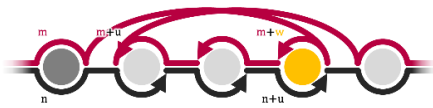
$Node_{m+w,PATH_X} = Node_{n+u,PIVOT}$ and is the first common Step since the disruption.

First common Step



2. 3. 5.

6. Is it the beginning of a chain inversion?



Are there symmetric inversions?

Let $v \in [0, u]$, does $\forall v$:

$$\left\{ \begin{array}{l} Node_{m+u-v,PATH_X} = Node_{n+1+v,PIVOT} \\ Step_{m+u+v,PATH_X}(strand) \neq Step_{n+1-v,PIVOT}(strand) \end{array} \right. ?$$

NO | YES

The Step is followed by an **INVERSION CHAIN**



2. 3. 5.

7. Then, is it a simple insertion?



$$Node_{m+w,PATH_X} = Node_{n+1,PIVOT} ?$$

NO | YES

The Step is followed by an **INSERTION**



2. 3. 5.

8. Then, is it a simple deletion?



$$Node_{m+1,PATH_X} = Node_{n+u,PIVOT} ?$$

NO | YES

The Step is followed by at least one **DELETION**

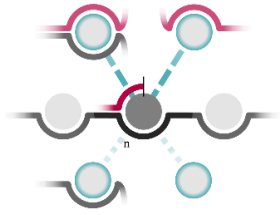


Appendix CV: SV Annot Decision Tree 2/3 – ...it will then look for Synteny disruptions...

SaVanache – SV Annotation storage



Cooccurrence: { length, stepIdxInPath }



length: integer – the maximum length of consecutive cooccurrent nodes.
stepIdxInPath: array – ordered array of the index of the cooccurrent nodes found in $PATH_X$.

Deletion: { length, endIdx }



length: integer – the cumulative length of nodes not in $PATH_X$ but present in $PIVOT$.
endIdx: integer – index $n + u$ of the first common node after the DELETION.

A DELETION is a subtype of SWAP.

Insertion: { length }



length: integer – the cumulative length of nodes present in $PATH_X$ but not in $PIVOT$.

An INSERTION is a subtype of SWAP.

Appendix CVIII: SV Annot Storage 2/3 – ...that would contain SV objects only at specific indices...

SaVanache – SV Annotation storage

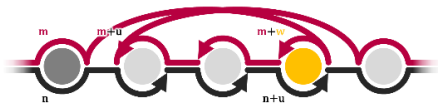


Inversion: { length }



length: integer – the length of $Step_{n,PIVOT}$.

Inversion Chain: { length, endIdx }



length: integer – the cumulative length of the successive inverted nodes: $[Step_{m+w,PATH_X}, \dots, Step_{m+u,PATH_X}]$.
endIdx: integer – index $n + u + 1$ of the first $PIVOT$ node without the INVERSION CHAIN.


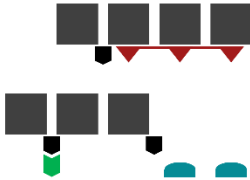
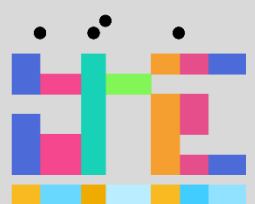
SYNTENY DISRUPTIONS within an INVERSION CHAIN should not be taken into account

Swap: { length }

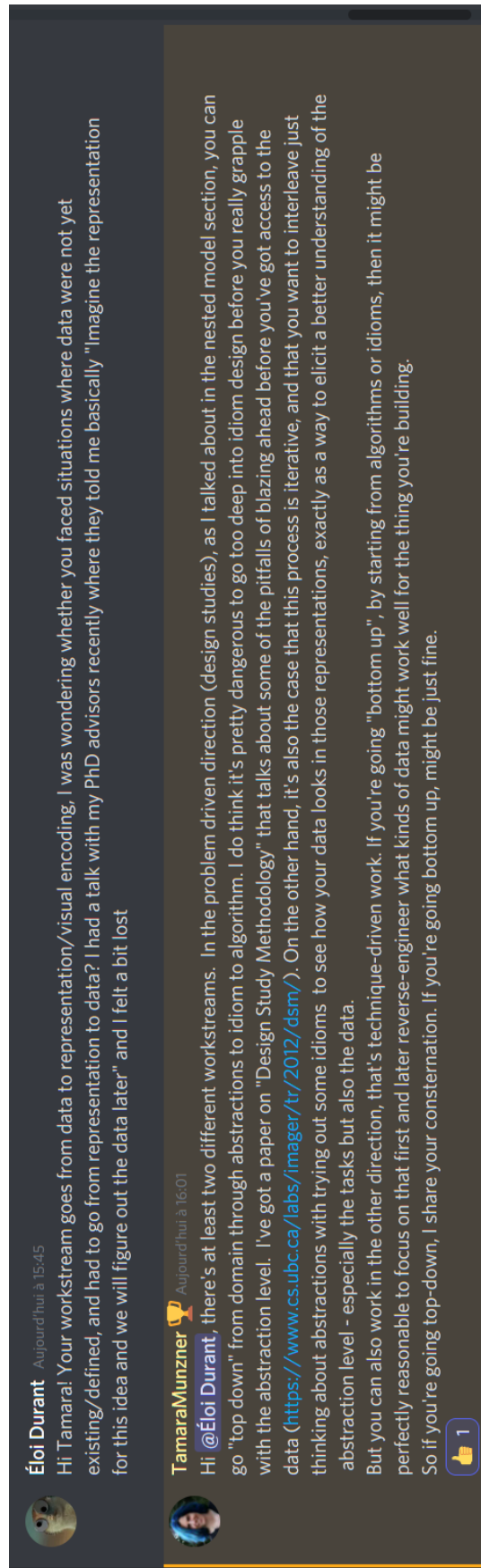


length: integer – the maximum cumulative length of nodes between $]Step_{n,PIVOT}, Step_{n+u,PIVOT}[$ and $]Step_{m,PATH_X}, Step_{m+w,PATH_X}[$.
endIdx: integer – index $n + u$ of the first common node after the SWAP.

Appendix CIX: SV Annot Storage 3/3 – ...with each their own properties

File generation	Input file(s)	View
<p>Positioned Assemblies: PCA, UMAP, whatever...</p>	.tsv (xPos, yPos, metadata)	 <p>↓ Pangenome Selection</p>
<p>Pangenome Detail: Parsed from available pangenomes</p>	.tsv (pangenome name and the assemblies within)	
<p>Conversion? from Syngenta's file ??? What public option ??? Nguyet's Biograph x Parrot? Gfa2json pre-conversion?</p>	.json (panSkeleton + paths) ? .tsv for panCircos?	 <p>↓ Step Selection</p>
<p>« Blind » use case: Reformatted PAV, script from odgi graphs (cf Brett Script), ...</p>	.pav/.tsv (BED-like + pav matrix) .gff (version 3, opt.) .nwk/.txt (opt.)	
<p>« Preformatted » use case: From BEDPAV and gff, use companion script within Panache's repository</p>	.json (from pav and gff) .nwk/.txt (opt.)	

Appendix CX: All views within SaVanache would be connecting different files and formats



Appendix CXI: Usually design studies follow a top-down approach, going from the data to the abstraction; Screenshot of a public conversation on Discord with Tamara Munzner as part of IEEEVIS 2021, October. Tamara Munzner gave a presentation about design studies and how to make visualization tools, following a nested workflow described in her book "Visualization Analysis and Design" [248]. Her workflow starts from domain and data / task abstractions and later focuses on visual idioms and tool implementation. I asked for advice as it was unclear to me how to handle a scenario where data were not properly defined.

Résumé

La démocratisation des technologies de séquençage lors des vingt dernières années a entraîné une explosion du nombre de génomes séquencés. La diversité des génomes de référence ainsi disponible a mis en évidence les biais induits par l'utilisation d'une unique référence, qui n'est pas suffisante pour donner accès à la diversité au sein d'une espèce. Chez les plantes, de nombreux exemples de variations intraspécifiques ont été recensés, notamment de la variation en présence / absence ou en nombre de copies de gènes. Ces variations peuvent exercer une forte influence sur le phénotype des plantes, par exemple chez le riz où la présence d'un gène Sub1A est associée à une tolérance à l'inondation. Pour une meilleure intégration de ces variations en génomique, le concept de pangénome s'est progressivement développé. Un pangénome peut être construit aussi bien pour recenser des gènes que pour tous types de fragments génomiques présents au sein d'un groupe, et est utile pour comparer la répartition de ces éléments entre plusieurs individus. Plusieurs catégories d'éléments existent selon le taux de présence ; les deux principales recensent les éléments présents chez tous les individus (les éléments 'core') et ceux présents seulement chez certains (les éléments 'variable'). La pangénomique souffre encore d'un manque d'outils, notamment pour sa visualisation. Ce manque est particulièrement vrai pour les eucaryotes (dont les plantes), aux génomes plus gros et complexes que les bactéries, premier domaine d'application des pangénomes et dont les outils existants ne supportent pas facilement le passage à l'échelle vers des génomes plus volumineux. Mes travaux de thèse ont donc porté sur la création de nouvelles représentations visuelles ainsi que la création d'outils de visualisation utilisables pour la visualisation de pangénomes de plantes, et d'eucaryotes en général.

Dans ce manuscrit de thèse, je présente l'état de l'art de la pangénomique : j'y fais la distinction entre la notion de pan-gène atlas et de pangénome, le second étant souvent représenté sous la forme d'un graphe où chaque séquence forme un nœud et chaque succession observée de séquence forme des liens entre ces nœuds ; j'identifie également des outils de visualisation non spécifiques, qualitatifs, positionnés, structuraux, et enfin composites. Le premier chapitre recueille dix conseils pour créer un outil de visualisation de données génomiques, à l'attention de futurs chercheur·se·s en biologie ou bio-informatique qui s'intéresseraient à la data visualisation. Le second chapitre, décrit mon premier outil de visualisation de pangénome, publié dans le journal *Bioinformatics* sous le titre « *Panache : a Web Browser-Based Viewer for Linearized Pangenomes* ». J'y détaille la représentation visuelle utilisée dans Panache, jusqu'à la création d'une application web développée en JavaScript permettant l'exploration dynamique de données pangénomiques. Le troisième et dernier chapitre détaille le travail de conception d'un outil composite de visualisation de pangénomes, appelé SaVanache, permettant la navigation entre plusieurs niveaux d'échelle pangénomique. Quatre vues ont été identifiées : une vue de la diversité globale ; une vue des variations structurales ; une vue dédiée à la variation en présence / absence ; et une dernière vue dédiée aux variations nucléotidiques. Je propose une nouvelle approche pour l'annotation et la représentation visuelle de variations structurales au sein d'un graphe de pangénome, axée autour de la définition d'un chemin pivot servant de système de coordonnées principal.

Abstract

The popularization of sequencing technologies in the past twenty years led to a high increase of the number of sequenced genomes. The diversity of the newly sequenced reference genomes highlighted the biases of using a single reference, which is not enough to access all the diversity within a species. There are many examples of intraspecific variations within plants, including presence / absence and copy number variations. These variations can have a strong effect on plant phenotypes, as exemplified by the African rice in which the presence of the gene Sub1A is linked to drought tolerance. The concept of pangenome appeared to better integrate these variations within genomics approaches. A pangenome can be built from genes only or from any genomic fragments found within a group, and is useful to compare their distributions between multiple individuals. Depending on the presence rate, many categories of elements can be defined; the main ones are the elements present in all the individuals (part of the 'core' genome) and these absent in at least one of them (part of the 'variable' genome). Pangenomics still lacks tools, especially for visualization. This is particularly true for eukaryotes (including plants) which have larger and more complex genomes than bacteria. Pangenomes were first built for bacteria, but their related tools cannot properly work on bigger genomes. My PhD investigated the creation of novel visual representations and tools for the visualization of plant pangenomes (and eukaryotes in general).

Within this dissertation, I introduce the state of the art of pangenome visualization: I distinguish pan-gene from pangenomes, the latter often being represented by pangenome graphs where each sequence is a node and each observed sequence succession forms an edge; I also identify unspecific, qualifying, positioned, structural and composite visualization tools. The first chapter introduce ten principles for creating a genomic visualization tool, for future biology or bioinformatics scientists interested in datavisualization. The second chapter describes my first pangenome visualization, published in the journal *Bioinformatics* under the name '*Panache: a Web Browser-Based Viewer for Linearized Pangenomes*'. I detail the visual representation used within Panache and the creation of the resulting web application built in JavaScript, enabling the dynamic exploration of pangenomic data. The third and final chapter details the design of a composite visualization tools for pangenomes, called SaVanache, and enabling the navigation between four view scales. There are four of them: one for global diversity, one for structural variations, one for the presence / absence variations, and one for nucleotide variations. I propose a new approach for the annotation and visual representation of structural variations within a pangenome graph, based on a pivot path within the graph used as a reference coordinate system.