MUFFIN: a suite of tools for the analysis of functional sequencing data

Pierre De langen, Benoit Ballester

NAR Genomics and Bioinformatics, 2024, **6**, lqae051
https://doi.org/10.1093/nargab/lqae051
Advance access publication date: 14 May 2024
Pipelines and Workflows for Biological Data Analysis

OXFORD

MUFFIN: a suite of tools for the analysis of functional sequencing data

Pierre de Langen and Benoit Ballester*

Aix-Marseille Univ, INSERM, TAGC, Marseille, France

*To whom correspondence should be addressed. Tel: +33 4 91 82 87 28; Fax: +33 4 91 82 87 01; Email: benoit.ballester@inserm.fr

Abstract

The large diversity of functional genomic assays allows for the characterization of non-coding and coding events at the tissue level or at a single-cell resolution. However, this diversity also leads to protocol differences, widely varying sequencing depths, substantial disparities in sample sizes, and number of features. In this work, we have built a Python package, MUFFIN, which offers a wide variety of tools suitable for a broad range of genomic assays and brings many tools that were missing from the Python ecosystem. First, MUFFIN has specialized tools for the exploration of the non-coding regions of genomes, such as a function to identify consensus peaks in peak-called assays, as well as linking genomic regions to genes and performing Gene Set Enrichment Analyses. MUFFIN also possesses a robust and flexible count table processing pipeline, comprising normalization, count transformation, dimensionality reduction, Differential Expression, and clustering. Our tools were tested on three widely different scRNA-seq, ChIP-seq and ATAC-seq datasets. MUFFIN integrates with the popular Scanpy ecosystem and is available on Conda and at https://github.com/pdelangen/Muffin.

Introduction

The diversity of whole-genome functional sequencing protocols allows us to measure a wide variety of regulatory activities across the genome. These range from transcriptomic signals, which can be detected using techniques like RNA sequencing or Cap Analysis of Gene Expression, to epigenetic signals, which can be identified with methods like ChIP-seq or ATAC-seq. The latter respectively quantify the binding of target proteins to the DNA and regions of open chromatin. The number of observations (e.g. cells or tissue samples) can also vary significantly, depending on the experimental design, from a standard two-condition triplicate protocol to thousands of observations in large-scale integrative analyses or sequencing at single-cell resolution.

These whole-genome sequencing protocols involve the mapping of sequencing reads onto a reference genome. These are then counted at specific genomic locations to effectively serve as molecular counters. The genomic locations typically originate either from reference databases (for gene-centric analyses) or are identified de-novo through peak-calling algorithms (for epigenetic or regulatory non-coding events analyses). The measured sequencing signal's intensity can also be highly diverse, ranging from a few thousand UMIs per observation in single-cell analyses to hundreds of millions of reads in bulk sequencing. However, a common aspect across all these whole-genome sequencing protocols is that the sequencing signal per observation ultimately gets quantified through counts in a large number of features (e.g. genes or genomic peaks) resulting in a count table.

To analyze such a variety of assays, we built MUFFIN, a modular Python package. First, MUFFIN provides a robust and flexible count table processing pipeline centered around the use of the residuals of a regularized Negative Binomial statistical model. We demonstrate that this generic methodology can be employed to conduct between-observation comparisons in a wide variety of experimental designs, with low to high observation counts, and shallow to deep sequencing. We also extend this procedure to account for sequencing protocols with input or control sequencing. MUFFIN also provides several native and wrapper functions regarding normalization, count transformation, dimensionality reduction, differential expression and clustering.

MUFFIN also has tools specifically tailored for epigenetic or peak-called assays such as ATAC-seq or ChIP-seq. It allows to identify consensus peaks across multiple peak-called experiments at a high resolution, permitting between-sample comparison. It also offers a statistically robust tool to perform Gene Set Enrichment Analyses on arbitrary genomic regions such as sequencing peaks.

Our tool is fully interoperable with the functions of the popular Scanpy package and its ecosystem, which was initially designed for single-cell analysis, but whose tools can be adapted for broader applications.

Materials and methods

Sequencing signal in arbitrary genomic features

Our framework uses raw, non-normalized read or UMI count values over genomic features. For read-based data, we provide a simple wrapper to FeatureCounts (1) to retrieve counts in BAM files located in query genomic features provided in BED, GTF or SAF format using the *muffin.load.dataset_from_bam function*. We support any type of genomic feature, as the user can use genes or reference maps for regulatory elements such as the ENCODE CCREs (2) or DNase hypersensitive regions

Received: December 12, 2023. **Revised:** April 10, 2024. **Editorial Decision:** April 22, 2024. **Accepted:** April 27, 2024

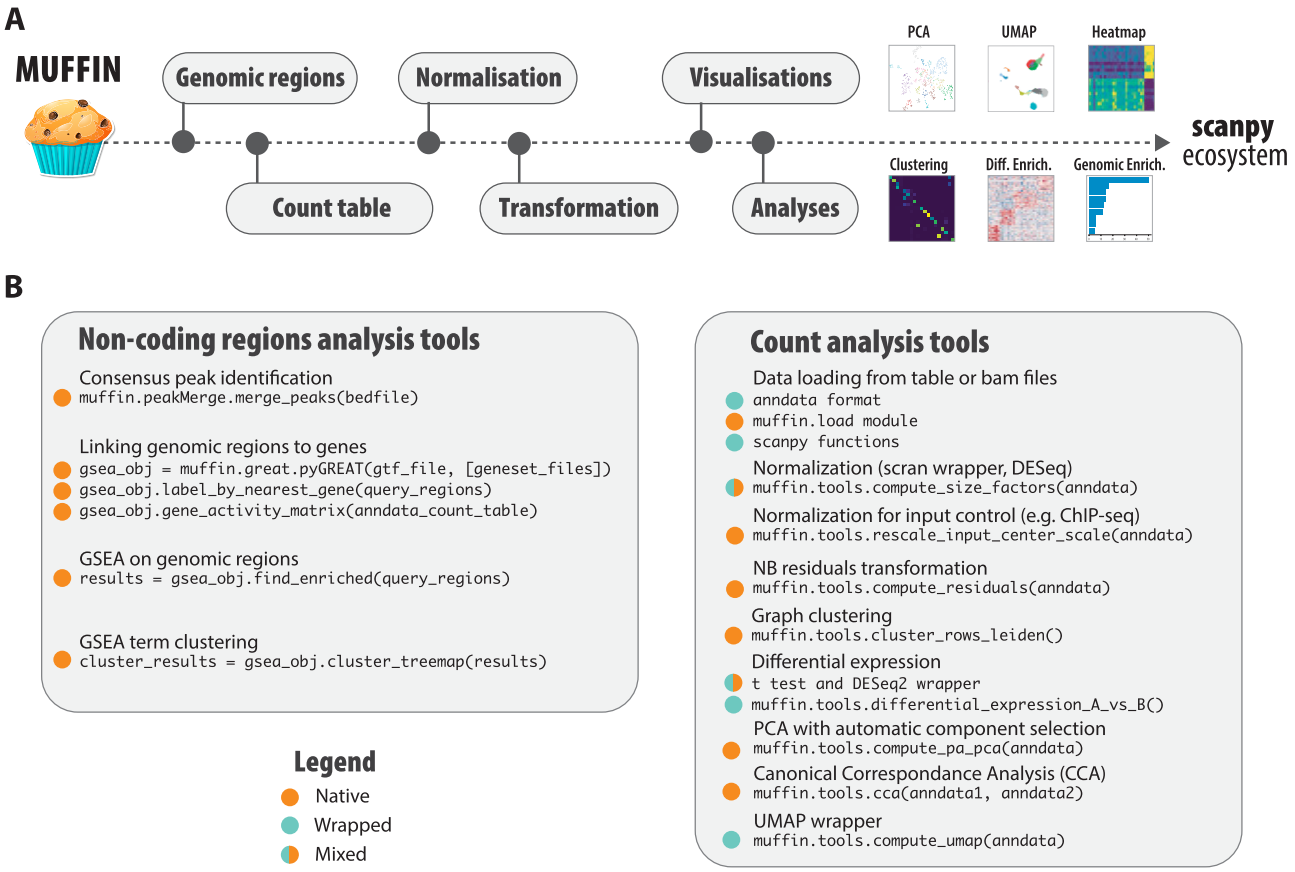

Figure 1. Overview of the MUFFIN suite of tools. (**A**) MUFFIN covers multiple aspects of the analysis of several types of functional sequencing data, from the identification of consensus peaks, to differential expression and Gene Set Enrichment Analysis. (**B**) List of the main tools offered by MUFFIN. For an exhaustive list and detailed instructions, we recommend to visit our ReadTheDoc website.

catalogue (3). For the de novo discovery of genomic elements, typically done after using a peak-calling algorithm, we provide a simple tool detailed in the next paragraph for the identification of consensus peaks across a large number of experiments (muffin.peakMerge.merge_peaks function). Alternatively, the user can provide their own count tables, which can be easily formatted in the standardised Python AnnData format (4) using the provided helper functions such as *muffin.load.dataset_from_arrays*. An overview of our tools is available in Figure 1, and a detailed description of the provided tools is given in the methods below.

High resolution consensus peak identification in large datasets

When working with sequencing data that target regulatory elements, the signal typically manifests in the form of peaks in sequencing protocols such as ChIP-seq, ATAC-seq or CAGE. Various peak-callers have been developed to identify these signal regions on genomic tracks (5,6). To integrate observation-level peaks into consensus peaks, which serve as key sampling points for sequencing reads and downstream analyses, we propose a simple tool. This tool (called with *muffin.peakMerge.merge_peaks*), which we used in previous work (7), accepts per-experiment peak-calling results (from an external tool) as input and outputs consensus peaks based on the genome-wide peak density. While identifying consensus peaks based on simple overlaps is an acceptable approach when working with a small number of experiments, it becomes problematic with a larger number of experiments. In such cases, some regions of the genome can be almost entirely covered by peaks, making the high-resolution identification of consensus peaks impossible. To merge peaks into consensus peaks, we first compute the density function of the peak summits (the single base pair genomic location with the maximum signal of the peak, we use the centre position if not available) across each chromosome. To do this, we use a kernel density estimate, employing σ = Median peak size/8 as the bandwidth. We then delineate consensus peaks at each local minimum of the density function. Abnormally small artifactual consensus peaks with a size smaller than the bandwidth are discarded, as well as irreproducible peaks (consensus peaks formed by only one experiment, can be modified). Finally, we restrict the boundaries of the newly defined consensus peak to those of the farthest peaks (Supplementary Figure S6). When working with stranded peak data (e.g., CAGE or RAMPAGE), we simply run these steps once for each strand.

We compared our approach to the naïve merge on overlap method, as well as an iterative approach used in the ArchR single-cell package (8) in Supplementary Figure S2. The latter sorts peaks per signal intensity, chooses the strongest peak as lead peak and merges every peaks which summit overlaps the lead peak. The procedure is repeated until each peak has been merged or chosen as lead peak. We use a custom implementation, as the use of this tool is tedious outside of its single-cell analysis pipeline. Our comparison show that our method accurately identifies consensus peaks at various peaks sizes and counts, whereas the naïve approach is not resolu-

tive at all; the iterative approach yields a substantial improvement over the naïve method, but often does not align properly with the actual peak shape, especially with wide peaks. This is likely caused by the greedy heuristic of choosing the peak with the strongest score as the 'lead peak' for merging subsequent peaks, which may not be optimal. Our method also guarantees non-overlapping consensus peaks, which is important in order to avoid merging signals originating from different regulatory elements.

Linking genomic regions to genes and functional annotations

Genes are much more studied and functionally annotated compared to regulatory elements of the genome. As regulatory elements are typically located near the gene they regulate, or as transcripts are frequently grouped into co-regulated clusters with similar functions, we propose tools to assist in functionally annotating regions of interest in the *muffin.great* module of our package. We provide an utility to rename them according to their nearest gene, as well as a tool to perform Gene Set Enrichment Analysis based on genomic regions. We use a statistical framework that assumes the query regions are a subset of background regions (e.g. a set of differentially expressed/bound/accessible regions across two conditions is a subset of all the regions considered for differential testing). Our approach is similar to Chip-Enrich (9) and Poly-Enrich (10), which have shown that gene-wise modelling is required to reduce false discoveries. However, these two methods do not offer a model for the case where the query regions are a subset of a set of background regions, which is important as the background regions typically have a gene set bias (e.g. a two-condition ATAC-seq experiment carried on liver will have enriched liver-related gene sets, which are not necessarily related to the differences in conditions).

GREAT (11) uses a two-test approach using a binomial and a hypergeometric test: the first one is robust to randomly chosen regions of the genome while the latter is robust to random genesets. However, when querying a large number of regions, the hypergeometric test saturates and cannot find enriched genesets, as most genes of the genome are getting tagged. Some tools and approaches perform GSEA of genomic regions using only the hypergeometric test on a list of tagged genes, and we strongly advocate against using this approach as this is not robust to randomly chosen regions of the genome (see Supplementary Table S1). We instead recommend to use our approach, GREAT (if query size is suitable) or Poly-Enrich.

To assign query genomic regions to genes, we use the same heuristic as GREAT at default settings: a basal domain of 5 kb upstream and 1 kb downstream, extended in both directions up to 1 Mb or to the nearest basal domain (whichever is the closest). Alternatively, the user can provide his custom regulatory regions for each gene as a bed file. For each gene i, we obtain the number of genomic regions in its regulatory region, for all the background regions (n_i) and the subset of interest (k_i).

To compute Gene Set enrichments, we fit a Negative Binomial Generalized Linear Model per gene set s (e.g. the set of genes with the 'B cell activation' GO term) , which predicts the expected number of genomic regions of the subset of interest $\mu_{i,s}$ in the regulatory region of a gene i: $\ln(\mu_{i,s}) = \beta_{0,s} + \beta_{1,s} \times G_s + ln(E_i)$, where G_s is equal to 1 if the studied gene belongs to the Gene Set s of interest and 0 otherwise. The term E_i multiplicatively corrects for the intersection bias of the background regions, with E_i being the expected number of hits for a particular gene: $E_i = n_i \times \frac{K}{N}$, with K being the number of query regions, and N the number of background regions. We test for each gene set s whether $\beta_{1,s}$ is greater than zero (i.e., whether genes in the gene set have more hits than those not in the gene set) using a Wald test and apply the Benjamini–Hochberg FDR correction.

We show that our approach is statistically robust in Supplementary Table S1. Poisson and binomial models do not handle properly the overdispersion seen in real genomic data, as query regions are often grouped together, resulting in large gene hit counts, leaving a large number of genes with very few or no hits. While these models are robust under random regions, as this does follow a Binomial process, these are not robust to random genesets, as gene-regions association counts are still overdispersed with real genomic data. We also note that when using background regions, GREAT drops its Hypergeometric test on gene lists and uses instead an hypergeometric test that is similar to the Poisson or Binomial test. We thus recommend to not use GREAT with background regions.

The top enriched genesets are often correlated due to shared enriched genes, displaying redundant information. To reduce redundancy between gene sets, gene sets can be clustered according to the similarities between their annotated genes. This allows to display the results as de-correlated groups of gene sets. We use the Leiden (12) algorithm on a kNN graph to cluster gene sets, which can be represented as a gene, gene set binary matrix of association. We show one example of clustered gene sets in Supplementary Figure S3.

Count modelling and transformation for between observation comparisons

Following previous work (13,14), we assume that for an observation i (e.g. sample or cell), for a variable j (e.g. genomic region or genes), the observed number of counts $Ki_{,j}$ (reads or UMIs) can be modelled with a Negative Binomial (NB) distribution with mean $\mu_{i,j}$ and overdispersion α_j:

$$K_{i,j} \sim NB(\mu_{i,j}, \alpha_j)$$

This family of distribution is discrete, heteroscedastic, skewed and with long tails which often cause issues with approaches that assume normality. To circumvent this issue, normalizing transformations have been developed, the most popular being the log1p transform (or logCPM), which typically appears under the form:

$$log_a\left(1 + c \times \frac{K_{i,j}}{s_i}\right), \ a \text{ and } c \text{ being constants.}$$

While this approach has been found to work well for large counts with few zeroes, it raises issues for small counts where it can distort the underlying distribution, especially between zeroes and small counts. Furthermore, the choice of the a and c constants is often arbitrary but does have a large effect on the balance between skewness and zero-distortion (15). Recent work suggests using the residuals of a null (i.e. constant expression) NB model that has its overdispersion parameter α_j being constrained to be a function of the mean (14,16,17):

$$ln(\mu_{i,j}) = \beta_j X_i + ln(s_{i,j})$$
$$\alpha_j^{reg} = f(\mu_j)$$

With $s_{i,j}$ being the multiplicative normalization factor for variable j and observation i; α_j^{reg} the regressed model coefficients for variable j, X the design matrix containing variables to regress out (i.e. a single intercept if there is no unwanted variables to regress). Briefly, the idea behind this approach is to model what would be the expected count distribution of a variable according to its mean expression for a constant gene/feature, then compute how much each observed count $K_{i,j}$ is deviating from the expected fitted distribution $NB(\mu_{i,j}, \alpha_j^{reg})$ of the variable j.

In this work, we fit $f(\mu_{i,j})$ by randomly sampling up to 2000 variables (to speed up computations as there is no need to fit each variable to obtain a trendline), fit models using Maximum Likelihood Estimation with the Statsmodels (18) Nelder-Mead solver (without constraint on the overdispersion parameter). To obtain the regularized estimate of overdispersion we use a rolling median of the mean-sorted overdispersion values. Finally, we re-fit for each variable a new model with its regularised j using the Statsmodels IRLS solver, then compute residuals between the predicted distribution $NB(\mu_{i,j}, \alpha_j^{reg})$ and the observed counts $K_{i,j}$.

The popular SCTransform approach suggests using Pearson residuals, which are a linear transformation of the observed counts:

$$r_{i,j}^{p} = \frac{K_{i,j} - \mu_{i,j}}{\sqrt{\mu_{i,j} + \alpha_j^{reg} \times \mu_{i,j}^2}}$$

Here, we use Anscombe residuals, as implemented in Statsmodels, which are a nonlinear transformation of the observed counts, and are asymptotically following a standard normal distribution for a properly specified model.

$$r_{i,j}^{a} = \frac{K_{i,j}^{2/3} \times H(-\alpha_j^{reg} \times K_{i,j}) - \mu_{i,j}^{2/3} \times H(-\alpha_j^{reg} \times \mu_{i,j})}{(\mu_{i,j} + \alpha_j^{reg} \times \mu_{i,j}^2)^{1/6}}$$

where $H(x) = H_2F_1(2/3, 1/3, 5/3, x)$, H_2F_1 being the Gauss Hypergeometric function.

We found Pearson residuals to be much more sensitive to outliers, and as they are still skewed and heavy-tailed due to the linear nature of the transformation, their values have to be clipped to prevent large values from driving a majority of the variance. The clipping value is a sensitive hyperparameter and has to be adjusted per dataset, with too large values being sensitive to outliers, and too small values removing biological signal. Additionally, SCTransform uses an arbitrary lower bound on the variance estimate to prevent weakly expressed genes to drive a majority of the variance. We found that Anscombe residuals do not require clipping (Supplementary Figure S4), give low weights to weakly expressed features (Supplementary Figure S7A), as well as having better known theoretical statistical properties concerning normality. In our package, we implement both methods and use Anscombe residuals as the default transform.

Compared to log1p transforms, residuals were also found to generate less library size-related batch effects (14,15). We confirm in Supplementary Figure S5 that our implementation is as well less sensitive to library size variations.

To perform the residual transformation, the user simply has to call *muffin.tools.compute_residuals*. The matrix X containing variables to regress out (e.g. age) can be supplied through *muffin.load.set_design_matrix*.

Between sample normalization

We implement a few popular methods of normalization on top of library size normalization, such as DESeq2 (13) median of ratios, Upper Quartile normalization, and implement the scran pooling and deconvolution method (19) through a rpy2 wrapper to the scran library. These functions are located within *muffin.tools.compute_size_factors*, and the user can choose which method to use via the 'method' argument. The median of ratios approach is recommended for data with large counts, while the scran approach is better suited for small counts and a large number of observations. A generic method of normalization that is independent of sample size and sequencing depth is still an area of open research. We note that our model for count distribution allows not only for a per observation multiplicative size factor but also for a per feature, per observation normalization factor ($s_{i,j}$). This could be exploited to use normalization strategies that aim to correct sequence-content bias in the model. Here, we use this property to extend the model to sequencing data with input counts.

Extending the model to sequencing data with input counts

Input sequencing is used to estimate sequence-content biases, such as PCR bias, sonication, mappability, or CNV alterations, and is used widely, for example in ChIP-seq. In these experiments, the log of the fold change (LFC), or the enrichment p-value against the input is typically used as metrics for the signal. However, the LFC does not take into account the significance of the enrichment, and on the other hand, the *P*-value is difficult to interpret nor displays the effect size of the enrichment; both measurements can be difficult to use for between sample comparisons. Most current methods simply ignore the amount of Input (20) and assume the sequence bias is the same across all conditions, which can be incorrect in case of differential chromatin accessibility or difference in the immunoprecipitation efficiency. As the amount of input has a multiplicative effect on the observed counts, the null count model simply becomes:

$$ln(\mu_{i,j}) = \beta_j X_i + ln(Normalized\ Input_{i,j})$$

However, there is a need to normalize the input counts, and previous research has found that library size normalization is insufficient, as the observed (ChIP) signal contains both true signal and noise reads. The signal-to-noise ratio has also been found to fluctuate strongly between experiments, mainly due to varying immunoprecipitation quality. We present a novel two-step approach, which consists of centering and then scaling the input counts. The centering step aims to find the zero of the log fold change, i.e., finds a multiplicative factor at the observation level which normalizes background regions with no signal between the ChIP and input. Here, we use the Signal Extraction Scaling (21) (SES) approach on counts sampled from random regions of the genome (10 000 per default) to estimate a centering factor c_i. While the centering step is sufficient to perform peak calling, another scaling step is required to normalize fold changes, as samples with a more successful immunoprecipitation will have larger fold changes. We compute the per observation, median centered,

signal enrichment factor $SEFM_i$ as:

$$SEF_i = \frac{\sum_{j \in variables} K_{i,j}}{\sum_{j \in variables} c_i \times I_{i,j}}$$

$$SEFM_i = SEF_i / median(SEF)$$

I being the input count matrix (note that here, we do not use the counts at random genomic regions to compute SEF_i, but the actual count matrices K and I).

Intuitively, this means that, for example, for an observation i with a two times stronger signal enrichment factor, with an observed fold change over input of 1.5 at feature j, its input should be scaled to obtain a fold change of 1.25. Formally, this translates into:

$$fc_{i,j} = \frac{K_{i,j}}{I_{i,j} \times c_i}$$

$$r_{i,j} = SEFM_i \times (fc_{i,j} - 1) + 1, \quad fc_{i,j} >= 1$$

$$r_{i,j} = \frac{1}{SEFM_i \times (1/fc_{i,j} - 1) + 1}, \quad fc_{i,j} < 1$$

$$Normalized\ Input_{i,j} = I_{i,j} \times c_i \times \frac{fc_{i,j}}{r_{i,j}}$$

Additionally, to avoid removing features having at least a single zero as input counts (as it causes a division by 0 or taking the log of zero in the model), we replace zeroes by the mean input count for this feature (taking into account the input library size). This avoids dropping an extremely large number of variables when dealing with a large number of observations while keeping a meaningful value for the input counts.

Note that we considered ChIP-seq as an example in this paragraph, but our framework should work with experimental designs that are using a control track with an expected multiplicative effect on the resulting counts. Supplementary Figure S6 highlights the effect of the two steps of our input-based normalisation. This normalization procedure is implemented in *muffin.tools.rescale_input_center_scale*.

Differential expression

We implement DESeq2 with Log Fold Change (LFC) shrinkage (22) via a simplified rpy2 wrapper to conduct differential expression between two conditions, as it uses a statistical model similar to ours, supporting a per feature, per observation normalisation factor. However, DESeq2 can be computationally intensive for large datasets. When working with more than 50 samples, which provides sufficient statistical power, we transition to a Welch's t-test performed on the NB residuals, which has been shown to work well for single-cell data (14). It should be noted, though, that the t-test offers less statistical power than DESeq2. These functions are located within *muffin.tools.differential_expression_A_vs_B*, and the user can choose which method to use via the 'method' argument. It should be noted that multiplicative normalization/size factors $s_{i,j}$ are automatically passed to DESeq2, as well as the design matrix containing the variables to regress out that are also used in the computation of the residuals. These functions also output additional information that are required for scanpy's visualization tools to work properly.

Feature selection for downstream tasks

Feature selection in scRNA-seq (single-cell RNA sequencing) is a step that facilitates the elimination of a substantial portion of likely uninformative variables, i.e. those with very low expression that are mostly contaminated by technical noise, or those with ubiquitous expression, which do not provide insight into the sample/cell biology (15,16). Removing these can enhance the quality of downstream tasks, such as classification, clustering or visualization. Typically, around 2000–3000 genes are retained in scRNA-seq experiments, but this number is generally hand-tuned for each experiment. We use the sum of squared residuals (SSR) as a criterion for selecting variables, which serves as an indicator of the goodness of fit for the null NB model. Large residuals suggest a poor fit of the null model of constant expression, likely caused by differential expression between biological conditions that are not observed in the null model. In Supplementary Figure S7A, we show that the Anscombe residuals have a higher variance in sufficiently expressed, highly variable features.

Optimal number of components for principal component analysis

To automatically identify the optimal number of Principal Components, we utilize Horn's Parallel Permutation Analysis, which has proven to be one of the most effective methods to determine the number of components in factor analysis (23). This approach involves generating row-wise permutations for each feature, computing PCA on these permuted datasets, and selecting the number of components at the threshold where the eigenvalues from the randomized dataset exceed those from the actual dataset. By default, we use the residuals as input to the PCA, and due to the computational cost of this approach, we only perform three permutations by default. This is justifiable, as the randomized eigenvalues are very stable on large matrices (Supplementary Figure S7B). To compute PCAs, we use the fast 'randomized' solver from the Python sklearn library. The user can run this tool using *muffin.tools.compute_pa_pca*.

Graph clustering

We implement the Shared Nearest Neighbour (SNN) Graph Clustering approach to identify clusters. This approach is common in single-cell RNA sequencing (scRNA-seq) analyses to identify clusters of cells without a priori on the number of clusters. To scale to a large number of points to cluster, we use an Approximate Nearest Neighbour (ANN) method to build the NN graph (python library PyNNDescent (24)). This approach avoids the quadratic time complexity of building exact nearest neighbours, can use any metric and runs in an almost linear time complexity. By default, we use the PCA representation with Pearson correlation as the metric to build the NN graph. In the SNN graph, vertices are weighted by the number of shared nearest neighbours between the two nodes. To identify communities in the SNN graph, we used the Leiden graph clustering algorithm (12) implemented in the Python Leidenalg library. The clustering can be performed using *muffin.tools.cluster_rows_leiden*.

Tools for dataset integration

We labelled a scATAC-seq dataset using scRNA-seq labels in Supplementary Figure S9. To do so, we summed the the

counts of scATAC peaks overlapping gene bodies (extended by 1 kb) in each cell. We computed NB residuals on the resulting count table and performed Canonical Correspondence Analysis (CCA) as described in the Seurat paper (25) using the NB residuals of the genes identified in both scRNA-seq and scATAC-seq. As a rule of thumb, we used the same number of components as identified in the PCA step of the scRNA-seq dataset (22, Seurat default being 20). We used harmony (26) to correct batch effects in the CCA space. Then, we fit a random forest classifier on the scRNA-seq CCA space, predicting cell labels; which is then used to predict cell labels from the scATAC-seq CCA space. Finally, as this process is quite noisy, we smooth the transferred labels of each cell by using the most represented label within its nearest neighbors in the scATAC-seq PCA space (fitted using only scATAC-seq information).

CCA is implemented in *muffin.tools.cca* and label transfer in *muffin.tools.transfer_categorical_labels*.

Data visualization

For visualizing the similarity among observations in datasets with a large number of features and observations, dimensionality reduction methods such as UMAP (Uniform Manifold Approximation and Projection) or t-SNE (27) (t-Distributed Stochastic Neighbor Embedding) have become standard, particularly for the analysis of single-cell data. Here, we provide a wrapper for UMAP, which is, by default, performed on the PCA representation. By default, Pearson correlation is used as the metric when there are more than 10 input dimensions; otherwise, the Euclidean distance is used. This can be done using *muffin.tools.compute_umap*.

Analysis of benchmark datasets

We manually retrieved matching peak calling BED files, ChIP BAM files, and Input BAM files from ENCODE (2) (see Supplementary Tables). We selected only those H3K4Me3 ChIP-seq experiments that were mapped onto hg38, and removed samples with any audit errors. In cases where multiple replicates of the input file were present, we retained only the most deeply sequenced one and used it across all biological replicates. We used a coarse annotation (the same as in the ENCODE data portal) instead of the detailed cell types as there is a large number of precise cell types with no replicates. The list of genes associated with Gene Ontology terms was retrieved from the g:ProfileR website (28).

Peaks used for the identification of consensus peaks in Supplementary Figure S2 were retrieved from ENCODE for the CAGE and H3K27Ac ChiP-seq peaks, and ReMap for the CTCF ChIP-seq peaks.

We obtained the filtered 10k PBMC scRNA-seq and scATAC-seq count tables from 10X Genomics (https://www.10xgenomics.com/resources/datasets/10k-human-pbmcs-3-ht-v3-1-chromium-x-3-1-high, https://cf.10xgenomics.com/samples/cell-atac/1.0.1/atac_v1_pbmc_10k/atac_v1_pbmc_10k_filtered_peak_bc_matrix.h5). We also analyzed the dataset using Seurat with SCTransform V2 as a reference, with the default settings from its tutorial vignette. We compared the similarity between our method and SCTransform using the jaccard index of similarities between clusters, and using Adjusted Mutual Information (AMI) to compare similarities across all clusters.

TCGA ATAC-seq (29) count tables were obtained from the NIH (https://gdc.cancer.gov/about-data/publications/ATACseq-AWG). The list of candidate genes associated with specific cancer hallmarks was obtained from the CHG database (30).

All analyses were conducted using the default settings, with the exception of the normalization method: for the single-cell dataset, we employed scran normalization; for the TCGA ATAC-seq dataset, we applied DESeq's median of ratios; and for the ChIP-seq dataset, we used our custom normalization method. Furthermore, we utilized logistic regression via Scanpy to identify markers across multiple classes in both the single-cell and the ATAC datasets.

Results

MUFFIN suite: applications and validation in diverse biological contexts

Here we introduce the comprehensive MUFFIN suite, encompassing a diverse array of tools tailored for analyses of a large spectrum of sequencing experimental designs. The origin of this suite can be traced back to the creation and analysis of an RNA Polymerase II Atlas, involving the development and use of various tools in the context of 900 ChIP-seq experiments and approximately 28 000 RNA-seq samples (7).

MUFFIN offers two main categories of tools: first, a set of tools to work with non-coding regions that can identify and link regulatory regions to gene and functional annotations; second, a robust and flexible data-driven count processing pipeline that minimizes the number of parameters that has to be chosen by the user (see Figure 1). Our tools primarily bring improvements to existing approaches (e.g. GSEA of genomic regions, NB residuals), or port missing tools existing only in the R ecosystem to the Scanpy Python ecosystem through wrappers and re-implementation of existing approaches (e.g. scran and DESeq2 wrappers, NB residuals). A detailed description of the tools with comparisons to existing approaches is available in the Methods.

To validate the suite's efficacy, it was tested in three largely different datasets: (i) the ENCODE Epigenetic Atlas of Immune Cells using H3K4Me3 ChIP-seq, (ii) 10 000 PBMCs sequenced with 10× scRNA-seq and (iii) 400 cancer samples sequenced via ATAC-seq originating from TCGA. These applications showcase the versatility and robustness of the MUFFIN suite across diverse biological and technical contexts. MUFFIN is thus likely to be useful for the analysis of most count-based sequencing dataset, and shows that several methodologies initially developed for scRNA-seq can be applied to a broader range of functional sequencing assays.

ENCODE epigenetic atlas of immune cells through H3K4Me3 ChIP-seq

In order to evaluate the general applicability of our proposed methodology, we utilized the H3K4Me3 ChIP-seq dataset from the comprehensive ENCODE epigenetic atlas of immune cells. H3K4me3, an epigenetic modification known for marking active promoters, is a crucial mediator of transcriptional activity and plays a significant role in cellular differentiation and function. Here, multiple difficulties are linked to this dataset: there are no reference regions to sample sequencing signals from, the sample sizes are rather small, and the sequencing protocol uses a control track.

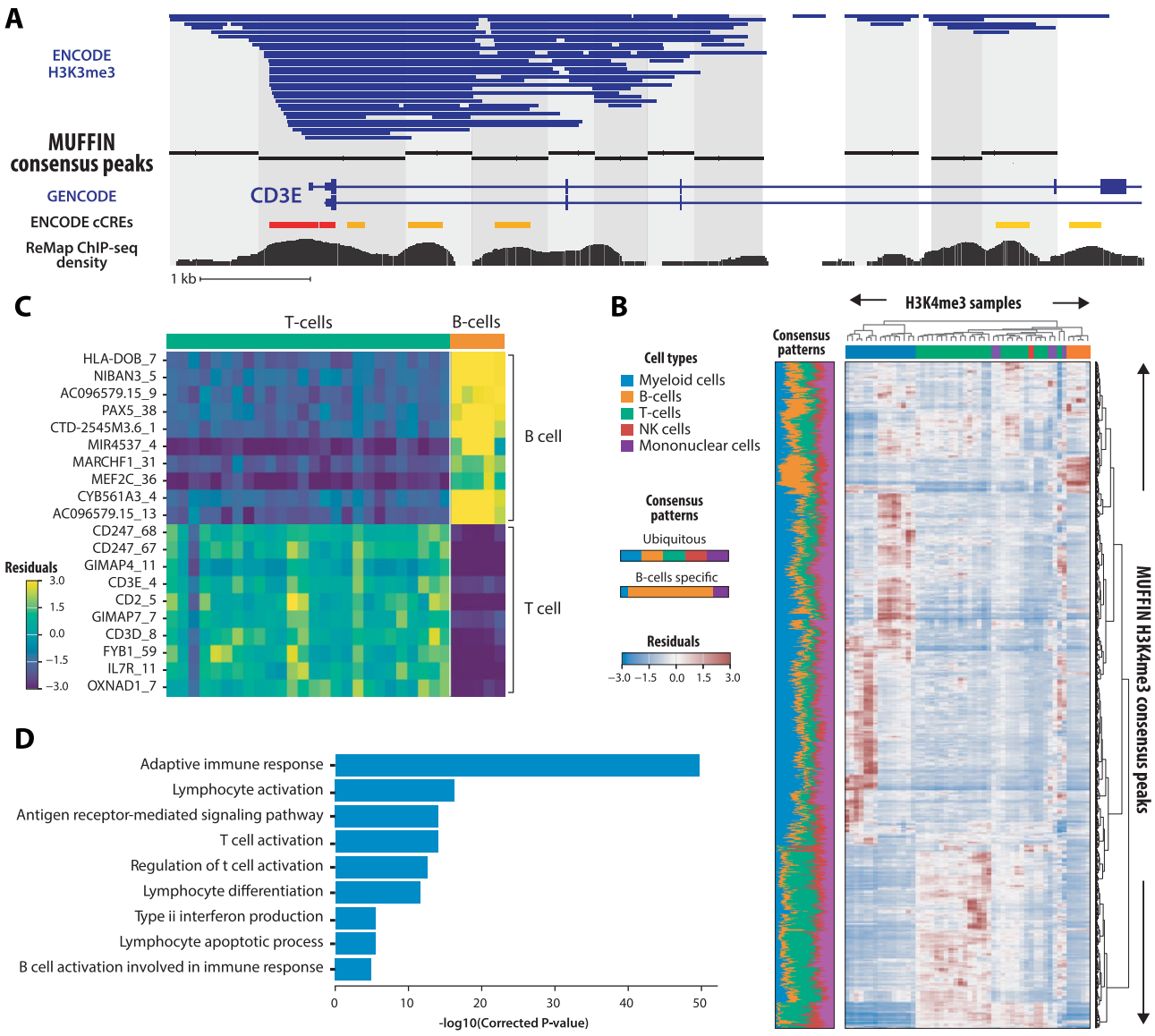

Figure 2. Analysis of the ENCODE epigenetic atlas of Immune cells through H3K4Me3 ChIP-seq. (**A**) Result of the consensus peak identification in a 10kb region around the CD3E gene. Top to bottom rows indicates: ReMap Transcription Factor binding density, Consensus peaks, Peaks from all experiments aggregated, GENCODE V43 annotation. The alternating colors highlight the coverage of each consensus peak. (**B**) Heatmap of the Anscombe residuals. Rows correspond to samples and columns to consensus peaks. Bottom panel indicates the weighted proportion of the total Sum of Squared Residuals carried by each class. (**C**) Heatmap of the Anscombe residuals, for the top 10 most differentially marked consensus peaks (each renamed according to its nearest gene) in either B-cells or T-cells. (**D**) Clustered GO terms enrichments of genes near differentially marked consensus peaks between B-cells and T-cells. Terms are clustered by gene similarity and only the term with the strongest enrichment is displayed.

Our first task was to identify which regions to consider for quantifying H3K4Me3 occupancy. For this purpose, we employed our density-based approach to identify consensus peaks from per-experiment peak calling results. Our approach identifies the consensus peaks at high resolution in dense regions, and correlates with the Transcription Factor binding density from the external ReMap database (31) (Figure 2A). Subsequently, we counted reads in immuno-precipitated and input sample pairs at consensus peak locations. Ultimately, the counts are normalized using our centering and scaling approach (Methods), transformed using Anscombe residuals, and can be processed through a visualization or dimensionality reduction routine. As shown in Figure 2B, a heatmap view shows a clear distinction between immune cell types and reveals distinct H3K4Me3 occupancy across different cell types.

Next, we sought to identify regions differentially marked by H3K4Me3 between B-cells ($n = 5$) and T-cells ($n = 26$). We found that the regions most differentially bound in B-cells are indeed located near B-cell marker genes such as HLA-DOB, which is expressed in Antigen Presenting Cells, or PAX5, a transcription factor involved in B-cell differentiation. Conversely, for T-cells, we also retrieved T-cell markers like CD247 (T-cell surface glycoprotein CD3 zeta chain) or CD3E (CD3 Epsilon Subunit Of T-Cell Receptor Complex). Using our tool of Genomic Regions Enrichment Analysis, we find that these differentially occupied regions are located near genes annotated with immune and

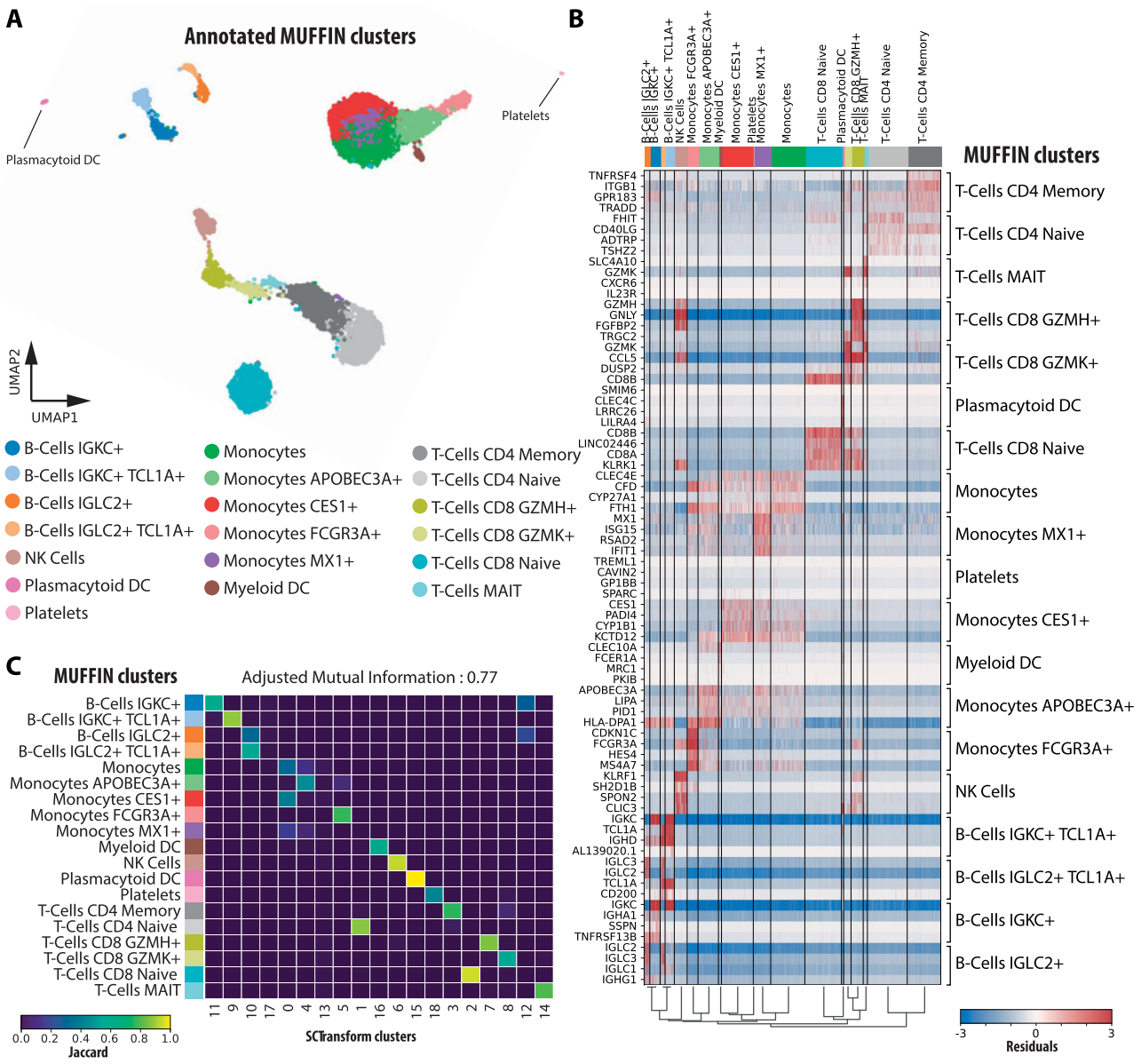

Figure 3. Analysis of a 10k PBMCs single-cell RNA-seq dataset. (**A**) UMAP visualization of the annotated clustered cells. (**B**) Heatmap of the Anscombe residuals, per cell, for the four best markers of each cluster, identified via the coefficients of a multivariate, multi-class logistic regression. (**C**) Heatmap of the Jaccard indexes between clusters identified via SCTransform and MUFFIN.

lymphocyte-related Gene Ontology terms (Figure 2C, D and Supplementary Figure S3. As a whole, this confirms that our methods are able to analyze ChIP-seq data and are likely to work well with other similar datasets that are using control sequencing.

Cell-type clustering of 10 000 PBMCs sequenced by 10× scRNA-seq

After investigating the differential occupancy of H3K4Me3 across various immune cell types, we sought to evaluate the robustness and versatility of our methodology using another widely adopted dataset. Hence, we turned our focus to a single-cell RNA-sequencing (scRNA-seq) dataset, which offers a distinct set of challenges due to the large number of observations, inherent sparsity, shallow sequencing depth and high dimensionality of the data. The 10× Genomics scRNA-seq dataset of Peripheral Blood Mononuclear Cells (PBMCs), a mixed population of various immune cells, is often used as a benchmark case study. The PBMC dataset encompasses numerous immune cell types, allowing us to assess the efficacy of our approach in identifying and distinguishing between different cellular phenotypes.

Here, count quantification per gene, per cell has already been performed. We used UMAP to visualise the transcriptomic similarities between individual cells and applied graph clustering to pinpoint cell types (Figure 3A). We clearly identify canonical cell types, such as Monocytes, B-cells, T-cells or platelets. Their markers, identified via the coefficients of a logistic regression, are also coherent (Figure 3B), with canonical markers such as CD8B for CD8⁺ T-Cells, or GP1BB for platelets. Furthermore, we are able to identify cell types at a fine resolution, highlighting clusters of cells with similar

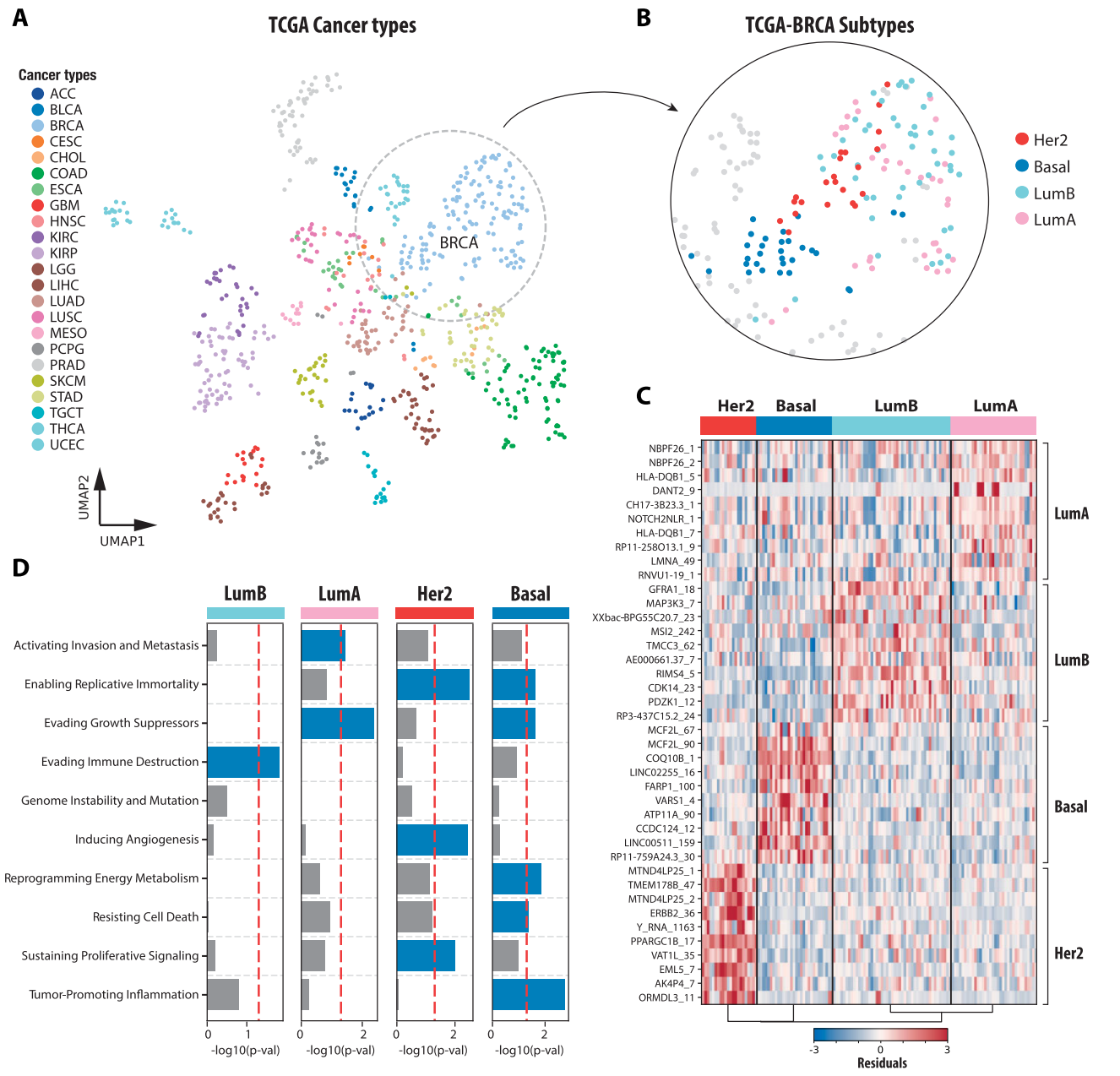

Figure 4. Analysis of cancer samples sequenced with ATAC-seq. (**A**) UMAP visualisation of each tissue sample, annotated by cancer type. (**B**) Subtype for the TCGA-BRCA breast cancer samples. (**C**) Heatmap of the Anscombe residuals, per sample, for the 10 best markers of each breast cancer subtype, identified via the coefficients of a multivariate, multi-class logistic regression. (**D**) Enrichment ($-\log_{10}$(*P*-value)) in cancer hallmarks for genes nearby the 5% most discriminative ATAC peaks for each breast cancer subtype, identified via the coefficients of a multivariate, multi-class logistic regression.

transcriptomes, but with a few very specific markers, especially in the B-cell clusters and the monocyte clusters. Overall, our analysis pipeline was able to provide clusters coherent to those obtained with the reference SCTransform method (Figure 3C), as well as the popular scRNA-seq analysis packages Scanpy and Seurat (Supplementary Figure S8), but similarly to SCTransform is less sensitive to library sizes (Supplementary Figure S5); compared to SCTransform, our approach does not require to tune the clipping parameter which can drastically change the outcome of an analysis (Supplementary Figure S8).

We also analyzed PBMC scATAC-seq data and successfully inferred clusters labels from scRNA-seq to scATAC-seq in Supplementary Figure S9.

Together, these results show that our approaches are also well-suited for the analysis of datasets with low sequencing depth that are typically found in single-cell sequencing.

The Cancer Genome Atlas open chromatin landscape of cancers via ATAC-seq

Finally, we studied the open chromatin landscape in cancers using Assay for Transposase-Accessible Chromatin using sequencing (ATAC-seq) in tissues. ATAC-seq is a technique used to assess the chromatin accessibility, thereby providing insights into the regulatory regions active in a particular cell type, tissue or condition. This sequencing protocol typically

generates deeply sequenced datasets with an extremely large number of features.

The read count table per ATAC peak, per sample was already provided, but a consensus peak identification as performed with the ChIP-seq dataset could have been performed. In Figure 4A, using UMAP, we can clearly distinguish the different cancer types, and even distinguish between different breast cancer molecular subtypes. When focusing solely on breast cancer samples, we can identify markers for each subtype (Figure 4B, C). While Luminal A and B subtypes do not exhibit strongly characteristic signatures, the Basal and Her2 subtypes display characteristic regions of open chromatin. Notably, in the Her2 subtype, an open chromatin region located near the main marker of this cancer, ERBB2 (or Her2) can be identified. Using our tool of Genomic Regions Enrichment Analysis, we observe that characteristic open chromatin regions for each breast cancer subtype locate near cancer hallmark genes (Figure 4D). Notably, the Her2+ breast cancer subtype is known to have elevated levels of Vascular Endothelial Growth Factor (VEGF) gene expression, inducing angiogenesis (32,33).

Discussion

In conclusion, our proposed framework, MUFFIN, represents a valuable and generic toolset for the processing and analysis of count data derived from a variety of high-throughput sequencing experiments. We have demonstrated its performance across diverse datasets, from epigenetic to transcriptomic, and from bulk to single-cell resolution. The underlying model is highly flexible and can adapt to a wide range of experimental designs, and relies on already well-tested, state-of-art statistical approaches.

One of MUFFIN's key strengths lies in its ability to address multiple facets of count-based genomic assays in a generic way with minimal hand-tuning: normalization, count transformation, feature selection, dimensionality reduction, visualization, differential expression, and clustering. Additionally, it offers specialized tools for analyzing data that do not rely on gene annotations. This includes a tool for identifying consensus peaks at a high resolution, and another one for performing functional enrichment of genes located near genomic regions of interest. Furthermore, MUFFIN seamlessly integrates in the existing Python Scanpy ecosystem and its diverse range of tools. The modular architecture of MUFFIN could also allow future packages to be built on top of it.

However, it's worth noting a potential limitation of MUFFIN, which is its compatibility with extremely large, sparse datasets. As its count transformation does not support sparse data formats, this may restrict its use in situations where memory capacity is a significant concern, such as in the analysis of extremely large integrative single-cell datasets consisting of millions of cells. MUFFIN also does not offer all the tools necessary to perform multi-omics analyses or cross-dataset integrations. Nonetheless, we have shown that the NB residuals and the PCA or CCA space produced by MUFFIN can still serve as robust fundations for tools such as Scanorama (25), Harmony (26) or Muon (34), which are all available within the Scanpy ecosystem.

In summary, MUFFIN offers generic tools to analyze high-throughput sequencing count data and complements the existing tools available in Scanpy and the Python ecosystem. We anticipate that it will be of strong interest for bioinformaticians working with functional genomic data, at the tissue or single-cell level.

Data availability

MUFFIN integrates with the popular Scanpy ecosystem and is available on Conda, at https://github.com/pdelangen/Muffin and Zenodo 10.5281/zenodo.11066511 (code), 10.5281/zenodo.10561580 (data).

Supplementary data

Supplementary Data are available at NARGAB Online.

Acknowledgements

We would like to thank Dr Lionel Spinelli for engaging in discussions that have contributed positively to our work.

This work was supported with PhD Fellowship to P.D.L. from the French Ministry of Higher Education and Research (MESR); Institut National de la Santé et de la Recherche Médicale (INSERM); The Core Cluster of the Institut Français de Bioinformatique (IFB) (ANR-11-INBS-0013) for granting access to its high performance computing resources. The results shown here are based upon data generated by the TCGA Research Network, the ENCODE Consortium and the ENCODE production laboratories.

Author contributions: P.D.L. took the lead in developing the methods, coding, and analyzing the datasets. Both P.D.L. and B.B. contributed to the writing and reviewing of the manuscript. All authors edited and approved the article.

Funding

No external funding.

Conflict of interest statement

None declared.

References

1. Liao,Y., Smyth,G.K. and Shi,W. (2014) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, **30**, 923–930.
2. Moore,J.E., Purcaro,M.J., Pratt,H.E., Epstein,C.B., Shoresh,N., Adrian,J., Kawli,T., Davis,C.A., Dobin,A., Kaul,R., *et al.* (2020) Expanded encyclopaedias of DNA elements in the human and mouse genomes. *Nature*, **583**, 699–710.
3. Meuleman,W., Muratov,A., Rynes,E., Halow,J., Lee,K., Bates,D., Diegel,M., Dunn,D., Neri,F., Teodosiadis,A., *et al.* (2020) Index and biological spectrum of human DNase I hypersensitive sites. *Nature*, **584**, 244–251.
4. Wolf,F.A., Angerer,P. and Theis,F.J. (2018) SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.*, **19**, 15.
5. Zhang,Y., Liu,T., Meyer,C.A., Eeckhoute,J., Johnson,D.S., Bernstein,B.E., Nusbaum,C., Myers,R.M., Brown,M., Li,W., *et al.* (2008) Model-based analysis of ChIP-Seq (MACS). *Genome Biol.*, **9**, R137.
6. Tarbell,E.D. and Liu,T. (2019) HMMRATAC: a Hidden Markov ModeleR for ATAC-seq. *Nucleic Acids Res.*, **47**, e91.
7. Langen,P.D., Hammal,F., GuÃ©ret,E., Mouren,J.-C., Spinelli,L. and Ballester,B. (2023) Characterizing intergenic transcription at RNA polymerase II binding sites in normal and cancer tissues. *Cell Genomics*, **3**, 100411.

8. Granja,J.M., Corces,M.R., Pierce,S.E., Bagdatli,S.T., Choudhry,H., Chang,H.Y. and Greenleaf,W.J. (2021) ArchR is a scalable software package for integrative single-cell chromatin accessibility analysis. *Nat. Genet.*, **53**, 403–411.
9. Welch,R.P., Lee,C., Imbriano,P.M., Patil,S., Weymouth,T.E., Smith,R.A., Scott,L.J. and Sartor,M.A. (2014) ChIP-Enrich: gene set enrichment testing for ChIP-seq data. *Nucleic Acids Res.*, **42**, e105.
10. Lee,C.T., Cavalcante,R.G., Lee,C., Qin,T., Patil,S., Wang,S., Tsai,Z., Boyle,A.P. and Sartor,M.A. (2020) Poly-Enrich: count-based methods for gene set enrichment testing with genomic regions. *NAR Genomics Bioinform.*, **2**, lqaa006.
11. McLean,C.Y., Bristor,D., Hiller,M., Clarke,S.L., Schaar,B.T., Lowe,C.B., Wenger,A.M. and Bejerano,G. (2010) GREAT improves functional interpretation of cis-regulatory regions. *Nat. Biotechnol.*, **28**, 495–501.
12. Traag,V.A., Waltman,L. and van Eck,N.J. (2019) From Louvain to Leiden: guaranteeing well-connected communities. *Sci. Rep.*, **9**, 5233.
13. Love,M.I., Huber,W. and Anders,S. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.*, **15**, 550.
14. Choudhary,S. and Satija,R. (2022) Comparison and evaluation of statistical error models for scRNA-seq. *Genome Biol.*, **23**, 1–20.
15. Townes,F.W., Hicks,S.C., Aryee,M.J. and Irizarry,R.A. (2019) Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. *Genome Biol.*, **20**, 295.
16. Lause,J., Berens,P. and Kobak,D. (2021) Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data. *Genome Biol.*, **22**, 1–20.
17. Hafemeister,C. and Satija,R. (2019) Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biol.*, **20**, 1–15.
18. Seabold,S. and Perktold,J. (2010) Statsmodels: econometric and statistical modeling with Python. In: *Proceedings of the 9th Python in Science Conference*. pp. 92–96.
19. L.Lun,A.T., Bach,K. and Marioni,J.C. (2016) Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biol.*, **17**, 75.
20. Eder,T. and Grebien,F. (2022) Comprehensive assessment of differential ChIP-seq tools guides optimal algorithm selection. *Genome Biol.*, **23**, 119.
21. Diaz,A., Park,K., Lim,D.A. and Song,J.S. (2012) Normalization, bias correction, and peak calling for ChIP-seq. *Stat. Appl. Genet. Mol. Biol.*, **11**, Article 9.
22. Zhu,A., Ibrahim,J.G. and Love,M.I. (2019) Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences. *Bioinformatics*, **35**, 2084–2092.
23. Buja,A. and Eyuboglu,N. (1992) Remarks on parallel analysis. *Multivar. Behav. Res.*, **27**, 509–540.
24. Dong,W., Moses,C. and Li,K. (2011) Efficient k-nearest neighbor graph construction for generic similarity measures. In: *Proceedings of the 20th International Conference on World Wide Web*. Association for Computing Machinery WWW '11, NY, pp. 577–586.
25. Stuart,T., Butler,A., Hoffman,P., Hafemeister,C., Papalexi,E., Mauck,W.M., Hao,Y., Stoeckius,M., Smibert,P. and Satija,R. (2019) Comprehensive integration of single-cell data. *Cell*, **177**, 1888–1902.
26. Korsunsky,I., Millard,N., Fan,J., Slowikowski,K., Zhang,F., Wei,K., Baglaenko,Y., Brenner,M., Loh,P.-R. and Raychaudhuri,S. (2019) Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat. Methods*, **16**, 1289–1296.
27. Maaten,L. v.d. and Hinton,G. (2008) Visualizing data using t-SNE. *J. Mach. Learn. Res.*, **9**, 2579–2605.
28. Raudvere,U., Kolberg,L., Kuzmin,I., Arak,T., Adler,P., Peterson,H. and Vilo,J. (2019) g:Profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update). *Nucleic Acids Res.*, **47**, W191–W198.
29. Corces,M.R., Granja,J.M., Shams,S., Louie,B.H., Seoane,J.A., Zhou,W., Silva,T.C., Groeneveld,C., Wong,C.K., Cho,S.W., *et al.* (2018) The chromatin accessibility landscape of primary human cancers. *Science (New York, N.Y.)*, **362**, eaav1898.
30. Zhang,D., Huo,D., Xie,H., Wu,L., Zhang,J., Liu,L., Jin,Q. and Chen,X. (2020) CHG: a systematically integrated database of cancer hallmark genes. *Front. Genet.*, **11**, 11–29.
31. Hammal,F., de Langen,P., Bergon,A., Lopez,F. and Ballester,B. (2022) ReMap 2022: a database of Human, Mouse, Drosophila and Arabidopsis regulatory regions from an integrative analysis of DNA-binding sequencing experiments. *Nucleic Acids Res.*, **50**, D316–D325.
32. Nasir,A., Holzer,T.R., Chen,M., Man,M.Z. and Schade,A.E. (2017) Differential expression of VEGFR2 protein in HER2 positive primary human breast cancer: potential relevance to anti-angiogenic therapies. *Cancer Cell Int.*, **17**, 56.
33. Kumar,R. and Yarmand-Bagheri,R. (2001) The role of HER2 in angiogenesis. *Semin. Oncol.*, **28**, 27–32.
34. Bredikhin,D., Kats,I. and Stegle,O. (2022) MUON: multimodal omics analysis framework. *Genome Biol.*, **23**, 42.

Received: December 12, 2023. **Revised:** April 10, 2024. **Editorial Decision:** April 22, 2024. **Accepted:** April 27, 2024