Inference of Gene Networks from Single Cell Data through Quantified Inductive Logic Programming

Samuel Buchet, Francesco Carbone, Morgan Magnin, Mickaël Ménager, Olivier Roux

Inference of Gene Networks from Single Cell Data through Quantified Inductive Logic Programming

Samuel Buchet
samuel.buchet@ec-nantes.fr
LS2N, UMR CNRS 6004, École Centrale de Nantes
Nantes, France

Francesco Carbone
francesco.carbone@institutimagine.org
Université de Paris, Imagine Institute, Laboratory of Inflammatory Responses and Transcriptomic Networks in Diseases, Atip-Avenir Team, INSERM UMR 1163
Paris, France

Morgan Magnin
morgan.magnin@ec-nantes.fr
LS2N, UMR CNRS 6004, École Centrale de Nantes
Nantes, France

Mickaël Ménager
mickael.menager@institutimagine.org
Université de Paris, Imagine Institute, Laboratory of Inflammatory Responses and Transcriptomic Networks in Diseases, Atip-Avenir Team, INSERM UMR 1163
Paris, France

Olivier Roux
olivier.roux@ec-nantes.fr
LS2N, UMR CNRS 6004, École Centrale de Nantes
Nantes, France

ABSTRACT

Single cell sequencing technologies represent a unique opportunity to appreciate all the heterogeneity of gene expressions within specific biological cell types. While these data are sparse and especially noisy, it remains possible to perform multiple analysis tasks such as identifying sub cellular types and biological markers. Beyond revealing distinct sub cell populations, single cell gene expressions usually involve complex gene interactions, which may often be interpreted as an underlying gene network. In this context, logical computational approaches are particularly attractive as they provide models that are easy to interpret and verify. However, the noise is especially important in single cell sequencing data. This may appear as a limit for symbolic methods as they usually fail in addressing the statistical aspect necessary to handle efficiently such noise. In this work, we propose a computational approach based on symbolic modeling to identify gene connections from single cell *RNA* sequencing data. Our algorithm, *LOLH*, is based on Inductive Logic Programming, and intends to rapidly identify potential gene interactions by formulating discrete classification problems, which are solved through discrete optimization. By combining symbolic modeling with optimization techniques, we aim to provide an interpretable model that still fits properly on sparse and noisy data. We apply our method to the unsupervised inference of a gene correlation network from a concrete single cell dataset. We show that the output of our algorithm can be interpreted by using the data itself, and we use additional biological knowledge to validate the approach.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CSBio2021, October 14–15, 2021, Virtual (GMT+7 Bangkok Time), Thailand
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8510-7/21/10…$15.00
https://doi.org/10.1145/3486713.3486746

CCS CONCEPTS

• **Theory of computation → Constraint and logic programming**; • **Mathematics of computing →** *Combinatorial optimization*; • **Applied computing → Biological networks**; *Computational genomics*; • **Computing methodologies → Inductive logic learning**; **Rule learning**.

KEYWORDS

Systems biology, Gene Correlation Networks, Single cell, Inductive Logic Programming, Optimization, Machine Learning.

ACM Reference Format:
Samuel Buchet, Francesco Carbone, Morgan Magnin, Mickaël Ménager, and Olivier Roux. 2021. Inference of Gene Networks from Single Cell Data through Quantified Inductive Logic Programming. In *The 12th International Conference on Computational Systems-Biology and Bioinformatics (CSBio2021), October 14–15, 2021, Virtual (GMT+7 Bangkok Time), Thailand.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3486713.3486746

1 INTRODUCTION

1.1 Context and scientific challenge

Single cell sequencing technologies allow to capture the gene expressions of individual cells coming from a biological sample. These recent technologies are now widely used in molecular biology research. Although single cell gene expressions are particularly noisy, their analysis provides insightful knowledge about biological systems. For instance, they allow the identification of specific sub cell populations and biological markers. The precision of single cell data makes it particularly interesting for the inference of gene networks. In comparison, data coming from traditional sequencing techniques can be seen as an average expression profile of different cells. Thus the heterogeneity of single cell data can be expected to reveal more complex gene interactions. However, the noise and the technical artefacts complicate the design of computational tools, and original computational methods are required to entirely exploit the biological signal available. Despite of numerous tools being developed in single cell analysis, the inference of gene networks remains a difficult task [15].

In this context, symbolic and logical approaches are especially interesting as they help to study genomics data as expression patterns, and allow to easily represent the combined effect of multiple genes. Moreover, gene expression data can be easily abstracted as: expressed/not expressed. However, the noisy aspect is not always considered with this type of algorithms and existing methods would not perform properly on them. In this paper, we propose a computational approach based on logic modeling, to automatically induce logical models from single cell gene expressions. Our algorithm called *Learning Optimized Logical Hypothesis* (abbreviated as *LOLH*) is an extension of existing algorithms in Inductive Logic Programming (ILP), which is a form of machine learning applied to Logic Programming. More precisely, *LOLH* is specifically dedicated to account for the noise by extending the definitions of existing methods, and by using notions from discrete optimization. To our knowledge, such a combination of ILP and statistical machine learning to learn single-cell data has not been investigated before. This conjunction allows us to get a logical output that has the advantage to be a simplification of the input data while being fully interpretable. To show its relevance, we apply our algorithm to compute biological properties on the cells by tackling the induction of co-expression relations between the genes. We propose an interpretation of the resulting logical relations as a graph in order to identify cells of interest, and we perform a comparison with existing single cell tools.

1.2 Originality of the contribution

We are interested in the problem of identifying gene expression patterns, representing the combined effect of several genes in order to reveal interesting properties of a biological system such as complex gene interactions. Several scientific barriers are encountered to address this question. On the experimental side, advanced technologies are needed to obtain a reasonable amount of data, with a precision that is sufficient enough to enable the capture of such patterns. This challenge seems well addressed by single cell technologies as they provide an unprecedented amount of data that is precise enough to reveal complex relations such as sub cellular types. On the computational side, the difficulties lie in the combinatorial and large amount of input data, as well as a need for explanation of machine learning methods. To address this problem, we propose a new method combining inductive logic programming (ILP) and statistical machine learning. This combination aims at overcoming the respective limitations of each of these two approaches taken individually. Inductive logic programming (ILP) leads to interpretable logic rules, which are however very sensitive to noise in the input data. At the same time, classical machine learning approaches are efficient in handling large data sets, but struggle to be explainable. The algorithm proposed in this paper, named Learning Optimized Logical Hypothesis (*LOLH*), achieves a simplification of the input data while providing an interpretable representation of the learned features.

We will now describe the general principle of *LOLH*. As input, we process gene expression data for multiple cells. The input cells are assumed to represent a binary classification problem, they are thus separated into two sets: positive examples, i.e. expression data corresponding to the class that is learned; and negative examples, i.e. expression data not corresponding to this class. This separation that has been proposed by the so-called *Learning From Interpretation Transition* framework (abbreviated hereafter as *LFIT*) [10] to learn dynamic data expressed as successions of states-transitions allows us to reduce every learning task to a binary classification. Although considering classification problems may be seen as reductive regarding the complexity of the biological properties hidden in the data, our assumption is that such binary classification tasks, when judiciously identified from the input data, are able to represent complex knowledge. Given the encouraging results obtained by *LFIT*, our proposition thus consists in an extension to handle high noise. To do so, we introduce a new notion which was not previously included in the existing *LFIT* framework, namely a score for each rule that is optimized regarding the positive and negative examples in order to provide generality of the model on the data. Focusing on the generality of the rules ensure the ability of the algorithm to avoid overfitting on the data. This approach then leads to a unique optimal rule, which is the aggregation of a reduced number of genes. This rule, which can be interpreted, is in a way the signature of a given learned concept (e.g. cell type, co-expression). It can thus serve as a predictor for the positive concept. To put the approach into practice, we show several possibilities to identify the positive and negative examples from a single-cell dataset. We propose a first example to perform cellular classification using labels provided in the dataset as a supervised learning task. We also demonstrate the application of our method to learn co-expression relations between the genes, as an unsupervised approach. This latter application is developed in order to obtain interesting properties about the dataset that may not be revealed by existing computational approaches for single cell data.

1.3 Overview of the paper

In this paper, we first show how to adapt existing Inductive Logic Programming approaches on noisy data, in order to account for the noise and still provide a formal and explainable model. To this aim, the logical definitions are refined for the quantification of a model error on the data, and optimization techniques are used to select the most relevant features. The method is then applied to a single cell dataset provided by Imagine Institute. It is first illustrated through the induction of gene markers for the identification of cell types. We show that our approach account for the noise and recover biologically relevant information from the gene expressions. Finally, we demonstrate its application to the inference of gene co-expression (correlation) networks, in order to discover relevant biological properties. The inference is performed unsupervisedly on all the genes and without the use of background knowledge, and different approaches are proposed to verify the relevance of the network by using the data itself and already known biological markers. Our implementation of *LOLH* used to produce all the results in this paper is available online at http://doi.org/10.5281/zenodo.4738850.

2 SINGLE CELL DATA AND RELATED WORKS

In this section, we introduce the single cell sequencing technologies and the different types of data produced in this context. We introduce more specifically the dataset used in this study. We then review multiple approaches developed for the inference of gene networks and logical models from single cell data.

2.1 Single cell sequencing technologies

Single cell genomics represent a set of techniques for the isolation and the measurement of a large amount of cells. Different technologies have been developed, allowing to measure several biological aspects of the cells such as proteomics, spatial genomics and chromatin accessibility. Due to the technical difficulty of isolating the cells and measuring small amount of molecules, these data are especially sparse and noisy. Even so, they allow to study a biological system at a very precise level and they have become widely used in biological research.

In this work, we focus on transcriptomic data, i.e. the sequencing of the *mRNA* at the single cell level, abbreviated hereafter as *scRNA-seq*. *scRNA-seq* data usually consists in a matrix containing the expressions (real values) of multiple genes for several cells. Our method is intended to be applied to various datasets but we focus on one specific *scRNA-seq* dataset, containing mononuclear cells extracted from blood samples (*pbmc*). The dataset already processed has been provided by Imagine Institute[1] (the processing was performed with *Seurat*[2] [8]). More details about the data are available in Appendix B. Each cell of the dataset has been provided with a label designating the cell type, which have been identified beforehand using cell clustering combined with manual annotation. A two dimensional representation of the cells, computed with the *UMAP* method [17] have also been provided through *2d* coordinates for all the cells, and is designated hereafter as the *UMAP* representation of the cells. Fig. 1 shows the *UMAP* representation of the cells with colors indicating the labels of the cell types provided with the data. For the application of our method, we have empirically discretized the genes of the normalized (i.e. processed) data into multiple discrete values, using the gene expression distributions. It results into 14030 genes having 2 discrete values, 134 genes with 3 values and 42 genes with 4 values. The script used to discretize the data is available within the implementation.

2.2 Modeling gene networks in systems biology

Modeling gene networks has always represented a main challenge in systems biology. The availability of large scale genomics data encourages the development of models and computational tools to better understand biological systems, but the consideration of up to thousands of variables makes this task especially difficult

[1]The dataset is available online at http://doi.org/10.5281/zenodo.4730807
[2]*Seurat* is available at https://satijalab.org/seurat

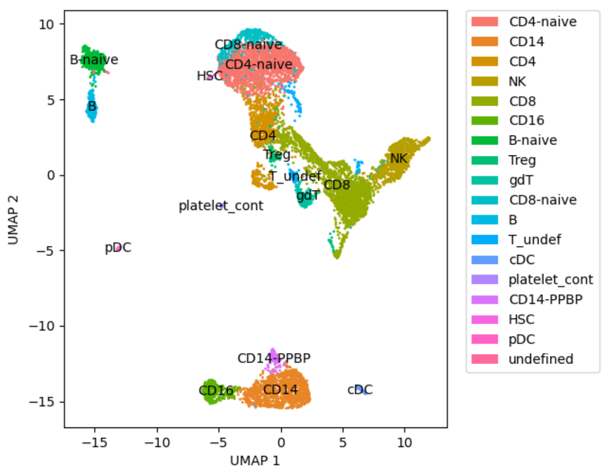

Figure 1: UMAP visualization of the 9198 cells with pre-identified cell types. Each point represents one cell, and is colored according to its corresponding cell type.

on both the modeling and the computational aspects [22]. Many approaches based on gene networks have been proposed to understand the relationships between the genes [18]. Other approaches such as Boolean Networks use a symbolic/discrete representation of the gene expression and propose logical relations to model their dynamical behaviors [12, 24]. In this context, single cell sequencing is especially interesting since it allows to obtain the gene expressions of thousands of cells in one single experiment.

Although network inference methods originally designed for traditional genomics data have been applied to single cell data [19], several methods have been specifically created for *scRNA-seq*. Most of them focus on the inference of directed edges between the genes, representing direct influences. Some approaches like *SCODE* [16] are based on Ordinary Differential Equations, and rely on pseudotime information provided by algorithms such as *Monocle* [25]. Other approaches such as *SCENIC* [1] rely on statistical and machine learning methods. Since single cell technologies have also been developed for other type of omics data, some network inference methods also consider alternative data such as *scATAC-seq*, and their combination with *scRNA-seq* [9]. Lastly, Boolean Networks have also been applied to single cell data analysis with different tools such as *SCNS* [26] and *BTR* [14]. Benchmarking methodologies have been designed to evaluate and compare these different methods on artificial and concrete datasets [4, 11, 19].

Our framework *LOLH* presented in this paper relies on multi-valued logic programs, a form of logic modeling which allows the use of more than two discrete values per gene. It intends to take advantage of both logical modeling and statistical machine learning aspects to produce a formal model while accounting for the noise in the data. Like *BTR*, a scoring function is used to induce the model, but our method performs locally instead of considering the state space of the system. This paper presents one possible application of our framework comparable to the application of *SCENIC*, through the unsupervised inference of gene correlation networks. However, several other uses of *LOLH* are also considered, including the inference of dynamical relations between the genes.

3 EXTENSION OF INDUCTIVE LOGIC PROGRAMMING TO SINGLE CELL GENE EXPRESSIONS DATA

Inductive Logic Programming is a research topic consisting in the application of machine learning to the field of Logic Programming. One motivation of learning logic programs is the ability to perform formal verification on the induced programs, in order to analyse its properties. This aspect is particularly interesting in systems biology since it would be difficult to exploit and to verify the biological properties of a *blackbox* model.

Inductive Logic Programming algorithms consider as input a set of positive and negative examples, and background knowledge on the system [5]. Several algorithms tackling different goals have been proposed to infer the Logic Programs. In this work, we focus on the framework *LFIT* (Learning From Interpretation Transition), which intends to infer a Multi Valued Logic Program from time series data. This framework has been previously applied on biological systems to model their dynamical behavior [10]. Up to now, the experimental and noisy aspect of genomics data has not been properly studied in *LFIT*. As single cell data is particularly sparse and noisy, we discuss the behavior of existing algorithms and we propose an extension that aims to be more adapted.

The next sub-section introduces the formal definitions of the *LFIT* framework. Then, we study the application of existing *LFIT* algorithms on a *scRNA-seq* data, and we propose a new approach that intends to better account for the specificity of the data. The different approaches are compared on one practical induction task, formulated from the single cell dataset introduced in subsection 2.1.

3.1 Learning biological systems from Interpretation Transitions

LFIT algorithms aim to model a dynamical discrete system as a multi-valued dynamical logic program. *LFIT* takes as input a set of discrete transitions, represented as pairs of discrete states usually extracted from time series data. The output of the algorithm is a multi-valued logic program, i.e. a set of logic rules representing the dynamical behavior of each discrete variable. In this work, we focus especially on the induction of the logic rules. Although *LFIT* intends to learn the dynamics of a system, the learning of every variable is always transformed into a binary classification problem, from which the logic rules are induced. By focusing on properly identified classification problems on *scRNA-seq* data, we hope to improve existing algorithms as a first step. The following definitions formalizes the different notions used in *LFIT* algorithms.

3.1.1 Atoms and discrete states. We consider a set of multi-valued logic variables $\mathcal{V}$, with $|\mathcal{V}| = n$, representing genes with multiple discrete values. The discrete values of a variable are represented by logical atoms. For instance, if a variable $a \in \mathcal{V}$ has 3 discrete values, the corresponding atoms are denoted: $\{a_{0/2}, a_{1/2}, a_{2/2}\}$. Note that the maximum discrete value of a is mentioned in its atoms for purposes of clarity. A discrete state s of the system is a vector containing one atom per variable (e.g. one discrete value per gene), for example $s = (a_{1/1}, b_{0/1}, c_{1/1})$. The set of all states of the system is denoted as $\mathcal{S}$. The notation $s(x) = x_{i/j}$ is used to represent the atom corresponding to the variable x in the state s (e.g. $s(b) = b_{0/1}$ in the previous example).

3.1.2 Multi-valued logic rules. The logic rules forming the logic program have the following form: $r = a_{1/1} \longleftarrow a_{0/1}, b_{1/1}, c_{1/1}$, Where $a_{1/1}$ is called the head of r, denoted $head(r)$ and $a_{0/1}, b_{1/1}, c_{1/1}$ is called the body of r, denoted $body(r)$. A rule r matches on a state $s \in \mathcal{S}$, denoted $match(s, r) = \top$, if $x_{i/j} = s(x) \quad \forall x_{i/j} \in body(r)$. Usually, multiple rules are proposed with the same head in order to represent complex knowledge.

3.1.3 Learning task in LFIT. In the *LFIT* framework, the logic rules are induced independently for each variables. To do so, the discrete states of the system are separated into positive and negative examples, denoted $\mathcal{S}^+$ and $\mathcal{S}^-$ respectively. The positive examples aim to represent the data on which the rules are supposed to match, and the negative examples represent the data on which they should not.

3.2 Extension for noisy data: application of *LFIT* to *scRNA-seq*

To extend the application of the *LFIT* framework to noisy discrete data, we focus on the rule induction task, formulated from positive ($\mathcal{S}^+$) and negative ($\mathcal{S}^-$) examples. We consider the behaviour of two general and recent implementations: *GULA* [20], and *PRIDE* [21], and discuss their relevance when applied to positive and negative examples identified from noisy discrete data. We then propose a new algorithm that intends to overcome the difficulties presented by this new application.

3.2.1 GULA and PRIDE algorithms. Let $\mathcal{P}$ be a logic program, i.e. a set of logic rules created from $\mathcal{S}^+$ and $\mathcal{S}^-$. As the logic rules of $\mathcal{P}$ should be consistent with $\mathcal{S}^+$ and $\mathcal{S}^-$, the following property is used as a starting point: every positive example must be matched at least once, and no negative example must be matched, i.e: $\left(\forall s^+ \in \mathcal{S}^+, \exists r \in \mathcal{P} : match(r, s^+) = \top\right) \wedge \left(\forall s^- \in \mathcal{S}^-, \nexists r \in \mathcal{P} :. match(r, s^-) = \top\right)$ However, many different sets of rules are consistent with this definition. For example, it is possible to create one rule per positive example, which results into an *overfitting* on the data. For this reason, an optimality criterion is used to specify the desired properties. In *GULA*, $\mathcal{P}$ is optimal if its rules match on as many examples as possible, while being consistent with negative examples [20]. In this case, $\mathcal{P}$ is proved to be unique. Since the computation of such program is expensive, a heuristic version named *PRIDE* has also been proposed, where the optimality is not formally proven.

We identify two main issues with these *LFIT* implementations. First, the logical matching definition between rules and states makes the generalization over noisy states difficult. Indeed, assuming a gene expression pattern composed of multiple genes is visible in the data, it is not expected to observe any positive state verifying completely the pattern due to the noise. Thus, a rule matching on this pattern is not likely to be proposed since it would not completely match on any positive examples. Similarly, a mismatch of a rule by only one condition in its body does not help to properly account for the negative examples, i.e. a rule proposed by *GULA* could be very close from matching the negative examples in the data. The second issue comes from the definition of optimality from *GULA*, and its application to a system that is only partially observed. Indeed, assuming that many negative states are not observed, inducing the most general rules will tend to *overfit* on the negative examples. In this case, the generalization would go beyond the observed positive examples and the rules would not be able to correctly represent the patterns showing up in the input data.

3.2.2 Extension through rule-state matching error. To overcome the difficulties identified previously, we propose a new definition of the matching predicate on a logic rule. In order to identify a gene pattern from noisy discrete states, we introduce the *matchingError* function between a rule and a state, that aims to estimate how well the pattern is present in the state. The matching error corresponds to the number of atoms from the body that do not match in the state, and it is defined as follows:

$$matchingError(r,s) = \sum_{x_{i/j} \in body(r)} \left(1 - \mathbb{1}_{\{s(x)\}}(x_{i/j})\right) \quad (1)$$

With $\mathbb{1}_A(x) = 1$ if $x \in A$, 0 otherwise, alternatively formulated as $matchingError(r,s) = |\{x_{i/j} \in body(r) \mid x_{i/j} \neq s(x)\}|$. From this definition, we introduce a new optimization criterion in order to search for the most interesting rule. Since the rule intends to match as much as possible on the positive examples, the matching error should be the smallest possible on these states. On the contrary, the error should be as big as possible on the negative examples. Using the mean of the matching error as a global score over multiple states, this leads to the following bi-objective optimization problem on the rule:

$$\begin{aligned} &\text{- } \min_{\boldsymbol{r}} \quad \frac{1}{|\mathcal{S}^+|} \sum_{s^+ \in \mathcal{S}^+} matchingError(\boldsymbol{r}, s^+) \\ &\text{- } \max_{\boldsymbol{r}} \quad \frac{1}{|\mathcal{S}^-|} \sum_{s^- \in \mathcal{S}^-} matchingError(\boldsymbol{r}, s^-) \end{aligned} \quad (2)$$

To simplify this problem, we propose to maximize the difference between the two mean matching errors of the rule. Thus the positive and the negative examples are equally considered for the global score. This leads to the following criterion to maximize (where *mError* stands for *matchingError*):

$$\max_{\boldsymbol{r}} \left(\frac{1}{|\mathcal{S}^-|} \sum_{s^- \in \mathcal{S}^-} mError(\boldsymbol{r}, s^-) - \frac{1}{|\mathcal{S}^+|} \sum_{s^+ \in \mathcal{S}^+} mError(\boldsymbol{r}, s^+) \right) \quad (3)$$

By developing the *matchingError* term, one can notice that the mean matching error can be decomposed by computing an individual error for each atom, as it is shown below (where $*$ corresponds to $+$ for the positive examples and $-$ for the negative examples):

$$\frac{1}{|\mathcal{S}^*|} \sum_{s \in \mathcal{S}^*} mError(r,s) = \frac{1}{|\mathcal{S}^*|} \sum_{x_{i/j} \in body(r)} \sum_{s \in \mathcal{S}^*} \left(1 - \mathbb{1}_{\{s(x)\}}(x_{i/j})\right)$$

From this expression, our optimization criteria can be re-written as the sum of individual atom scores that need to be optimized. Thus, we define $score(x_{i/j}) \in [-1, 1]$ as follows:

$$score(x_{i/j}) = \frac{error_-(x_{i/j})}{|\mathcal{S}^-|} - \frac{error_+(x_{i/j})}{|\mathcal{S}^+|} \quad (4)$$

With:

$$error_+(x_{i/j}) = \sum_{s^+ \in \mathcal{S}^+} \left(1 - \mathbb{1}_{\{s^+(x)\}}(x_{i/j})\right)$$
$$error_-(x_{i/j}) = \sum_{s^- \in \mathcal{S}^-} \left(1 - \mathbb{1}_{\{s^-(x)\}}(x_{i/j})\right)$$

These scores can be efficiently computed from the dataset by iterating over all the examples. As an intermediary step, we fix the number of atoms of the body to a constant k. The choice of this constant is discussed hereafter. We finally obtain the following optimization criterion equivalent to expression 3, for the inference of a logic rule with k atoms:

$$\max_{r, |body(r)|=k} \sum_{x_{i/j} \in body(r)} score(x_{i/j}) \quad (5)$$

When the number k of atoms is fixed, this method allows for the efficient computation of one rule which generalizes over the positive and the negative examples at the same time, by sorting the atoms according to their scores. For the sake of clarity, the next example illustrates the computation of the atom scores using the previous definitions on a small dataset, in order to compose the body of an optimal rule.

Example 3.1. Let $S^+ = \{(a_{0/1}, b_{1/1}, c_{0/1}, d_{1/1}),$ $(a_{0/1}, b_{0/1}, c_{1/1}, d_{0/1}), (a_{0/1}, b_{1/1}, c_{0/1}, d_{1/1})(a_{0/1}, b_{1/1}, c_{1/1}, d_{1/1})\}$ and $S^- = \{(a_{1/1}, b_{1/1}, c_{1/1}, d_{1/1}),$ $(a_{1/1}, b_{1/1}, c_{0/1}, d_{1/1}), (a_{0/1}, b_{1/1}, c_{1/1}, d_{1/1}), (a_{1/1}, b_{1/1}, c_{0/1}, d_{1/1})\}$ be a dataset of 4 positive and 4 negative examples for a set of 4 discrete variables $\mathcal{V} = \{a, b, c, d\}$ (with 2 discrete values each). In this dataset, one can notice that the value 0 is dominant for a in $\mathcal{S}^+$ and 1 is dominant for b in $\mathcal{S}^-$. Thus, the atoms $a_{1/1}$ and $b_{0/1}$ have the best scores, and a and b are efficient to differentiate between $\mathcal{S}^+$ from $\mathcal{S}^-$. For instance, $error^+(a_{0/1}) = 0$, $error^-(a_{0/1}) = 3$, thus $score(a_{0/1}) = \frac{error^-(a_{0/1})}{4} - \frac{error^+(a_{0/1})}{4} = 3/4 = 0.75$. Regarding variable b, $score(b_{1/1}) = \frac{3}{4} - \frac{0}{4} = 0.75$ On the contrary, c has no dominant value in $\mathcal{S}^+$ and $\mathcal{S}^-$, thus it cannot be used to differentiate between the two subsets, e.g $error^+(c_{0/1}) = 2$, $error^-(c_{0/1}) = 2$, and $score(c_{0/1}) = 0$. Lastly, d is almost constant across $\mathcal{S}^+$ and $\mathcal{S}^-$. Thus, d is not effective as well to classify $\mathcal{S}^+$ from $\mathcal{S}^-$, which can be verified on the score: $score(d_{1/1}) = 0/4 - 1/4 = -0.25$. Thus, the most interesting rule in this example can be created with the atoms $a_{0/1}$ and $b_{1/1}$ which have a score close to 1, i.e. $r = concl. \leftarrow a_{0/1}, b_{1/1}$, where *concl.* (conclusion) is the head, representing the logical concept described by $\mathcal{S}^+$ and $\mathcal{S}^-$.

Expression 5 can be used to compute an optimal rule when the length of the body is fixed. However, deciding about the number of atoms to include in a rule can be difficult. For instance, one may try to compare the scores of two rules having different lengths. However, when the rules do not have the same number of atoms in their body, their score are not comparable (since the score is in the range of 0 to the number of atoms). Although it is possible to normalize the score in expression 5, the optimal rule regarding the normalized score would always contain only one single atom. Indeed, adding an additional atom can only worsen a normalised score, as logical atoms are sorted out beforehand. A logic rule containing only one atom would not be robust for the classification considering the noise of the data, thus it is preferable to select multiple atoms to form a rule. As atoms with scores close to 0 should be avoided to, we propose to select the atoms based on a threshold t on their score. Such threshold intends to find a compromise between the quality and the robustness of the prediction. The body of the optimal rule r_{opt} is then created by selecting the best atoms according to the threshold t: $body(r_{opt}) = \{x_{i/j} \mid score(x_{i/j}) \geq t\}$. A more detailed example illustrates the application of *LOLH* on a toy dataset in appendix A with the computation of a rule from the atom scores by using a threshold.

3.3 Comparisons through cell type classification on *scRNA-seq*

In this sub-section, we compare the performance of *LOLH* with an existing *LFIT* implementation, through a cell type classification task. In this case, the goal is to predict the type of a cell from the expressions of the genes. For the illustration, we selected the *NK* cells (*natural killer*) previously identified in the data (see Fig. 1). This cell type is interesting to evaluate the method since these cells have gene expressions that are similar to the *CD8* cells. While the ultimate goal of inducing logic programs is the representation of gene interactions, cell classification is a relevant point of comparison since biological knowledge, such as gene markers, can be used to interpret the output of the algorithms.

Since the *GULA* algorithm would not be applicable to an entire dataset due to its computational complexity, we use the heuristic version *PRIDE*, with the python implementation *pylfit*[3]. From the cell type labels provided with the single cell matrix, a binary classification problem can be formulated on the discretized cells, which leads to 829 positive examples $\mathcal{S}^+$ (cells labeled as *NK*), and 8369 negative examples $\mathcal{S}^-$ (cells not labeled as *NK*) over 14206 genes. Thus in this example $|\mathcal{S}^+| = 829$ and $|\mathcal{S}^-| = 8369$.

This binary classification problem has been given as input to *PRIDE* and to *LOLH*. *PRIDE* returned a logic program with 593 rules, presenting up to 96 atoms from the dataset in their bodies. On the other side, *LOLH* applied with a threshold of 0.55 returned one optimized rule containing the following 21 atoms in the body:
$GNLY_{1/1}$, $GZMB_{1/1}$, $NKG7_{1/1}$, $PRF1_{1/1}$, $KLRD1_{1/1}$, $CTSW_{1/1}$, $KLRF1_{1/1}$, $CST7_{1/1}$, $GZMA_{1/1}$, $HOPX_{1/1}$, $KLRB1_{1/1}$, $IL2RB_{1/1}$, $TYROBP_{1/1}$, $CD7_{1/1}$, $CD247_{1/1}$, $CMC1_{1/1}$, $CLIC3_{1/1}$, $SPON2_{1/1}$, $MATK_{1/1}$, $FCER1G_{1/1}$, $B2M_{2/3}$
The matching error of *LOLH* rule on the data can be visualized as histograms in Appendix D.1. In the following, the logic program returned by *PRIDE* is denoted as $\mathcal{P}_{PRIDE}$ and the logic program returned by our optimization method is denoted as $\mathcal{P}_{LOLH}$ (consequently $|\mathcal{P}_{PRIDE}| = 593$ and $|\mathcal{P}_{LOLH}| = 1$).

[3] *pylfit* package can be found at https://github.com/Tony-sama/pylfit.git

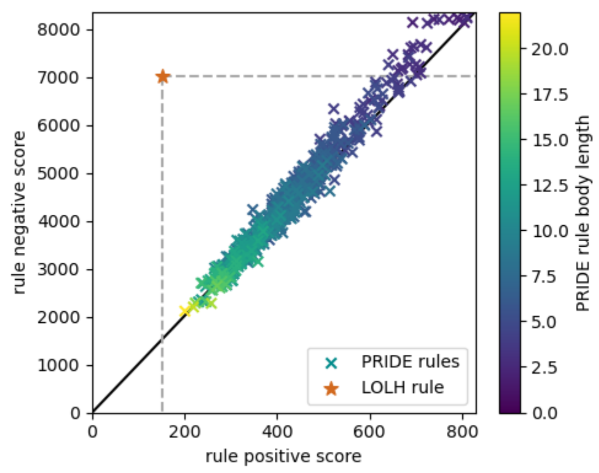

Figure 2: Comparison between *LOLH* rule (in orange) and *PRIDE* rules (colored crosses). Brightest *PRIDE* rules indicates the longest bodies. The dotted lines delimit all the points for which both positive and negative scores are worst than the *LOLH* score. The black line indicates proportionally equal positive and negative score.

3.3.1 *Comparison of mean rule matching scores.*

A first comparison is proposed through the mean matching error over the positive and the negative examples of the rules from $\mathcal{P}_{PRIDE}$ and $\mathcal{P}_{LOLH}$. Since the different rules may not have the same number of atoms in their bodies, the positive and negative means are normalized by the number of atoms in every body. Using the definitions previously introduced, the scores (i.e. normalized error) of each rule r on the positive and negative examples are defined as:

$$\begin{aligned} &\text{- } score_+(r) = \left(\sum_{s \in \mathcal{S}^+} matchingError(s, r)\right) / |body(r)| \\ &\text{- } score_-(r) = \left(\sum_{s \in \mathcal{S}^-} matchingError(s, r)\right) / |body(r)| \end{aligned} \tag{6}$$

The scores of all the rules are shown in Fig. 2, where the black line corresponds to the rules having proportional scores on $\mathcal{S}^+$ and $\mathcal{S}^-$. Since it is intended to minimize the score on $\mathcal{S}^+$ and maximize the score on $\mathcal{S}^-$, the dotted lines delimit all the rules that perform less than the rule in $\mathcal{P}_{LOLH}$. It can be noticed that most of the rules from $\mathcal{P}_{PRIDE}$ are inside the area delimited by these lines, indicating that according to mean matching error defined previously, they perform significantly less than the rule in $\mathcal{P}_{LOLH}$. Moreover, they mostly lie on the black line, meaning that they fail at differentiating between the positive and negative examples. It is still possible to identify two effects induced by the search performed by *PRIDE*: rules with a largest body deviates from the black line, indicating that they are slightly better, and rules with the shortest bodies have also the smallest scores on both $\mathcal{S}^+$ and $\mathcal{S}^-$.

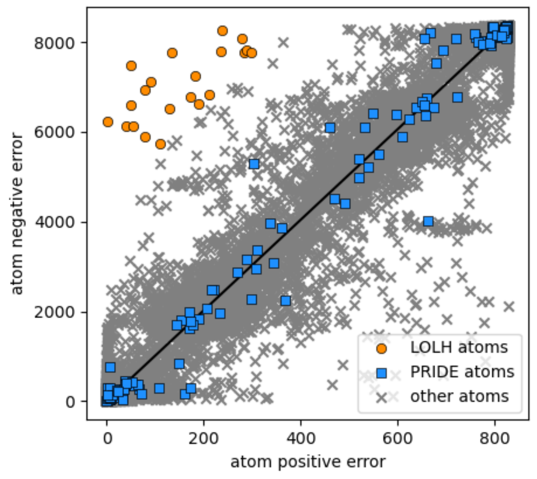

Figure 3: Visualization of the individual scores of all atoms. The blue atoms are the atoms selected by *PRIDE* and the orange atoms are the atoms selected by *LOLH*. The grey atoms are the remaining atoms from the dataset. The black line indicates atoms having proportionally the same error on the positive and negative examples.

3.3.2 Comparison on each atom. Since it remains difficult to analyse individually each of the 593 rules of $\mathcal{P}_{PRIDE}$, we alternatively propose to compare all the atoms from the bodies of the rules in $\mathcal{P}_{PRIDE}$ and $\mathcal{P}_{LOLH}$. The comparison can be performed on their individual positive and negative errors, as defined in subsection 3.2.2. Fig 3 shows the positive and negative errors of all possible atoms from the data (i.e. one atom per discrete value, per gene). In this graph, the black line also represents the points for which the positive and negative errors are proportionally equal. It can be directly noticed that the orange circles, corresponding to the atoms from $\mathcal{P}_{LOLH}$, are the most distanced from the back line, as a result of the optimization of the weighted difference between the two scores performed by *LOLH*. On the contrary, the atoms from $\mathcal{P}_{PRIDE}$, indicated in blue, mostly lie on the black line. It shows that they do not help to differentiate between the positive and the negative examples. While this does not inform about the effect of combining the atoms together into the different rules, this explains the location of the rule mean errors in Fig. 2 and the better performance of $\mathcal{P}_{LOLH}$ rule regarding the rule matching error.

3.3.3 Comparison with differential expression analysis and biomarkers. To further assess the relevance of the atoms selected by *PRIDE* and *LOLH*, it is possible to rely on existing analysis tools in single cell genomics, such as differential expression (DE) analysis. Given two groups of cells, (DE) analysis tools compute genes that are the most differentially expressed between these two groups. We performed (DE) analysis on the dataset with *Seurat* (with FindMarkers function), using the cells labeled as *NK* as the first group, and the remaining cells as the second group (the script used for this analysis is available within our *LOLH* implementation). The resulting table containing the 30 most differentially expressed genes, sorted by the third column, can be found in Appendix D.2. In this table, the second and the last columns indicate the statistical significance of the result, and the third column indicates how much the gene is differentially expressed between the two groups of cells (positive values indicate that the gene is more expressed in the first group, i.e. in the *NK* cells in this case) (more details are available in the documentation of *Seurat*). We can notice that most of the genes used in $\mathcal{P}_{LOLH}$ rule are in the table, except *TYROBP*, *MATK*, *FCER1G* and *B2M*. On the contrary, genes selected in $\mathcal{P}_{PRIDE}$ are not found in the table, which seems consistent with the observation in Fig. 3. Lastly, using known biomarkers, the (DE) analysis can be used to justify that the *NK* cells are indeed correctly labeled in the dataset, and that *LOLH* selected relevant genes for the classification. Appendix C shows a list of genes associated to different type of cells. The genes *NKG7*, *PRF1* and *IL2RB* are indeed mainly found in *NK* cells. In addition, other genes that are usually expressed (not only) in *NK* cells are found, such as *GNLY*, *GZMB*, *KLRD1*, *CST7* and *FCER1G*. None of the genes selected by *PRIDE* have been found to be relevant to characterise the *NK* cells.

4 INFERENCE OF GENE CO-EXPRESSION NETWORKS

Subsection 3.3 has shown the application of *LOLH* on positive and negative examples obtained from already identified cell types. Yet, this framework becomes especially interesting when it is applied in an unsupervised way. In order to create a model of gene interactions, we propose to formulate the classification problems directly from the discrete expressions of the genes, in order to link together genes that are globally expressed in the same cells. By capturing correlations between the genes in the dataset, the resulting model can be expected to bring interesting insights about the structure of the dataset.

Considering an atom $x_{i/j}$ from the dataset, representing the discrete value i of the gene x, the sets of positive and negative examples are formed from the following definition: $\mathcal{S}^+_{x_{i/j}} = \{s \in \mathcal{S} \mid s(x) = x_{i/j}\}$ and $\mathcal{S}^-_{x_{i/j}} = \mathcal{S} \setminus \mathcal{S}^+_{x_{i/j}}$, i.e. the positive examples are the cells where the gene x has the discrete value i and the negative examples are the cells where x does not have the value i. In this classification task, the gene x is discarded from the data, so that $x_{i/j}$ is not selected by *LOLH*. We propose to use *LOLH* on this classification problem with an appropriate threshold, and to interpret the resulting logic rule as edges between the atom $x_{i/j}$ and the atoms in the rule's body. Thus, creating one classification problem per atom leads to the inference of a global weighted correlation/co-expression network between all the atoms, over all the cells of the dataset. It is also interesting to quantify the influence of each atom selected in the body of the rule, thus expression 4 is used to compute a weight for each edge of the network. The analysis of the resulting network can lead to useful information about the way genes are expressed in the cells, as it is shown in the following sub-section.

4.1 Identification of biological knowledge through gene clustering

Using the methodology introduced previously, a global graph of all the atoms has been generated from the complete discrete single cell matrix. In order to account for relevant correlations while having a sufficient number of edges in the graph, a threshold of 0.35 have been selected to compute the logic rules with *LOLH*. Additionally, constant genes are usually not informative, thus genes varying in less than 200 cells are not considered. Lastly, mitochondrial and ribosomal genes are also discarded since they are not expected to be informative and they may complicate the analysis of the graph.

The resulting network can by analyzed by performing clustering on the atoms. The graph based clustering algorithm *Louvain* [3] has been used in this purpose, through the python implementation *python-louvain*[4]. In this graph, the application of *Louvain* clustering leads to 6 clusters of atoms. The content of the clusters can be found in Appendix E.1. Some symmetries are observed between the clusters, i.e. some clusters contain globally the same genes with opposite discrete values. Thus, the following pairs of clusters can be considered together: clusters 0 and 4, clusters 1 and 3, and clusters 2 and 5. For this reason, we focus our analysis on clusters 3, 4 and 5, which mainly contain atoms corresponding to expressed genes.

To associate the cells of the dataset with each cluster of atoms, the clusters can be interpreted as the bodies of logic rules in order to compute a matching error on all the cells using Expression 1. The resulting cells with the lowest matching error are then considered to be representative of the clusters. Fig. 4 allows to visualize the matching error of the clusters on all cells as colors on the *UMAP* representation. In this figure, the symmetries are clearly visible, and the clusters can be easily associated with some cell types identified in Fig. 1 and Fig. 8, additionally with the help of the biological markers from Appendix C as done previously. Thus, cluster 3 seems to match on differentiated *T* cells and *NK* cells, which is confirmed by the presence of $GNLY_{1/1}$, $IL2RB_{1/1}$, $GZMB_{1/1}$, $CST7_{1/1}$, $NKG7_{1/1}$, $CCL5_{1/1}$, $KLRD1_{1/1}$, $PRF1_{1/1}$, $GZMH_{1/1}$, $CD8A_{1/1}$ and $CD8B_{1/1}$. Cluster 4 can be associated with the *myeloids*, confirmed by the presence of $AIF1_{1/1}$, $CD14_{1/1}$, $CST3_{1/1}$, $FCER1G_{1/1}$, $FCN1_{1/1}$, $LGALS3_{1/1}$, $LST1_{1/1}$, $LYZ_{1/1}$, $S100A4_{2/2}$, $S100A9_{1/1}$ and $TYROBP_{1/1}$, as well as the following atoms indicating the absence of genes associated with *T* cells: $CD3E_{0/2}$, $IL32_{0/2}$, and $IL7R_{0/1}$. Lastly, cluster 5 matches on *B* cells, confirmed by the presence of $BLK_{1/1}$, $CD37_{2/2}$, $CD74_{1/1}$, $CD79A_{1/1}$, $CD79B_{1/1}$, $HLA\text{-}DMA_{1/1}$, $HLA\text{-}DQA1_{1/1}$, $MS4A1_{1/1}$, $PAX5_{1/1}$, $SPIB_{1/1}$, $TCL1A_{1/1}$ and $VPREB3_{1/1}$. However, cluster 3 and cluster 4 in Fig. 4

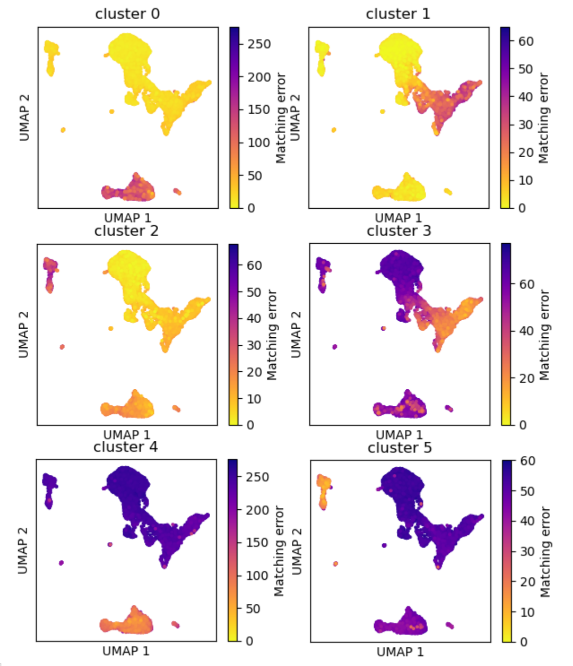

Figure 4: Visualization of the matching error of the atoms from each cluster on all the cells on the UMAP representation.

[4] *python-louvain* is available at https://github.com/taynaud/python-louvain.git

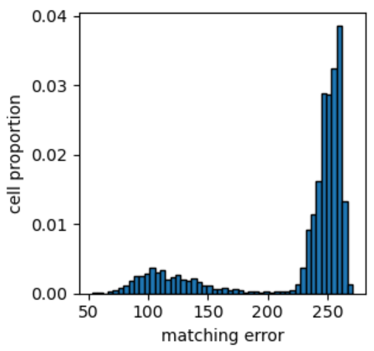

Figure 5: Histogram of the matching error of cluster 4 on all the cells.

seems to indicate the presence of wrongly classified cells in the *myeloids*. To improve the characterization of the clusters, and to further study the cell heterogeneity within clusters, it is possible to perform network refinement through cell selection. We illustrate this by selecting cells matching on cluster 5, corresponding to the *myeloids*. The selection is performed by selecting cells with a matching error ≤ 193 on cluster 5, which can be visualized in the histogram illustrated in Fig. 5.

4.2 Network refinement on the *myeloids* cells

Using the previous cell selection method, we created a new network by considering only the *myeloids*. The application of *Louvain* algorithm on this network gave 13 clusters of atoms. Since symmetries are also found with these clusters, we focus our analysis on clusters 1, 4, 5, 8, 9 and 10, which can be found in Appendix E.2. Similarly to the previous analysis, the matching error between these clusters and the cells can be visualized on Fig. 6 (where 11 cells were discarded for clarity purpose). By using the biological markers, cluster 1 can be associated with the *CD16* cell type, through the presence of $CD14_{0/1}$, $CSF1R_{1/1}$, $FCGR3A_{1/1}$, $LYZ_{0/1}$, $S100A4_{2/2}$ and $S100A9_{0/1}$. Cluster 4 can be associated with *T* cells, with $CD3E_{1/1}$, $CD8A_{1/1}$, $GZMH_{1/1}$, $GZMK_{1/1}$, $IL32_{1/1}$, $IL7R_{1/1}$ and $LEF1_{1/1}$. Cluster 5 can be associated with *cDC* cells, with $CD74_{1/1}$, $FCER1A_{1/1}$, $HLA\text{-}DQA1_{1/1}$ and $S100A4_{0/2}$. Cluster 9 relates to *B* cells, with the presence of $CD79A_{1/1}$, $MS4A1_{1/1}$, $PAX5_{1/1}$ and $TCL1A_{1/1}$. Lastly, cluster 10 can be association to *NK* cells, with $CCL5_{1/1}$, $CST7_{1/1}$, $GNLY_{1/1}$,

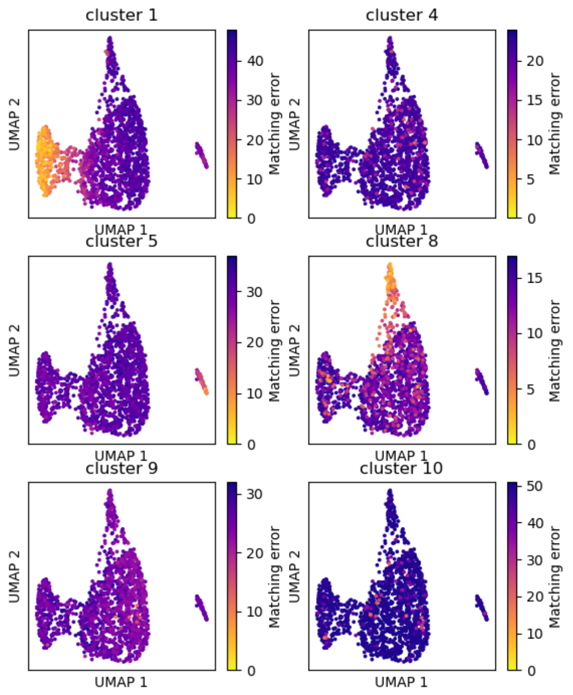

Figure 6: Visualization of the matching error of the atoms from each cluster on the previously selected cells.

GZMB$_{1/1}$, *IL2RB*$_{1/1}$, *KLRC1*$_{1/1}$, *KLRD1*$_{1/1}$, *NCAM1*$_{1/1}$, *NKG7*$_{1/1}$ and *PRF1*$_{1/1}$.

The matching error of the different clusters can also be visualized on all the cells of the dataset on Fig. 11 in Appendix E.3. The colors seem to confirm that the clusters 4, 9 and 10 do not represent *myeloids*. This indicates that both our algorithm and the processing previously performed with *Seurat* wrongly associated few cells with the *myeloids*. To verify this more precisely, we focused on cluster 10 and performed a sub selection of the cells from the *myeloids*, with the matching error. We performed (DE) analysis similarly to sub-section 3.3.3, by using these cells as the first group, and the remaining *myeloids* as the second group. The resulting table in Appendix E.4 seems to confirm that the corresponding cells are *NK* cells wrongly associated with *myeloids*.

Lastly, cluster 8 seems also interesting since it matches not only on *myeloids*, as seen in Fig. 11. This cluster is enriched for *PPBP*, a growth factor marker for platelets, but found to be associated with *monocyte* (*CD14*) migration and autocrine, receptor-desensitising chemokine ligand release [6, 23]. To confirm the identity of these cells, we selected the cells matching with cluster 8 from the complete dataset and performed (DE) analysis, with the the first group of cells being the cells that also belong to the *myeloids*, and the second group being the remaining cells. The resulting table in Appendix E.4 indeed indicates that these cells correspond to monocytes, which is also consistent with the identification shown in Fig. 1, and where *PPBP* can be a sign of inflammation. The other cells matching with cluster 8 might also be associated with inflammation.

5 CONCLUSION AND FUTURE WORKS

In conclusion, we have presented a method derived from inductive logic programming that intends to compute logical models describing combined gene interactions from noisy gene expressions data. Our method, extended from the *LFIT* framework, ensure robustness from the noise as well as interpretability of the gene relations, as it has been illustrated through an example of cell classification. We have applied our algorithm *LOLH* on a representative single cell dataset to compute correlation networks of the different genes. The structure of the graph, more especially the gene clusters, have been demonstrated to be informative about biological aspects of the cells. We have shown how *LOLH* can be used to identify cells of interest, and to compute refined models capturing more detailed characteristics (e.g. by focusing on one specific cell type). By clustering genes instead of cells, we have also been able to characterize cells that were previously wrongly identified using classical single cell analysis tools. Although the method has been applied on *scRNA-seq* data in this work, we expect it to be relevant on other type of single cell sequencing based dataset, when the discretization can be performed.

Through this application, gene relations have been identified within the same cells. However, it might be interesting to recover dynamical relations, by using the initial *LFIT* formulation on transitions data. While transitions are not initially available in *scRNA-seq* data, we consider a reconstruction of this information using graph based-methods. Alternatively, other types of data containing dynamical information may be used in that purpose, such as *RNA-velocity* based approaches [2, 13]. In order to compare this approach to existing tools for the inference of gene networks from single cell, we also consider using the benchmarking methodology presented in [19]. Lastly, we consider refining the optimization task performed by *LOLH*, in order to infer multiple rules from one classification instance.

ACKNOWLEDGMENTS

We would like to thank Dr Tony Ribeiro for his valuable comments on this paper and Imagine institute for providing and processing the dataset used in this work.

REFERENCES

[1] Sara Aibar, Carmen Bravo González-Blas, Thomas Moerman, Vân Anh Huynh-Thu, Hana Imrichova, Gert Hulselmans, Florian Rambow, Jean-Christophe Marine, Pierre Geurts, Jan Aerts, Joost van den Oord, Zeynep Kalender Atak, Jasper Wouters, and Stein Aerts. 2017. SCENIC: single-cell regulatory network inference and clustering. *Nature Methods* 14, 11 (Nov. 2017), 1083–1086. https://doi.org/10.1038/nmeth.4463
[2] Volker Bergen, Marius Lange, Stefan Peidli, F. Alexander Wolf, and Fabian J. Theis. 2020. Generalizing RNA velocity to transient cell states through dynamical modeling. *Nature Biotechnology* 38, 12 (Dec. 2020), 1408–1414. https://doi.org/10.1038/s41587-020-0591-3
[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (Oct. 2008), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008
[4] Shuonan Chen and Jessica C. Mar. 2018. Evaluating methods of inferring gene regulatory networks highlights their lack of performance for single cell gene

expression data. *BMC Bioinformatics* 19, 1 (Dec. 2018), 232. https://doi.org/10.1186/s12859-018-2217-z

[5] Andrew Cropper, Sebastijan Dumančić, and Stephen H. Muggleton. 2020. Turning 30: New Ideas in Inductive Logic Programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4833–4839. https://doi.org/10.24963/ijcai.2020/673 Survey track.

[6] A. Elmesmari, A. R. Fraser, C. Wood, D. Gilchrist, D. Vaughan, L. Stewart, C. McSharry, I. B. McInnes, and M. Kurowska-Stolarska. 2016. MicroRNA-155 regulates monocyte chemokine and chemokine receptor expression in Rheumatoid Arthritis. *Rheumatology (Oxford)* 55, 11 (Nov 2016), 2056–2065.

[7] Christoph Hafemeister and Rahul Satija. 2019. Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biology* 20, 1 (Dec. 2019). https://doi.org/10.1186/s13059-019-1874-1

[8] Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, William M. Mauck, Shiwei Zheng, Andrew Butler, Maddie J. Lee, Aaron J. Wilk, Charlotte Darby, Michael Zagar, Paul Hoffman, Marlon Stoeckius, Efthymia Papalexi, Eleni P. Mimitou, Jaison Jain, Avi Srivastava, Tim Stuart, Lamar B. Fleming, Bertrand Yeung, Angela J. Rogers, Juliana M. McElrath, Catherine A. Blish, Raphael Gottardo, Peter Smibert, and Rahul Satija. 2020. Integrated analysis of multimodal single-cell data. *bioRxiv* (2020). https://doi.org/10.1101/2020.10.12.335331

[9] Xinlin Hu, Yaohua Hu, Fanjie Wu, Ricky Wai Tak Leung, and Jing Qin. 2020. Integration of single-cell multi-omics for gene regulatory network inference. *Computational and Structural Biotechnology Journal* 18 (2020), 1925–1938. https://doi.org/10.1016/j.csbj.2020.06.033

[10] Katsumi Inoue, Tony Ribeiro, and Chiaki Sakama. 2014. Learning from interpretation transition. *Machine Learning* 94, 1 (Jan. 2014), 51–79. https://doi.org/10.1007/s10994-013-5353-8

[11] Yoonjee Kang, Denis Thieffry, and Laura Cantini. 2021. Evaluating the Reproducibility of Single-Cell Gene Regulatory Network Inference Algorithms. *Frontiers in Genetics* 12 (2021), 362. https://doi.org/10.3389/fgene.2021.617282

[12] S.A. Kauffman. 1969. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology* 22, 3 (1969), 437–467. https://doi.org/10.1016/0022-5193(69)90015-0

[13] Gioele La Manno, Ruslan Soldatov, Hannah Hochgerner, Amit Zeisel, Viktor Petukhov, Maria E. Kastriti, Peter Lönnerberg, Alessandro Furlan, Jean Fan, Zehua Liu, David van Bruggen, Jimin Guo, Erik Sundström, Gonçalo Castelo-Branco, Igor Adameyko, Sten Linnarsson, and Peter V. Kharchenko. 2018. RNA velocity in single cells. *Nature* 560 (2018), 494–498. https://doi.org/10.1038/s41586-018-0414-6

[14] Chee Yee Lim, Huange Wang, Steven Woodhouse, Nir Piterman, Lorenz Wernisch, Jasmin Fisher, and Berthold Göttgens. 2016. BTR: training asynchronous Boolean models using single-cell expression data. *BMC Bioinformatics* 17, 1 (Dec. 2016). https://doi.org/10.1186/s12859-016-1235-y

[15] Malte D Luecken and Fabian J Theis. 2019. Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular Systems Biology* 15, 6 (June 2019). https://doi.org/10.15252/msb.20188746

[16] Hirotaka Matsumoto, Hisanori Kiryu, Chikara Furusawa, Minoru S H Ko, Shigeru B H Ko, Norio Gouda, Tetsutaro Hayashi, and Itoshi Nikaido. 2017. SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation. *Bioinformatics* 33, 15 (Aug. 2017), 2314–2321. https://doi.org/10.1093/bioinformatics/btx194

[17] Leland McInnes, John Healy, and James Melville. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints 1802.03426* (2018). arXiv:1802.03426 [stat.ML]

[18] Daniele Mercatelli, Laura Scalambra, Luca Triboli, Forest Ray, and Federico M. Giorgi. 2020. Gene regulatory network inference resources: A practical overview. *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms* 1863, 6 (2020), 194430. https://doi.org/10.1016/j.bbagrm.2019.194430 Transcriptional Profiles and Regulatory Gene Networks.

[19] Aditya Pratapa, Amogh P. Jalihal, Jeffrey N. Law, Aditya Bharadwaj, and T. M. Murali. 2020. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nat Methods* 17 (June 2020), 147–154. https://doi.org/10.1038/s41592-019-0690-6

[20] Tony Ribeiro, Maxime Folschette, Morgan Magnin, Olivier Roux, and Katsumi Inoue. 2018. Learning Dynamics with Synchronous, Asynchronous and General Semantics. In *Inductive Logic Programming*, Fabrizio Riguzzi, Elena Bellodi, and Riccardo Zese (Eds.). Vol. 11105. Springer International Publishing, Cham, 118–140. https://doi.org/10.1007/978-3-319-99960-9_8

[21] Tony Ribeiro, Maxime Folschette, Laurent Trilling, Nicolas Glade, Katsumi Inoue, Morgan Magnin, and Olivier Roux. 2020. Les enjeux de l'inférence de modèles dynamiques des systèmes biologiques à partir de séries temporelles. (May 2020). https://hal.archives-ouvertes.fr/hal-02634235 to appear.

[22] Alan Scheinine, Wieslawa I. Mentzen, Giorgio Fotia, Enrico Pieroni, Fabio Maggio, Gianmaria Mancosu, and Alberto De La Fuente. 2009. Inferring Gene Networks: Dream or Nightmare?: Part 2: Challenges 4 and 5. *Annals of the New York Academy of Sciences* 1158, 1 (March 2009), 287–301. https://doi.org/10.1111/j.1749-6632.2008.04100.x

[23] F. Schwartzkopff, F. Petersen, T. A. Grimm, and E. Brandt. 2012. CXC chemokine ligand 4 (CXCL4) down-regulates CC chemokine receptor expression on human monocytes. *Innate Immun* 18, 1 (Feb 2012), 124–139.

[24] René Thomas. 1973. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology* 42, 3 (1973), 563–585. https://doi.org/10.1016/0022-5193(73)90247-6

[25] Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. 2014. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature Biotechnology* 32, 4 (April 2014), 381–386. https://doi.org/10.1038/nbt.2859

[26] Steven Woodhouse, Nir Piterman, Christoph M. Wintersteiger, Berthold Göttgens, and Jasmin Fisher. 2018. SCNS: a graphical tool for reconstructing executable regulatory networks from single-cell genomic data. *BMC Systems Biology* 12, 1 (Dec. 2018). https://doi.org/10.1186/s12918-018-0581-y

A TOY EXAMPLE WITH *LOLH* (FROM AN ARTIFICIAL DATASET)

We illustrate here the application of our algorithm *LOLH* on an artificial example over 4 bi-valued discrete variables. In this example, the dataset consists of 4 positive and 4 negative examples, and *LOLH* is used to create a logic rule able to recognise the positive examples from the negative ones.

A.1 Artificial dataset

This artificial example contains observations over a set of 4 discrete variables: $\mathcal{V} = \{a, b, c, d\}$. 8 observations (e.g. cells) are given, labeled from s_0 to s_7, and are separated into positive and negative examples as follows: $\mathcal{S}^+ = \{s_0, s_1, s_2, s_3\}$ and $\mathcal{S}^- = \{s_4, s_5, s_6, s_7\}$ The next tables contain the values of the variables in each observation. Rows correspond to the observations and columns correspond to the variables.

Table 1: Positive examples

	a	b	c	d
s_0	1	0	0	0
s_1	1	0	1	0
s_2	1	0	1	1
s_3	0	0	1	0

Table 2: Negative examples

	a	b	c	d
s_4	1	1	1	1
s_5	0	1	0	1
s_6	1	1	0	1
s_7	1	0	0	1

It can be noticed that the variables b and d are mostly equal to 0 in the positive examples and they are mostly equal to 1 in the negative examples. Additionally, c has the inverse behaviour. Thus, the atoms associated to b, c and d are expected to have the best scores. On the contrary, the variable a has no dominant value between the positive and negative examples, thus a should not be used in a logic rule.

Table 3: Atom positive errors, negative errors and scores

	$error_+$	$error_-$	$score$
$a_{0/1}$	3	3	0
$a_{1/1}$	1	1	0
$b_{0/1}$	0	3	0.75
$b_{1/1}$	4	1	-0.75
$c_{0/1}$	3	1	-0.5
$c_{1/1}$	1	3	0.5
$d_{0/1}$	1	4	0.75
$d_{1/1}$	3	0	-0.75

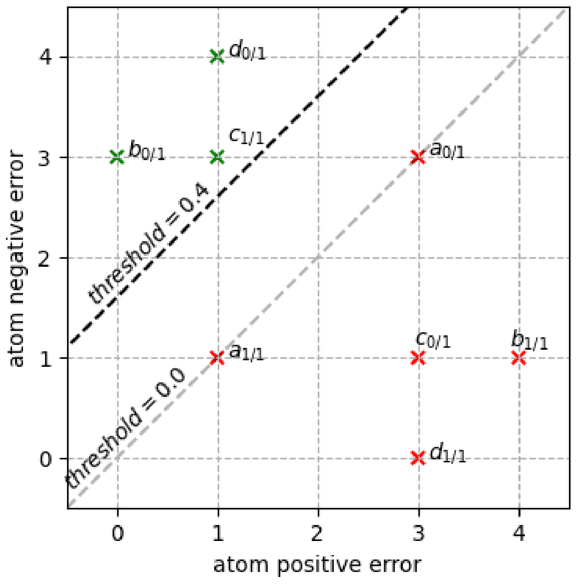

Figure 7: Positive and negative errors of all atoms.

A.2 Computation of the atom scores

From this dataset, one can define one logical atom for each discrete value of each variable ($4 * 2 = 8$ atoms in total), in order to form the logic rules. This leads to the following set of atoms: $\mathcal{A} = \{a_{0/1}, a_{1/1}, b_{0/1}, b_{1/1}, c_{0/1}, c_{1/1}, d_{0/1}, d_{1/1}\}$

For each atom, it is possible to compute its positive and negative errors according to the definitions introduced in part 3.2.2. For instance, the score of atom $b_{0/1}$ is computed as follows:

- $error_+(b_{0/1}) = |\{s_i, \mathbf{i} \in \{\mathbf{0}, .., \mathbf{3}\} \mid s_i(b) \neq b_{0/1}\}| = 0$
- $error_-(b_{0/1}) = |\{s_i, i \in \{\mathbf{4}, .., \mathbf{7}\} \mid s_i(b) \neq b_{0/1}\}| = 3$
- $score(b_{0/1}) = \frac{error_-(b_{0/1})}{|\mathcal{S}^-|} - \frac{error_+(b_{0/1})}{|\mathcal{S}^+|} = 3/4 - 0/4 = 0.75$

Table 3 gives the positive errors, negative errors and scores of all atoms from the artificial dataset. Fig. 7 shows the errors as 2d coordinates. In this plot, it is possible to visualize the scores as diagonal lines. Atoms with the best scores (i.e. closer to 1) are located at the top left part of the plot.

A.3 Induction of *LOLH* rule

In this example, the threshold 0.4 has been chosen to select the atoms. In Fig.7, the selected atoms (in green) are the atoms located above line corresponding to the score threshold: $b_{0/1}$, $d_{0/1}$ and $c_{1/1}$. Thus the corresponding *LOLH* rule is: $concl. \leftarrow b_{0/1}, d_{0/1}, c_{1/1}$
It can be noticed that the atoms $b_{0/1}$ and $d_{0/1}$ have the same score, thus they are located on the same diagonal line. The atom $c_{1/1}$ is slightly worst, but it is still selected according to the threshold. As the variables have 2 discrete values each, the errors are symmetric. As a result, the atoms $d_{1/1}$, $b_{1/1}$ and $c_{0/1}$ can be used to form a rule that matches on the negative examples. Lastly, it can be noticed in the dataset that the variable a does not behave differently between the positive and the negative example, i.e. the errors of the atoms are the same in $\mathcal{S}^+$ and $\mathcal{S}^-$. This is also visible in the plot as $a_{0/1}$ and $a_{1/1}$ are located on the same line corresponding to the score 0.

B TECHNICAL INFORMATION ABOUT THE DATASET

The dataset used in this study is a single cell RNA-sequencing matrix composed of 9198 peripheral blood mononuclear cells (pbmc: *T*, *B*, *NK* and *myeloids*) extracted from two healthy donors. The dataset has been provided and processed by Imagine Institute. Cells labeled as C26 come from a 30 years old female and cells labeled as C27 come from a 53 years old male. Cells have been isolated from blood using ficoll. Samples were sequenced using standard 3' v3 chemistry protocols by 10x genomics. Cellranger v4.0.0 was used for the processing, and reads were aligned to the ensembl GRCg38 human genome (GRCg38_r98-ensembl_Sept2019). QC metrics were calculated on the count matrix generated by cellranger (filtered_feature_bc_matrix). Cells with less than 3 genes per cells, less than 500 reads per cell and more than 20% of mithocondrial genes were discarded.

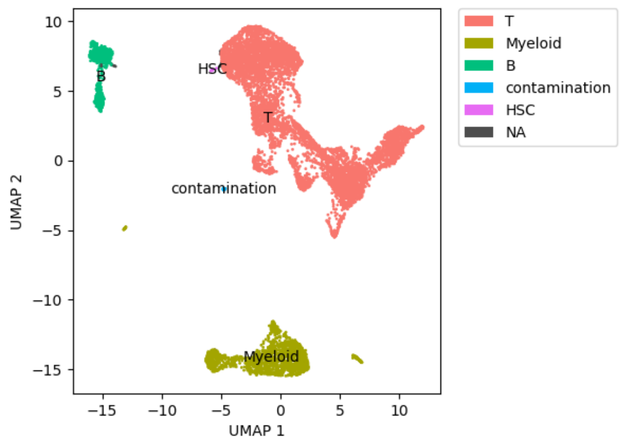

Figure 8: Visualization of cells through the *UMAP* coordinates with the macro cell-types pre-identified in the data.

The processing steps were performed with the R package Seurat[5], including sample integration, data normalisation and scaling, dimensional reduction, and clustering. *SCTransform* method [7] was adopted for the normalisation and scaling steps. The clustered cells were manually annotated using known cell type markers (Fig. 1). Fig. 8 shows the general cell types identified from the data on the UMAP representation.

C KNOWN GENE MARKERS

Table 4: List of genes expressed for different type of cells.

genes	B cells	NK	T cells	CD4	CD8
BLK	✓				
CCL5		✓			✓
CCR6	✓				
CD180	✓				
CD19	✓				
CD37	✓				
CD3D			✓	✓	✓
CD3E			✓	✓	✓
CD40LG				✓	
CD72	✓				
CD74	✓				
CD79A	✓				
CD79B	✓				
CD80	✓				
CD8A					✓
CD8B					✓
CD96		✓	✓	✓	✓
CST7		✓			✓
CTLA4				✓	
CXCR5	✓				
FASLG		✓			
FCER1G		✓			
FCGR3A		✓			
FCRL2	✓				
GNLY		✓			
GZMB		✓		✓	✓
GZMH		✓		✓	✓
HLA-DMA	✓				
HLA-DQA1	✓				
HTR3A	✓				
IL2RB		✓			
IL32			✓	✓	✓
IL6R				✓	
IL7R			✓	✓	✓
ITM2A			✓	✓	✓
KIR2DL3		✓			✓
KLRC1		✓			
KLRD1		✓			✓
KLRG1					✓
MS4A1	✓				
NCAM1		✓			
NKG7		✓			
PAX5	✓				
PIKFYVE	✓				
PNOC	✓				
PRF1		✓			
SPIB	✓				
STAP1	✓				
TBX21		✓		✓	✓
TCL1A	✓				
TCL1B	✓				
TLR7	✓				
TNFRSF13B	✓				
VPREB3	✓				
ZBTB16		✓			

Table 4: List of genes expressed for different type of cells (continued).

Table 5: List of genes expressed in myeloids cells.

genes	CD14	CD16	cDC
AIF1			✓
CCL2	✓		
CD14	✓		
CD16		✓	
CD74			✓
CD83			✓
CSF1R		✓	
FCER1A			✓
FCER1G			✓
FCGR3A		✓	
FCN1	✓		✓
FLT3			✓
GPR183			✓
HLA-DMA			✓
HLA-DQA1			✓
IL1R2			✓
ITGAX			✓
LGALS2	✓		✓
LGALS3			✓
LST1	✓		✓
LYZ	✓		
S100A4			✓
S100A9	✓		
TYROBP			✓

[5]https://satijalab.org/seurat/

D SUPPLEMENTARY MATERIALS ON NK CELL CLASSIFICATION

D.1 Visualizations of the $\mathcal{P}_{LOLH}$ rule on the *NK* cells

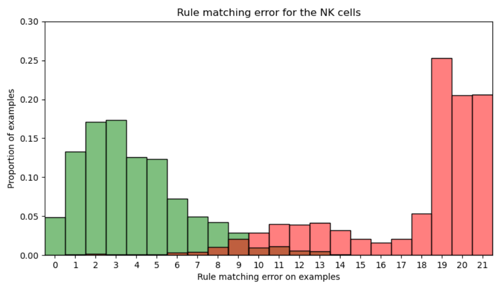

Figure 9: Rule matching error histogram on positive examples (in green) and negative examples (in red) for $\mathcal{P}_{LOLH}$ rule.

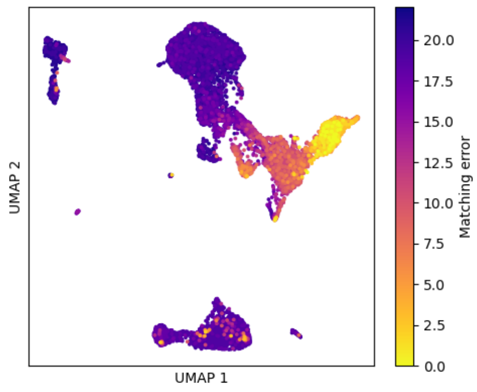

Figure 10: Matching error between the cells and $\mathcal{P}_{LOLH}$ rule visualized as colors on the *UMAP* representation. Brighter cells match the most with the rule.

D.2 (DE) analysis on the NK cells

Table 6: Result of (DE) analysis performed on the *NK* cells versus all the other cells, sorted by avg_log2FC.

gene	p_val	avg_log2FC	p_val_adj
GNLY	0	4.48	0
GZMB	0	3.18	0
KLRF1	0	2.75	0
SPON2	0	2.54	0
PRF1	0	2.53	0
CTSW	0	2.48	0

Table 6: Result of (DE) analysis performed on the *NK* cells versus all the other cells, sorted by avg_log2FC (continued).

gene	p_val	avg_log2FC	p_val_adj
HOPX	0	2.48	0
PTGDS	$1.33e-155$	2.43	$1.99e-151$
IL2RB	0	2.38	0
CLIC3	0	2.37	0
KLRB1	0	2.36	0
KLRD1	0	2.27	0
CMC1	0	2.25	0
NKG7	0	2.11	0
CD7	0	2.09	0
CST7	0	2.06	0
TRDC	0	1.94	0
FGFBP2	$1.38e-294$	1.92	$2.06e-290$
CD247	0	1.91	0
GZMA	0	1.78	0
MAP3K8	$2.46e-252$	1.72	$3.68e-248$
KLRC1	0	1.69	0
CD63	0	1.67	0
IFITM2	$9.92e-273$	1.62	$1.48e-268$
AREG	$1.71e-279$	1.57	$2.56e-275$
RUNX3	$8.1e-256$	1.54	$1.21e-251$
EFHD2	$7.93e-258$	1.53	$1.19e-253$
FCGR3A	0	1.52	0
GZMM	$3.77e-241$	1.51	$5.63e-237$
METRNL	$5.31e-290$	1.51	$7.94e-286$

E SUPPLEMENTARY MATERIALS FOR THE COEXPRESSION ANALYSIS

E.1 Atom clusters from the first network

- Cluster 0: (275 atoms)

$HES4_{0/1}$, $MS4A7_{0/1}$, $CDKN1C_{0/1}$, $IFITM3_{0/1}$, $SERPINA1_{0/1}$, $CFD_{0/1}$, $SIGLEC10_{0/1}$, $PSAP_{0/1}$, $SAT1_{0/1}$, $AIF1_{0/1}$, $SPI1_{0/1}$, $CTSS_{0/1}$, $FTL_{0/1}$, $CD68_{0/1}$, $LST1_{0/1}$, $COTL1_{0/1}$, $CST3_{0/1}$, $LYN_{0/1}$, $CSF1R_{0/1}$, $ASAH1_{0/1}$, $WARS_{0/1}$, $TYMP_{0/1}$, $VSIR_{0/1}$, $LILRB2_{0/1}$, $TNFRSF1B_{0/1}$, $NPC2_{0/1}$, $FGL2_{0/1}$, $FCER1G_{0/1}$, $BRI3_{0/1}$, $PILRA_{0/1}$, $TKT_{0/1}$, $PECAM1_{0/1}$, $TCF7L2_{0/1}$, $TYROBP_{0/1}$, $RHOC_{0/1}$, $LRRC25_{0/1}$, $CYBB_{0/1}$, $IER5_{0/1}$, $STXBP2_{0/1}$, $CXCL16_{0/1}$, $LGALS1_{0/1}$, $CEBPB_{0/1}$, $FGR_{0/1}$, $BID_{0/1}$, $RNF130_{0/1}$, $CTSZ_{0/1}$, $LILRA5_{0/1}$, $CLEC7A_{0/1}$, $TIMP1_{0/1}$, $CUX1_{0/1}$, $S100A11_{0/1}$, $NAMPT_{0/1}$, $CLEC12A_{0/1}$, $HSBP1_{0/1}$, $FKBP1A_{0/1}$, $FCN1_{0/1}$, $TCIRG1_{0/1}$, $DUSP1_{0/1}$, $LGALS9_{0/1}$, $NINJ1_{0/1}$, $PYCARD_{0/1}$, $CTSL_{0/1}$, $ANXA2_{0/1}$, $NCF2_{0/1}$, $SH3BP2_{0/1}$, $SEC14L1_{0/1}$, $ANXA5_{0/1}$, $CAMK1_{0/1}$, $LYST_{0/1}$, $PRELID1_{0/1}$, $MARCKS_{0/1}$, $CTSB_{0/1}$, $THEMIS2_{0/1}$, $ZFAND5_{0/1}$, $ZNF703_{0/1}$, $SNX10_{0/1}$, $PLEKHO1_{0/1}$, $CASP1_{0/1}$, $SOD2_{0/1}$, $CFP_{0/1}$, $HMOX1_{0/1}$, $GSTP1_{0/1}$, $C5AR1_{0/1}$, $GRINA_{0/1}$, $ZYX_{0/1}$, $LGALS3_{0/1}$, $GDI2_{0/1}$, $FCGRT_{0/1}$, $EHBP1L1_{0/1}$, $HCK_{0/1}$, $ARPC1B_{0/1}$, $PGK1_{0/1}$, $EVI2B_{0/1}$, $APOBEC3A_{0/1}$, $LTA4H_{0/1}$, $ARRB2_{0/1}$, $NOTCH2_{0/1}$, $MGAT1_{0/1}$, $SKAP2_{0/1}$, $GPBAR1_{0/1}$, $ARPC5_{0/1}$, $MIDN_{0/1}$, $SLC7A7_{0/1}$, $C1orf162_{0/1}$, $STX11_{0/1}$, $CD4_{0/1}$, $ZEB2_{0/1}$, $AP1S2_{0/1}$, $RILPL2_{0/1}$, $ADA2_{0/1}$, $IFI30_{0/1}$, $CAP1_{0/1}$, $HK3_{0/1}$, $BATF3_{0/1}$, $RRAS_{0/1}$, $PIK3AP1_{0/1}$, $MYOF_{0/1}$, $CHP1_{0/1}$, $FLNA_{0/1}$, $RIN3_{0/1}$, $NCOA4_{0/1}$, $TSPO_{0/1}$, $CPPED1_{0/1}$, $PPT1_{0/1}$, $LAP3_{0/1}$, $C19orf38_{0/1}$, *C1QA*$_{0/1}$, *LMO2*$_{0/1}$, *LILRB1*$_{0/1}$, *YBX3*$_{0/1}$, *H2AFY*$_{0/1}$, *TPP1*$_{0/1}$, *MAFB*$_{0/1}$, *SH3BGRL*$_{0/1}$, *NUMB*$_{0/1}$, *MS4A4A*$_{0/1}$, *OAS1*$_{0/1}$, *S100A4*$_{0/2}$, *ATP6V0D1*$_{0/1}$, *NR4A1*$_{0/1}$, *JAML*$_{0/1}$, *TMEM176B*$_{0/1}$, *CKB*$_{0/1}$, *QKI*$_{0/1}$, *RGS18*$_{0/1}$, *RTN3*$_{0/1}$, *AGTRAP*$_{0/1}$, *NUP214*$_{0/1}$, *ACTR2*$_{0/1}$, *APLP2*$_{0/1}$, *ALDH3B1*$_{0/1}$, *RAC1*$_{0/1}$, *DPYSL2*$_{0/1}$, *IRAK3*$_{0/1}$, *S100A9*$_{0/1}$, *CPVL*$_{0/1}$, *TNFSF10*$_{0/1}$, *IFNGR1*$_{0/1}$, *GABARAP*$_{0/1}$, *MYO1F*$_{0/1}$, *CSTA*$_{0/1}$, *SCO2*$_{0/1}$, *MPEG1*$_{0/1}$, *MNDA*$_{0/1}$, *GRN*$_{0/1}$, *LYZ*$_{0/1}$, *S100A8*$_{0/1}$, *NCF1*$_{0/1}$, *FOS*$_{0/1}$, *KLF10*$_{0/1}$, *CTSD*$_{0/1}$, *RGS2*$_{0/1}$, *LRRK2*$_{0/1}$, *TNFSF13B*$_{0/1}$, *ZFP36*$_{0/1}$, *MS4A6A*$_{0/1}$, *VCAN*$_{0/1}$, *MCL1*$_{0/1}$, *CEBPD*$_{0/1}$, *CSF3R*$_{0/1}$, *KCTD12*$_{0/1}$, *PTPRE*$_{0/1}$, *AHNAK*$_{0/1}$, *IER2*$_{0/1}$, *CD14*$_{0/1}$, *S100A12*$_{0/1}$, *CD36*$_{0/1}$, *EGR1*$_{0/1}$, *CYP1B1*$_{0/1}$, *SDCBP*$_{0/1}$, *MEGF9*$_{0/1}$, *PGD*$_{0/1}$, *HIF1A*$_{0/1}$, *CD44*$_{0/1}$, *NFKBIA*$_{0/1}$, *SERPINB1*$_{0/1}$, *IRS2*$_{0/1}$, *FPR1*$_{0/1}$, *OGFRL1*$_{0/1}$, *DPYD*$_{0/1}$, *PICALM*$_{0/1}$, *PLBD1*$_{0/1}$, *RAB31*$_{0/1}$, *ANXA1*$_{0/1}$, *LAMP2*$_{0/1}$, *DMXL2*$_{0/1}$, *CMIP*$_{0/1}$, *THBS1*$_{0/1}$, *CD302*$_{0/1}$, *G0S2*$_{0/1}$, *IQGAP1*$_{0/1}$, *VIM*$_{0/1}$, *GCA*$_{0/1}$, *GASK1B*$_{0/1}$, *FOSB*$_{0/1}$, *ALDH2*$_{0/1}$, *CRISPLD2*$_{0/1}$, *ANPEP*$_{0/1}$, *SCPEP1*$_{0/1}$, *FCAR*$_{0/1}$, *CD93*$_{0/1}$, *MYADM*$_{0/1}$, *SLC11A1*$_{0/1}$, *TGFBI*$_{0/1}$, *PLXDC2*$_{0/1}$, *ALDH1A1*$_{0/1}$, *CRTAP*$_{0/1}$, *CD163*$_{0/1}$, *DUSP6*$_{0/1}$, *SGK1*$_{0/1}$, *TREM1*$_{0/1}$, *MARCH1*$_{0/1}$, *AOAH*$_{0/1}$, *BLVRB*$_{0/1}$, *GLUL*$_{0/1}$, *TNFAIP2*$_{0/1}$, *TFEC*$_{0/1}$, *CAPG*$_{0/1}$, *KLF4*$_{0/1}$, *SIRPA*$_{0/1}$, *FOSL2*$_{0/1}$, *SAMHD1*$_{0/1}$, *IFNGR2*$_{0/1}$, *F13A1*$_{0/1}$, *TLR4*$_{0/1}$, *FCGR1A*$_{0/1}$, *CSF2RA*$_{0/1}$, *LGALS2*$_{0/1}$, *TLR8*$_{0/1}$, *CD300E*$_{0/1}$, *CTSH*$_{0/1}$, *FCGR2A*$_{0/1}$, *SRGN*$_{0/1}$, *TMEM176A*$_{0/1}$, *HLA-DRB5*$_{0/1}$, *PLSCR1*$_{0/1}$, *GAS7*$_{0/1}$, *CCDC88A*$_{0/1}$, *JDP2*$_{0/1}$, *HBEGF*$_{0/1}$, *ACTB*$_{1/3}$, *CD3E*$_{1/2}$, *S100A6*$_{1/2}$, *IL32*$_{1/2}$, *LTB*$_{1/2}$, *EEF1B2*$_{2/3}$, *TRAC*$_{1/1}$, *IL7R*$_{1/1}$, *CD2*$_{1/1}$, *LAT*$_{1/1}$, *C12orf57*$_{1/1}$, *CD6*$_{1/1}$, *PRKCH*$_{1/1}$, *BCL2*$_{1/1}$, *CD27*$_{1/1}$

- Cluster 1 (64 atoms)

FCGR3A$_{0/1}$, *METRNL*$_{0/1}$, *EFHD2*$_{0/1}$, *ITGAL*$_{0/1}$, *CLIC1*$_{0/1}$, *CD99*$_{0/1}$, *ARL4C*$_{0/1}$, *GNLY*$_{0/1}$, *KLRB1*$_{0/1}$, *KLRF1*$_{0/1}$, *SYTL3*$_{0/1}$, *IL2RB*$_{0/1}$, *CTSW*$_{0/1}$, *GZMB*$_{0/1}$, *CST7*$_{0/1}$, *NKG7*$_{0/1}$, *GZMA*$_{0/1}$, *CCL5*$_{0/1}$, *KLRD1*$_{0/1}$, *GZMM*$_{0/1}$, *PRF1*$_{0/1}$, *FGFBP2*$_{0/1}$, *ABHD17A*$_{0/1}$, *MATK*$_{0/1}$, *SPON2*$_{0/1}$, *GZMH*$_{0/1}$, *APMAP*$_{0/1}$, *CLIC3*$_{0/1}$, *PYHIN1*$_{0/1}$, *CHST12*$_{0/1}$, *DUSP2*$_{0/1}$, *RUNX3*$_{0/1}$, *LITAF*$_{0/1}$, *HOPX*$_{0/1}$, *CD81*$_{0/1}$, *C12orf75*$_{0/1}$, *ACTN4*$_{0/1}$, *PLAAT4*$_{0/1}$, *SYNE2*$_{0/1}$, *ID2*$_{0/1}$, *MAP3K8*$_{0/1}$, *SYNE1*$_{0/1}$, *CD8A*$_{0/1}$, *APOBEC3G*$_{0/1}$, *CD8B*$_{0/1}$, *IL2RG*$_{0/1}$, *SAMD3*$_{0/1}$, *PPP2R5C*$_{0/1}$, *LYAR*$_{0/1}$, *PPIB*$_{0/1}$, *TPST2*$_{0/1}$, *CCL4*$_{0/1}$, *CMC1*$_{0/1}$, *TRDC*$_{0/1}$, *SH2D1B*$_{0/1}$, *B3GNT7*$_{0/1}$, *PTPN22*$_{0/1}$, *CLEC2B*$_{0/1}$, *TRGC1*$_{0/1}$, *EEF1A1*$_{1/2}$, *B2M*$_{1/3}$, *HLA-A*$_{1/3}$, *HLA-C*$_{1/3}$, *TPT1*$_{2/3}$

- Cluster 2 (67 atoms)

HLA-DRA$_{0/1}$, *POU2F2*$_{0/1}$, *HLA-DPA1*$_{0/1}$, *HLA-DRB1*$_{0/1}$, *HLA-DQB1*$_{0/1}$, *HLA-DPB1*$_{0/1}$, *SNX2*$_{0/1}$, *HLA-DQA1*$_{0/1}$, *HLA-DMB*$_{0/1}$, *HLA-DMA*$_{0/1}$, *CD79A*$_{0/1}$, *MS4A1*$_{0/1}$, *CD74*$_{0/1}$, *TNFRSF13C*$_{0/1}$, *IGHM*$_{0/1}$, *IGHD*$_{0/1}$, *BANK1*$_{0/1}$, *FCRL1*$_{0/1}$, *CD22*$_{0/1}$, *CD79B*$_{0/1}$, *PAX5*$_{0/1}$, *NIBAN3*$_{0/1}$, *TCF4*$_{0/1}$, *ADAM28*$_{0/1}$, *IGKC*$_{0/1}$, *RALGPS2*$_{0/1}$, *GNG7*$_{0/1}$, *BCL11A*$_{0/1}$, *BLK*$_{0/1}$, *MEF2C*$_{0/1}$, *ARHGAP24*$_{0/1}$, *FCER2*$_{0/1}$, *EBF1*$_{0/1}$, *COBLL1*$_{0/1}$, *SPIB*$_{0/1}$, *HLA-DOB*$_{0/1}$, *IGLC2*$_{0/1}$, *AFF3*$_{0/1}$, *TCL1A*$_{0/1}$, *BCL7A*$_{0/1}$, *VPREB3*$_{0/1}$, *TSPAN13*$_{0/1}$, *PKIG*$_{0/1}$, *JCHAIN*$_{0/1}$, *CD40*$_{0/1}$, *BLNK*$_{0/1}$, *FCRLA*$_{0/1}$, *POU2AF1*$_{0/1}$, *SNX22*$_{0/1}$, *IGLC3*$_{0/1}$, *HVCN1*$_{0/1}$, *CXCR4*$_{0/1}$, *PLPP5*$_{0/1}$, *IL4R*$_{0/1}$, *S100A4*$_{1/2}$, *CD37*$_{1/2}$, *MGAT4A*$_{1/1}$, *TCF7*$_{1/1}$, *LEF1*$_{1/1}$, *CCR7*$_{1/1}$, *MAL*$_{1/1}$, *TRABD2A*$_{1/1}$, *PIK3IP1*$_{1/1}$, *NOSIP*$_{1/1}$, *CAMK4*$_{1/1}$, *RCAN3*$_{1/1}$, *LEPROTL1*$_{1/1}$

- Cluster 3 (76 atoms)

TCF7$_{0/1}$, *LEF1*$_{0/1}$, *MAL*$_{0/1}$, *CCR7*$_{0/1}$, *NOSIP*$_{0/1}$, *CAMK4*$_{0/1}$, *RCAN3*$_{0/1}$, *TRABD2A*$_{0/1}$, *TPT1*$_{1/3}$, *EEF1B2*$_{1/3}$, *B2M*$_{2/3}$, *HLA-C*$_{2/3}$, *HLA-A*$_{2/3}$, *CD99*$_{1/1}$, *ARL4C*$_{1/1}$, *FCGR3A*$_{1/1}$, *METRNL*$_{1/1}$, *FLNA*$_{1/1}$, *EFHD2*$_{1/1}$, *ITGAL*$_{1/1}$, *CLIC1*$_{1/1}$, *GNLY*$_{1/1}$, *KLRB1*$_{1/1}$, , *KLRF1*$_{1/1}$, *SYTL3*$_{1/1}$, *IL2RB*$_{1/1}$, *CTSW*$_{1/1}$, *GZMB*$_{1/1}$, *CST7*$_{1/1}$, *NKG7*$_{1/1}$, *GZMA*$_{1/1}$, *CCL5*$_{1/1}$, *KLRD1*$_{1/1}$, *GZMM*$_{1/1}$, *PRF1*$_{1/1}$, *FGFBP2*$_{1/1}$, *ABHD17A*$_{1/1}$, *MATK*$_{1/1}$, *SPON2*$_{1/1}$, *GZMH*$_{1/1}$, *APMAP*$_{1/1}$, *CLIC3*$_{1/1}$, *PYHIN1*$_{1/1}$, *CHST12*$_{1/1}$, *DUSP2*$_{1/1}$, *RUNX3*$_{1/1}$, *LITAF*$_{1/1}$, *HOPX*$_{1/1}$, *CD81*$_{1/1}$, *C12orf75*$_{1/1}$, *PLAAT4*$_{1/1}$, *SYNE2*$_{1/1}$, *ID2*$_{1/1}$, *ACTN4*$_{1/1}$, *MAP3K8*$_{1/1}$, *SYNE1*$_{1/1}$, *CD8A*$_{1/1}$, *CD8B*$_{1/1}$, *IL2RG*$_{1/1}$, *SAMD3*$_{1/1}$, *APOBEC3G*$_{1/1}$, *PPIB*$_{1/1}$, *PPP2R5C*$_{1/1}$, *LYAR*$_{1/1}$, *TPST2*$_{1/1}$, *CCL4*$_{1/1}$, *CMC1*$_{1/1}$, *TRDC*$_{1/1}$, *SH2D1B*$_{1/1}$, *B3GNT7*$_{1/1}$, *PTPN22*$_{1/1}$, *CLEC2B*$_{1/1}$, *TRGC1*$_{1/1}$, *PFN1*$_{2/2}$, *ACTG1*$_{2/2}$, *CFL1*$_{2/2}$

- Cluster 4 (275 atoms)

TRAC$_{0/1}$, *IL7R*$_{0/1}$, *CD2*$_{0/1}$, *LAT*$_{0/1}$, *C12orf57*$_{0/1}$, *CD6*$_{0/1}$, *PRKCH*$_{0/1}$, *BCL2*$_{0/1}$, *CD27*$_{0/1}$, *CD3E*$_{0/2}$, *IL32*$_{0/2}$, *TLE5*$_{0/2}$, *LTB*$_{0/2}$, *ACTB*$_{2/3}$, *HES4*$_{1/1}$, *MS4A7*$_{1/1}$, *CDKN1C*$_{1/1}$, *IFITM3*$_{1/1}$, *SERPINA1*$_{1/1}$, *CFD*$_{1/1}$, *SIGLEC10*$_{1/1}$, *PSAP*$_{1/1}$, *FTH1*$_{2/2}$, *SAT1*$_{1/1}$, *AIF1*$_{1/1}$, *SPI1*$_{1/1}$, *CTSS*$_{1/1}$, *FTL*$_{1/1}$, *CD68*$_{1/1}$, *LST1*$_{1/1}$, *COTL1*$_{1/1}$, *CST3*$_{1/1}$, *LYN*$_{1/1}$, *CSF1R*$_{1/1}$, *ASAH1*$_{1/1}$, *WARS*$_{1/1}$, *TYMP*$_{1/1}$, *VSIR*$_{1/1}$, *LILRB2*$_{1/1}$, *TNFRSF1B*$_{1/1}$, *NPC2*$_{1/1}$, *FGL2*$_{1/1}$, *FCER1G*$_{1/1}$, *BRI3*$_{1/1}$, *PILRA*$_{1/1}$, *TKT*$_{1/1}$, *PECAM1*$_{1/1}$, *TCF7L2*$_{1/1}$, *TYROBP*$_{1/1}$, *RHOC*$_{1/1}$, *LRRC25*$_{1/1}$, *CYBB*$_{1/1}$, *IER5*$_{1/1}$, *STXBP2*$_{1/1}$, *CXCL16*$_{1/1}$, *LGALS1*$_{1/1}$, *CEBPB*$_{1/1}$, *FGR*$_{1/1}$, *BID*$_{1/1}$, *RNF130*$_{1/1}$, *CTSZ*$_{1/1}$, *LILRA5*$_{1/1}$, *CLEC7A*$_{1/1}$, *TIMP1*$_{1/1}$, *CUX1*$_{1/1}$, *S100A11*$_{1/1}$, *NAMPT*$_{1/1}$, *S100A4*$_{2/2}$, *CLEC12A*$_{1/1}$, *HSBP1*$_{1/1}$, *FKBP1A*$_{1/1}$, *FCN1*$_{1/1}$, *TCIRG1*$_{1/1}$, *DUSP1*$_{1/1}$, *LGALS9*$_{1/1}$, *NINJ1*$_{1/1}$, *PYCARD*$_{1/1}$, *CTSL*$_{1/1}$, *ANXA2*$_{1/1}$, *NCF2*$_{1/1}$, *SH3BP2*$_{1/1}$, *SEC14L1*$_{1/1}$, *S100A6*$_{2/2}$, *ANXA5*$_{1/1}$, *CAMK1*$_{1/1}$, *LYST*$_{1/1}$, *PRELID1*$_{1/1}$, *MARCKS*$_{1/1}$, *CTSB*$_{1/1}$, *THEMIS2*$_{1/1}$, *ZFAND5*$_{1/1}$, *ZNF703*$_{1/1}$, *SNX10*$_{1/1}$, *PLEKHO1*$_{1/1}$, *CASP1*$_{1/1}$, *SOD2*$_{1/1}$, *CFP*$_{1/1}$, *HMOX1*$_{1/1}$, *GSTP1*$_{1/1}$, *C5AR1*$_{1/1}$, *GRINA*$_{1/1}$, *ZYX*$_{1/1}$, *LGALS3*$_{1/1}$, *GDI2*$_{1/1}$, *FCGRT*$_{1/1}$, *EHBP1L1*$_{1/1}$, *HCK*$_{1/1}$, *PGK1*$_{1/1}$, *NOTCH2*$_{1/1}$, *APOBEC3A*$_{1/1}$, *LTA4H*$_{1/1}$, *ARRB2*$_{1/1}$, *EVI2B*$_{1/1}$, *MGAT1*$_{1/1}$, *SKAP2*$_{1/1}$, *GPBAR1*$_{1/1}$, *ARPC5*$_{1/1}$, *MIDN*$_{1/1}$, *SLC7A7*$_{1/1}$, *C1orf162*$_{1/1}$, *STX11*$_{1/1}$, *CD4*$_{1/1}$, *ZEB2*$_{1/1}$, *AP1S2*$_{1/1}$, *RILPL2*$_{1/1}$, *CTSC*$_{1/1}$, *ADA2*$_{1/1}$, *IFI30*$_{1/1}$, *CAP1*$_{1/1}$, *HK3*$_{1/1}$, *BATF3*$_{1/1}$, *RRAS*$_{1/1}$, *PIK3AP1*$_{1/1}$, *MYOF*$_{1/1}$, *CHP1*$_{1/1}$, *RIN3*$_{1/1}$, *NCOA4*$_{1/1}$, *TSPO*$_{1/1}$, *CPPED1*$_{1/1}$, *PPT1*$_{1/1}$, *LAP3*$_{1/1}$, *LMO2*$_{1/1}$, *C19orf38*$_{1/1}$, *LILRB1*$_{1/1}$, *C1QA*$_{1/1}$, *YBX3*$_{1/1}$, *H2AFY*$_{1/1}$, *TPP1*$_{1/1}$, *MAFB*$_{1/1}$, *SH3BGRL*$_{1/1}$, *NUMB*$_{1/1}$, *MS4A4A*$_{1/1}$, *OAS1*$_{1/1}$, *ATP6V0D1*$_{1/1}$, *NR4A1*$_{1/1}$, *JAML*$_{1/1}$, *TMEM176B*$_{1/1}$, *CKB*$_{1/1}$, *QKI*$_{1/1}$, *RGS18*$_{1/1}$, *RTN3*$_{1/1}$, *AGTRAP*$_{1/1}$, *NUP214*$_{1/1}$, *ACTR2*$_{1/1}$, *APLP2*$_{1/1}$, *ALDH3B1*$_{1/1}$, *RAC1*$_{1/1}$, *DPYSL2*$_{1/1}$, *IRAK3*$_{1/1}$, *S100A9*$_{1/1}$, *GABARAP*$_{1/1}$, *TNFSF10*$_{1/1}$, *IFNGR1*$_{1/1}$, *CPVL*$_{1/1}$, *MYO1F*$_{1/1}$, *CSTA*$_{1/1}$, *SCO2*$_{1/1}$, *MPEG1*$_{1/1}$, *MNDA*$_{1/1}$, *GRN*$_{1/1}$, *LYZ*$_{1/1}$, *S100A8*$_{1/1}$, *NCF1*$_{1/1}$, *FOS*$_{1/1}$, *KLF10*$_{1/1}$, *CTSD*$_{1/1}$, *RGS2*$_{1/1}$, *LRRK2*$_{1/1}$, *TNFSF13B*$_{1/1}$, *MS4A6A*$_{1/1}$, *ZFP36*$_{1/1}$, *VCAN*$_{1/1}$, *MCL1*$_{1/1}$, *CEBPD*$_{1/1}$, *CSF3R*$_{1/1}$, *KCTD12*$_{1/1}$, *PTPRE*$_{1/1}$, *IER2*$_{1/1}$, *CD14*$_{1/1}$, *S100A12*$_{1/1}$, *CD36*$_{1/1}$, *EGR1*$_{1/1}$, *CYP1B1*$_{1/1}$, *SDCBP*$_{1/1}$, *MEGF9*$_{1/1}$, *PGD*$_{1/1}$, *HIF1A*$_{1/1}$, *CD44*$_{1/1}$, *NFKBIA*$_{1/1}$, *SERPINB1*$_{1/1}$, *IRS2*$_{1/1}$, *FPR1*$_{1/1}$, *OGFRL1*$_{1/1}$, *DPYD*$_{1/1}$, *PICALM*$_{1/1}$, *PLBD1*$_{1/1}$, *RAB31*$_{1/1}$, *ANXA1*$_{1/1}$, *LAMP2*$_{1/1}$, *DMXL2*$_{1/1}$, *CMIP*$_{1/1}$, *G0S2*$_{1/1}$, *CD302*$_{1/1}$, *THBS1*$_{1/1}$, *IQGAP1*$_{1/1}$, *VIM*$_{1/1}$, *GCA*$_{1/1}$, *CRISPLD2*$_{1/1}$, *AHNAK*$_{1/1}$, *FOSB*$_{1/1}$, *GASK1B*$_{1/1}$, *ALDH2*$_{1/1}$, *ANPEP*$_{1/1}$, *SCPEP1*$_{1/1}$, *FCAR*$_{1/1}$, *CD93*$_{1/1}$, *MYADM*$_{1/1}$, *SLC11A1*$_{1/1}$, *TGFBI*$_{1/1}$, *PLXDC2*$_{1/1}$, *ALDH1A1*$_{1/1}$, *CRTAP*$_{1/1}$, *CD163*$_{1/1}$, *DUSP6*$_{1/1}$, *SGK1*$_{1/1}$, *TREM1*$_{1/1}$, *MARCH1*$_{1/1}$, *AOAH*$_{1/1}$, *BLVRB*$_{1/1}$, *GLUL*$_{1/1}$, *TNFAIP2*$_{1/1}$, *TFEC*$_{1/1}$, *CAPG*$_{1/1}$, *KLF4*$_{1/1}$, *SIRPA*$_{1/1}$, *SAMHD1*$_{1/1}$, *FOSL2*$_{1/1}$, *IFNGR2*$_{1/1}$, *F13A1*$_{1/1}$, *TLR4*$_{1/1}$, *CSF2RA*$_{1/1}$, *FCGR1A*$_{1/1}$, *LGALS2*$_{1/1}$, *TLR8*$_{1/1}$, *CD300E*$_{1/1}$, *CTSH*$_{1/1}$, *FCGR2A*$_{1/1}$, *SRGN*$_{1/1}$, *JDP2*$_{1/1}$, *HLA-DRB5*$_{1/1}$, *PLSCR1*$_{1/1}$, *GAS7*$_{1/1}$, *CCDC88A*$_{1/1}$, *TMEM176A*$_{1/1}$, *HBEGF*$_{1/1}$

- Cluster 5 (59 atoms)

MGAT4A$_{0/1}$, *PIK3IP1*$_{0/1}$, *LEPROTL1*$_{0/1}$, *HLA-DRA*$_{1/1}$, *POU2F2*$_{1/1}$, *HLA-DPA1*$_{1/1}$, *HLA-DRB1*$_{1/1}$, *HLA-DQB1*$_{1/1}$, *HLA-DPB1*$_{1/1}$, *SNX2*$_{1/1}$, *HLA-DQA1*$_{1/1}$, *HLA-DMB*$_{1/1}$, *HLA-DMA*$_{1/1}$, *CD79A*$_{1/1}$, *MS4A1*$_{1/1}$, *CD74*$_{1/1}$, *TNFRSF13C*$_{1/1}$, *IGHM*$_{1/1}$, *IGHD*$_{1/1}$, *BANK1*$_{1/1}$, *FCRL1*$_{1/1}$, *CD22*$_{1/1}$, *CD79B*$_{1/1}$, *PAX5*$_{1/1}$, *NIBAN3*$_{1/1}$, *TCF4*$_{1/1}$, *ADAM28*$_{1/1}$, *CD37*$_{2/2}$, *IGKC*$_{1/1}$, *EBF1*$_{1/1}$, *GNG7*$_{1/1}$, *BCL11A*$_{1/1}$, *BLK*$_{1/1}$, *MEF2C*$_{1/1}$, *ARHGAP24*$_{1/1}$, *FCER2*$_{1/1}$, *RALGPS2*$_{1/1}$, *COBLL1*$_{1/1}$, *SPIB*$_{1/1}$, *AFF3*$_{1/1}$, *IGLC2*$_{1/1}$, *HLA-DOB*$_{1/1}$, *TCL1A*$_{1/1}$, *BCL7A*$_{1/1}$, *VPREB3*$_{1/1}$, *TSPAN13*$_{1/1}$, *PKIG*$_{1/1}$, *JCHAIN*$_{1/1}$, *CD40*$_{1/1}$, *BLNK*$_{1/1}$, *FCRLA*$_{1/1}$, *POU2AF1*$_{1/1}$, *SNX22*$_{1/1}$, *IGLC3*$_{1/1}$, *HVCN1*$_{1/1}$, *CXCR4*$_{1/1}$, *PLPP5*$_{1/1}$, *BTG1*$_{2/2}$, *IL4R*$_{1/1}$

E.2 Atom clusters on the second (refined) network

- Cluster 1 (47 atoms)

FOS$_{0/1}$, *VCAN*$_{0/1}$, *S100A12*$_{0/1}$, *S100A8*$_{0/1}$, *MS4A6A*$_{0/1}$, *LYZ*$_{0/1}$, *CSF3R*$_{0/1}$, *CD14*$_{0/1}$, *SLC2A3*$_{0/1}$, *CD36*$_{0/1}$, *EGR1*$_{0/1}$, *S100A9*$_{0/1}$, *PILRA*$_{1/1}$, *HES4*$_{1/1}$, *FCGR3A*$_{1/1}$, *CDKN1C*$_{1/1}$, *RHOC*$_{1/1}$, *SIGLEC10*$_{1/1}$, *TCF7L2*$_{1/1}$, *CSF1R*$_{1/1}$, *IFITM2*$_{1/1}$, *MS4A7*$_{1/1}$, *CTSL*$_{1/1}$, *LRRC25*$_{1/1}$, *IFITM3*$_{1/1}$, *PAG1*$_{1/1}$, *DRAP1*$_{1/1}$, *LILRB2*$_{1/1}$, *SPN*$_{1/1}$, *HMOX1*$_{1/1}$, *PTP4A3*$_{1/1}$, *ZNF703*$_{1/1}$, *LY6E*$_{1/1}$, *RRAS*$_{1/1}$, *VMO1*$_{1/1}$, *PPM1N*$_{1/1}$, *C1QA*$_{1/1}$, *CKB*$_{1/1}$, *OAS1*$_{1/1}$, *NAP1L1*$_{1/1}$, *MS4A4A*$_{1/1}$, *UNC119*$_{1/1}$, *IL3RA*$_{1/1}$, *C1QC*$_{1/1}$, *ICAM4*$_{1/1}$, *C1QB*$_{1/1}$, *S100A4*$_{2/2}$

- Cluster 4 (26 atoms)

EMB$_{1/1}$, *ISG15*$_{1/1}$, *SLC38A1*$_{1/1}$, *TESPA1*$_{1/1}$, *TCF7*$_{1/1}$, *INPP4B*$_{1/1}$, *IL32*$_{1/1}$, *IL7R*$_{1/1}$, *CD3E*$_{1/1}$, *CD2*$_{1/1}$, *GZMK*$_{1/1}$, *C12orf75*$_{1/1}$, *TRAC*$_{1/1}$, *LEF1*$_{1/1}$, *GZMH*$_{1/1}$, *CD8A*$_{1/1}$, *ABLIM1*$_{1/1}$, *HOOK1*$_{1/1}$, *PLEKHG1*$_{1/1}$, *CFAP54*$_{1/1}$, *BIRC5*$_{1/1}$, *STX1B*$_{1/1}$, *TPX2*$_{1/1}$

- Cluster 5 (36 atoms)

WARS$_{0/1}$, *NCF1*$_{0/1}$, *IER2*$_{0/1}$, *MNDA*$_{0/1}$, *APOBEC3A*$_{0/1}$, *GBP4*$_{0/1}$, *CTSS*$_{0/1}$, *GBP2*$_{0/1}$, *FTL*$_{0/1}$, *CYBB*$_{0/1}$, *S100A4*$_{0/2}$, *BATF3*$_{1/1}$, *HLA-DQA1*$_{1/1}$, *S100A6*$_{1/1}$, *CD74*$_{1/1}$, *HLA-DRB5*$_{1/1}$, *CLIC2*$_{1/1}$, *FCER1A*$_{1/1}$, *ENHO*$_{1/1}$, *BEND5*$_{1/1}$, *AREG*$_{1/1}$, *NDRG2*$_{1/1}$, *CD1C*$_{1/1}$, *ENPP1*$_{1/1}$, *CLEC9A*$_{1/1}$, *GCSAM*$_{1/1}$, *CADM1*$_{1/1}$, *TACSTD2*$_{1/1}$, *CLNK*$_{1/1}$, *TMEM14A*$_{1/1}$, *UHRF1*$_{1/1}$, *CXCR3*$_{1/1}$, *DUSP4*$_{1/1}$, *BTLA*$_{1/1}$, *MCM2*$_{1/1}$, *TYMS*$_{1/1}$

- Cluster 8 (16 atoms)

CD163$_{1/1}$, *PPBP*$_{1/1}$, *NRGN*$_{1/1}$, *CAVIN2*$_{1/1}$, *CLU*$_{1/1}$, *PF4*$_{1/1}$, *HIST1H2AC*$_{1/1}$, *MPIG6B*$_{1/1}$, *SPARC*$_{1/1}$, *GNG11*$_{1/1}$, *TMEM40*$_{1/1}$, *GP9*$_{1/1}$, *TUBB1*$_{1/1}$, *PTCRA*$_{1/1}$, *TDRP*$_{1/1}$, *GATA2*$_{1/1}$

- Cluster 9 (31 atoms)

IFI6$_{0/1}$, *EPSTI1*$_{0/1}$, *PSME2*$_{0/1}$, *STAT1*$_{0/1}$, *TNFSF10*$_{0/1}$, *GBP1*$_{0/1}$, *MEGF9*$_{1/1}$, *SLC44A2*$_{1/1}$, *IKZF3*$_{1/1}$, *RHOH*$_{1/1}$, *LTB*$_{1/1}$, *IRS2*$_{1/1}$, *THBS1*$_{1/1}$, *CD22*$_{1/1}$, *FCRL5*$_{1/1}$, *IGKC*$_{1/1}$, *CD79A*$_{1/1}$, *MS4A1*$_{1/1}$, *IGHM*$_{1/1}$, *PAX5*$_{1/1}$, *BANK1*$_{1/1}$, *IGHG3*$_{1/1}$, *TTN*$_{1/1}$, *IGHG1*$_{1/1}$, *TCL1A*$_{1/1}$, *EBF1*$_{1/1}$, *FCRL1*$_{1/1}$, *IGHD*$_{1/1}$, *NIBAN3*$_{1/1}$, *PCDH9*$_{1/1}$, *CARMIL2*$_{1/1}$

- Cluster 10 (50 atoms)

SYTL1$_{1/1}$, *TNFRSF18*$_{1/1}$, *GZMA*$_{1/1}$, *PRKCH*$_{1/1}$, *IL2RB*$_{1/1}$, *CTSW*$_{1/1}$, *GNLY*$_{1/1}$, *SYNE2*$_{1/1}$, *STAT4*$_{1/1}$, *CST7*$_{1/1}$, *NKG7*$_{1/1}$, *SYNE1*$_{1/1}$, *NCAM1*$_{1/1}$, *AKR1C3*$_{1/1}$, *B3GNT7*$_{1/1}$, *CD247*$_{1/1}$, *GZMB*$_{1/1}$, *CLIC3*$_{1/1}$, *SPON2*$_{1/1}$, *GZMM*$_{1/1}$, *KLRB1*$_{1/1}$, *MATK*$_{1/1}$, *KLRD1*$_{1/1}$, *SKAP1*$_{1/1}$, *CCL5*$_{1/1}$, *KLRF1*$_{1/1}$, *FEZ1*$_{1/1}$, *HOPX*$_{1/1}$, *PRF1*$_{1/1}$, *CMC1*$_{1/1}$, *LYAR*$_{1/1}$, *XCL2*$_{1/1}$, *CCL4*$_{1/1}$, *TRGC1*$_{1/1}$, *FGFBP2*$_{1/1}$, *SH2D1A*$_{1/1}$, *PYHIN1*$_{1/1}$, *PTCH1*$_{1/1}$, *NCR1*$_{1/1}$, *ADGRG1*$_{1/1}$, *NCR3*$_{1/1}$, *PDGFD*$_{1/1}$, *RRM2*$_{1/1}$, *TOGARAM2*$_{1/1}$, *SOCS2*$_{1/1}$, *RNF165*$_{1/1}$, *KIR3DL1*$_{1/1}$, *KIR2DL1*$_{1/1}$, *KLRC1*$_{1/1}$, *SLA2*$_{1/1}$

E.3 Rule matching error of the sub network clusters on all the cells

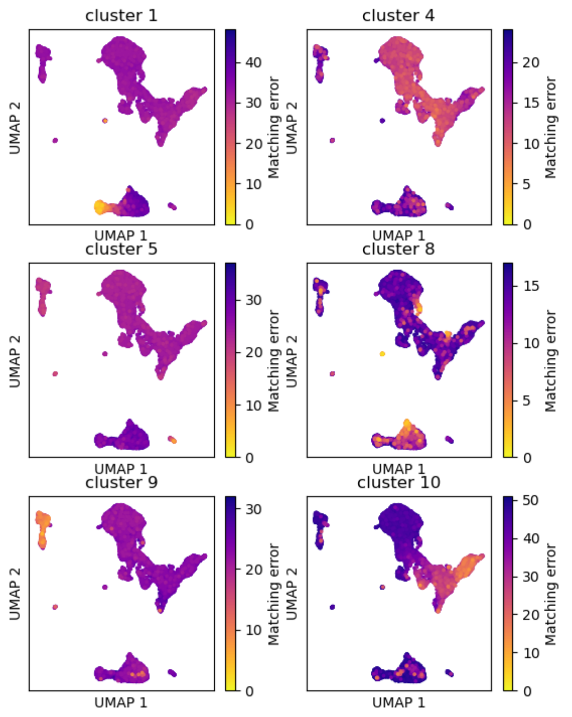

Figure 11: Rule matching error of the sub network clusters on all the cells.

E.4 (DE) analysis on the refined network

Table 7: Result of (DE) analysis performed on cells selected from cluster 10 in the refined network. The 20 first and 20 last genes sorted by the third column (avg_log2FC), indicating the most differentially expressed genes between the two groups.

gene	p_val	avg_log2FC	p_val_adj
GNLY	$1.78e-71$	4.84	$2.65e-67$
NKG7	$7.48e-52$	4.08	$1.12e-47$
CCL5	$1.17e-36$	3.21	$1.76e-32$
GZMA	$1.06e-141$	2.84	$1.59e-137$
CST7	$3.62e-120$	2.44	$5.41e-116$
GZMB	$5.6e-171$	2.42	$8.37e-167$
IL32	$3.71e-30$	2.32	$5.55e-26$
KLRB1	$1.1e-137$	2.28	$1.64e-133$
CTSW	$1.94e-102$	2.23	$2.9e-98$
GZMH	$2.52e-94$	1.98	$3.76e-90$
FGFBP2	$1.06e-120$	1.97	$1.58e-116$
PRF1	$3.63e-139$	1.95	$5.43e-135$
CLIC3	$1.77e-93$	1.93	$2.64e-89$
KLRD1	$1.22e-164$	1.93	$1.82e-160$
SYNE2	$2.97e-73$	1.89	$4.43e-69$
GZMM	$2.12e-107$	1.79	$3.17e-103$
ETS1	$7.13e-61$	1.77	$1.07e-56$
CD247	$1.18e-102$	1.74	$1.76e-98$
SPON2	$1.27e-85$	1.68	$1.9e-81$
CMC1	$5.15e-34$	1.66	$7.71e-30$
...	...	...	...
ORMDL3	$3.38e-10$	0.253	$5.05e-06$
MNAT1	$8.35e-07$	0.251	0.0125
OAZ1	$1.65e-08$	−0.511	0.000247
LAPTM5	$1.01e-06$	−0.521	0.0151
GNAI2	$6.64e-07$	−0.536	0.00992
CST3	$3.28e-06$	−0.546	0.049
S100A6	$2.83e-07$	−0.579	0.00424
BRI3	$2.03e-07$	−0.612	0.00304
FTH1	$4.19e-07$	−0.614	0.00626
FKBP1A	$1.32e-06$	−0.618	0.0197
FTL	$2.03e-10$	−0.67	$3.04e-06$
COTL1	$1.22e-09$	−0.683	$1.82e-05$
ASAH1	$9.13e-07$	−0.701	0.0136
CTSS	$6.64e-12$	−0.755	$9.93e-08$
VIM	$1.3e-07$	−0.762	0.00194
LST1	$3.2e-07$	−0.801	0.00478
LYZ	$6.22e-07$	−0.854	0.0093
FCN1	$2.68e-09$	−0.882	$4.01e-05$
S100A11	$6.85e-07$	−0.899	0.0102
SPI1	$3.1e-09$	−0.941	$4.64e-05$

Table 8: (DE) analysis performed on cells selected from cluster 8 in the refined network. The first group correspond to the myeloids and the second group is the remaining cells selected. Genes are sorted by the third column.

gene	p_val	avg_log2FC	p_val_adj
LYZ	$7.23e-43$	6.13	$1.08e-38$
S100A9	$4.28e-41$	5.57	$6.4e-37$
FOS	$1.57e-39$	4.83	$2.35e-35$
S100A8	$4.84e-36$	4.74	$7.23e-32$
CTSS	$4.51e-41$	4.27	$6.75e-37$
VCAN	$4.63e-38$	4.12	$6.92e-34$
FCN1	$3.87e-42$	3.92	$5.79e-38$
AIF1	$4.07e-40$	3.48	$6.09e-36$
LST1	$1.28e-39$	3.46	$1.92e-35$
FGL2	$4.79e-42$	3.42	$7.16e-38$
CYBB	$2.59e-40$	3.27	$3.87e-36$
NAMPT	$1.34e-36$	3.22	$2e-32$
S100A12	$2.58e-30$	3.18	$3.85e-26$
DUSP1	$1.56e-38$	3.11	$2.33e-34$
SERPINA1	$4.88e-42$	3.05	$7.3e-38$
MNDA	$3.86e-38$	2.97	$5.77e-34$
PSAP	$5.05e-39$	2.75	$7.55e-35$
KLF10	$4.74e-39$	2.74	$7.09e-35$
LGALS1	$1.8e-35$	2.71	$2.69e-31$
TYMP	$6.4e-36$	2.71	$9.57e-32$
TYROBP	$1.57e-35$	2.68	$2.35e-31$
SPI1	$6.55e-39$	2.64	$9.8e-35$
EGR1	$1.93e-25$	2.63	$2.89e-21$
S100A6	$1.69e-37$	2.59	$2.52e-33$
APLP2	$1.14e-39$	2.56	$1.71e-35$
CSTA	$1.76e-36$	2.54	$2.63e-32$
CLEC7A	$2.5e-38$	2.53	$3.74e-34$
CEBPD	$1.17e-35$	2.47	$1.75e-31$
CD14	$1.1e-31$	2.46	$1.65e-27$
COTL1	$2.67e-36$	2.44	$3.99e-32$