Chapter 10

Interrogating the Accessible Chromatin Landscape of Eukaryote Genomes Using ATAC-seq

Georgi K. Marinov and Zohar Shipony

Abstract

The ATAC-seq assay has emerged as the most useful, versatile, and widely adaptable method for profiling accessible chromatin regions and tracking the activity of *cis*-regulatory elements (cREs) in eukaryotes. Thanks to its great utility, it is now being applied to map active chromatin in the context of a very wide diversity of biological systems and questions. In the course of these studies, considerable experience working with ATAC-seq data has accumulated and a standard set of computational tasks that need to be carried for most ATAC-seq analyses has emerged. Here, we review and provide examples of common such analytical procedures (including data processing, quality control, peak calling, identifying differentially accessible open chromatin regions, and variable transcription factor (TF) motif accessibility) and discuss recommended optimal practices.

Key words Regulatory elements, Transcription factors, Chromatin accessibility, ATAC-seq, High-throughput sequencing

1 Introduction

In most eukaryotic cells, the genome is packaged by nucleosomal octamer particles comprised of the four core nucleosomal histones H3, H4, H2A, and H2B. Nucleosomes have a refractory effect to transcription and to the binding to DNA by most regulatory proteins. Thus, active *cis*-regulatory elements in eukaryotes tend to be depleted of nucleosomes. This is a highly useful property as it allows for active cREs to be specifically labeled and mapped in a variety of ways, as first recognized decades ago when the hypersensitivity to cleavage by DNase enzymes of enhancer and promoter elements was initially reported [1–3]. DNase hypersensitivity continued to be the primary way of mapping cREs into the genomic era, first, by coupling it to microarrays [4–6], and later to high-throughput massively parallel sequencing [7–9]. Numerous alternative and complementary methods have been also developed in recent years, based on the preferential enzymatic/chemical cleavage/

Noam Shomron (ed.), *Deep Sequencing Data Analysis*, Methods in Molecular Biology, vol. 2243,
https://doi.org/10.1007/978-1-0716-1103-6_10,

modification of accessible DNA. In order to map open chromatin regions in the genome, these assays employ methyltransferases [10–14], restriction enzymes [15], nicking enzymes [16], small molecules [17], viral integration [18], and the preferential insertion into unprotected DNA by transposomes [19].

The latter approach, in the form of the ATAC-seq assay [19], has emerged as the most convenient, versatile, and widely used method for studying the chromatin state of the eukaryotic cell. In an ATAC-seq reaction (Fig. 1), isolated nuclei are subjected to

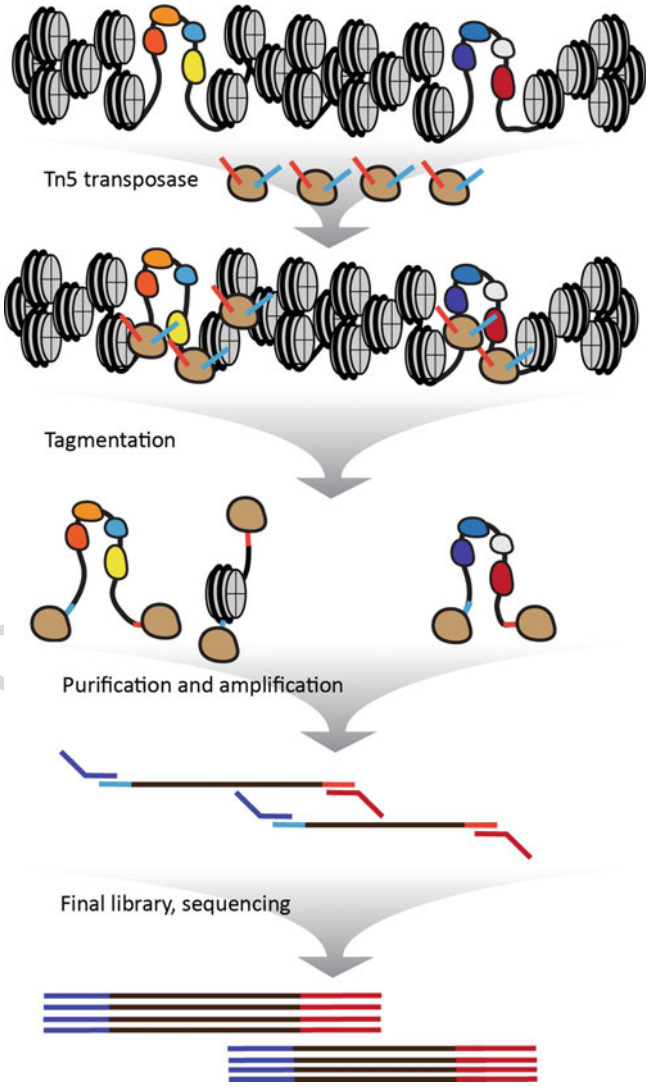

Fig. 1 Overview of the ATAC-seq experimental protocol. Chromatin is subjected to incubation with an active Tn5 transposase carrying adapter sequences that can be directly used for PCR amplification. The transposase preferentially inserts into accessible regions in the genome, such as active cREs. DNA is then purified and PCR amplification is carried out from the adapter sequences deposited by Tn5

treatment with a Tn5 transposase enzyme carrying DNA adapters that are inserted into DNA where DNA is accessible. These adapters then enable the direct amplification of open chromatin fragments, eliminating most of the complex intermediate enzymatic conversion steps that were part of previous approaches such as DNase-seq. This allows for the whole protocol to be completed in just a few hours. It also dramatically lowers the input requirements (50,000 mammalian cells are typically used for an ATAC reaction), including down to single-cell level [20, 21].

Due to these advantages, ATAC-seq has become the method of choice for profiling open chromatin. In the process, a standard set of processing, quality assessment, and downstream analyses practices has begun to emerge (Fig. 2). In this chapter, we review

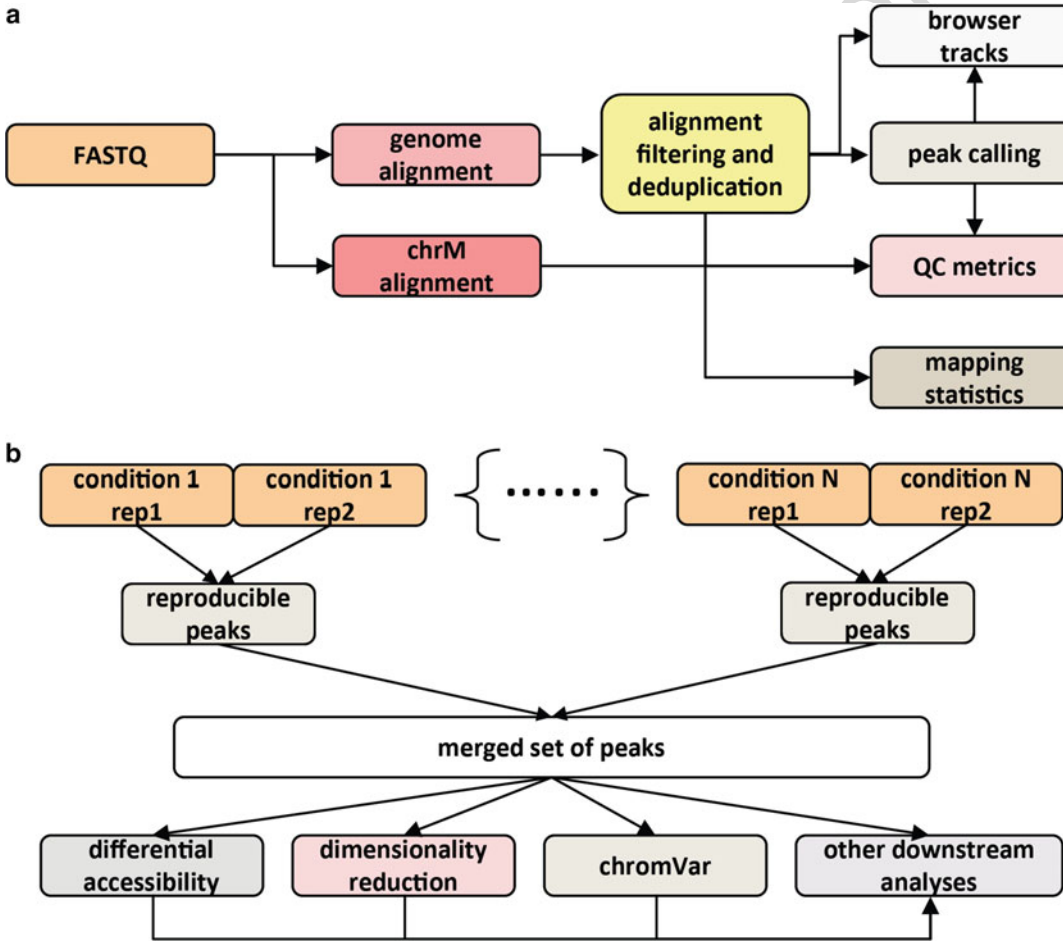

Fig. 2 Overview of a general ATAC-seq analysis workflow. (**a**) Individual samples are aligned both against the nuclear genome and also against the mitochondrial genome (the latter is for quality control purposes as described in the main text), alignments are filtered, and peak calls, browser tracks and mapping statistics and quality metrics are compiled. (**b**) For multiple samples and replicates in a study, reproducible peaks are identified, then combined to derive a unified merged set of peaks. This set of peaks is used to carry out most downstream analyses, including differential accessibility, dimensionality reduction, variable motif accessibility, and others

the optimal approaches to carrying out these tasks, and illustrate their application using publicly available ATAC-seq datasets from the ENCODE Project Consortium [22] as examples.

2 Materials

The analyses described are designed to run on standard Linux systems through the UNIX command line. The maximal memory usage depends on the size of the datasets but is usually less than 15GB.

2.1 Genomic Sequence and Annotation Files

1. A FASTA file containing the GRCh38 version of the human genome can be downloaded from the UCSC Genome Browser at http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz. Genome files can also be obtained from ENSEMBL (http://ensemblgenomes.org/) and from the NCBI website (http://www.ncbi.nlm.nih.gov/assembly/). However, it has to be noted that in the case of the human genome, reference FASTA files available in public repositories contain alternative haplotype contigs, i.e. alternative versions of sequences already present in the assembly. The inclusion of such sequences means that their version in the main chromosomes loses its unique mappability with short reads, and becomes effectively "invisible" to downstream analysis. This is, in almost all cases, an undesirable behavior, thus alternative haplotypes should be removed from reference files before use. The ENCODE Project [22] provides such filtered files from its portal at https://www.encodeproject.org/data-standards/reference-sequences/. For the purposes of ATAC-seq processing, a separate fasta file containing only the mitochondrial genome is also needed; this sequence can be extracted from the whole-genome FASTA file.
2. The same page on the ENCODE Portal also provides "blacklist" regions [23], i.e. locations in the genome that are artifactually enriched in sequencing assays and should be filtered out from peak call sets (discussed further below).
3. Genome annotations in GTF format can be obtained from UCSC, ENSEMBL, NCBI, or ENCODE. For the purposes of ATAC-seq analysis, we prefer to work with the RefSeq annotation, which can be obtained from https://www.ncbi.nlm.nih.gov/projects/genome/guide/human/index.shtml. See discussion in the relevant section below for more details.

2.2 Software Packages

1. `Bowtie` [24] (http://bowtie-bio.sourceforge.net/index.shtml) or `Bowtie2` [25] (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml). Also see the discussion on alignment below for more details.

2. `samtools` [26]: http://www.htslib.org/
3. `PicardTools` https://broadinstitute.github.io/picard/
4. `MACS 2.1.0` [27]: https://github.com/taoliu/MACS/
5. IDR [28] analysis code (version 2.0.4): https://github.com/kundajelab/idr
6. UCSC Genome Browser [29, 30] utilities: http://hgdownload.cse.ucsc.edu/admin/exe/
7. R: https://www.r-project.org/
8. Python (version 2.7 or higher) https://www.python.org/
9. UMAP https://umap-learn.readthedocs.io/en/latest/
10. DESeq2 [31] https://bioconductor.org/packages/release/bioc/html/DESeq2.html
11. chromVAR [32]: https://github.com/GreenleafLab/chromVAR
12. `deepTools` [33]: https://deeptools.readthedocs.io/en/develop/index.html
13. `BEDtools` [34]: https://bedtools.readthedocs.io/en/latest/#
14. featureCounts [35] (from the Subread package): http://subread.sourceforge.net/
15. JASPAR2018 : https://bioconductor.org/packages/release/data/annotation/html/JASPAR2018.html
16. pheatmap : https://cran.r-project.org/web/packages/pheatmap/index.html
17. TFBSTools : https://bioconductor.org/packages/release/bioc/html/TFBSTools.html
18. BSgenome.Hsapiens.UCSC.hg38 : https://bioconductor.org/packages/release/data/annotation/html/BSgenome.Hsapiens.UCSC.hg38.html
19. Additional scripts: https://github.com/georgimarinov/GeorgiScripts. Contains python scripts used in the examples shown below; some of the scripts depend on having `pysam` (https://pysam.readthedocs.io/en/latest/index.html) and `pyBigWig` (https://github.com/deeptools/pyBigWig) installed.
20. TGL Kmeans: https://github.com/tanaylab/tglkmeans

3 Methods

The typical ATAC-seq analysis procedure is outlined in Fig. 2, and can be split in two parts—first, processing, evaluation and analysis at the level of individual samples, then followed by integrative analysis of multiple samples.

Individual sample processing consists of the following steps:

1. Aligning reads against the whole genome.
2. Aligning reads against the mitochondrial genome alone.
3. Filtering poor quality and multireads alignments.
4. Deduplication of alignments.
5. Generation of genome browser tracks.
6. Calculation of mapping statistics and other quality control metrics.
7. Per-replicate/sample peak calling.

Typical multisample analysis tasks include:

1. IDR
2. Merging peaks
3. Dimensionality reduction and data exploration
4. Identifying clusters of accessible regions that behave similarly across samples
5. Identifying regions that are differentially accessible between conditions
6. Analyzing variable motif occupancy

In addition, one might also be interested in examining ATAC-seq footprints around transcription factor binding sites and nucleosome occupancy around particular genomic landmarks.

Procedures and considerations for carrying out these task are described in the subsequent sections.

3.1 Preparation of Genomic Files

1. Download and uncompress genome reference files:

```
wget https://www.encodeproject.org/files/
GRCh38_no_alt_analysis_set_GCA_000001405.15/
@@download/GRCh38_no_alt_analysis_set_GCA_000001405.15.fasta.gz
 -O hg38_no_alt.fasta.gz

 gunzip hg38_no_alt.fasta.gz
```

2. Create a `bowtie` genome index file:

```
mkdir genomes/hg38/bowtie-indexes
cd genomes/hg38/bowtie-indexes
ln ../hg38_no_alt.fa
bowtie-build -f hg38_no_alt.fa hg38_no_alt
```

With `Bowtie2`:

```
mkdir genomes/hg38/bowtie2-indexes
cd genomes/hg38/bowtie2-indexes
ln ../hg38_no_alt.fa
bowtie2-build -f hg38_no_alt.fa hg38_no_alt
```

3. Create a `bowtie` mitogenome ("chrM") index file using only the mitochondrial genome as input:

```
mkdir genomes/hg38/bowtie-indexes
cd genomes/hg38/bowtie-indexes
ln ../chrM.fa
bowtie-build -f chrM.fa chrM
```

With `Bowtie2`:

```
mkdir genomes/hg38/bowtie2-indexes
cd genomes/hg38/bowtie2-indexes
ln ../chrM.fa
bowtie2-build -f chrM.fa chrM
```

4. Create chromosome size info (`chrom.sizes`) files:

```
python makeChromSizesFromFasta.py
              hg38_no_alt.fa hg38_no_alt.chrom.sizes
```

Chromosome size files consist of one line per chromosome in the following format:

```
chr <tab> chromosome_size
```

They identify the end points of chromosomes/contigs and are used at multiple steps in the processing of high-throughput sequencing data.

3.2 Read Mapping

The currently most widely used sequencing platforms are the Illumina NextSeq, HiSeq, MiSeq, and NovaSeq instruments. Sequencing kits sufficient for 75-, 100-, 150-, 300-, and 600-cycle runs are available for these machines in various configurations. In the case of ATAC-seq data, it is strongly recommended that sequencing runs are carried in a paired-end format, first, because the information provided by having two insertion sites per fragment rather than one is helpful for a number of analyses (such as TF footprinting), and second, because fragment distribution is a standard part of the quality assessment of ATAC-seq libraries.

Another point to consider is that while it is often common to see ATAC-seq libraries sequenced as 2×75mers or longer (driven either by the thinking that longer sequences provide better

mapping of short reads, by logistic constraints at sequencing facilities, or by other factors), this is in fact not necessary and only increases the cost of sequencing (by at least twofold). This is because the fragment length distribution of ATAC-seq libraries usually peaks at around 45–50 bp (Fig. 5), and as a result, a majority of fragments are already fully covered by 2×36mer reads.

For these reasons, we carry out all our ATAC-seq sequencing runs as 2×36mers, and we also analyze all ATAC-seq datasets as 2×36mers even if they were sequenced as 2×75mers (or longer), in order to maintain uniformity across all datasets we work with (as longer reads do indeed map uniquely more often than shorter reads when fragments originate from areas of the genome that are not uniquely mappable, the possibility exists for mappability and alignment bias in some samples to generate misleading results if read length is not uniform).

However, under certain circumstances (e.g. when examining allele-biased accessibility or the effect of sequence variants on accessibility) it can be beneficial to use longer reads and to use the full length of fragments. In those cases, reads need to be trimmed of adapters, which can be done using the Trimmomatic [36] or TrimGalore/Cutadapt [37] programs.

Read mapping can in principle be carried out with any of the numerous short read aligners developed over the years, with equivalent results, but two of them—Bowtie2 [25] and BWA [38]—have emerged as the standard tools for carrying this task. In our practice we use primarily Bowtie, as follows.

1. Trim reads (both ends) to 36mers:

```
zcat SAMPLE.end1.fastq.gz | python trimfastq.py - 36 -stdout |
    gzip > SAMPLE.end1.36mers.fastq.gz
zcat SAMPLE.end2.fastq.gz | python trimfastq.py - 36 -stdout |
    gzip > SAMPLE.end2.36mers.fastq.gz
```

2. Map 2×36mer reads to whole genome.

   With Bowtie:

```
python PEFastqToTabDelimited.py
  SAMPLE.end1.36mers.fastq.gz SAMPLE.end2.36mers.fastq.gz |
  bowtie hg38/bowtie-indexes/hg38_no_alt -p 16 -v 2 -k 2 -m 1
  -t --best --strata -q --sam-nh -X 1000 --sam --12 - |
  samtools view -F 4 -bT hg38/sequence/hg38_no_alt.fa - |
  samtools sort - SAMPLE.2x36mers.unique
```

   This retains uniquely mapping read pairs with up to 2 mismatches relative to the reference.

   With Bowtie2:

```
bowtie2 -x hg38/bowtie2-indexes/hg38_no_alt
  -1 SAMPLE.end1.36mers.fastq.gz -2 end2.fastq.gz -p 16 -t
  -X 1000 --no-mixed --no-discordant -
  | samtools -F 1804  view -bT hg38/sequence/hg38_no_alt.fa - |
    samtools sort - SAMPLE.2x36mers.unique
```

This command also filters out all alignments with poor quality and non-unique alignments.

Alignments are stored in the BAM format (a binary version of the SAM format [26]) for all subsequent analyses.

3. Map 2×36mer reads to the mitochondrial genome.

   This step is necessary for the proper counting of mitochondrial reads.

   As ATAC-seq relies on the preferential insertion of Tn5 into accessible DNA, the mitochondrial genome tends to be extremely strongly enriched in ATAC-seq libraries. This is in part because there are hundreds to thousands of copies of it in each mammalian cells, but primarily because it is not packaged by nucleosomes and is thus highly susceptible to transposase insertion. As a result, in early versions of the ATAC-seq protocol mitochondrial reads often constituted the majority of the library. The assay has subsequently been optimized to greatly reduce mitochondrial contamination [39], but estimating the chrM-mapping reads is still a core part of the quality assessment of ATAC libraries.

   The simplest approach to estimating mitochondrial contamination is to calculate the number of alignments mapping to chrM. However, this underestimates the actual number of such reads, and does so to an extent that greatly varies between species and even different assemblies of the same species. This is because of the presence of the so-called NUMTs (**NU**clear **M**i**T**ochondrial sequences) in nuclear genomes due to the still ongoing process of endosymbiotic gene transfer (EGT), in which DNA from the mitochondrion (or from other endosymbionts in eukaryotic cells) is inserted into the nuclear genome [40]. Very recent NUMT insertions have essentially the same sequence as the mitochondrial genome, and as a result make the homologous regions of chrM not uniquely mappable. Depending on the exact content of an assembly, this effect can affect from a minor fraction to almost the entire mitochondrial genome. Examination of chrM unique mappability in different species shows, for example, that up to half of the mouse and nearly the whole *Drosophila melanogaster* mitochondrial genomes are not uniquely mappable with short reads [41].

   For these reasons, if mitomapping reads are to be accurately counted, it is optimal to carry out a separate alignment to

the mitochondrial genome alone, as the great majority of reads mapping to it are expected to derive from the mitochondrion and not from NUMTs (which are chromatinized and individually at most diploid in copy number, compared to the thousands of copies of the mitochondrial genome in the cell). In addition, because there can be sequences that are not uniquely mappable even within the mitochondrial genome itself (this is not the case with mammalian mitogenomes, but does happen quite frequently in the organellar genomes of other lineages), this alignment step can be carried out allowing for multimapping reads.

With Bowtie1:

```
python PEFastqToTabDelimited.py
  SAMPLE.end1.36mers.fastq.gz SAMPLE.end2.36mers.fastq.gz |
  bowtie hg38/bowtie-indexes/chrM -p 16 -v 2 -a
  -t --best --strata -q --sam-nh -X 1000 --sam --12 - |
  samtools view -F 4 -bT hg38/sequence/hg38_no_alt.fa - |
  samtools sort - SAMPLE.2x36mers.chrM
```

With Bowtie2:

```
bowtie2 -x hg38/bowtie2-indexes/chrM
  -1 SAMPLE.end1.36mers.fastq.gz -2 end2.fastq.gz -p 16 -t
  -X 1000 --no-mixed --no-discordant -
  | samtools -F 1804  view -bT hg38/sequence/hg38_no_alt.fa - |
    samtools sort - SAMPLE.2x36mers.chrM
```

4. Index bam files with `samtools`:

```
samtools index SAMPLE.2x36mers.unique.bam
samtools index SAMPLE.2x36mers.chrM.bam
```

3.3 Filtering and Deduplicating Alignments

As it is the nuclear genome that is typically of interest in an ATAC-seq dataset, reads mapping to the mitochondrion represent a confounding factor, as they affect the total library size and normalization factors if retained for downstream analyses. For these reasons, mitochondrial reads are filtered out of BAM files after the initial alignment step, as follows:

1. Filter out mitochondrial reads.

```
samtools view SAMPLE.2x36mers.unique.bam | egrep -v chrM |
samtools view -bT hg38/sequence/hg38_no_alt.fa - -o
SAMPLE.2x36mers.unique.nochrM.bam
```

Note that, depending on the species one is working with, the mitochondrial chromosome need not be named "chrM," need not be a single chromosome (multipartite mitochondrial genomes are found in numerous species [42]), and need not be the only organellar genome to be filtered out (for example, photosynthesizing eukaryotes also have plastids). Change the filtering command accordingly, if necessary.

2. Index the resulting BAM file:

```
samtools index SAMPLE.2x36mers.unique.nochrM.bam
```

3. Remove duplicate alignments.

   As ATAC-seq is typically performed on a very small number of cells (the equivalent of 50,000 mammalian cells), meaning that a limited initial population of original molecules is used as a starting point for library construction, and because it is sequenced in a paired-end format, fragments with exactly the same coordinates are more likely than not to represent PCR duplicates rather than different original fragments. Thus it is a standard step in ATAC-seq processing to remove duplicate fragments. This is carried out using the `MarkDuplicates` program in the `PicardTools` suite, as follows:

```
module load java; java -Xmx4G -jar
picard-tools-1.99/MarkDuplicates.jar
INPUT=SAMPLE.2x36mers.unique.nochrM.bam
OUTPUT=SAMPLE.2x36mers.unique.nochrM.dedup.bam
METRICS_FILE=SAMPLE.2x36mers.unique.nochrM.dedup.metrics
VALIDATION_STRINGENCY=LENIENT
ASSUME_SORTED=true REMOVE_DUPLICATES=true
```

4. Index the resulting BAM file:

```
samtools index SAMPLE.2x36mers.unique.nochrM.dedup.bam
```

3.4 Genome Browser Track Generation

The next step in the processing is to generate genome-wide coverage tracks. Two types of tracks can be generated, a "coverage" track that adds to the score of each base that a mapped fragment covers, and "5′" tracks, which only represent the end points (or "cute sites") of fragments. While the latter type of tracks is also used in the analysis of DNAse-seq, ChIP-exo, and other sequencing-based functional genomic assays, there is a small caveat when working with ATAC-seq datasets—as the transposase itself has a footprint of about 9 bp [19], fragment ends are shifted by 5 bp or 4 bp (depending on which strand they map to) in order to more accurately represent the actual "cut site."

For the purpose of allowing comparison between different samples, it is optimal to normalize the tracks relative to the total set of mapped and dedupped reads, e.g. in RPM (Reads Per Million mapped reads) units.

1. Generate RPM-normalized coverage tracks (using the `bamCoverage` program in `deepTools`):

```
bamCoverage --bam SAMPLE.2x36mers.unique.nochrM.dedup.bam
-o SAMPLE.2x36mers.unique.nochrM.dedup.coverage.bigWig
--binSize 100 --normalizeUsingRPKM --extendReads
--numberOfProcessors {threads}
```

2. Generate RPM-normalized "5′" tracks (using the `alignmentSieve` and `bamCoverage` programs in `deepTools`):

```
alignmentSieve --numberOfProcessors {threads}
--ATACshift --bam SAMPLE.2x36mers.unique.nochrM.dedup.bam
-o tmp.bam

samtools sort -O bam -o
SAMPLE.2x36mers.unique.nochrM.dedup.shifted.bam tmp.bam

samtools index SAMPLE.2x36mers.unique.nochrM.dedup.shifted.bam

bamCoverage --bam
SAMPLE.2x36mers.unique.nochrM.dedup.shifted.bam
-o SAMPLE.2x36mers.unique.nochrM.dedup.shifted.coverage.bigWig
--binSize 100 --normalizeUsingRPKM --extendReads
--numberOfProcessors {threads}

rm tmp.bam
```

Examples of coverage and 5′ tracks are shown for the *IVL* locus in the context of keratinocyte differentiation in Fig. 3.

3.5 Mapping Statistics and ATAC-seq Quality Assessment

How well the experiment worked and whether its properties could negatively affect data analyses and interpretation are critical questions about every high-throughput sequencing library. Quality control (QC) evaluation is therefore an essential step of processing pipelines. Some of these metrics are common across most functional genomic assays, e.g. estimating the molecular complexity of a library (generally, the fewer original molecules are represented in the final library, the worse the dataset is) and calculating read mapping statistics, while others are specific to the nature of the data type at hand.

In addition to these more general QC statistics, several assay-specific metrics are employed when working with ATAC-seq data.

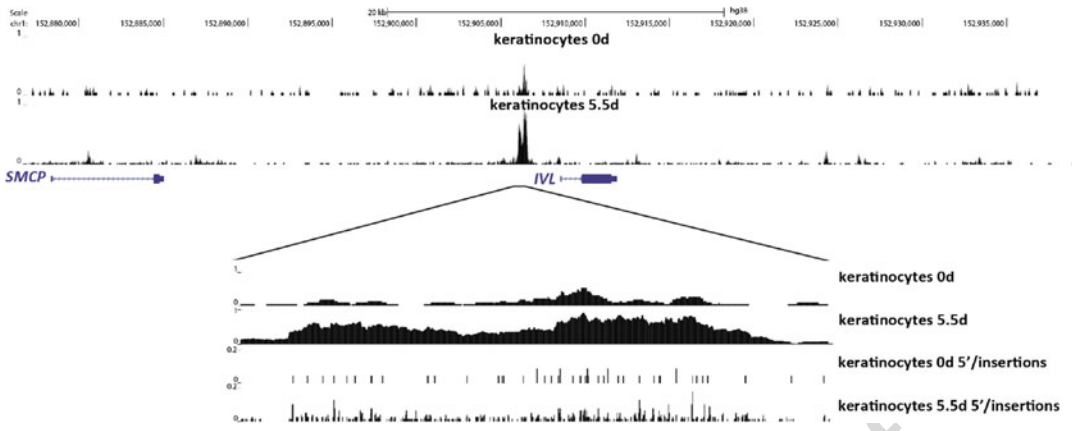

Fig. 3 Example of ATAC-seq coverage and 5′ tracks. Shown are UCSC browser snapshots displaying the changes in chromatin accessibility during keratinocyte differentiation around the *IVL* (involucrin) gene. The lower panel shows both coverage and Tn5 insertion/fragment 5′ tracks

These include the fragment length distribution, the fraction of mitochondrial reads, and transcription start site (TSS) enrichment. The typical goals of ATAC-seq QC are to evaluate the extent of mitochondrial contamination, the fragment length distribution and the molecular complexity of libraries, and, most importantly, how strongly enriched for open chromatin regions an ATAC library is.

1. Count raw reads:

```
zcat SAMPLE.fastq.gz | wc -l
```

   Divide by 4 to get the number of reads (as each read is represented by 4 lines in a FASTQ file).

2. Calculate mapping statistics for the unfiltered BAM file:

```
python SAMstats.py SAMPLE.2x36mers.unique.bam
SAMstats-SAMPLE.2x36mers.unique -bam hg38.chrom.sizes
samtools -paired
```

3. Calculate mapping statistics for the chrM-mapping BAM file:

```
python SAMstats.py SAMPLE.2x36mers.chrM.bam
SAMstats-SAMPLE.2x36mers.chrM -bam hg38.chrom.sizes
samtools -paired
```

   Record the total number of reads mapping to the mitochondrion ($|R_M|$).

4. Calculate mapping statistics for the chrM-filtered pre-deduplication BAM file:

```
python SAMstats.py SAMPLE.2x36mers.unique.nochrM.bam
SAMstats-SAMPLE.2x36mers.unique.nochrM
-bam hg38.chrom.sizes samtools -paired
```

Record the total number of reads mapping to the nuclear genome ($|R_N|$).

5. Calculate mapping statistics for the chrM-filtered post-deduplication BAM file:

```
python SAMstats.py SAMPLE.2x36mers.unique.nochrM.dedup.bam
SAMstats-SAMPLE.2x36mers.unique.nochrM.dedup
-bam hg38.chrom.sizes samtools -paired
```

6. Estimate library complexity and total library size.

Several simple metrics have been used in the literature to characterize the apparent molecular complexity of sequencing libraries [43], such as the Non-Redundant read Fraction *NRF* and the PCR Bottlenecking Coefficients *PBC*1 and *PBC*2, defined as follows:

$$NRF = U_P / U_R \quad (1)$$

Where U_P is the set of genomic positions to which 5′ ends of reads map uniquely and U_R is the total number of uniquely mapped reads.

$$PBC1 = M_1 / M_0 \quad (2)$$

$$PBC2 = M_1 / M_2 \quad (3)$$

Where M_0, M_1, and M_2 are the numbers of distinct genomic locations to which at least one, exactly one, and exactly two reads map uniquely, respectively.

These three metrics should be calculated on the chrM-filtered pre-deduplication BAM file (as the deduplication procedure eliminates redundant reads with the same coordinates).

However, as ATAC-seq is generally carried out from the same amount of starting material, the direct estimation of absolute library size can be a useful metric that can be directly compared across different datasets. Multiple, more or less advanced in their mathematical sophistication approaches have been presented for estimating absolute library complexity (e.g. Preseq [44]). As ground truth is inherently difficult to establish in this case, it is not clear to what extent the estimates that these methods provide are accurate, but we have found

them useful in our practice as rough guides. We estimate absolute library size using the `EstimateLibraryComplexity` program in `PicardTools` as follows:

```
module load java; java -Xmx4G -jar
picard-tools-1.99/EstimateLibraryComplexity.jar
INPUT=SAMPLE.2x36mers.hg38-no-haps.unique.nochrM.bam
OUTPUT=SAMPLE.2x36mers.unique.nochrM.est_lib_complex_metrics.txt
```

Generally, high library size values are desired.

Total library sizes for the example ENCODE datasets used for illustration here are shown in Fig. 4.

7. Calculating the extent of mitochondrial contamination.

   The fraction of mitochondrial reads is calculated according to the following formula:

$$MRF = \frac{|R_M|}{|R_M| + |R_N|} \tag{4}$$

   Where R_M and R_N are as defined above.

   While low *MRF* values are generally desirable, it should be pointed out that a high (though perhaps not extremely high) fraction of mitochondrial contaminants does not directly correspond to low degree of enrichment for open chromatin regions. However, it is a highly useful metric for assessing the performance of the experimental protocol and/or the properties of the cells studied (highly metabolically active cell types, e.g. muscle cells and some cancer cell lines, tend to have many more mitochondria in each cell, and accordingly exhibit higher degrees of mitochondrial contamination in ATAC-seq datasets [41]), which can be used to suggest improvements in the experimental procedures used leading to significant cost savings in terms of sequencing expenditures.

   MRF values for illustrative ENCODE datasets are shown in Fig. 6b.

8. Estimating the fragment length distribution.

   The fragment length distribution of libraries is evaluated based on the chrM-filtered post-deduplication BAM file (including the mitochondrial-mapping fragments can result in misleading results, as mitochondria are not packaged in nucleosomal particles). It is carried out as follows:

```
python PEInsertDistFromBAM.py
SAMPLE.2x36mers.unique.nochrM.dedup.bam hg38.chrom.sizes
SAMPLE.2x36mers.unique.nochrM.dedup.InsLen
-uniqueBAM -normalize
```

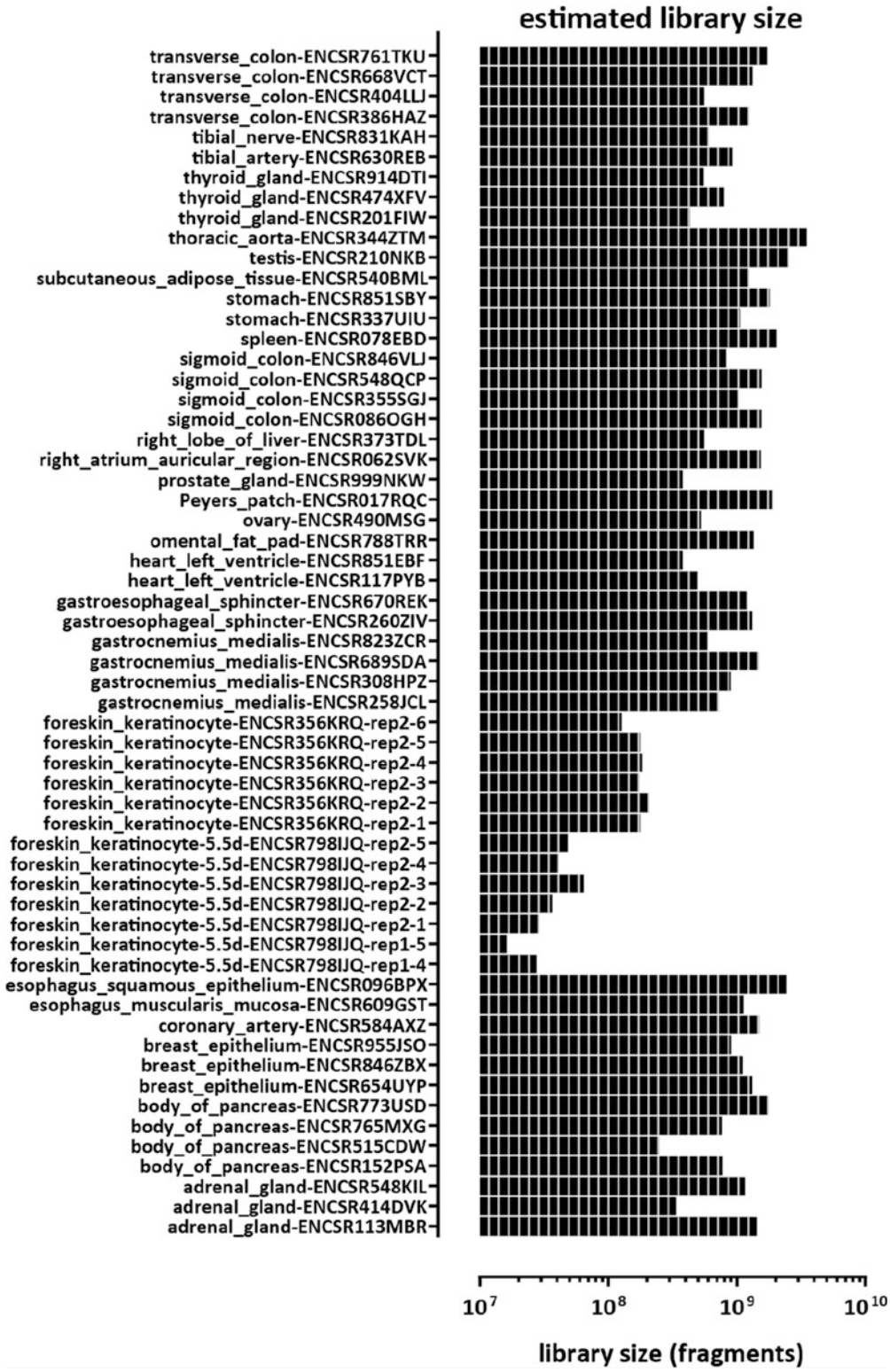

Fig. 4 Estimated absolute library sizes in example ATAC-seq datasets from the ENCODE Project Consortium. Values were calculated using PicardTools

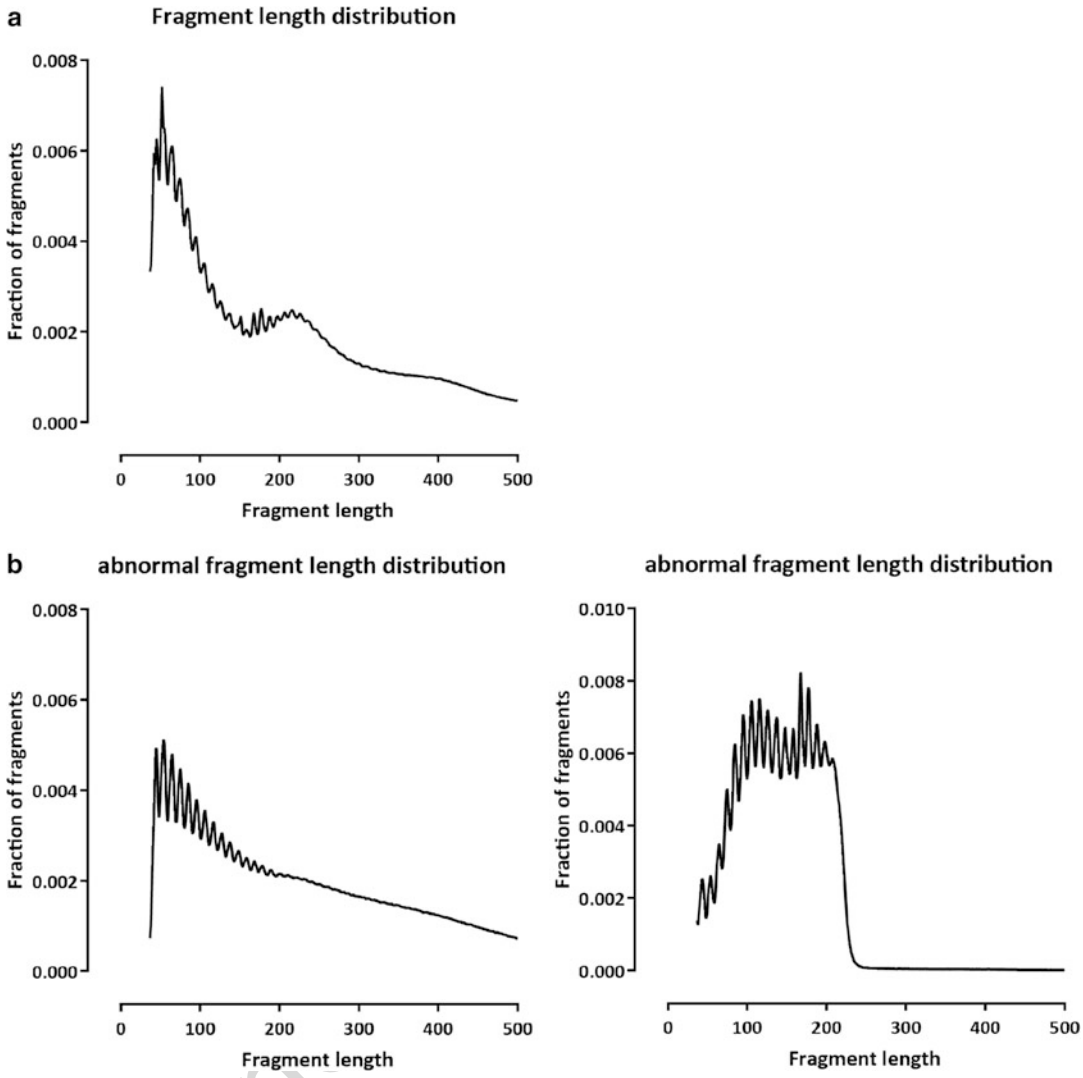

Fig. 5 Fragment length distribution in ATAC-seq datasets. (**a**) Typical ATAC-seq fragment length distributions display a prominent subnucleosomal peak and a peak corresponding to fragments encompassing one nucleosome, as well as a much smaller peak corresponding to dinucleosomal fragments (the example shown is ENCODE accession ID ENCSR404LLJ). (**b**) Examples of abnormal fragment length distributions (ENCODE accession IDs ENCSR031HDN and ENCSR939XWM)

A typical ATAC-seq fragment length distribution (Fig. 5a) is characterized by a prominent subnucleosomal component as well as smaller peaks corresponding to mononucleosomal and dinucleosomal fragments. In addition, a 10-basepair periodicity with a smaller amplitude is overlaid onto this general pattern. It corresponds to the length of the helical turn of DNA in the context of the wrapping of DNA molecules by nucleosomes and other proteins.

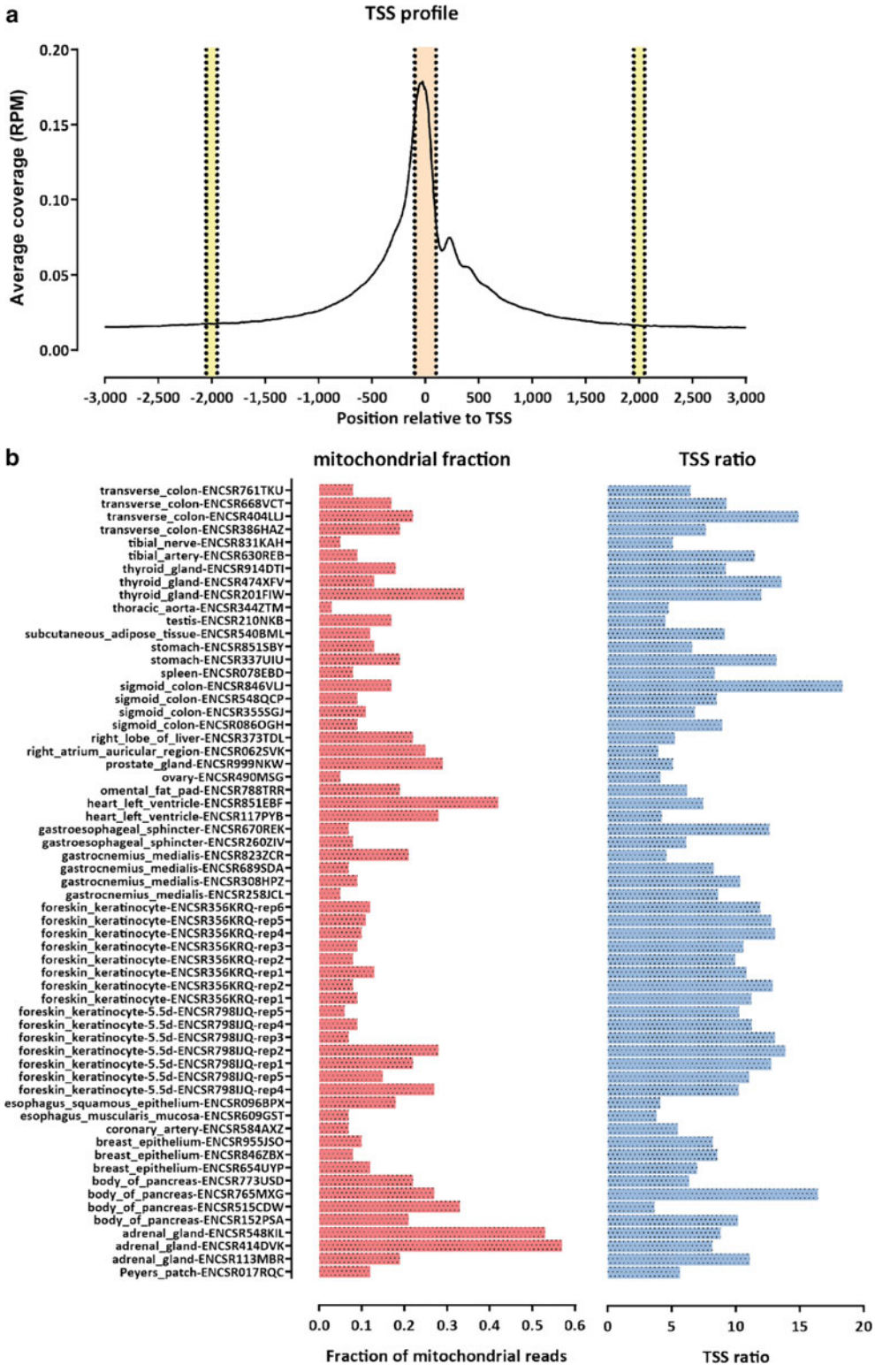

Fig. 6 ATAC-seq quality assessment metrics. (**a**) The TSS ratio quantifies the extent of enrichment in an ATAC-seq library in an internally controlled, independent from peak calling thresholds way. It is calculated by compiling an aggregate TSS profile plot around annotated protein coding TSSs, then dividing the integrated accessibility signal in the 100-bp radius-window around the TSS point to the integrated accessibility signal observed in 100-bp windows at the points ± 2 kbp away from the TSS. (**b**) Mitochondrial read fractions and TSS ratios in selected ENCODE ATAC-seq datasets

Abnormal fragment length distributions can be related to poor enrichment for open chromatin regions, though this is not necessarily always the case. They are, however, a sign of some deviation from standard practices in the experimental protocol and estimating them is thus highly useful for optimizing wet lab procedures. Examples of atypical/abnormal fragment length distributions are shown in Fig. 5b.

9. Evaluating the degree of enrichment for open chromatin.

The most important for downstream analysis QC metrics concern the extent of enrichment for accessible regions of the genome in the library.

The simplest such metric is FRiP [43] (the Fraction of Reads in Peaks), which calculates the fraction of reads in a library that fall within called peaks. However, it depends on the specific thresholds and peak definitions employed by the peak calling algorithm used, which makes it suboptimal for evaluating enrichment across many datasets.

In the context of the ChIP-seq assay, very helpful peak calling-independent metrics for assessing enrichment have been developed based on cross-correlation analysis [43, 45]. However, they are not applicable to ATAC-seq as there are no characteristic strand asymmetry patterns around fixed points in the genome in ATAC-seq datasets.

Instead, the most powerful peak calling-independent enrichment metric for ATAC-seq is TSS enrichment, which is based on creating an aggregate profile curve around the transcription start sites of protein coding genes, and calculating the ratio of the average signal in a small (typically 100-bp radius) window around the TSS versus the combined average signal in the two 100-bp long windows flanking the TSS at a distance of 2 kbp, i.e. as follows (also illustrated in Fig. 6a):

$$TSS_E = \frac{|R \in [TSS \pm 100]|}{\begin{array}{c}|R \in [TSS - 2050, TSS - 1950]| + \\ |R \in [TSS + 1950, TSS + 2050]|\end{array}} \quad (5)$$

Another advantage of the TSS enrichment metric is that it is largely independent of sequencing depth—just a few thousand reads can be used to quite accurately evaluate the enrichment of a given library. Thus it is ideally suited for small initial QC-focused sequencing runs, before commitment to deep sequencing of many libraries, and it can also be applied at the level of individual cells in the context of scATAC-seq.

Several cautionary notes need to be mentioned regarding the metric. As it is calculated relative to annotated TSSs, it is sensitive to the annotation used.

First, highly complex annotations may not be optimal for the purpose of calculating the TSS enrichment as they contain

numerous noncoding transcripts and alternative promoters. More reliable sets of TSSs, including only protein coding genes and few alternative isoforms per gene, are typically the optimal choice.

Second, different species can exhibit widely varying TSS_E scores, depending both on the properties of their genomes and the available annotations. High-quality ATAC-seq datasets in mouse and human exhibit TSS_E scores ≥ 10, which are also the species for which the vast majority of ATAC-seq datasets are generated. The TSS_E scores and the calibrations developed so far are a reliable way to evaluate ATAC libraries from these two organisms. These guidelines are, however, not similarly applicable even for other mammals due to the absence of reliable gene annotations. Very often 5′ UTRs are either incorrectly annotated or completely missing, leading to an artificial depression of the apparent TSS_E scores as there is no proper centering around the accessibility peak at the TSS. Species with smaller, more compact genomes also tend to exhibit lower TSS_E scores (e.g. in the $TSS_E = 2$–3 range of yeast and flies), due to a combination of poor TSS annotation and generally higher and denser transcriptional activity.

Yet when the same species is analyzed with the same annotation, the metric is consistent, robust, and the most reliable way to evaluate the enrichment of an ATAC-seq library. It can also be applied to other open chromatin enrichment assays such as DNase-seq.

As a one-time step, create a TSS 0-radius BED file, as follows (in this case, using the refSeq annotation for the human genome):

```
python TSS_bed_FromGTF.py refSeq.gtf 0 0 refSeq.TSS-0bp.bed
```

For each sample, generate an average profile around TSSs, e.g. as follows, or using equivalent `deepTools` commands:

```
python signalAroundCoordinate-BW.py
refSeq.TSS-0bp.bed 0 1 3 4000
SAMPLE.2x36mers.unique.nochrM.dedup.coverage.bigWig
SAMPLE.2x36mers.unique.nochrM.dedup.coverage.TSS_profile
 -normalize
```

Then calculate TSS_E values:

```
python ATACTSSscore.py
SAMPLE.2x36mers.unique.nochrM.dedup.coverage.TSS_profile
 100 2000 >> ATACTSSscore.txt
```

TSS_E values for illustrative ENCODE datasets are shown in Fig. 6b.

3.6 Peak Calling and Identification of Reproducible Peaks

Once the quality of libraries is ensured, the core analysis steps (Fig. 2b) in a typical ATAC-seq workflow can be carried out. Identifying open chromatin regions/peaks is the first step of that process most of the time. Peak calling tools specifically designed with ATAC-seq in mind are now becoming available [46], but they remain to be thoroughly benchmarked. The MACS2 peak caller [27], originally developed for ChIP-seq datasets, has been the workhorse for ATAC-seq peak calling, with some modifications applied to the settings it is run with.

However, the default output of MACS2 is typically not used as the final set of peak calls. The reason is that it, as is also true for all other peak callers used in isolation, sets arbitrary and not necessarily optimal thresholds to define peak calls. In addition, a properly designed and executed experiment has two or more replicate libraries, and the analysis would ideally incorporate information from these replicates to identify robustly reproducible peaks.

In order to accomplish these goals, the IDR (Irreproducible Discovery Rate) framework has been developed, as part of the efforts of the ENCODE Project Consortium [28]. The objective of IDR analysis is to identify and separate the set of peaks that are reproducible between replicates from those that are not reproducible using the peak ranks (according to any metric suitable for the purpose of ranking peaks). This operation can be carried out in isolation on any two sets of peaks, but the actual procedure used to derive a final set of peaks involves running several different IDR comparison steps, as described below (Fig. 7).

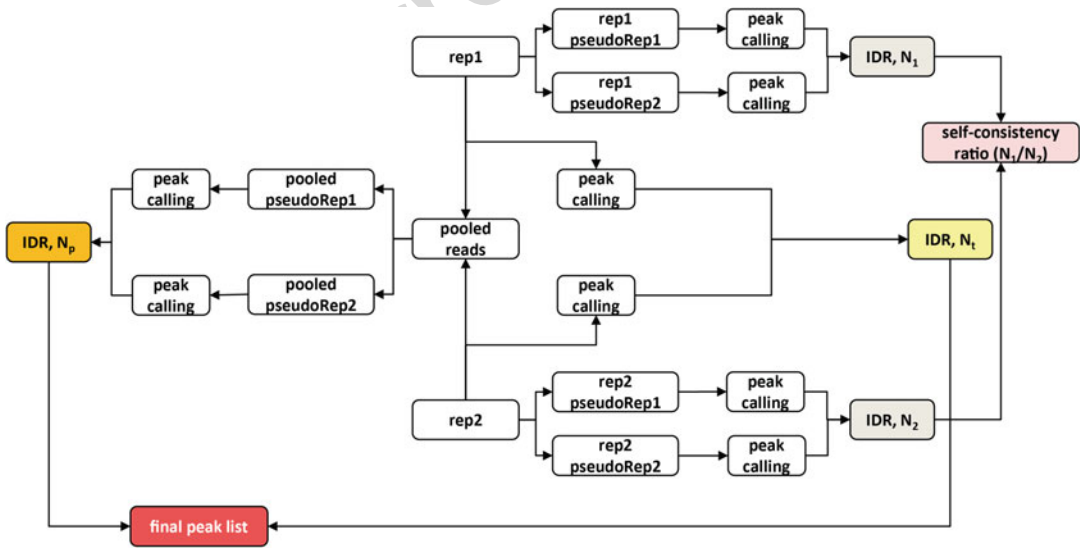

Fig. 7 Overview of the IDR analysis procedure. IDR is run both on pooled pseudoreplicates and on individual replicates, and these runs are used to identify the final set of reproducible across replicates peaks. IDR is also run on individual pseudoreplicates in order to identify discrepancies between the individual replicates used as input

An important caveat is that for IDR to work correctly, the algorithm needs to be presented with sufficiently large samples of both the reproducible and the irreproducible components of the peaks [43]. For this reason, peak calls used as input to IDR are generated with extremely relaxed peak calling settings (in order to allow irreproducible peaks into the peak call set).

Starting from two replicate ATAC-seq libraries, the standard IDR procedure for identifying reproducible peaks can be summarized as follows (Fig. 7). First, peaks are called with relaxed settings on each of the individual replicates and IDR is run to derive a "true replicate" set of peaks N_t. Then reads from the two replicates are pooled and "pseudoreplicate" BAM files are created by randomly dividing the pooled set of reads into two halves. Peaks are called with relaxed settings on each of the two pseudoreplicates, then IDR is run to derive a set of peaks N_p reproducible between pseudoreplicates. Peaks are also called with relaxed settings on the pooled set of reads, and these peak calls are used to extract the final set of peaks, typically by taking the top $max(N_t, N_p)$ peaks, though one can also use the usually more conservative N_t cutoff.

The use of pseudoreplicates allows one to compensate for cases when there are significant discrepancies in the sequencing depth and/or quality of one of the replicates. It also enables a peak caller threshold-independent approach to calling peaks at the level of an individual replicate, by splitting it into individual pseudoreplicates and identifying peaks that are reproducible between them. This step is a standard part of the IDR pipeline where it is used as a quality control measure to identify discordant sets of replicates (i.e. where there are significant differences in the N_1 and N_2 individual pseudoreplicate peak call sets).

3.6.1 IDR Analysis Pipeline

1. Run MACS2 on individual replicates:

```
macs2 callpeak -t Rep1.unique.nochrM.dedup.bam -n Rep1.MACS2
-g hs -f BAMPE --to-large -p 1e-1 --keep-dup all --nomodel
macs2 callpeak -t Rep2.unique.nochrM.dedup.bam -n Rep2.MACS2
-g hs -f BAMPE --to-large -p 1e-1 --keep-dup all --nomodel
```

2. Merge BAM files for the individual replicates:

```
samtools merge Rep1Rep2.pooled.bam
               Rep1.unique.nochrM.dedup.bam
               Rep2.unique.nochrM.dedup.bam
```

3. Sort the merged BAM files:

```
samtools sort Rep1Rep2.pooled.bam Rep1Rep2.pooled.sorted.bam
```

4. Generate pseudoreplicates for the pooled replicates:

```
python BAMPseudoReps.py Rep1Rep2.pooled.sorted.bam
       samtools hg38/sequence/hg38_no_alt.fa
```

5. Generate pseudoreplicates for each individual replicate:

```
python BAMPseudoReps.py Rep1.unique.nochrM.dedup.bam
       samtools hg38/sequence/hg38_no_alt.fa
python BAMPseudoReps.py Rep2.unique.nochrM.dedup.bam
       samtools hg38/sequence/hg38_no_alt.fa
```

6. Call peaks on the pooled dataset as previously shown.
7. Call peaks on the pooled pseudoreplicates as previously shown.
8. Call peaks on individual pseudoreplicates as previously shown.
9. Take the top 300,000 of each of the relaxed peak call sets using the MACS2 *p*-value as a ranking measure, e.g. for replicate 1:

```
cat Rep1.MACS2_peaks.narrowPeak | sort -k 8nr,8nr |
   awk 'BEGIN{OFS="\t"}{$4="Peak_"NR ; print $0}' |
   head -300000 | gzip -c >
   Rep1.MACS2_peaks.narrowPeak.sorted.300K.gz
```

10. Run IDR on individual replicates:

```
idr-2.0.4.2/bin/idr --samples
Rep1.MACS-2.1.0.p1e-1_peaks.narrowPeak.sorted.300K.gz
Rep2.MACS-2.1.0.p1e-1_peaks.narrowPeak.sorted.300K.gz
--input-file-type narrowPeak --output-file Rep1Rep2.indRep.
IDR
--peak-list Rep1Rep2.pooled.sorted.narrowPeak.sorted.300K.gz
--rank p.value --soft-idr-threshold 0.05 --plot
```

11. Run IDR on pooled pseudoreplicates as above.
12. Run IDR on individual pseudoreplicates as above.

Example of IDR analysis output is shown in Fig. 8.

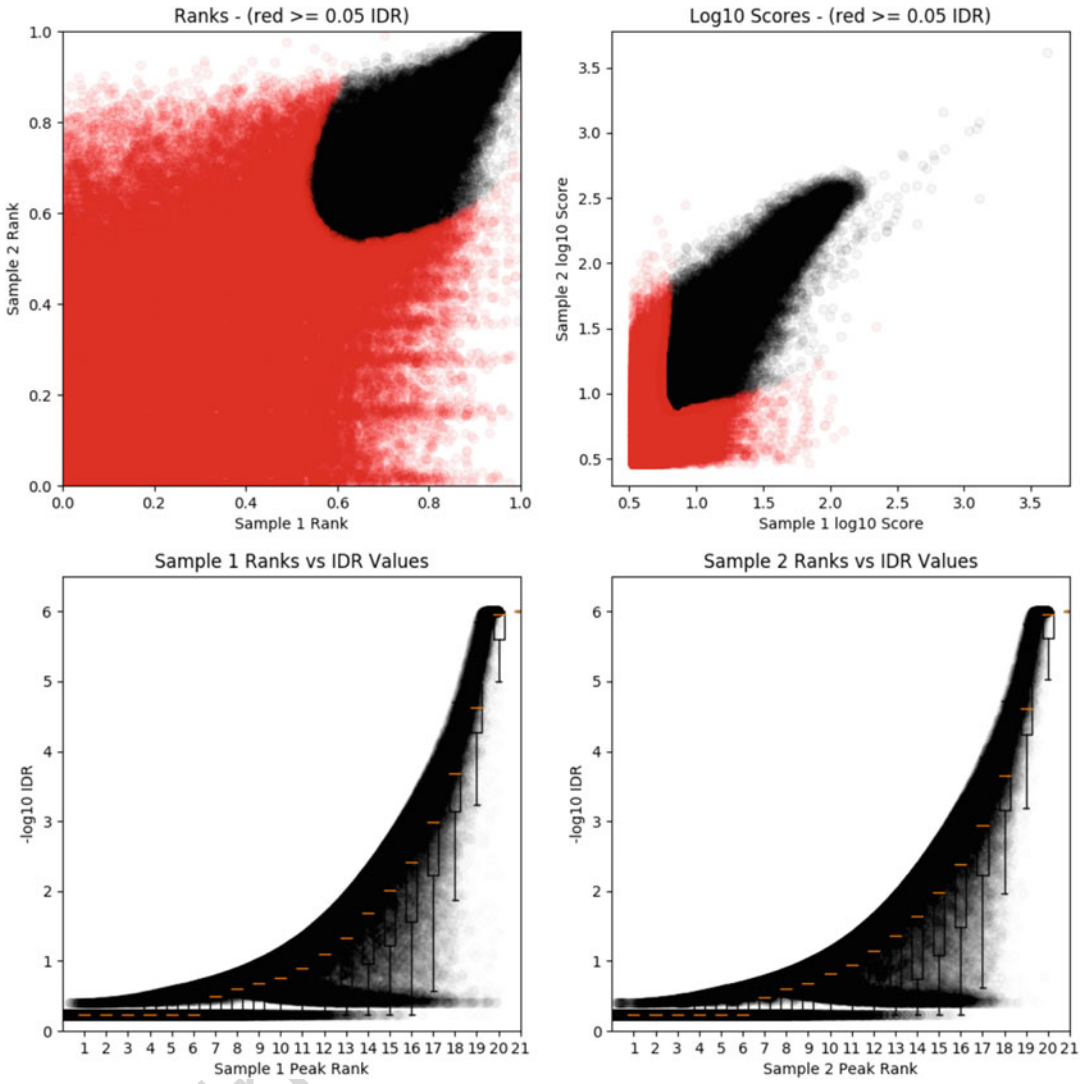

Fig. 8 Example IDR analysis output showing peak reproducibility as a functioning of peak ranks. A true replicate comparison for ENCODE dataset ENCSR260ZIV is shown

13. Examine the IDR output files to determine the N_t, N_p, N_1, N_2 values, e.g. for N_t:

```
awk '$11 <= 0.02 {print $0}' Rep1Rep2.indRep.IDR | wc -l
```

The examples provided here use an individual replicate IDR cutoff of 0.05 which works reasonably well for most purposes. If more stringent peaks are wanted, individual replicate thresholds of 0.02 or 0.01 can also be applied.

14. Pick the top $max(N_t, N_p)$ peaks from the peak calls generated from the pooled replicates:

```
zcat Rep1Rep2.pooled.sorted.narrowPeak.sorted.300K.gz |
  head -n $max(N_t,N_p) | cat > Rep1Rep2.IDR0.05.bed
```

If more than two biological replicate samples are available, IDR can be run between all pairs of replicates to determine the maximum set of reproducible peaks.

3.6.2 Removing Known Artifacts

Most genome assemblies include problematic regions that generate strong artifactual enrichment in most sequencing-based assays, e.g. due to the presence of collapsed repeats that are included as a single copy in the assembly but exist in numerous copies in real cells. In order to avoid biases introduced by including such regions in downstream analyses, it is necessary to exclude these so-called blacklist regions. Blacklist sets have been generated for multiple species by the ENCODE Project Consortium [23], and are readily available from its portal.

The post-IDR set of peaks is filtered against blacklists as follows:

```
bedtools intersect -v -a Rep1Rep2.IDR0.05.bed
-b hg38_blacklist.bed -wa > Rep1Rep2.IDR0.05.noBlacklist.bed
```

3.7 Merging Peaks and Creating Multisample Data Matrices

The typical next step in the analysis workflow is to create a data matrix that combines all samples that are to be analyzed together. To this end, the regions derived from the peak calling procedure in individual samples/conditions, each of which has different coordinates even when overlapping a peak from another sample/condition, need to be merged together so that only one set of coordinates remains for each location of enrichment in the genome. There is no consensus strategy for merging peaks and multiple approaches have been used with generally equal success. For ATAC-seq, perhaps the simplest strategy is to split the genome in bins, e.g. 500 bp each, and to only retain bins that overlap called peaks. Another strategy is to iteratively merge peaks if their summits are within a specified distance from each other, with a new summit corresponding to the summit of the stronger of the two peaks; the final list of peaks is derived by extending the resulting final list of summits by a fixed distance.

Each of these procedures results in a uniform set of coordinates, which are used to compile a final data matrix. Such a matrix can contain read counts, RPM values, or some other measure.

3.8 Identifying Differentially Accessible Regions

A common task in ATAC-seq analysis is to identify regions that are differentially accessible between two conditions. Standard tools originally developed for the analysis of RNA-seq datasets, such as DESeq2 [31], edgeR [47], limma [48], and others can be used for this purpose. Most such packages require read counts as inputs, as the statistical models that they employ are based on discrete distributions (e.g. negative binomial in the case of DESeq/edgeR). The major difference between differential analysis in the context of RNA-seq and its application to ATAC-seq is that stable static gene annotations are used in the former case, while for ATAC-seq no such set of features is available, and it has to be compiled through the peak merging procedures discussed in the previous section.

Once peaks have been merged, read counts can be generated for all samples combined as follows:

```
featureCounts -F SAF -a Merged_peaks.saf -o All_samples.counts
Sample1.bam....SampleN.bam
```

Note: when using `featureCount`, a file in `.saf` format is needed, i.e. coordinates are specified as follows:

```
ID <tab> chr <tab> start  <tab> end
```

Once a read count matrix has been compiled, it can be used as input to DESeq/DESeq2. While DESeq2 allows for arbitrarily complex design matrices to be employed for differential analysis, here we illustrate its use with a simple two-condition two-replicates-per-condition use case.

Within `R`, we first invoke the DESeq2 library:

```
library("DESeq2")
```

Then the count matrix is converted into an `R` object:

```
pasillaCountTable <- read.table("samples.counts.table",
                                header=TRUE, row.names=1)
samples <- data.frame(row.names=c(
           "condition1-rep1", "condition1-rep2",
           "condition2-rep1", "condition2-rep2"),
           condition=c("condition1", "condition2"))
```

After that, mean and dispersion estimates for the two conditions are derived:

```
dds <- DESeqDataSetFromMatrix(countData = pasillaCountTable,
                       colData=samples, design=~condition)
dds_1 <- DESeq(dds)
```

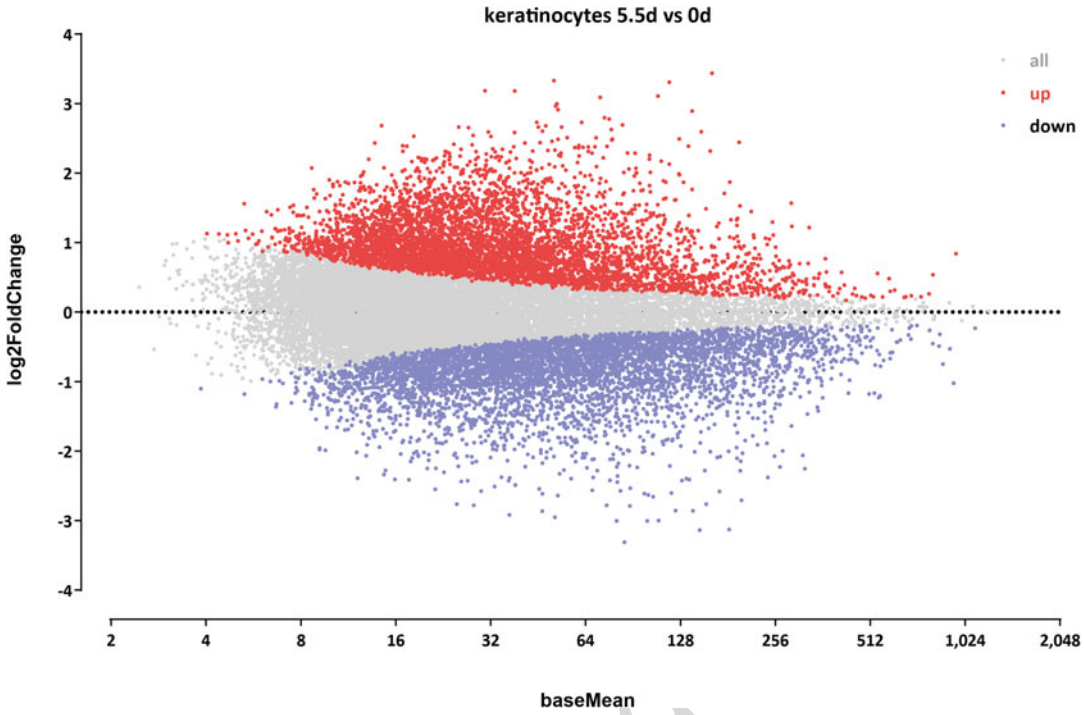

Fig. 9 MA plot showing an example of differential accessibility analysis results. Differentially accessible regions were identified between the 0-day and the 5.5-day time points of a skin differentiation times course (ENCODE accessions ENCSR798IJQ and ENCSR356KRQ)

Finally, differential fold change and significance values are generated for downstream filtering and analysis:

```
res <- results(dds_1, contrast=c(
            "condition", "condition1", "condition2" ))
write.table(res, "samples.condition1-vs-condition2.csv")
```

An example of the results from differential accessibility analysis is shown in Figs. 9 and 10 using the ENCODE keratinocyte differentiation time course ATAC-seq dataset.

3.9 Visualizing Signal Around Genomic Features Using Heatmaps

We use the case of differential accessibility to demonstrate another common task encountered in the analysis of ATAC-seq (and other functional genomic data), visualizing ATAC signal around a set of genomic features, often across multiple datasets.

In this case, we start with modified DESeq output files in the following format, one each for the set of "up" and "down" peaks:

```
#chr left right baseMean log2FoldChange
```

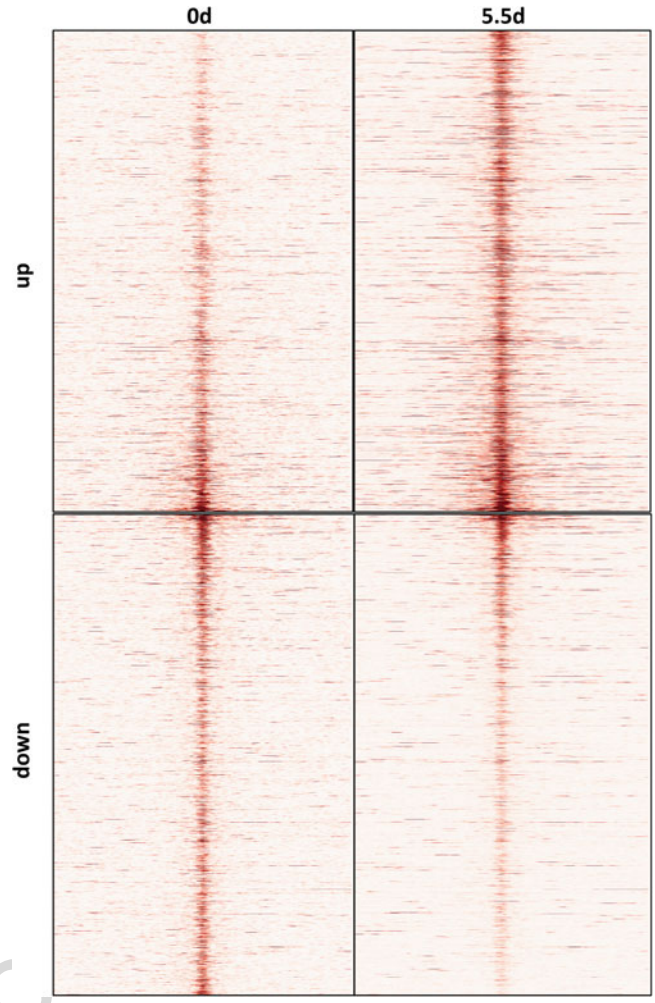

Fig. 10 Heatmaps of normalized ATAC-seq signal for differentially accessible regions during keratinocyte differentiation. Differentially accessible regions were identified between the 0-day and the 5.5-day time points of a skin differentiation times course (ENCODE accessions ENCSR798IJQ and ENCSR356KRQ). Regions were sorted according to their fold change values across the two conditions as estimated by DESeq2

We calculate a coverage matrix containing the ±2 kb profile around the midpoint of each differentially accessible peak (sorting the matrix by the $\log_2$(FoldChange) value):

```
python signalAroundCoordinate-individual.py
   foreskin_keratinocyte.0d-vs-5.5d.up.bed
   0 midPoint -noStrand 2000 2000
     foreskin_keratinocyte-5.5.2x36mers.unique.nochrM.dedup.
bigWig
   foreskin_keratinocyte.0d-vs-5.5d.up.5.5d.matrix -sortby 4
```

We also generate the same matrices for the "down" set of peaks and for the "0d" condition. The resulting matrices can then be visualized (Fig. 10) using a number of tools (`R`, `matplotlib`, and others). A similar procedure can also be implemented using functions from the `deepTools` package.

3.10 Data Exploration

In order to examine the relationship between samples in a high-dimensional dataset, it is usually useful to apply a dimensionality reduction technique that performs a mapping of the data onto a lower-dimension space (e.g. one with two or three dimensions) that is possible to visualize.

The most well-known such technique is PCA (Principle Component Analysis), which carries out a linear transformation of the data in such a way that the variance along each of the principle components identified is maximized in a decreasing order. Data points are then projected along some combination of principle components in a low-dimensional space (most often, for data exploratory purposes, the first and the second).

A variety of PCA implementations exist in all statistical and programming languages. A PCA analysis can be carried out in `R` as follows:

```
library(ggfortify)
library(tidyverse)
rpm_fixed = read_tsv("samples.RPM.table", skip = 1)
pca = prcomp((as.matrix(rpm_fixed))

autoplot(pca, x = 1, y = 2)
```

Figure 11 shows the first two PCA projections for ENCODE ATAC-seq datasets.

With the advent of single-cell genomic methods (such as scRNA-seq and scATAC-seq), datasets of increasingly high dimensionality and complex structure have become available, for which simple dimensionality reduction methods such as PCA are ill suited. Instead, novel methods have been employed (e.g. *t*-SNE, or *t*-distributed Stochastic Neighbor Embedding, [49]), or specifically developed for the purpose of working with such datasets (e.g. UMAP, or Uniform Manifold Approximation and Projection for Dimension Reduction, [50]).

These techniques can also be applied to bulk ATAC-seq datasets. The main tool the field used for several years was *t*-SNE. However, it has now been largely replaced by UMAP as field standard, as UMAP is much faster and less memory intensive, and, most importantly, it preserves global data structure and relationships between data points, unlike *t*-SNE.

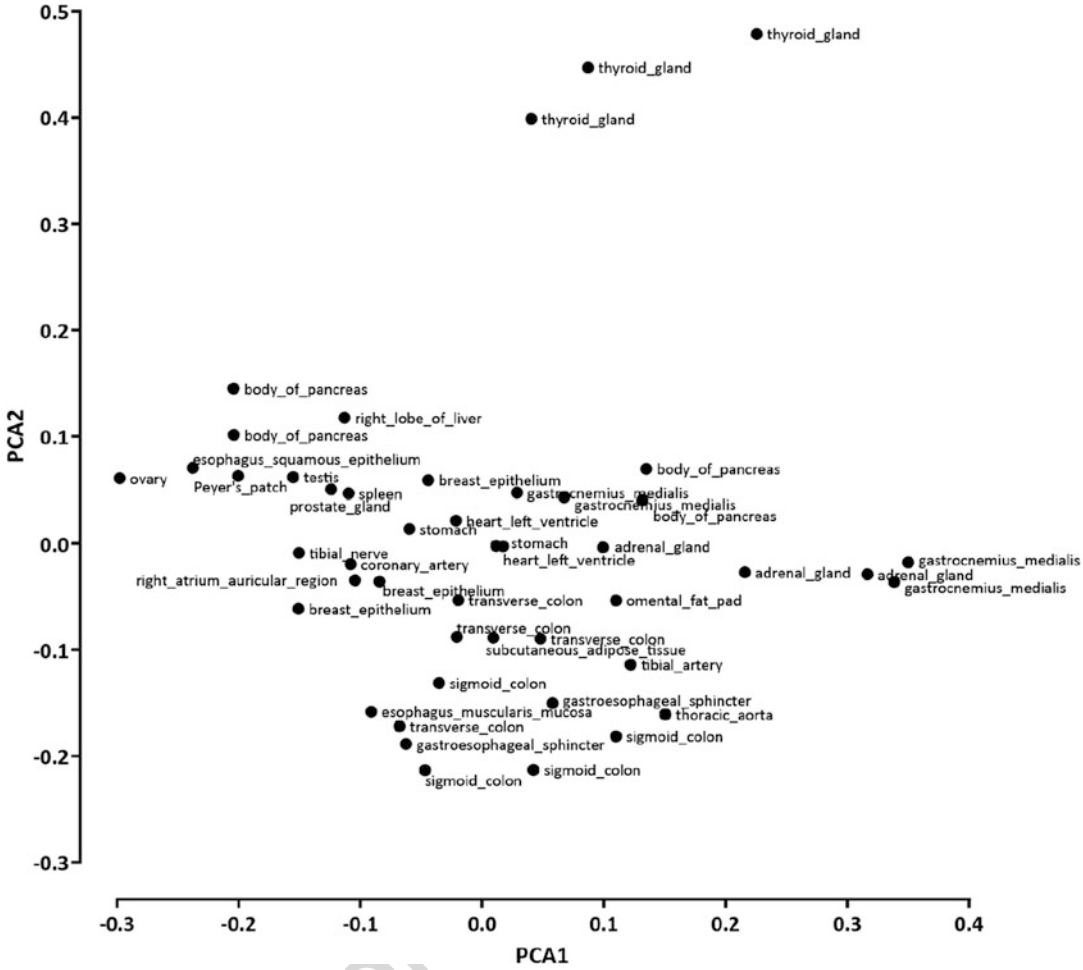

Fig. 11 First two PCA projections for human ENCODE ATAC-seq datasets

UMAP can be, in its simplest form, run from inside Python as follows:

First, UMAP and `numpy` are loaded:

```
import numpy as np
import umap
```

Second, a `numpy` array object containing the ATAC matrix is created.

Then UMAP is run:

```
reducer = umap.UMAP()
embedding = reducer.fit_transform(data)
```

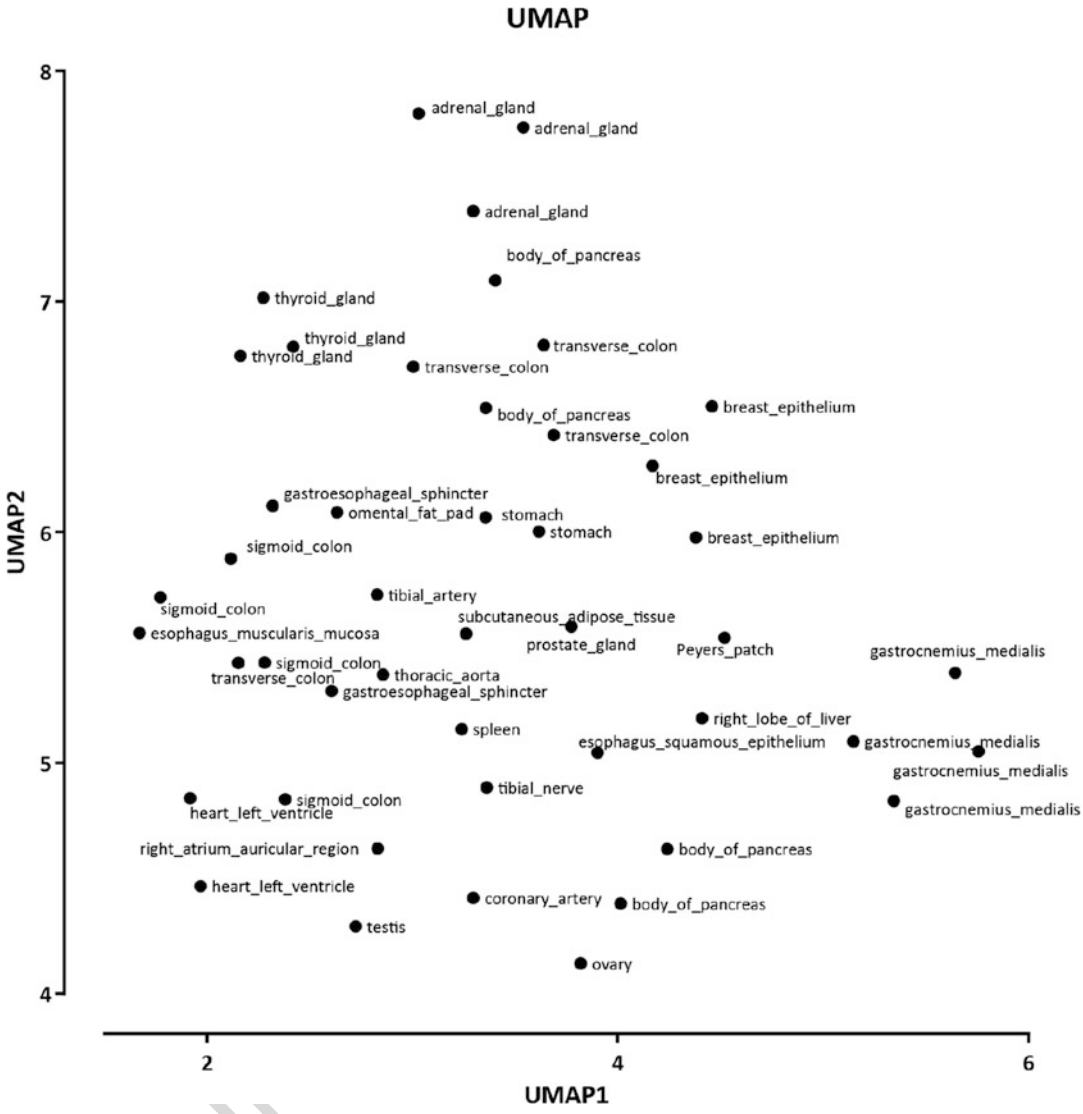

Fig. 12 UMAP projections for human ENCODE ATAC-seq datasets

The `embedding` object contains the projections along the first two UMAP dimensions. It should be noted that while default settings often work well, UMAP output depends on several hyper-parameters; for a more detailed discussion of these the reader is referred to the online UMAP documentation.

UMAP projections for ENCODE ATAC-seq datasets are shown in Fig. 12.

3.11 Clustering of Accessible Regions Across Conditions/Cell Types

Clustering of ATAC-seq peaks across cell lines, tissues, or times can identify sets of genomic regions exhibiting unique biologically relevant dynamic accessibility patterns. Subsequent analyses can then focus more deeply on such subsets of peaks, e.g. by analyzing their sequence properties.

It is not recommended to perform clustering on all peaks, as most of them typically show no differences between conditions. Filtering peaks and retaining the most variable ones are thus usually performed prior to clustering. One approach for doing this is to take peaks that are called as differential by DESeq2/edgeR between different conditions.

Before clustering the differential peaks, it is recommended to normalize the read counts, e.g. as $\log_2$(Reads/Counts Per Million), both for visualization purposes and in order to correct for differences in sequencing depth.

From within R:

```
library(viridis)
library(tidyverse)
library(tglkmeans)

res = read_csv("samples.condition1-vs-condition2.csv")
rpm_fixed = read_tsv("samples.RPM.table")

rpm_filter = rpm_fixed[order(res$padj),
                       4:dim(rpm_fixed)[2]][1:35000]
rpm_filter = log2(rpm_filter+1)
```

Next we cluster using TGL Kmeans. Note that the number of clusters *K* to be used has to be decided *a priori*, and usually there is no prior reliable guess of what it should be. One possibility is to determine *K* using the elbow method:[1]

```
km = TGL_kmeans(rpm_filter,12,"euclid",reorder_func =
                "hclust",id_column=FALSE)

image(t(as.matrix(rpm_filter[order(km$cluster),])),,xaxt="n",
           yaxt="n",col=viridis(12),zlim=c(0,8),main="TITLE")

cur_y = 0
tot_y = 1
for (i in 1:K) {
    cur_y = cur_y + km$size[i]/sum(km$size)
```

[1] https://www.r-bloggers.com/finding-optimal-number-of-clusters/.

```
    abline(a=cur_y, b=0, col="yellow",lwd=2)
    mtext(i,side=2,line=1,at=(cur_y/tot_y)-
    ((km$size[i]/sum(km$size))/2),adj=1,cex=2,las=2)
}
```

The results from the clustering above is shown in Fig. 13.

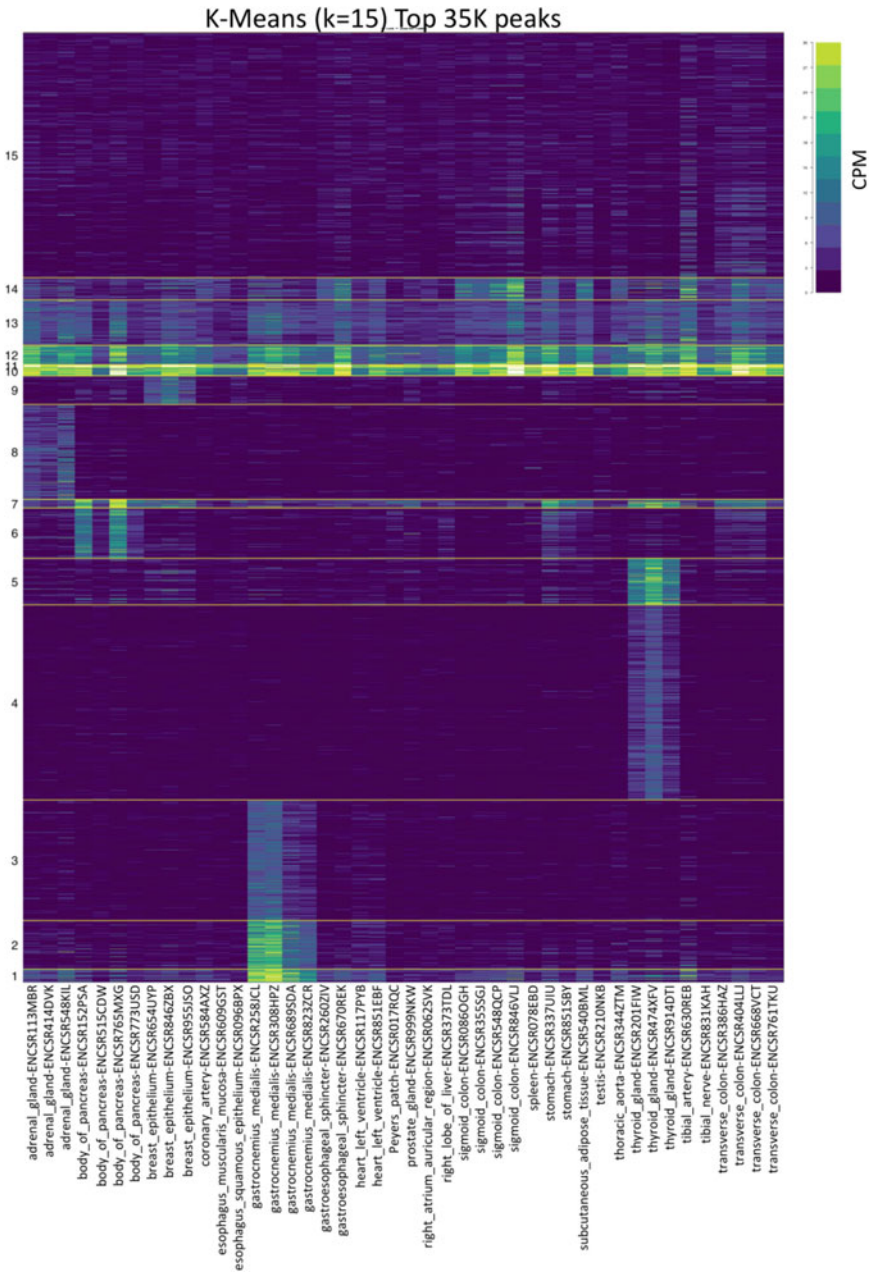

Fig. 13 K-means clustering analysis applied to human ENCODE ATAC-seq datasets. K-means clustering ($k = 15$) was carried out on the top 35,000 most variable peaks across the human ENCODE ATAC-seq datasets

3.12 Analyzing Variable Motif Accessibility Using chromVAR

Accessibility peaks are defined by the regulatory input of transcription factors, and therefore there is a relationship between the motif content of open chromatin regions and active regulatory factors in a given cell or cell type. This relationship can be captured by analyzing variable accessibility associated with transcription factor motifs. The chromVAR algorithm and R package were created to provide that functionality [32]. It works by taking as input a set of accessibility peaks and read counts for each peak across all samples in a dataset together with a collection of mapped TF motif instances. The chromVAR algorithm then identifies "accessibility deviations" for each TF and cell/cell type after correcting for a number of biases (such as variable tagmentation, GC content, mean accessibility, and others).

Within R:

```
library(chromVAR)
library(motifmatchr)
library(JASPAR2018)
library(BSgenome.Hsapiens.UCSC.hg38)
library(TFBSTools)

opts <- list()
opts["species"] <- "Homo sapiens"
opts["collection"] <- "CORE"
motifs <- TFBSTools::getMatrixSet(JASPAR2018::JASPAR2018,
opts)
if (!isTRUE(all.equal(TFBSTools::name(motifs), names(mo-
tifs))))
names(motifs) <- paste(names(motifs), TFBSTools::name(mo-
tifs),
                                                    sep = "_")

genome = "BSgenome.Hsapiens.UCSC.hg38"
tf_num = 100

rowRanges = makeGRangesFromDataFrame(rpm_fixed[,1:3])
counts = as.matrix(rpm_fixed[,4:dim(rpm_fixed)[2]])
colData = DataFrame(Tissue = names(rpm_fixed),
                            row.names=colnames(counts))

rse <- SummarizedExperiment(assays=SimpleList(counts=counts),
                        rowRanges=rowRanges, colData=colData)

counts_filtered <- addGCBias(rse, genome = genome)
motif_ix <- matchMotifs(motifs, counts_filtered, genome =
genome)

dev_TF <- computeDeviations(object = counts_filtered,
```

```
                                annotations = motif_ix)
bg <- getBackgroundPeaks(object = counts_filtered)

expected <- computeExpectations(counts_filtered)
variability_TF <- computeVariability(dev_TF)
variabilty_plot_TF <- plotVariability(variability_TF,
                                      use_plotly = FALSE,n=8)

pdf("variability_plot_TF.pdf")
print(variabilty_plot_TF)
dev.off()

sample_cor_TF <- getSampleCorrelation(dev_TF)
dist_TF <- as.matrix(as.dist(sample_cor_TF))
TF_colData <- colData(dev_TF)
TF_colNames <- TF_colData$Treatment
colnames(dist_TF) <- TF_colNames
rownames(dist_TF) <- TF_colNames

pheatmap(dist_TF, clustering_distance_rows =
     as.dist(1-sample_cor_TF), clustering_distance_cols =
   as.dist(1-sample_cor_TF), filename = "Heatmap_TF_colnames.
pdf",
         height=10,width=10)

Highest_var_TF <- cbind(variability_TF, rownames(variabili-
ty_TF))
Highest_var_TF <- Highest_var_TF %>% group_by(name) %>%
                        dplyr::slice(which.max(variability))
Highest_var_TF <- Highest_var_TF[order(Highest_var_TF$varia-
bility,
                                decreasing = TRUE),]
Highest_var_TF <- Highest_var_TF[c(1:tf_num),]
Dev_Highest_var_TF <- assays(dev_TF)[[1]]
Dev_Highest_var_TF <- as.data.frame(Dev_Highest_var_TF)
Dev_Highest_var_TF <- cbind(rownames(variability_TF),
                                        variability_TF$name,
Dev_Highest_var_TF)
Dev_Highest_var_TF <- (subset(Dev_Highest_var_TF,
      Dev_Highest_var_TF$`rownames(variability_TF)` %in%
      Highest_var_TF$`rownames(variability_TF)`))
rownames(Dev_Highest_var_TF) <-
     Dev_Highest_var_TF$`variability_TF$name`
Dev_Highest_var_TF <- Dev_Highest_var_TF[,c(3:(dim(data)[2]
+2))]
pheatmap(Dev_Highest_var_TF, cluster_rows = T,
     cluster_cols = FALSE, scale = "row", filename =
     "Top100_TF_Heatmap.pdf",height=10,width=10)
```

The result is a map of transcription factor activity in each cell/cell type, as shown for ENCODE ATAC-seq datasets in Figs. 14 and 15.

3.13 Analyzing Transcription Factor Footprints

While not as strongly protective against enzymatic action as nucleosome occupancy, occupancy of DNA by other factors can preclude cleavage/modification too, resulting in protection "footprints." The physical association of individual transcription factor molecules with DNA lasts on the order of seconds for most factors (e.g. [51]), thus some skepticism regarding their ability to provide protection against enzymatic action is not unwarranted. Nevertheless, empirically such footprints are indeed observed [52–55], and their analysis is potentially tremendously powerful.

Two types of footprinting analysis can be carried out—globally and at the level of individual footprints. The former involves aggregate analysis across motif instances (often restricted to those within accessible regions of the genome), the latter aims at identifying individual footprints genome-wide.

Individual footprinting has been attempted by multiple studies using very deep DNase-seq data [52–55], and understandably, given its potential to map regulatory circuits, the computational problem of identifying such footprints has generated much interest by the bioinformatic community, with a large number of methods for how to optimally carry it out having been proposed [51, 56–68]. However, it is probably fair to say that the jury is still out regarding how robust footprinting is at the level of individual motif instances [69, 70].

First, very deeply sequenced datasets are necessary for individual motif footprints to become visible within accessible peaks. At least 300 million mapped reads have been typically used for this task in mammalian genomes. Second, DNase and other enzymes used to map open chromatin all have some non-negligible biases that need to be properly modeled if they are not to confound footprinting analysis.

TF footprints are also observed in ATAC-seq datasets (Fig. 16), but footprinting analysis in the ATAC-seq domain focuses primarily on global footprint profiles. As ATAC-seq libraries are usually generated starting from only 50,000 mammalian cells, they generally do not contain sufficient molecular complexity to support the depth of sequencing required to get to individual footprinting level, unless a large number of individual libraries for the same cell type are pooled together, which is rarely done.

Nevertheless, global footprinting profiles on their own are still highly informative about transcription factor activity across cell types.

These profiles are generating by calculating the average bias-corrected insertion (i.e. 5′ fragment ends) profiles around a set of

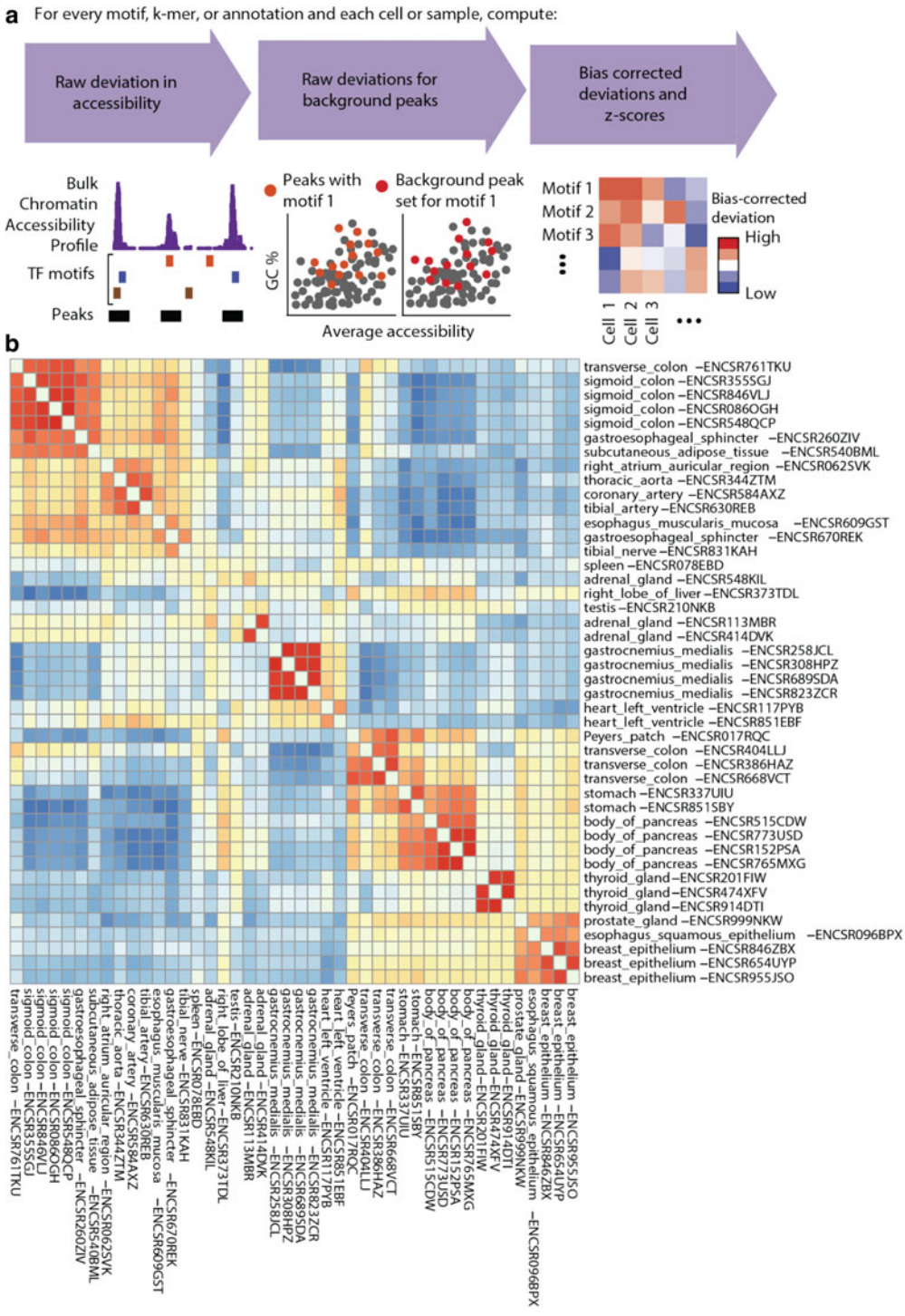

Fig. 14 Analysis of transcription factor activity variability using chromVar. (**a**) Overview of the chromVar algorithm (adapted from the original chromVar publication [32]. (**b**) Clustergram of ENCODE human ATAC-seq datasets based on transcription factor accessibility deviation scores estimated by chromVar

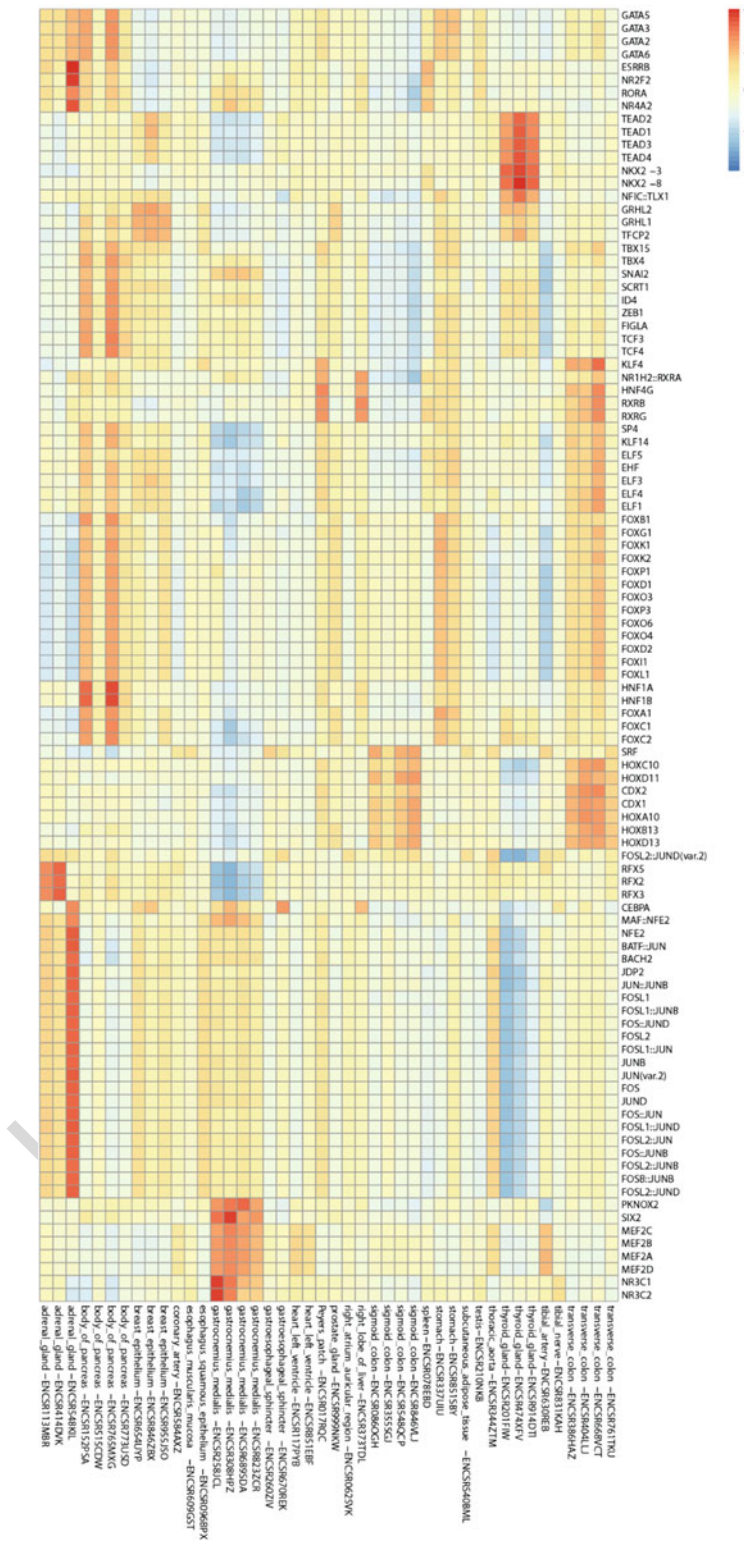

Fig. 15 Analysis of transcription factor activity variability using chromVar. Shown are transcription factor accessibility deviation scores across ENCODE human ATAC-seq datasets

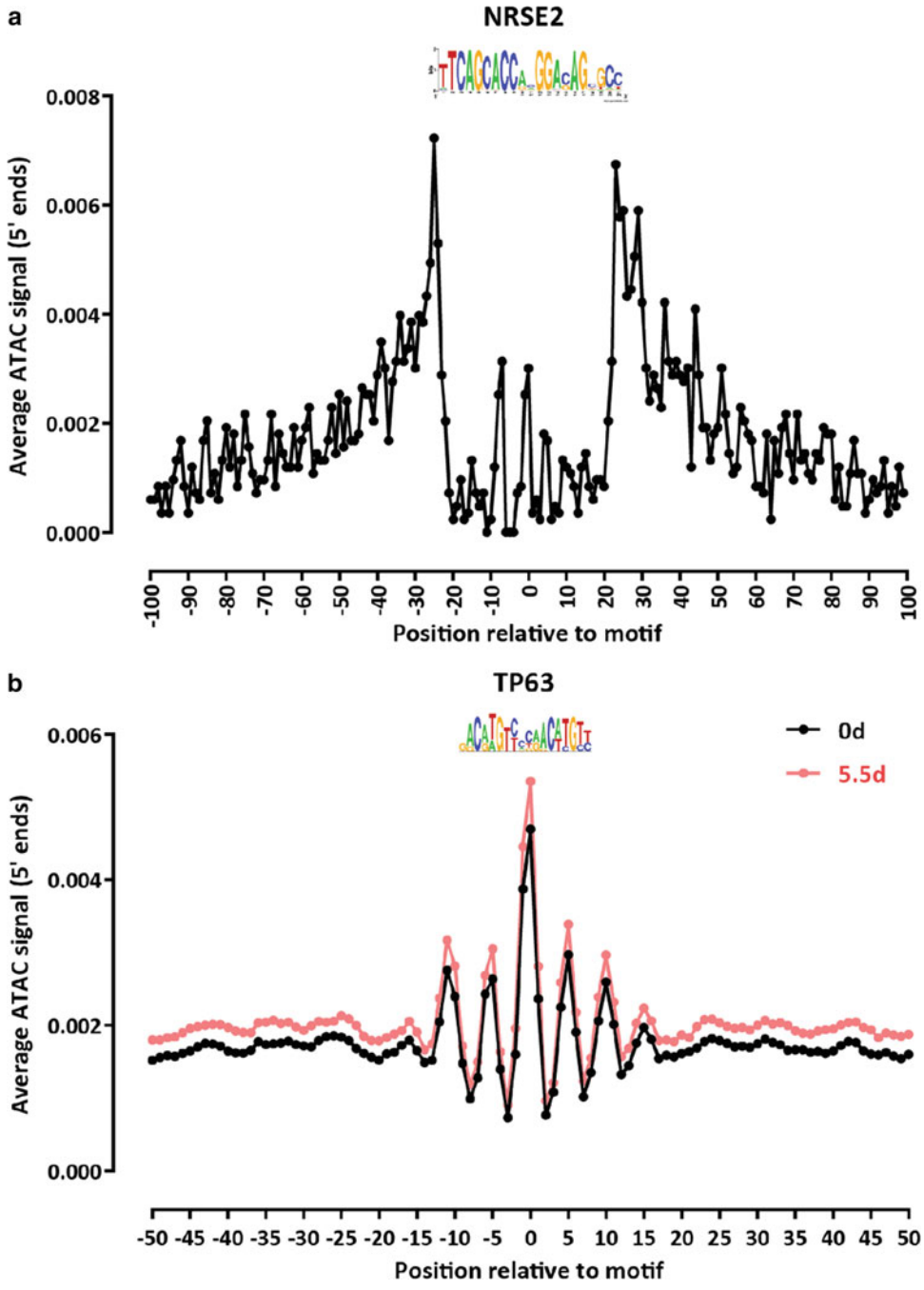

Fig. 16 Global footprinting of occupied NRSF/REST NRSE2 motifs [71] and of TP63 motifs in human keratinocytes (ENCODE accession ENCSR798IJQ). (**a**) NRSF/REST motifs were derived as previously described [71, 72], all instances of the NRSE2 motif identified genome-wife, and then intersected with publicly available ChIP-seq peaks (ENCODE accession ENCFF048JKT) to derive a set of occupied motifs. (**b**) TP63 motif definitions were obtained from the CIS-BP motif database [73] and mapped to the human genome using FIMO version 5.0.5 [74]

motif instances for each individual transcription factor after correction for the sequence bias of the Tn5 enzyme.

The transposase sequence bias is, in the simplest and most general procedure proposed [64], estimated from sequencing libraries generated by transposition of naked DNA. For a given value of k, usually $k=6$, the frequency F_i^{exp} of all k-mers in the mappable portion of genome is calculated, as is the observed frequency F_i^{obs}, corresponding to the k-mers centered on transposase insertion sites. The transposition propensity F_i^{obs}/F_i^{exp} is then calculated and used to adjust observed insertion counts in ATAC-seq datasets.

Examples of global ATAC-seq footprints for transcription factors are shown in Fig. 16. We note that the exact shape of the footprints depends greatly on the properties of the particular transcription factor. TFs with long motifs and tight and stable association with DNA (such as NRSF/REST in Fig. 16a) tend to exhibit deeper footprints than TFs that bind to shorter motif and/or occupy DNA more transiently.

3.14 V-plots

As ATAC-seq datasets contain fragments corresponding to subnucleosomal and nucleosomal fragments, they can be used to study nucleosome organization around certain genomic landmarks (such as TSSs, transcription factor binding sites, and others) based on the average fragment structure around such sites.

The V-plot [75] is the main tool for carrying out such analyses. V-plots are constructed from paired-end datasets by calculating the density of fragments on a map of the positions of sequencing fragment mid-points against the length of fragments, from the view point of a set of single-base pair genomic features.

Figure 17 shows such a V-plot centered on occupied (as measured by ChIP-seq) CTCF motifs. CTCF is known to be a strong nucleosome positioning factor [76], thus in this case it is not surprising to observe both mono- and dinucleosomal fragment density areas on the flanks relative to the CTCF sites, together with a high density of subnucleosomal fragments centered on the CTCF motif itself. In addition, the 10-bp periodicity characteristic to ATAC-seq datasets is also clearly visible as a persistent horizontal feature of the V-plot.

An example of running a custom-written script for generating V-plots is provided below:

```
python V-plot.py SAMPLE.2x36mers.unique.nochrM.dedup.bam
motifs.positions 0 1 500 500
SAMPLE.2x36mers.unique.nochrM.dedup.V-plot -stranded 2
```

Where the motifs file in this case is in the following format:

```
chr <tab> midPoint <tab> strand
```

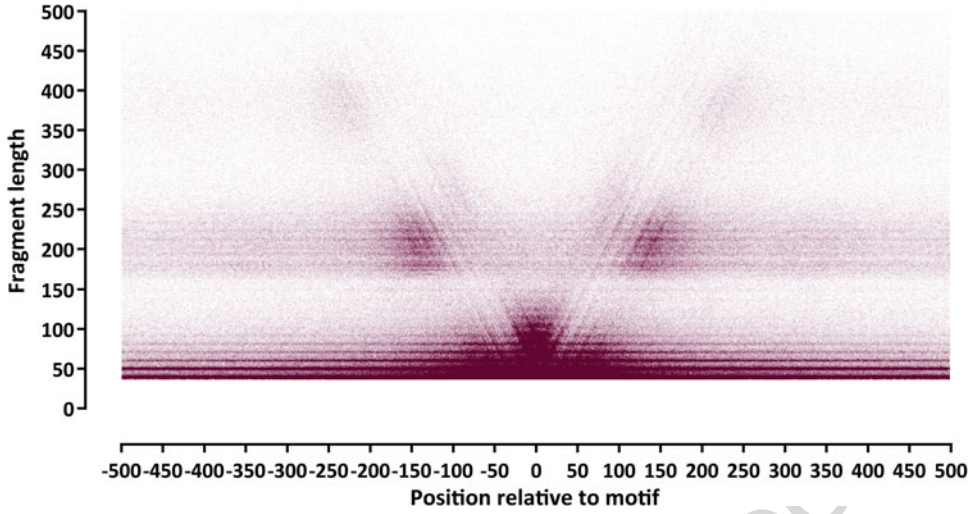

Fig. 17 ATAC-seq V-plot centered on occupied CTCF motifs. The ENCODE keratinocyte ATAC-seq dataset (0d time point) is used as an example

And the V-plot encompasses the ±500 bp neighborhood around the specified set of transcription factor motifs.

V-plots also make it possible to identify positioned nucleosomes within regulatory elements. As ATAC-seq strongly enriches for accessible chromatin, such an analysis can only be carried out in the immediate vicinity of open chromatin regions/*cis*-regulatory elements, but this is often highly informative given that cREs are where dynamics functionally relevant changes in chromatin states typically occur. The NucleoATAC algorithm [77] (based on identifying the positions in the genome in the immediate vicinity for open chromatin regions that maximize nucleosomal V-plot signatures, i.e. a V-plot in which a nucleosome-length fragment is observed centered on the V-plot midpoint) was developed to carry out this task (though we note that it is, as of the writing of this text, no longer maintained and no alternative algorithms are available).

Acknowledgements

The authors thank members of the Greenleaf and Kundaje labs for many helpful discussions. Z.S. is supported by EMBO Long-Term Fellowship EMBO ALTF 1119–2016 and by Human Frontier Science Program Long-Term Fellowship HFSP LT 000835/2017-L. G.K.M. was supported by the Stanford School of Medicine Dean's Fellowship.

References

1. Wu C (1980) The 5′ ends of *Drosophila* heat shock genes in chromatin are hypersensitive to DNase I. Nature 286(5776):854–860
2. Keene MA, Corces V, Lowenhaupt K et al (1981) DNase I hypersensitive sites in Drosophila chromatin occur at the 5′ ends of regions of transcription. Proc Natl Acad Sci USA 78:143–146
3. McGhee JD, Wood WI, Dolan M et al (1981) A 200 base pair region at the 5′ end of the chicken adult β-globin gene is accessible to nuclease digestion. Cell 27:45–55
4. Dorschner MO, Hawrylycz M, Humbert R et al (2004) High-throughput localization of functional elements by quantitative chromatin profiling. Nat Methods 1:219–225
5. Sabo PJ, Humbert R, Hawrylycz M et al (2004) Genome-wide identification of DNaseI hypersensitive sites using active chromatin sequence libraries. Proc Natl Acad Sci USA 101:4537–4542
6. Sabo PJ, Kuehn MS, Thurman R et al (2006) Genome-scale mapping of DNase I sensitivity in vivo using tiling DNA microarrays. Nat Methods 3:511–518
7. Crawford GE, Holt IE, Whittle J et al (2006) Genome-wide mapping of DNase hypersensitive sites using massively parallel signature sequencing (MPSS). Genome Res 16:123–131
8. Boyle AP, Davis S, Shulha HP et al (2008) High-resolution mapping and characterization of open chromatin across the genome. Cell 132 (2):311–322
9. Thurman RE, Rynes E, Humbert R et al (2012) The accessible chromatin landscape of the human genome. Nature 489 (7414):75–82.
10. Kelly TK, Liu Y, Lay FD et al (2012) Genome-wide mapping of nucleosome positioning and DNA methylation within individual DNA molecules. Genome Res 22(12):2497–2506
11. Krebs AR, Imanci D, Hoerner L, Gaidatzis D et al (2017) Genome-wide Single-Molecule Footprinting Reveals High RNA Polymerase II Turnover at Paused Promoters. Mol Cell 67(3):411–422.e4
12. Shipony Z, Marinov GK, Swaffer MP et al (2018) Long-range single-molecule mapping of chromatin accessibility in eukaryotes. bioRxiv 504662
13. Wang Y, Wang A, Liu Z et al (2019) Single-molecule long-read sequencing reveals the chromatin basis of gene expression. Genome Res 29(8):1329–1342
14. Aughey GN, Estacio Gomez A, Thomson J et al (2018) CATaDa reveals global remodelling of chromatin accessibility during stem cell differentiation in vivo. Elife 7:pii: e32341
15. Chereji RV, Eriksson PR, Ocampo J, Clark DJ (2019) DNA accessibility is not the primary determinant of chromatin-mediated gene regulation. bioRxiv 639971
16. Ponnaluri VKC, Zhang G, Estéve PO et al (2017) NicE-seq: high resolution open chromatin profiling. Genome Biol 18(1):122
17. Umeyama T, Ito T (2017) DMS-Seq for in vivo genome-wide mapping of protein-DNA interactions and nucleosome centers. Cell Rep 21(1):289–300
18. Timms RT, Tchasovnikarova IA, Lehner PJ (2019) Differential viral accessibility (DIVA) identifies alterations in chromatin architecture through large-scale mapping of lentiviral integration sites. Nat Protoc 14(1):153–170
19. Buenrostro JD, Giresi PG, Zaba LC et al (2013) Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. Nat Methods 10:1213–1218
20. Buenrostro JD, Wu B, Litzenburger UM et al (2015) Single-cell chromatin accessibility reveals principles of regulatory variation. Nature 523(7561):486–490
21. Cusanovich DA, Daza R, Adey A et al (2015) Multiplex single cell profiling of chromatin accessibility by combinatorial cellular indexing. Science 348(6237):910–914
22. ENCODE Project Consortium (2012) An integrated encyclopedia of DNA elements in the human genome. Nature 489:57–74
23. Amemiya HM, Kundaje A, Boyle AP (2019) The ENCODE Blacklist: Identification of Problematic Regions of the Genome. Sci Rep 9(1):9354
24. Langmead B, Trapnell C, Pop M et al (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10:R25
25. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9:357–359
26. Li H, Handsaker B, Wysoker A et al (2009) The Sequence Alignment/Map format and SAMtools. Bioinformatics 25:2078–2079
27. Feng J, Liu T, Qin B et al (2012) Identifying ChIP-seq enrichment using MACS. Nat Protoc 7:1728–1740

28. Li Q, Brown J, Huang H et al (2011) Measuring reproducibility of high-throughput experiments. Ann Appl Stat 5:1752–1779
29. Kuhn RM, Haussler D, Kent WJ (2013) The UCSC genome browser and associated tools. Brief Bioinform 14:144–161
30. Kent WJ, Zweig AS, Barber G et al (2010) BigWig and BigBed: enabling browsing of large distributed datasets. Bioinformatics 26:2204–2207
31. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15(12):550
32. Schep AN, Wu B, Buenrostro JD, Greenleaf WJ (2017) chromVAR: inferring transcription-factor-associated accessibility from single-cell epigenomic data. Nat Methods 14:975–978
33. Ramírez F, Ryan DP, Grüning B et al (2016) deepTools2: a next generation web server for deep-sequencing data analysis. Nucleic Acids Res 44(W1):W160–W165
34. Quinlan AR, Hall IM (2010) BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics 26(6):841–842
35. Liao Y, Smyth GK, Shi W. (2014) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics 30(7):923–930
36. Bolger AM, Lohse M, Usadel B (2014) Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics 30 (15):2114–2120
37. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet J 17(1):10–12
38. Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25:1754–1760
39. Corces MR, Trevino AE, Hamilton EG et al (2017) An improved ATAC-seq protocol reduces background and enables interrogation of frozen tissues. Nat Methods 14:959–962
40. Hazkani-Covo E, Zeller RM, Martin W (2010) Molecular poltergeists: mitochondrial DNA copies (numts) in sequenced nuclear genomes. PLoS Genet 6(2):e1000834
41. Marinov GK, Wang YE, Chan D, Wold BJ (2014) Evidence for site-specific occupancy of the mitochondrial genome by nuclear transcription factors. PLoS One 9(1):e84713
42. Smith DR, Keeling PJ (2015) Mitochondrial and plastid genome architecture: reoccurring themes, but significant differences at the extremes. Proc Natl Acad Sci USA 112 (33):10177–10184
43. Landt SG, Marinov GK, Kundaje A et al (2012) ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. Genome Res 22(9):1813–1831
44. Daley T, Smith AD (2013) Predicting the molecular complexity of sequencing libraries. Nat Methods 10(4):325–327
45. Marinov GK, Kundaje A, Park PJ, Wold BJ (2014) Large-scale quality analysis of published ChIP-seq data. G3 (Bethesda) 4(2):209–223
46. Tarbell ED, Liu T (2019) HMMRATAC: a Hidden Markov ModeleR for ATAC-seq. Nucleic Acids Res pii: gkz533
47. McCarthy DJ, Chen Y, Smyth GK (2012) Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Res 40 (10):4288–4297
48. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015) Limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res 43(7):e47
49. van der Maaten LJP, Hinton GE (2008) Visualizing high-dimensional data using t-SNE. J Mach Learn Res 9:2579–2605
50. Becht E, McInnes L, Healy J et al (2018) Dimensionality reduction for visualizing single-cell data using UMAP. Nat Biotechnol 37:38–44
51. Li Z, Schulz MH, Look T et al (2019) Identification of transcription factor binding sites using ATAC-seq. Genome Biol 20(1):45
52. Hesselberth JR, Chen X, Zhang Z et al (2009) Global mapping of protein-DNA interactions in vivo by digital genomic footprinting. Nat Methods 6(4):283–289
53. Neph S, Stergachis AB, Reynolds A et al (2012) Circuitry and dynamics of human transcription factor regulatory networks. Cell 150:1274–1286
54. Neph S, Vierstra J, Stergachis AB et al (2012) An expansive human regulatory lexicon encoded in transcription factor footprints. Nature 489:83–90
55. Stergachis AB, Neph S, Reynolds A et al (2013) Developmental fate and cellular maturity encoded in human regulatory DNA landscapes. Cell 154:888–903
56. Pique-Regi R, Degner JF, Pai AA et al (2011) Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data. Genome Res 21(3):447–455
57. Cuellar-Partida G, Buske FA, McLeay RC et al (2012) Epigenetic priors for identifying active transcription factor binding sites. Bioinformatics 28(1):56–62

58. Piper J, Elze MC, Cauchy P et al (2013) Wellington: a novel method for the accurate identification of digital genomic footprints from DNase-seq data. Nucleic Acids Res 41(21): e201
59. Sherwood RI, Hashimoto T, O'Donnell CW et al (2014) Discovery of directional and nondirectional pioneer transcription factors by modeling DNase profile magnitude and shape. Nat Biotechnol 32(2):171–178
60. He HH, Meyer CA, Hu SS et al (2014) Refined DNase-seq protocol and data analysis reveals intrinsic bias in transcription factor footprint identification. Nat Methods 11:73–78
61. Sung MH, Guertin MJ, Baek S, Hager GL. 2014. DNase footprint signatures are dictated by factor dynamics and DNA sequence. Mol Cell 56(2):275–285
62. Gusmao EG, Dieterich C, Zenke M, Costa IG (2014) Detection of active transcription factor binding sites with the combination of DNase hypersensitivity and histone modifications. Bioinformatics 30(22):3143–3151
63. Raj A, Shim H, Gilad Y et al (2015) msCentipede: modeling heterogeneity across genomic sites and replicates improves accuracy in the inference of transcription factor binding. PLoS One 10(9):e0138030
64. Yardimci GG, Frank CL, Crawford GE, Ohler U (2015) Explicit DNase sequence bias modeling enables high-resolution transcription factor footprint detection. Nucleic Acids Res 42 (19):11865–11878
65. Gusmao EG, Allhoff M, Zenke M, Costa IG (2016) Analysis of computational footprinting methods for DNase sequencing experiments. Nat Methods 13(4):303–309
66. Quach B, Furey TS (2017) DeFCoM: analysis and modeling of transcription factor binding sites using a motif-centric genomic footprinter. Bioinformatics 33(7):956–963
67. Baek S, Goldstein I, Hager GL (2017) Bivariate genomic footprinting detects changes in transcription factor activity. Cell Rep 19 (8):1710–1722
68. Karabacak Calviello A, Hirsekorn A, Wurmus R et al (2019) Reproducible inference of transcription factor footprints in ATAC-seq and DNase-seq datasets using protocol-specific bias modeling. Genome Biol 20(1):42
69. Sung MH, Baek S, Hager GL (2016) Genome-wide footprinting: ready for prime time? Nat Methods 13(3):222–228
70. Vierstra J, Stamatoyannopoulos JA (2016) Genomic footprinting. Nat Methods 13 (3):213–221
71. Mortazavi A, Leeper Thompson EC, Garcia ST et al (2006) Comparative genomics modeling of the NRSF/REST repressor network: from single conserved sites to genome-wide repertoire. Genome Res 16(10):1208–1221
72. Johnson DS, Mortazavi A, Myers RM, Wold B (2007) Genome-wide mapping of in vivo protein-DNA interactions. Science 316 (5830):1497–1502
73. Weirauch MT, Yang A, Albu M et al (2014) Determination and inference of eukaryotic transcription factor sequence specificity. Cell 158:1431–1443
74. Grant CE, Bailey TL, Noble WS (2011) FIMO: scanning for occurrences of a given motif. Bioinformatics 27:1017–1018
75. Henikoff JG, Belsky JA, Krassovsky K et al (2011) Epigenome characterization at single base-pair resolution. Proc Natl Acad Sci USA 108:18318–18323
76. Fu Y, Sinha M, Peterson CL, Weng Z (2008) The insulator binding protein CTCF positions 20 nucleosomes around its binding sites across the human genome. PLoS Genet 4:e1000138
77. Schep AN, Buenrostro JD, Denny SK et al (2015) Structured nucleosome fingerprints enable high-resolution mapping of chromatin architecture within regulatory regions. Genome Res 25:1757–1770