Chapter 2

Identification of Candidate Functional Elements in the Genome from ChIP-seq Data

Georgi K. Marinov

Abstract

ChIP-seq datasets provide a wealth of information for the identification of candidate regulatory elements in the genome. For this potential to be fully realized, methods for evaluating data quality and for distinguishing reproducible signal from technical and biological noise are necessary. Here, the computational methods for addressing these challenges developed by the ENCODE Consortium are described and the key considerations for analyzing and interpreting ChIP-seq data are discussed.

Key words Regulatory elements, Transcription factors, Histone modifications, Chromatin immunoprecipitation, High-throughput sequencing

1 Introduction

The high resolution, genome-wide coverage, and overall information richness characteristic to ChIP-seq datasets have made the assay the primary experimental tool for profiling protein-DNA interactions since its initial introduction in the second half of the 2000s [1–4]. The field has since entered maturity. A diverse array of analytical tools have been published, and much experience in working with ChIP-seq data has accumulated from the very large number of datasets that have been generated [5, 6], in particular in association with the activities of the ENCODE, modENCODE, and mouse ENCODE consortia [7–12]. These studies have helped clarify the most important characteristics of the data that need to be evaluated to ensure that biologically reliable results are obtained, and robust analytical pipelines for identifying reproducible regions of enrichment have been devised as a result. ChIP-seq data analysis can be broadly divided into two (not independent of each other) steps—data quality evaluation and identification of enriched regions (outlined in Fig. 1). The importance of quality assessment derives from the observation that there can be considerable variation in the quality of ChIP-seq datasets, sometimes even between

Sara Napoli (ed.), *Promoter Associated RNA: Methods and Protocols*, Methods in Molecular Biology, vol. 1543,
DOI 10.1007/978-1-4939-6716-2_2,

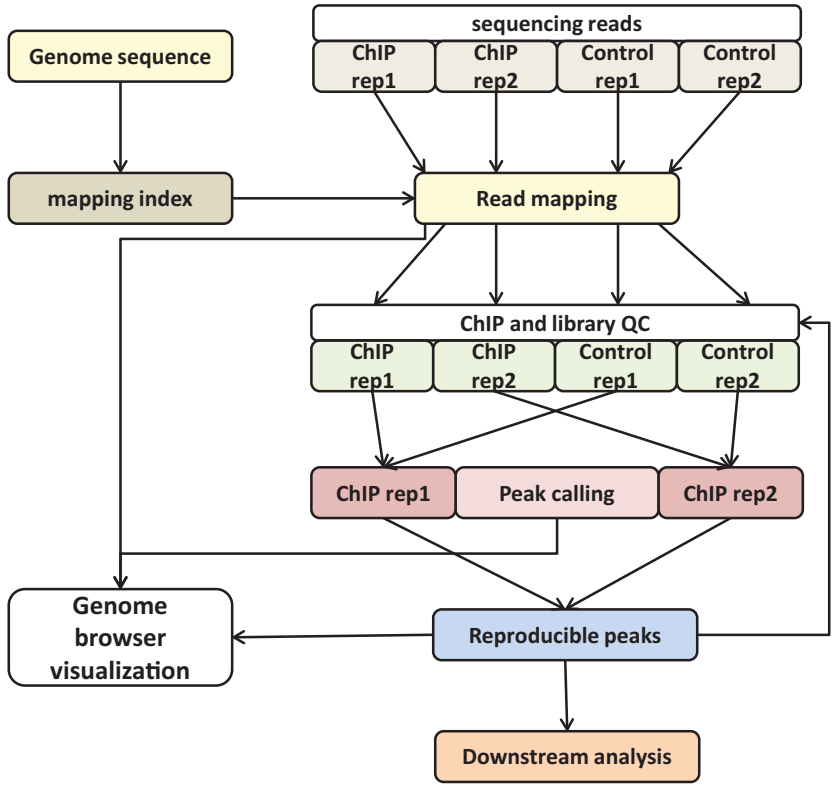


Fig. 1 Overview of the ChIP-seq analysis workflow. Reads from replicate ChIP and matching Control experiments are aligned against the genome and subjected to library and ChIP quality evaluation. Peak calling is then carried out followed by identification of consensus peaks using reproducibility analysis. The reproducibility analysis also serves as another QC step, if very low reproducibility is detected. Peaks and alignments can then be subjected to downstream analysis and/or visualized on a genome browser

what is essentially technical replicates of the same experiment. Poor-quality datasets negatively affect the ability to reliably identify occupancy regions (due to the presence of large numbers of false negatives and/or false positives), which can in turn confound biological interpretations of the data, in particular when multiple datasets are jointly analyzed (discussed in detail in [5, 6]). Low-quality ChIP-seq datasets are characterized by some combination of very low sequencing depth, low molecular complexity of the library, and low degree of enrichment of target-DNA complexes during the ChIP reaction. The latter manifests itself as the absence of strong localized read clustering (it should be noted that the presence of such clustering in control datasets, for which localized enrichment is not expected, is potentially problematic too [6]).

The main challenge when identifying regions of enrichment is distinguishing true signal from noise. It is commonly observed that regions of enrichment in ChIP-seq datasets follow an exponential-like distribution, with a small number of strong and a large number of weak sites, and no clear separation between the

very weak sites and noise. Dozens of peak calling algorithms have been published, each of them with its own specific, often tunable, approach toward thresholding. Significant differences can often be observed between their output, usually at the low end of the signal intensity spectrum (top sites tend to be reliably found by all algorithms but the inclusion of weaker sites in peak call lists is highly dependent on the particulars of the algorithms and thresholds applied). The threshold-independent IDR (Irreproducible Discovery Rate) analysis [13], which relies on the comparison of biological replicates, has recently emerged as the most robust way of identifying reproducibly occupied regions.

These computational procedures are described and discussed in this chapter.

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 for typical depths of sequencing. Some of the programs used are multithreaded and will therefore complete faster if run on multiple cores.

2.1 Genomic Sequence and Annotation Files

1. A FASTA file containing the hg19 version of the human genome can be downloaded from http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz. The more recent hg20 version can be found at http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/hg38.fa.gz. The UCSC Genome Browser also has genome files for many other species. Other rich genomic resources are ENSEMBL (http://ensemblgenomes.org/) and the NCBI website (http://www.ncbi.nlm.nih.gov/assembly/).
2. "Blacklist" regions (discussed recently in [14]). "Excludable" files containing catalogs of common artifacts in hg19 coordinates can be downloaded from http://hgdownload-test.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/.
3. The GENCODE annotation in GTF format: http://www.gencodegenes.org/.

2.2 Software Packages

1. Bowtie [15] (http://bowtie-bio.sourceforge.net/index.shtml) or Bowtie2 [16] (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) (*see* **Note 1** for discussion of alignment).
2. samtools [17]: http://www.htslib.org/.
3. Preseq [18]: http://smithlabresearch.org/software/preseq/.

4. FastQC: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/.
5. SRA Toolkit: http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software.
6. SPP [19]: https://code.google.com/p/phantompeakqualtools/.
7. IDR [13] analysis code: https://sites.google.com/site/anshulkundaje/projects/idr.
8. UCSC Genome Browser [20, 21] utilities: http://hgdownload.cse.ucsc.edu/admin/exe/.
9. Additional scripts: https://github.com/georgimarinov/GeorgiScripts. Contains the python scripts used in the examples shown below; some of the scripts depend on having pysam installed.

3 Methods

The general outline of the pipeline is presented in Fig. 1. As an initial and one-time step, a genome index file is prepared to be used during read mapping. Reads for ChIP and Control datasets are then mapped to the genome, and the library and read clustering characteristics of each dataset are evaluated, by calculating the number and fraction of mapped reads, estimating the library complexity, and running cross-correlation analysis [5, 6]. The datasets are then subjected to reproducibility analysis (Fig. 2), which consists of:

1. Calling peaks with very relaxed settings (*see* **Notes 2** and **3**).
2. Running IDR on the individual set of peaks. This gives the number of reproducible peaks between replicates *Nt*.
3. Pooling the reads for the two ChIP replicates and separately for the two controls.
4. Creating two sets of pooled pseudoreplicates with approximately equal number of randomly sampled reads from the pooled sets of ChIP and Control reads.
5. Calling peaks with relaxed settings on each pooled pseudoreplicate.
6. Running IDR on the pooled pseudoreplicates. The number of pseudoreplicate self-consistent peaks *Np* is obtained from this step.
7. Creating two sets of individual pseudoreplicates with approximately equal number of randomly sampled reads from each ChIP and Control replicate.
8. Calling peaks with relaxed settings on each individual pseudoreplicate.

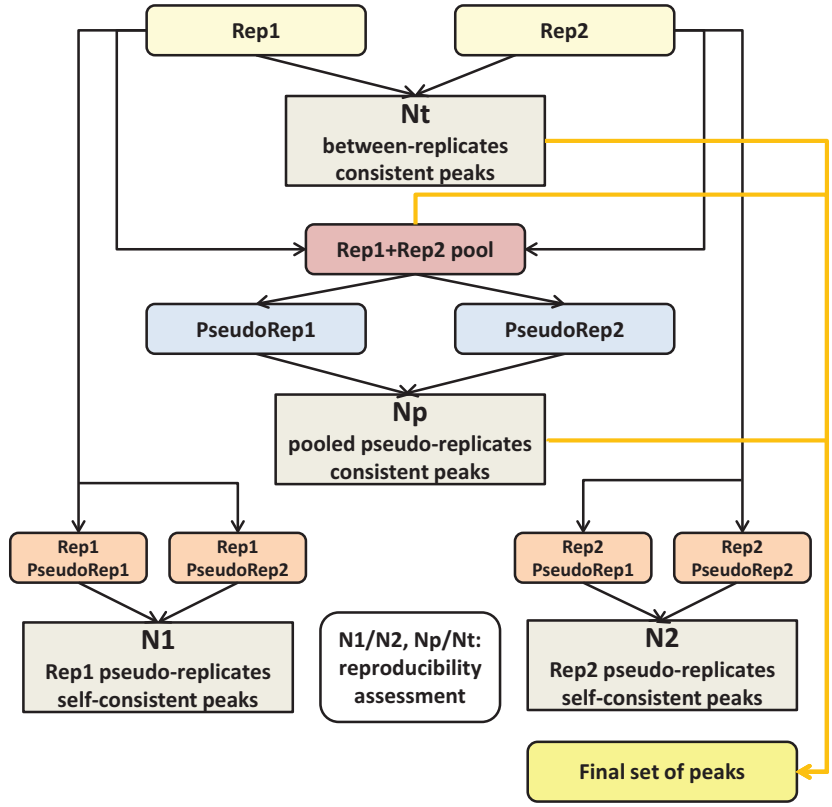


Fig. 2 Summary of the IDR-based reproducibility analysis pipeline. IDR is used to identify reproducible peaks between individual replicates (*Nt*), between pooled pseudoreplicates (*Np*), generated by pooling the two replicates and randomly splitting them in two, and between pseudoreplicates generated from each individual replicate (*N1* and *N2*). These values are used to assess reproducibility and to derive the final set of peaks (from the ranked peak calls derived from the pooled ChIP and Control datasets)

9. Running IDR on the two sets of individual pseudoreplicates. This provides the numbers of individual replicate self-consistent peaks *N1* and *N2*.
10. Checking for abnormalities in reproducibility. Specifically, when $N1/N2 > 2$, and/or $Np = Nt > 2$, the reproducibility between the two replicates is considered low [5] (*see* **Note 4**).
11. Peak calling with relaxed settings on the pooled sets of reads.
12. Using the number of reproducible peaks between replicates *Nt* and pooled pseudoreplicates *Np* to derive the final set of peaks by taking the top *max(Nt, Np)* peaks from the peak calls on the pooled sets of reads.

If more than two replicates are available, they can be analyzed in all possible pairs and the results merged at the level of the maximum *Nt* (*see* **Note 5**).

A step-by-step example using ENCODE data for the TAF1 protein (a component of the transcription preinitiation complex [22], and thus expected to be associated with promoter elements) in H1 human embryonic stem cells (H1-hESCs) is used as an illustration of the process throughout this chapter.

3.1 Preparation of Genomic Files

1. Download and uncompress genomic sequence files:

```
mkdir genomes; cd genomes; mkdir hg19; cd hg19;
mkdir sequence; cd sequence
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz
tar -xzvf chromFa.tar.gz
```

2. Create a fasta file without the "alt"/"hap" chromosomes (if starting from a larger fasta file, the fastaSubset.pyscript can be used for this purpose). The "random" contigs can also be removed (*see* **Note 6** for further discussion on this topic). Create a separate file with the Y chromosome removed if samples of known female origin are to be analyzed (the H1-hESC cells used here are male).

3. Create bowtie indexes (male version shown):

```
mkdir genomes/hg19/bowtie-indexes
cd genomes/hg19/bowtie-indexes
ln ../sequence/hg19-male.fa
bowtie-build -f hg19-male.fa hg19-male
```

   With Bowtie2:

```
mkdir genomes/hg19/bowtie2-indexes
cd genomes/hg19/bowtie2-indexes
ln ../sequence/hg19-male.fa
bowtie2-build -f hg19-male.fa hg19-male
```

4. Create chromosome size info (chrom.sizes) files (*see* **Note 7**):

```
python makeChromSizesFromFasta.py hg19-male.fa hg19-male.chrom.sizes
```

3.2 Read Mapping

1. Download reads. In this case, the following files:

```
wgEncodeHaibTfbsH1hescTaf1V0416102RawData-Rep1.fastq.gz
wgEncodeHaibTfbsH1hescTaf1V0416102RawData-Rep2.fastq.gz
wgEncodeHaibTfbsH1hescRxlchV0422111RawData-Rep1.fastq.gz
wgEncodeHaibTfbsH1hescRxlchV0422111RawData-Rep2.fastq.gz
```

   are downloaded from http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibTfbs/. The first

two files are the TAF1 ChIP samples, the last two are the Input Control samples.

2. Count raw reads:

```
gunzip -c SAMPLE.fastq.gz | wc -l
```

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

3. Read quality filtering (optional, *see* **Note 8**).

4. Decompress reads, trim the sequences to the desired read length (if necessary), align, convert to BAM, and sort the BAM file. This can be carried out in one step as follows:

```
gunzip -c wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.fastq.gz |
python trimfastq.py - 36 -stdout | bowtie
genomes/hg19/bowtie-indexes/hg19-male -p 16
-v 2 -k 2 -m 1 -t --best --strata -q --sam -
| samtools view -bT genomes/hg19/sequence/
hg19.fa - | samtools sort - wgEncodeHaibTfb-
sH1hescTaf1V0416102RawDataRep1.unique
```

This retains uniquely mapping reads with up to two mismatches relative to the reference.

With Bowtie2:

```
gunzip -c wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.fastq.gz | py-
thon trimfastq.py - 36 -stdout | bowtie2 -x
genomes/hg19/bowtie2-indexes/hg19-male -p
16 -t -q -U - | egrep -v 'XS:i:0' | | sam-
tools view -bT genomes/hg19/sequence/hg19.
fa - | samtools sort - wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.unique
```

This command filters out all alignments with a second equally good alignment (i.e., the equivalent of "multireads"; specified in the XS:itag).

If reads are downloaded from the Short Read Archive, they can be similarly directly streamed. The following command will print reads to standard output.

```
sratoolkit.2.4.0-1-ubuntu64/bin/fastq-
dump.2.4.0 -Z reads.sra
```

If paired-end reads are to be mapped with Bowtie1, this can be done from compressed *.gzor *bz2files as follows:

```
python PEFastqToTabDelimited.py end1.
fastq.gz end2.fastq.gz | bowtie genomes/
hg19/bowtie-indexes/hg19-male -p 16 -v 2 -k
2 -m 1 -t --best --strata -q -X 1000 --sam
--12 - | samtools view -bT genomes/hg19/se-
quence/hg19.fa - | samtools sort - sample.
PE.unique
```

With Bowtie2:

```
bowtie2 -x genomes/hg19/bowtie2-indexes/
hg19-male -1 end1.fastq.gz -2 end2.fastq.
gz -p 16 -t -X 1000 --no-mixed --no-
discordant - | egrep -v 'XS:i:0' | samtools
view -bT genomes/hg19/sequence/hg19.fa - |
samtools sort - sample.PE.unique
```

5. Index bam files with samtools:
```
samtools index wgEncodeHaibTfbsH1hescCtcf-
sc5916V0416102RawDataRep1.unique.bam
```
6. Evaluate sequencing quality with FastQC (*see* **Note 9**)
```
mkdir fastqc-wgEncodeHaibTfbsH1hescTaf1V041
6102RawDataRep1;
fastqc wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.fastq.gz -o
fastqc-wgEncodeHaibTfbsH1hescTaf1V0416102Ra
wDataRep1
```

The same procedure is carried out in parallel for all ChIP and Control samples. The resulting BAM files can be used as input for the next steps (*see* **Note 10**).

3.3 Library and ChIP/Control Quality Assessment

Library quality is evaluated here by calculating the apparent library complexity (NRF, or Non-Redundant Fraction of reads, [5, 6]), defined as:

$$NRF = \frac{\text{Number distinct unique reads}}{\text{Total number unique reads}}$$

More recently, ways to estimate the absolute molecular complexity of sequencing libraries have been developed, one example being the Preseqpackage [18]. Such knowledge is highly valuable but the NRF-based guidelines established previously (NRF∈[0:8; 1] => "high complexity," NRF < 0:5 => "low complexity" [5, 6]) are still useful for the typical sequencing depth of a ChIP-seq dataset (5×10^7reads; Fig. 3b, c).

The extent of read clustering, i.e., ChIP enrichment if the library is from a ChIP experiment, is evaluated using cross-correlation analysis [19] as described in detail previously [5, 6] based on the NSC and RSC coefficients and the cross-correlation plots (Fig. 4). The NSC and RSC coefficients are defined as follows:

$$NSC = \frac{CC(\text{fragment length})}{\min(CC)}$$

$$RSC = \frac{CC(\text{fragment length}) - \min(CC)}{CC(\text{read length}) - \min(CC)}$$

where CC refers to the cross-correlation function.

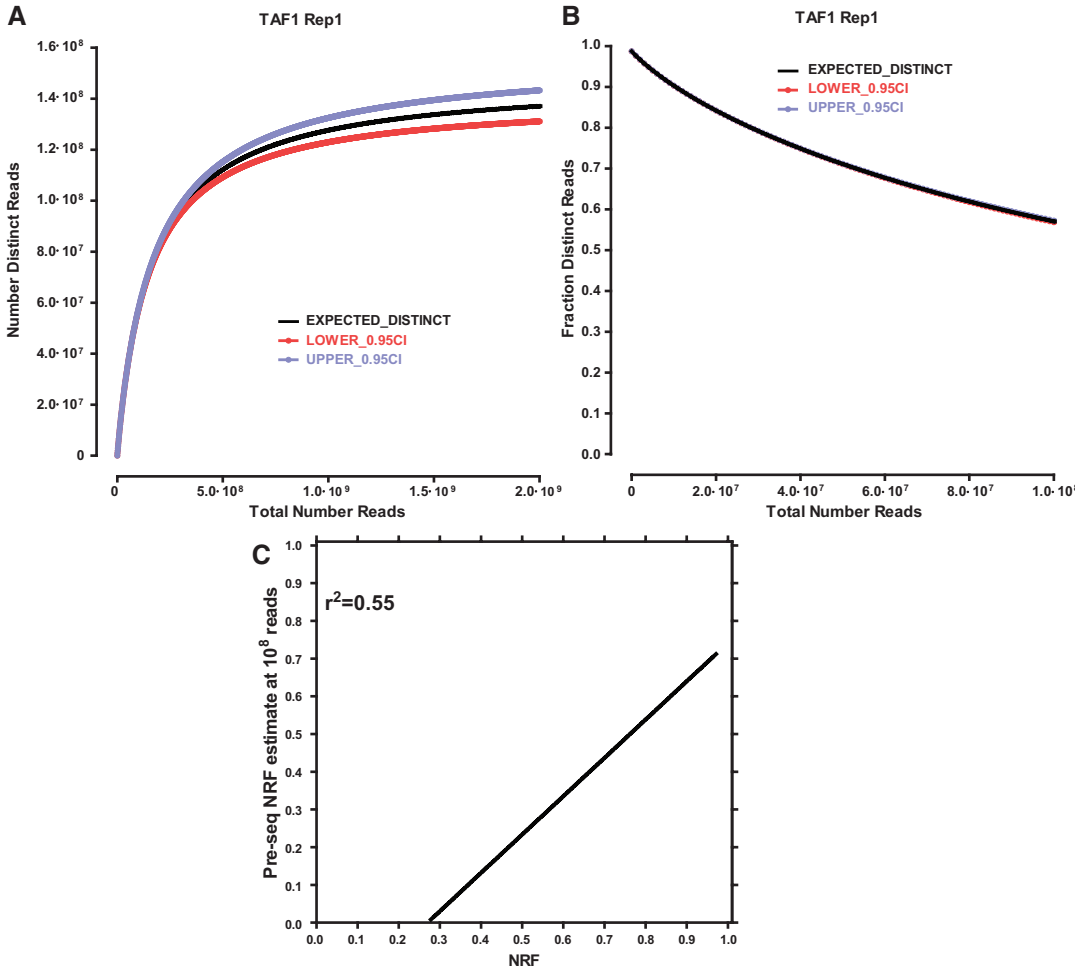
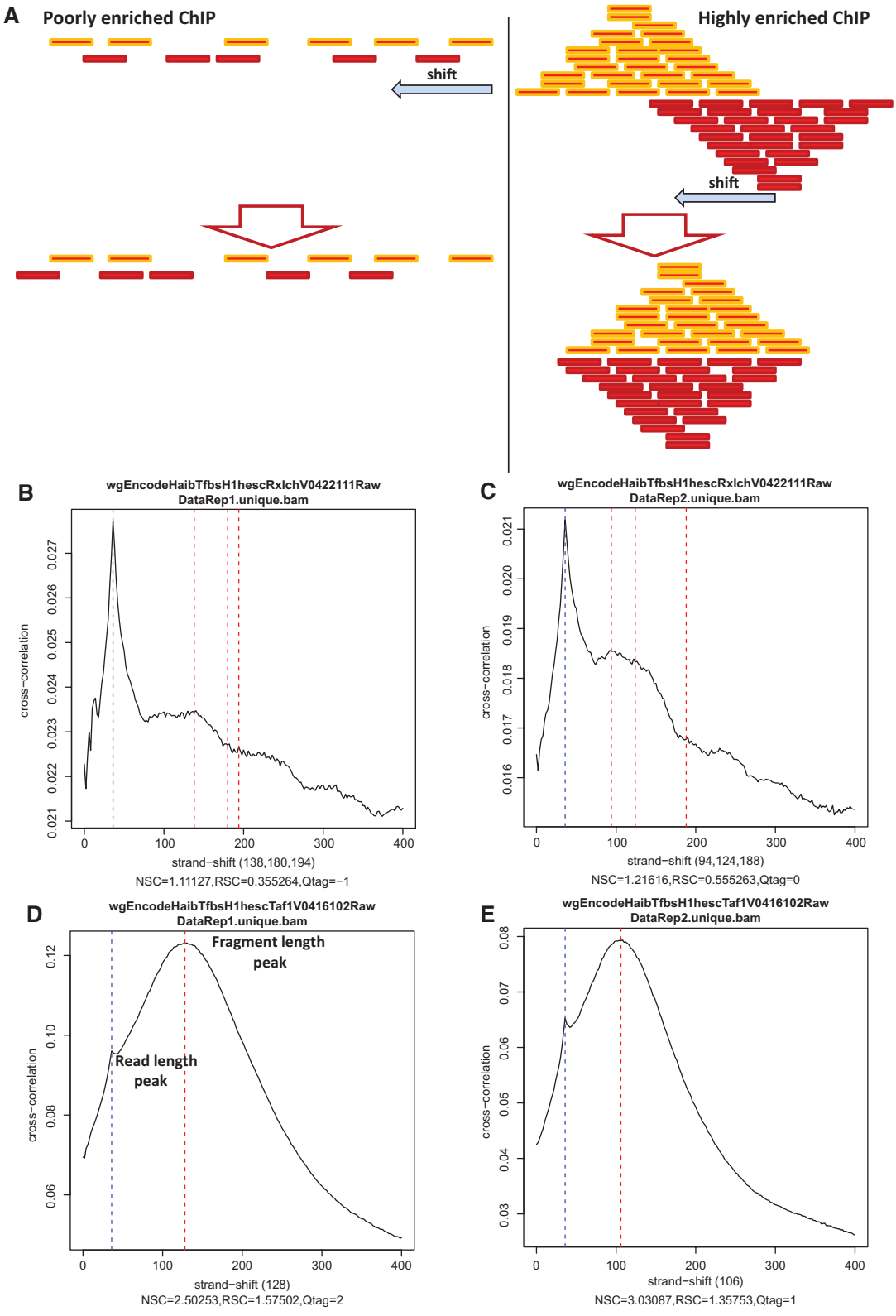


Fig. 3 Assessment of molecular complexity of libraries. (**a**) Preseq estimates for the total absolute molecular complexity of the HAIB TAF1 Rep1 dataset. (**b**) Preseq-estimated NRF fraction as a function of sequencing depth. (**c**) Correlation between empirical NRF values for all sequenced reads and Preseq-estimated NRF values at 1×10^8 reads for a subset of human ENCODE datasets (histone marks from the Broad Institute group)

See **Notes 11** and **12** for further discussion.

1. Calculate mapping statistics and apparent library complexity:
```
python SAMstats.py wgEncodeHaibTfbsH1hesc-
Ctcfsc5916V0416102RawDataRep1.unique.bam
SAMstats-H1hescCtcfscRep1 -bam genomes/hg19/
hg19.chrom.sizes samtools
```
2. Estimate the absolute library complexity using Preseq:
```
preseq-1.0.2.Linux_x86_64/preseq lc_extrap
-B wgEncodeHaibTfbsH1hescTaf1V0416102Raw-
DataRep1.unique.bam > wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.lc_extrap
```


A
Poorly enriched ChIP
shift
Highly enriched ChIP
shift
B
wgEncodeHaibTfbsH1hescRxlchV0422111Raw
DataRep1.unique.bam
cross−correlation
strand−shift (138,180,194)
NSC=1.11127,RSC=0.355264,Qtag=−1
C
wgEncodeHaibTfbsH1hescRxlchV0422111Raw
DataRep2.unique.bam
cross−correlation
strand−shift (94,124,188)
NSC=1.21616,RSC=0.555263,Qtag=0
D
wgEncodeHaibTfbsH1hescTaf1V0416102Raw
DataRep1.unique.bam
Fragment length
peak
Read length
peak
cross−correlation
strand−shift (128)
NSC=2.50253,RSC=1.57502,Qtag=2
E
wgEncodeHaibTfbsH1hescTaf1V0416102Raw
DataRep2.unique.bam
cross−correlation
strand−shift (106)
NSC=3.03087,RSC=1.35753,Qtag=1

3. Generate cross-correlation profiles and calculate NSC and RSC scores using SPP:

```
Rscript spp_package/run_spp.R
-c=wgEncodeHaibTfbsH1hescTaf1V0416102RawData
Rep1.unique.bam -p=16 -savp -rf -s=-0:2:400
-out=wgEncodeHaibTfbsH1hescTaf1V0416102RawD
ataRep1.QC
```

Make sure to manually examine the resulting plots to identify possible oddities in the cross-correlation profiles or instances of incorrect assignment of the fragment peak (*see* **Note 12**). *See* **Note 13** for discussion on quality score cutoffs and their application.

Figures 3a, b and 4 and Table 1 show the mapping statistics, QC metrics, and the cross-correlation plots for the datasets discussed here.

3.4 Peak Calling and Identification of Reproducible Peaks

Here, peak calling is carried out with very relaxed settings (requesting the top 3×10^5 peaks) with the SPP peak caller. (*See* **Notes 3** and **14** for discussion of peak calling algorithms. Detailed technical information is also available at https://sites.google.com/site/anshulkundaje/projects/idr). The commands in the pipeline can be quickly generated automatically for large numbers of datasets using the IDRSPPcommands.pyscript.

3.4.1 IDR Analysis Pipeline

1. Create output directories for each replicate:

```
mkdir SPP-300K-wgEncodeHaibTfbsH1hescTaf1V0
416102RawDataRep1
the pooled sample:
```

Table 1
Read mapping and dataset quality statistics for TAF1 example (Bowtie1 alignments)

Dataset	NRF	Number post-IDR peaks	post-IDR FRiP	NSC	RSC	QC	Read Length	Unique reads	Raw reads
Control-Rep1	0.96		1.111		0.355	−1	36	40,869,857	55,951,481
Control-Rep2	0.98		1.216		0.555	0	36	22,632,758	27,548,706
TAF1-Rep1	0.89	20,001	0.206	2.503	1.575	2	36	14,023,010	20,170,636
TAF1-Rep2	0.88	20,001	0.149	3.031	1.358	1	36	13,217,524	20,470,131

Fig. 4 Assessment of ChIP enrichment using cross-correlation analysis. (**a**) Overview of cross-correlation. The cross-correlation curve is generated by shifting reads on the two strands by an increasing number of base pairs relative to each other and calculating the correlation between the read profiles on the two strands. This results in two peaks—one at the read length ("phantom" peak) and one at the fragment length. The absolute height of the fragment peak and its height relative to the "phantom" peak provide information about the extent and shape of read clustering in the dataset. (**b** and **c**) Control datasets are not expected to display prominent fragment length peaks and are expected to have low NSC and RSC scores, in contrast to the high-quality ChIP-seq datasets (**d** and **e**)

```
mkdir SPP-300K-wgEncodeHaibTfbsH1hescTaf1V0
416102RawDataRep1
the pooled pseudoreplicates:
SPP-300K-wgEncodeHaibTfbsH1hescTaf1V0416102
RawDataPseudoRep1
SPP-300K-wgEncodeHaibTfbsH1hescTaf1V0416102
RawDataPseudoRep2
and the individual pseudoreplicates:
mkdir SPP-300K-wgEnco
deHaibTfbsH1hescTaf1V0416102RawDataRep1-
PseudoRep1
mkdir SPP-300K-wgEncodeHaibTfbsH1hescTaf1V0
416102RawDataRep2-PseudoRep1
mkdir SPP-300K-wgEncodeHaibTfbsH1hescTaf1V0
416102RawDataRep1-PseudoRep2
mkdir SPP-300K-wgEncodeHaibTfbsH1hescTaf1V0
416102RawDataRep2-PseudoRep2
```

2. Run SPP on individual replicates:

```
Rscript run_spp.R -c=wgEncodeHaibTfbsH1hesc
Taf1V0416102RawDataRep1.unique.bam -i=wgEnc
odeHaibTfbsH1hescRxlchV0422111RawDataRep1.
unique.bam -p=16
-npeak=300000 -savr -savp –rf -odir=SPP-
300K-wgEncodeHaibTfbsH1hescTaf1V0416102RawD
ataRep1
```

3. Merge the BAM files for the individual ChIP and Control replicates:

```
samtools merge wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.bam
wgEncodeHaibTfbsH1hescTaf1V0416102Raw-
DataRep1.unique.bam wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep2.unique.bam
```

4. Sort the merged BAM files:

```
samtools sort wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.bam wgEnco-
deHaibTfbsH1hescTaf1V0416102RawData.pooled.
sorted
```

5. Index the sorted BAM file:

```
samtools index wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.sorted.bam
```

6. Generate pseudoreplicates for the pooled datasets (ChIP and Control):

```
python BAMPseudoReps.py wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.sorted.bam
```

7. Generate pseudoreplicates for each individual replicate (ChIP and Control):

```
python BAMPseudoReps.py wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.unique.bam
python BAMPseudoReps.py wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep2.unique.bam
```

8. Call peaks on the pooled dataset as previously shown.
9. Call peaks on the pooled pseudoreplicates as previously shown.
10. Call peaks on individual pseudoreplicates as previously shown.
11. Uncompress peak calls:
    ```
    gunzip SPP-300K*/*gz
    ```
12. Copy IDR files and genome tables to the current working directory:
    ```
    cp idrCode/*.* .;
    cp idrCode/genome_tables/genome_table.hu-
    man.hg19.txt genome_table.txt
    ```
13. Run IDR on individual replicates:
    ```
    Rscript batch-consistency-analysis.r SPP-
    300KwgEncodeHaibTfbsH1hescTaf1V0416102RawDat
    aRep1/wgEncodeHaibTfbsH1hescTaf1V0416102Raw-
    DataRep1.unique_VS_wgEncodeHaibTfbsH1he-
    scRxlchV0422111RawDataRep1.unique.region-
    Peak SPP-300K-wgEncodeHaibTfbsH1hescTaf
    1V0416102RawDataRep2/wgEncodeHaibTfbsH-
    1hescTaf1V0416102RawDataRep2.unique_VS_wgEn-
    codeHaibTfbsH1hescRxlchV0422111RawDataRep2.
    unique.regionPeak -1 IDR-SPP-wgEncodeHaibTf
    bsH1hescTaf1V0416102RawData 0 F signal.value
    ```
14. Run IDR on pooled pseudoreplicates as above.
15. Run IDR on individual pseudoreplicates as above.
16. Create IDR consistency plots for each replicate, pooled pseudoreplicate, and individual pseudoreplicate run as follows:
    ```
    Rscript batch-consistency-plot.r 1 IDR-SPP-
    wgEncodeHaibTfbsH1hescTaf1V0416102RawData
    IDR-SPP-wgEncodeHaibTfbsH1hescTaf1V0416102R
    awData
    ```
17. Examine the *npeaks-aboveIDR.txtand *overlapped-peaks.txt files to determine the *Nt*, *Np*, *N1*, *N2* values. Here, an IDR threshold of 0.02 is used for between-replicates self-consistency and a0.005 threshold for pseudoreplicate self-consistency (*see* **Note 15**). High *N1/N2* (where *N1* > *N2*) and *Np/Nt* values (for example, >2 [5]) indicate poor reproducibility.
    Nt:
    ```
    awk '$11 <= 0.02 {print $0}' IDR-SPP-wgE
    ncodeHaibTfbsH1hescTaf1V0416102RawData-
    overlapped-peaks.txt | wc -l
    ```

Np:

```
awk '$11 <= 0.005 {print $0}' IDR-SPP-
Pooled-pseudoreps-wgEncodeHaibTfbsH1hescTaf1
V0416102RawData-overlapped-peaks.txt | wc -l
```

N1:

```
awk '$11 <= 0.005 {print $0}' IDR-SPP-
Individual-Pseudoreps-wgEncodeHaibTfbsH1hes
cTaf1V0416102RawDataRep1-overlapped-peaks.
txt | wc -l
```

N2:

```
awk '$11 <= 0.005 {print $0}' IDR-SPP-
Individual-Pseudoreps-wgEncodeHaibTfbsH1hes
cTaf1V0416102RawDataRep2-overlapped-peaks.
txt | wc –l
```

18. Pick the top *max (Nt; Np)* peaks from the peak calls generated from the pooled ChIP and Control datasets:

```
cat SPP-300K-wgEncodeHaibTfbsH1hescTaf1V
0416102RawData.pooled/wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.sorted_VS_
wgEncodeHaibTfbsH1hescRxlchV0422111RawData.
pooled.sorted.regionPeak | sort -k7nr,7nr |
head -n 20001 | cat > SPP-300K-wgEncodeHaib
TfbsH1hescTaf1V0416102RawData.pooled/wgEnco-
deHaibTfbsH1hescTaf1V0416102RawData.pooled.
IDR0.02.regionPeak
```

Figure 5 displays the results of the IDR pipeline for the datasets discussed here.

3.4.2 Removing Known Artifacts

The resulting set of peaks is filtered against the set of known regions of artifactual enrichment (so-called blacklists):

1. Concatenate the two sets of blacklisted regions:

```
cat wgEncodeDacMapabilityConsensusExclud-
able.bed wgEncodeDukeMapabilityRegionsEx-
cludable.bed > wgEncodeBlacklists.bed
```

2. Intersect the post-IDR peaks with the blacklisted regions:

```
python regionIntersection.py SPP-300K-wgEnco
deHaibTfbsH1hescTaf1V0416102RawData.pooled/
wgEncodeHaibTfbsH1hescTaf1V0416102RawData.
pooled.IDR0.02.regionPeak 0 wgEncodeBlack-
lists.bed 0 SPP-300K-wgEncodeHaibTfbsH1hescT
af1V0416102RawData.pooled/wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.IDR0.02-vs-
Blacklist
```

The *outsersect1file contains the wanted peaks.

3.4.3 Calculating FRiP Scores

The FRiP metric is defined as the fraction of peaks falling within called regions and can be a useful complement to the

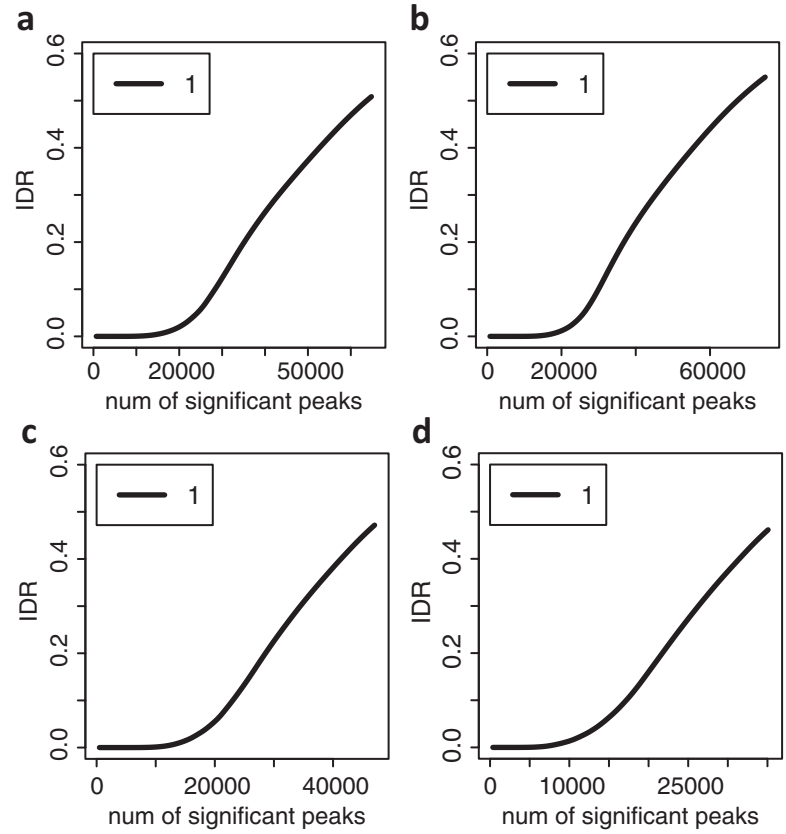


Fig. 5 Identifying reproducible peaks using IDR analysis. (**a**) Reproducible peaks between replicates (*Nt*); (**b**) Self-consistent peaks between pooled pseudoreplicates (*Np*); (**c** and **d**) Self-consistent peaks between individual replicates (*N1* and *N2*, respectively). Here, at IDR = 0.02, *Nt* = 20,001 (*Nt* = 17,367 at IDR = 0.01), and at IDR = 0.005, *Np* = 17,277, *N1* = 12,431, *N2* = 7973, *N1/N2* = 1.55 and *Nt/Np* = 1.15

cross-correlation metric even though it usually correlates well with it [5]. Here, it is calculated on the post-IDR set of peaks and for each replicate separately.

Calculate RPMs for each region and sum them:

```
python bedRPKMfromBAM.py SPP-300K-wgEncode
HaibTfbsH1hescTaf1V0416102RawData.pooled/
wgEncodeHaibTfbsH1hescTaf1V0416102Raw-
Data.pooled.IDR0.02-vs-Blacklist-
outersection1.sorted 0 wgEncodeHaibTfb-
sH1hescTaf1V0416102RawDataRep1.unique.
bam hg19.chrom.sizes wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.IDR0.02.
Rep1.RPM -RPM -printSum –uniqueBAM
```

The sum of RPM (Reads Per Million) values over all peaks is printed at the end of the output file. The FRiP value is equal to it divided by 1×10^6.

3.5 Data Visualization

The post-IDR and post-blacklist filtering peak calls are a generally reliable starting point for subsequent analysis, but it has to be kept in mind that individually each of them is still only a candidate regulatory region and orthogonal evidence is needed to confirm its status as such and clarify its role. The precise nature of the additional evidence varies depending on the type of regulatory element. The different strains of evidence include evolutionary conservation patterns, the presence of additional types of biochemical activity around the region, and others, with the gold standard being the direct experimental testing of biological function [23]. The detailed examination of data on a genome browser is a key component of this process. Here, the UCSC Genome Browser and binary bigWig and bigBed files [21] are used for this purpose, but the plain text files generated by the pipeline are general and can be uploaded to any genome browser. The advantage of bigWig and bigBed files is that they allow fast access to the current view on the browser without having to upload the whole track; they can be placed in a location visible to the internet and loaded on the browser as custom-track, bigBed/bigwig files, to be read directly from that location when needed. Note that if a large number of datasets are to be displayed, track data hubs [24] may be a more appropriate and convenient approach for organizing them, but they are beyond the scope of this chapter.

3.5.1 Genome Brower Data Tracks

1. Generate RPM-normalized total and strand-specific coverage tracks (*see* **Notes 16** and **17**):

```
python makewigglefromBAM-NH.py track_title
wgEncodeHaibTfbsH1hescTaf1V0416102RawDa-
taRep1.unique.bam genomes/hg19/hg19.chrom.
sizes wgEncodeHaibTfbsH1hescTaf1V0416102Raw-
DataRep1.unique.wig -uniqueBAM -RPM –notitle
python makewigglefromBAM-NH.py track_title
wgEncodeHaibTfbsH1hescTaf1V0416102RawDa-
taRep1.unique.bam genomes/hg19/hg19.chrom.
sizes wgEncodeHaibTfbsH1hescTaf1V0416102Raw-
DataRep1.unique.plus.wig -stranded +
-uniqueBAM -RPM -notitle
python makewigglefromBAM-NH.py track_title
wgEncodeHaibTfbsH1hescTaf1V0416102RawDa-
taRep1.unique.bam genomes/hg19/hg19.chrom.
sizes wgEncodeHaibTfbsH1hescTaf1V0416102Raw-
DataRep1.unique.minus.wig -stranded -
-uniqueBAM -RPM -notitle
```

2. Convert bedGraph/wig files to bigWig:

```
UCSC-tools/wigToBigWig wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.unique.plus.wig
genomes/hg19/hg19.chrom.sizes wgEncodeHai-
```

```
bTfbsH1hescTaf1V0416102RawDataRep1.unique.
plus.bigWig
```

3. Create bigBed files containing the final post-IDR peak call set. The input file needs to be first sorted by coordinates:

```
sort -k1,2V SPP-300K-wgEncodeHaibTfbsH1hesc
Taf1V0416102RawData.pooled/wgEncodeHaibTfb-
sH1hescTaf1V0416102RawData.pooled.IDR0.02-
vs-Blacklist-outersection1 > SPP-300K-wgEnco
deHaibTfbsH1hescTaf1V0416102RawData.pooled/
wgEncodeHaibTfbsH1hescTaf1V0416102RawData.
pooled.IDR0.02-vs-Blacklist-outersection1.
sorted
UCSC-tools/bedToBigBed -type=bed6+4 SPP-
300K-wgEncodeHaibTfbsH1hescTaf1V0416
102RawData.pooled/wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.IDR0.02-
vs-Blacklist-outersection1.sorted genomes/
hg19/hg19.chrom.sizes SPP-300K-wgEncodeH
aibTfbsH1hescTaf1V0416102RawData.pooled/
wgEncodeHaibTfbsH1hescTaf1V0416102RawData.
pooled.IDR0.02-vs-Blacklist-outersection1.
bigBed
```

4. Display the tracks on the UCSC Genome Browser through the custom track-loading interface as follows.

   bigBed files:

```
track type=bigBed name=IDR-SPP-HAIB-H1hesc-
Taf1 description=IDR-SPP-HAIB-H1hesc-
Taf1 visibility=full color=255,102,102
bigDataUrl=$URL/wgEncodeHaibTfbsH-
1hescTaf1V0416102RawData.pooled.IDR0.02-
vs-Blacklist-outersection1.bigBed
```

   bigWig files:

```
track type=bigWig name=HAIB-H1hesc-Taf1-
Rep1.minus description=HAIB-H1hesc-Taf1-
Rep1.minus visibility=full color=0,128,255
bigDataUrl=$URL/wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.unique.minus.
bigWig
track type=bigWig name=HAIB-H1hesc-Taf1-
Rep1.plus description=HAIB-H1hesc-Taf1-
Rep1.plus visibility=full color=255,51,51
bigDataUrl=$URL/wgEncodeHaibTfbsH-
1hescTaf1V0416102RawDataRep1.unique.plus.
bigWig
```

The color parameter can be varied (in RGB coordinates) to adjust the color display.

3.5.2 Mappability Tracks

Mappability tracks help visualize where in the genomes reads can and cannot be uniquely mapped, and thus better understand observed patterns of read coverage, especially in repetitive regions. Various approaches of differing sophistication have been proposed for evaluating mappability [25–27]. Here, a simple remapping of tilings of the genome back to itself is described in order to generate mappability tracks for the Bowtie aligner; this approach is generally sufficient for visually figuring out the repeat structure in a given region of the genome.

1. Generate synthetic reads (in this case 36 bases long) tiling the genome at every base pair for each chromosome (each chromosome is processed individually to avoid working with extremely large BAM files):
   ```
   python mappability-make_reads.py chr1.fa
   36 - | bzip2 > chr1.36mers.fa.bz2
   ```
2. Map the synthetic reads for each chromosome against the genome with the same settings used for ChIP-seq data:
   ```
   bzip2 -cd chr1.36mers.fa.bz2 | bowtie genomes/
   hg19/bowtie-indexes/hg19-male -p 16 -v 2 -k 2
   -m 1 -t --best --strata -f --sam - | samtools
   view -bT genomes/hg19/sequence/hg19.fa - |
   samtools sort - chr1.36mers.hg19-male.unique
   ```
3. Index the resulting BAM files:
   ```
   samtools index chr1.36mers.hg19-male.unique.
   bam
   ```
4. Generate mappability tracks for each chromosome (*see* **Note 18**)
   ```
   python makewigglefromBAM-NH.py track_title
   chr1.36mers.hg19-male.unique.bam /genomes/
   hg19/hg19.chrom.sizes chr1.36mers.hg19-male.
   unique.wig -notitle -uniqueBAM
   ```
   This may result in a very small number of alignments to other chromosomes, remove them as follows:
   ```
   grep -P 'chr1\t' chr1.36mers.hg19-male.
   unique.wig > chr1.36mers.hg19-male.unique.
   filtered.wig
   ```
5. Cat the individual chromosomes and convert to bigWig.
   ```
   cat chr*.36mers.hg19-male.unique.filtered.
   wig > hg19-male.36mers.unique.mappability.
   hg19-male.wig UCSC-tools/wigToBigWig hg19-
   male.36mers.unique.mappability.hg19-male.
   wig genomes/hg19/hg19.chrom.sizes hg19-
   male.36mers.unique.mappability.hg19-male.
   bigWig
   ```

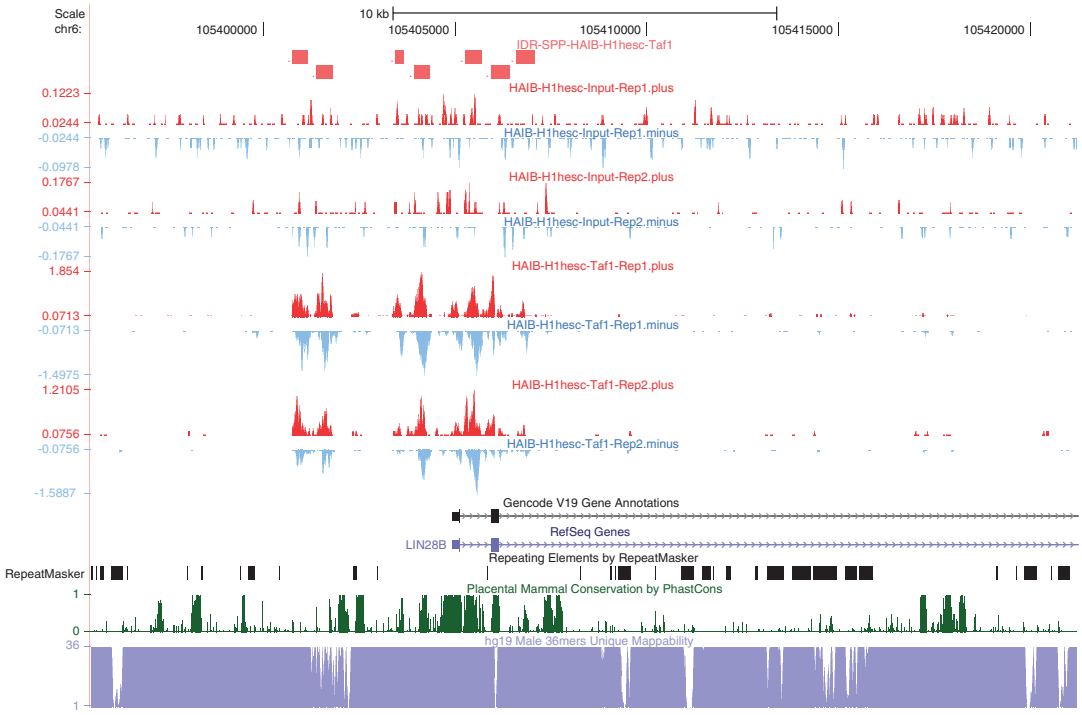


Fig. 6 Example of candidate novel regulatory elements. Shown is the 5′ region of the *LIN28B* gene (coding for an important regulator of stem cell renewal, which acts by repressing the *let-7* miRNA [28]). Multiple TAF1 peaks are observed upstream of the annotated transcription start site, representing possible alternative promoter elements

6. Display on the browser as shown before.

Figure 6 shows the region surrounding the annotated promoter of the LIN28B gene, with the post-IDR peak calls, the forward and reverse strand signal profiles for the two H1-hESC TAF1 ChIP and Control replicates, and the mappability track. Several possible alternative promoters, not present in either the refSeq or the GENCODE V19 annotations, are observed for this gene.

4 Notes

1. The choice of aligner is not absolutely crucial when ChIP-seq data is analyzed with the goal of identifying occupancy sites (it is of more significant importance when sequence polymorphisms in the data need to analyzed). The pipeline presented here is based on Bowtie1, but Bowtie2 [16] (also shown), BWA [29], as well as other aligners are also often used with mostly equally good results. The requirements for the alignments are that only unique alignments are retained (properly dealing with multimappers is a complicated subject beyond the

scope of this chapter) and soft-clipped alignments are not allowed (the latter is not an absolute requirement, but many of the steps described do assume a constant read length and it is preferable that the alignments conform to that assumption).

2. The underlying principle of IDR analysis is the separation of the two sets of features that are being compared into reproducible and irreproducible noise components. In order for this to be accomplished, there has to be a significant noise component present in the data. This requirement is opposed to the goal of the default settings of peak calling algorithms, which is to produce a set of peaks as devoid of noise as possible. It is therefore necessary to carry out peak calling with greatly relaxed relative to the default settings so that the noise component is present.
3. In this chapter, peak calling using SPP [19] is described; however, a number of other peak callers have also been successfully used with IDR (MACS2 [30], PeakSeq [31], GEM [32], and others). The applicability of peak callers to IDR analysis is determined by whether they can be run in the "overcall" mode required for the IDR algorithm to be able to separate the noise from reproducible components, and by how they rank individual peaks (for example, it is preferable that peak callers do not generate very large numbers of ties in rankings). The detailed parameters and potential output issues need to be optimized and characterized for each peak caller individually.
4. The reasons for poor reproducibility can be varied. A typical situation involves a pair of a strong and weak ChIP-seq datasets, in which the IDR analysis is dominated by the weak replicate. Alternatively, the experiments may not have been carried out under properly matched biological conditions, there might have been differences in the lots of antibodies used to carry out the ChIPs, etc. Such cases are cause for concern, should be investigated in detail, and the experiments repeated, if practically possible.
5. If more than two replicates are available, the final set of peaks can be derived by finding the maximum number of reproducible peaks *Nt* between all pairs of replicates and by using it to threshold on the peaks called on the pooled sets of reads.
6. The current human and mouse genome assemblies contain a number of "random" chromosomes, which represent sequences known to exist in the genome but not yet properly placed within the chromosomes. These are often not considered during alignment (for example, this was the policy of the ENCODE Consortium [7]). The hg19 and hg20 versions of the human genome also contain a number of "alt"/"hap" chromosomes, which represent alternative sequences for certain regions of the

genome that are commonly found in the population. Both the "random" and the "alt"/"hap" chromosomes are often places where reads can be aligned to. This includes both reads that would otherwise not align at all and reads that in their absence would erroneously align elsewhere (*see* [33] for a detailed discussion of the current status and significance of "alt"/"hap" segments). However, if the "alt"/"hap" sequences are included in the index, they also have the effect of erroneously making otherwise uniquely mappable reads multimappers, eliminating them from subsequent analysis. This is not the case with the random contigs, thus the optimal policy is to include the "random" chromosome and exclude the "alt"/"hap" chromosomes. When the sex of the source of the biological material assayed is known, it is preferable to align against the corresponding male or female version of the genome.

7. The chrom.sizes files are used to indicate the identity and end points of chromosomes to a number of programs. They are in the following format:
   `chr<tab> chromosome_size`
8. Removal and/or trimming of low-quality reads/bases from reads is essential for a number of genomic applications. It is not strictly necessary for ChIP-seq if the goal is to simply call peaks and the overall sequencing quality is as good as it usually is. It becomes important if additional information, such as allelic bias in occupancy, is to be extracted. In addition, it is preferable that the read length is kept constant, *see* also **Note 1**.
9. If most of the reads mapped (typically more than 80% of ChIP-seq reads align successfully to a mammalian genome, including the ones whose alignments are suppressed due to high multiplicity), then most likely there are no issues with read quality. While FastQC is very useful for spotting issues with reads in general, its application is particularly helpful in the cases when few reads align, due to issues such as general drop-offs in read quality after certain sequencing cycle, the presence of adapter and barcode sequences, and others, information about which is provided by FastQC. Identification of these problems can help rescue datasets by trimming reads accordingly prior to alignment (the trimfastq.py provides various utilities for trimming reads from either the 5′ or 3′ end).
10. It is a standard practice for a number of genomic applications to collapse apparent duplicate reads/fragments. Such groups of reads/fragments may represent potential PCR duplicates, but they can also be distinct but identical in sequence molecular fragments (the latter is more likely with short single-end reads). Such "dedup"-ing is sometimes performed on ChIP-seq datasets. However, as first, it is not necessary for the peak

calling procedures described here, and second, it defeats the purpose of the molecular complexity characterization steps if it is applied before them; here its application is advised against.

11. Cross-correlation analysis is based on the clustering of unpaired reads on opposite strands around sites in the genome enriched in the dataset [5, 6]. The assumption that the reads are unpaired is critical—if cross-correlation is run on a paired-end dataset, results will be meaningless as there will always be a paired read on the other strand at a distance in the neighborhood of the average fragment length. Another consideration to keep in mind is that if long read lengths are generated (e.g., 100 mers), the average fragment length can be very close to, or even shorter than the read length, in which case there will be no separation of the "phantom" peak and the fragment length peak. For these reasons, it is recommended that this step is carried out on single-end 36 bp or 50 bp alignments. Finally, the cross-correlation metrics have been calibrated for genomes with size and repeat structure similar to those of mammals. The baseline profile (i.e., what should be observed in input samples) can look very different in organisms whose genomes significantly deviate from these characteristics, and the metrics are not directly applicable in exactly the same form in such cases.
12. Cross-correlation analysis is carried out on both ChIP and Control samples. In ChIP samples, prominent fragment peaks and high values of the NSC and RSC metrics are desired; however, there are some caveats to be aware of as they can in fact be artifactual. For example, sometimes very high such peaks are observed in both the ChIP and the Control, suggesting that they originate from sources other than real ChIP enrichment. Such cases illustrate the need to carry out cross-correlation analysis of both ChIP and Control samples—in Control samples, prominent cross-correlation peaks are not expected, and if they are observed, this can be a sign that the enrichment patterns in the ChIP sample might be generated by a mixture of real ChIP signal and another source of enrichment, the latter being artifactual [6].
13. In prior studies [5, 6] cutoffs based on the RSC metric have been applied to divide datasets into high-, intermediate, and low-quality groups, for example: RSC $\in (0; 0.25)$ => QC = −2, RSC $\in (0.25; 0.5)$ => QC = −1, RSC $\in (0.5; 1)$ => QC = 0, RSC $\in (1;1:5)$ => QC = +1, RSC > 1.5 => QC = +2, with −2 corresponding to minimal read clustering and 2 to a highly clustered library (this is also the discretization used here). These are useful guidelines but are not on their own absolutely trustworthy metrics to be blindly used, first, because they represent discretizations of inherently continuous variables, and second, because various factors can lead to very high or very low scores

in an otherwise poor or high-quality dataset, respectively (*see* the issues discussed in **Notes 12** and **11**). For such reasons, there is no substitute for the manual examination of cross-correlation plots when evaluating libraries.

14. There are three different types of ChIP-seq datasets: "point-source," "broad-range," and mixed. Point-source datasets are best exemplified by sequence-specific factors, which bind to very precisely defined and short in length (in most cases 6–10 bp) sequences in the genome, generating the classic asymmetric read distribution pattern around binding sites. Most peak callers are tuned to find regions of enrichment with point-source characteristics. In contrast, the computational definition of "broad-range" regions of enrichment is more challenging. Such ChIP-seq datasets are classically observed when targeting histone modifications associated with transcriptional elongation (H3K36me3, H3K79me2) and with broad repression domains (H3K27me3, H3K9me3). Mixed-source datasets contain both regions with point-source and broad-range characteristics (the main representative of this group being RNA polymerase II). The pipeline described here is focused on the characterization of point-source enrichment regions. Fortunately, this characterizes most well-defined candidate regulatory regions (histone marks associated with active enhancers and promoters often approximate the point-source patterns, insulator proteins bind in a very localized manner, and sequence-specific transcription factor binding is the hallmark of enhancer and promoter elements).
15. With the SPP settings used here and mammalian-sized genomes, an IDR = 0.02 cutoff works fine, but the threshold can be increased or decreased if needed depending on the peak caller used (for example, if a smaller number of peaks are used as input to the IDR analysis) and the nature of the dataset. A more stringent threshold is applied for pseudoreplicates as the pooling and splitting process naturally leads to higher levels of reproducibility.
16. It is preferable to separate the forward and reverse strands when displaying ChIP-seq data as the patterns on the two strands provide important information about the nature of the observed enrichment patterns. Regions of artifactual enrichment tend to lack the strand asymmetry that characterizes classical occupancy sites, a difference that can only be readily identified when the signal profiles on the two strands are examined in parallel.
17. RPM normalization is important for the direct comparison of different datasets. Non-normalized tracks with the *y* axis corresponding to the total number of reads cannot be directly

compared to each other if the sequencing depth of the two datasets is significantly different.

18. Note that the mappability track generated as shown here has a range of scores $s \in [0;RL]$, where RL is the read length.

Acknowledgments

The author wishes to thank Anshul Kundaje, members of the Barbara Wold and Richard Myers labs and of the ENCODE Consortium for many helpful discussions, and Gilberto DeSalvo and Matthew D. Smalley for critical reading of the manuscript.

References

1. Shyh-Chang N, Daley GQ (2013) Lin28: primal regulator of growth and metabolism in stem cells. Cell Stem Cell 12:395–406
2. Barski A, Cuddapah S, Cui K et al (2007) High-resolution profiling of histone methylations in the human genome. Cell 129:823–837
3. Johnson DS, Mortazavi A, Myers RM et al (2007) Genome-wide mapping of in vivo protein-DNA interactions. Science 316: 1497–1502
4. Mikkelsen TS, Ku M, Jaffe DB et al (2007) Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. Nature 448:553–560
5. Robertson G, Hirst M, Bainbridge M et al (2007) Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. Nat Methods 4:651–657
6. Landt SG, Marinov GK, Kundaje A et al (2012) ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. Genome Res 22:1813–1831
7. Marinov GK, Kundaje A, Park PJ et al (2014) Large-scale quality analysis of published ChIP-seq data. G3 (Bethesda) 4:209–223
8. ENCODE Project Consortium (2012) An integrated encyclopedia of DNA elements in the human genome. Nature 489:57–74
9. Gerstein MB, Kundaje A, Hariharan M et al (2012) Architecture of the human regulatory network derived from ENCODE data. Nature 489:91–100
10. Gerstein MB, Lu ZJ, Van Nostrand EL et al (2010) Integrative analysis of the *Caenorhabditis elegans* genome by the modENCODE project. Science 330:1775–1787
11. modENCODE Consortium (2010) Identification of functional elements and regulatory circuits by *Drosophila* modENCODE. Science 330:1787–1797
12. The Mouse ENCODE Consortium (2014) A comparative encyclopedia of DNA elements in the mouse genome. Nature 515:355–364
13. Negre N, Brown CD, Ma L et al (2011) A cis-regulatory map of the *Drosophila* genome. Nature 471:527–531
14. Li Q, Brown J, Huang H et al (2011) Measuring reproducibility of high-throughput experiments. Ann Appl Stat 5:1752–1779
15. Carroll TS, Liang Z, Salama R, Stark R, de Santiago I (2014) Impact of artifact removal on ChIP quality metrics in ChIP-seq and ChIP-exo data. Front Genet 5:75
16. 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
17. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9:357–359
18. H L, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, 1000 Genome Project Data Processing Subgroup (2009) The Sequence Alignment/Map format and SAMtools. Bioinformatics 25:2078–2079
19. Daley T, Smith AD (2013) Predicting the molecular complexity of sequencing libraries. Nat Methods 10:325–327
20. Feng J, Liu T, Qin B et al (2012) Identifying ChIP-seq enrichment using MACS. Nat Protoc 7:1728–1740
21. Kuhn RM, Haussler D, Kent WJ (2013) The UCSC genome browser and associated tools. *Brief Bioinform* 14:144–161
22. Kent WJ, Zweig AS, Barber G et al (2010) BigWig and BigBed: enabling browsing of

large distributed datasets. Bioinformatics 26:2204–2207

23. Thomas MC, Chiang CM (2006) The general transcription machinery and general cofactors. Crit Rev Biochem Mol Biol 41:105–178
24. Kellis M, Hardison RC, Wold BJ et al (2014) Defining functional DNA elements in the human genome. *Proc Natl Acad Sci U S A* 111:6131–6138
25. Raney BJ, Dreszer TR, Barber GP et al (2014) Track data hubs enable visualization of user-defined genome-wide annotations on the UCSC Genome Browser. Bioinformatics 30: 1003–1005
26. Koehler R, Issac H, Cloonan N, Grimmond SM (2011) The uniqueome: a mappability resource for short-tag sequencing. Bioinformatics 27:272–274
27. Lee H, Schatz MC (2012) Genomic dark matter: the reliability of short read mapping illustrated by the genome mappability score. Bioinformatics 28:2097–2105
28. Derrien T, Estell´e J, Marco Sola S et al (2012) Fast computation and applications of genome mappability. PLoS One 7:e30377
29. Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25:1754–1760
30. Kharchenko PV, Tolstorukov MY, Park PJ (2008) Design and analysis of ChIP-seq experiments for DNA-binding proteins. Nat Biotechnol 26:1351–1359
31. Rozowsky J, Euskirchen G, Auerbach R et al (2009) PeakSeq enables systematic scoring of ChIP-seq experiments relative to controls. Nat Biotechnol 27:66–75
32. Guo Y, Mahony S, Gifford DK (2012) High resolution genome wide binding event finding and motif discovery reveals transcription factor spatial binding constraints. PLoS Comput Biol 8:e1002638
33. Church DM, Schneider VA, Steinberg KM et al (2015) Extending reference assembly models. Genome Biol 16:13

http://www.springer.com/978-1-4939-6714-8

Promoter Associated RNA
Methods and Protocols
Napoli, S. (Ed.)
2017, X, 287 p. 283 illus., 182 illus. in color., Hardcover
ISBN: 978-1-4939-6714-8
A product of Humana Press