Tutorial for "Improving Resolution of Protein Binding Sites by filtering conserved ultra-high signals in Arabidopsis"

Samantha Klasfeld[1] and Doris Wagner[1]

[1]Department of Biology, University of Pennsylvania

July 28, 2021

Contents

1 Methods

The pipeline used for this paper to generate green screens and perform ChIP-seq analysis is explained in detail below. A familiarity in the UNIX shell environment is encouraged. Code run using the shell environment of the console will be framed with orange lines, and a dollar sign ($) at the beginning of each line indicates lines of code, but a pound symbol (#) indicates a comment. Text files and scripts can be found in the github repository: https://github.com/sklasfeld/GreenscreenProject. Scripts are located in the 'scripts' directory in the repository, and text files are located in the 'meta' directory. A Dockerfile in the github repository contains the environment to run the following commands. All the paths listed below are based on this Docker environment.

1.1 Materials

The scripts are written in the following languages:

- BASH
  - Version 5.0.17(1)-release (x86_64-pc-linux-gnu)
- python
  - Python 3.8.10
- R
  - R version 4.0.0

External software includes:

- Docker v20.10.8: Generates a container containing necessary scripts and software environment [18]

Inside the docker container the following software should be installed:

- sratools v2.11.1: The NCBI SRA toolkit to download sequencing libraries
- FastQC v0.11.9: measures base and sequence quality [1]
- Trimmomatic v0.39: used to trim out adapters, trim low quality bases from the ends of reads, and remove low quality reads [3]
- samtools v1.9: modify sam/bam files, count mapped reads [16]
- Picard 2.26.0: identify duplicate reads [20]
  - openjdk version "1.8.0_191"
  - OpenJDK Runtime Environment (build 1.8.0_191-b12)

  - OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
- Bowtie2 2.4.4: maps reads to genome [12]
  - Compiler: gcc version 8.3.1 20170224 (experimental) (GCC)
- MACS2 2.2.7.1 [34]
- BEDTools v2.27.1 [27]
- kentUtils v419 [10]
- deeptools v3.5.1 [21]

1.2 Docker Workspace

To make this tutorial easier it is recommended to install git and Docker. To manage docker as a non-root user (and avoid using *sudo* for every docker command) see https://docs.docker.com/engine/install/linux-postinstall/. A Dockerfile in the github repository defines a container with the necessarily scripts, software, and environment. Before the docker file can be used, clone the github repository and enter the repository as your working directory.

```
# clone repository
$ git clone https://github.com/sklasfeld/GreenscreenProject.git
# enter repository as your working directory
$ cd GreenscreenProject
```

Build the docker image using the command: `docker build .`

The last string of letters and numbers in the output is the image ID. For example, the end of the output should look something like this:

```
 ---> Running in e60e511455d1
Removing intermediate container e60e511455d1
 ---> ef5749b6dfa5
Successfully built ef5749b6dfa5
```

Above the image ID is 'ef5749b6dfa5'. To confirm this image ID, use the following command: `docker image ls`. To run and generate an interactive container using this image type the following in to the command line (replace the text and brackets with the correct information):

```
$ docker run --name [CONTAINER_NAME] -i -t [IMAGE_ID] bash
```

In the example above [IMAGE_ID] is e60e511455d1. [CONTAINER_NAME] can be set to whatever name the user wants to give the new docker container.

The interactive environment will open to working directory '/home/'. Note that all commands specified are from within this working directory in the container unless specified otherwise. Files generated within the local or Docker container can always be transferred back and forth by opening a new terminal and running the `docker cp` command. The format to transfer a file from the container is:

```
$ docker cp [CONTAINER_NAME]:[PATH/IN/CONTAINER]
↪  [PATH/IN/LOCAL/COMPUTER]
```

The format to transfer a file into the container is the opposite:

```
$ docker cp [PATH/IN/LOCAL/COMPUTER]
↪  [CONTAINER_NAME]:[PATH/IN/CONTAINER]
```

If the docker container is exited (ie. your computer shuts down), one can always restart and reattach the interactive container with the following commads:

```
$ docker container start [CONTAINER_NAME]
$ docker attach [CONTAINER_NAME]
```

1.3 Green screen generation

1.3.1 Mapping the input samples

The raw sequences of the twenty inputs in Table S2 are retrieved using NCBI SRA Tools (`scripts/import_raw_fasta_inputs.sh`). To organize these fastq files, the files are compressed and moved into the 'fastq/raw' directory and renamed based on the sample ID given to then in Table S2 (`scripts/organize_raw_fasta_input.sh`). Note that InputJ contains three run files; these reads are concatenated and the individual run files are deleted.

Usually adapters are already trimmed before reading in the sequences. Adapters and overlap sequence quality are checked using FASTQC (`scripts/fastqc_raw_inputs.sh`). The FASTQC outputs in Inputs A,J,K,M,O,R,S showed TruSeq3 adapters remaining. Fortunately, Trimmomatic was granted permission to distribute Illumina adapter and other technical sequences that were copyrighted by Illumina. Therefore, trimmomatic provides adapter sequences called TruSeq2 (adapters normally used in GAII machines) and TruSeq3 (adapters normally used in HiSeq and MiSeq machines) adapters for both single-end and paired-end mode. An adapter directory is created in the working directory, and the TruSeq3 adapters by Trimmomatic are copied into the adapter directory.

```
$ mkdir adapters # create an adapter directory
# copy correct adapter file from Trimmomatic
$ cp /usr/src/Trimmomatic-0.39/adapters/TruSeq3-SE.fa \
    adapters
```

Trimmomatic is used to trim any low quality or N bases from the read ends and remove any reads less than 36bp long, and then FASTQC is run on the trimmomatic output (`scripts/trimmomatic_inputs.sh`). The Bowtie2 aligner was chosen for mapping the reads to the genome. Before mapping, Bowtie2 must first index a genome fasta file. The *Arabidopsis* genome fasta file, TAIR10_Chr.all.fasta, can be downloaded from the Araport's website (https://www.araport.org/downloads/TAIR10_genome_release) under the "TAIR10_genome_release" directory [6]. Note that the TAIR10 genome may need to be decompressed using the *gunzip* command. An output directory is created for the indexed genome files and then index the genome with Bowtie2.

```
# create directory for the indexed Bowtie2 files
$ mkdir -p meta/ArabidopsisGenome/bowtie2_genome_dir
# run Bowtie2 genome indexing
$ bowtie2-build meta/ArabidopsisGenome/TAIR10_Chr.all.fasta \
    meta/ArabidopsisGenome/bowtie2_genome_dir/TAIR10
```

Default parameters are used to map the reads with Bowtie2 which outputs a SAM-format file. When Bowtie2 is finished, the SAM file is sorted with samtools, the reads are compressed into BAM format, and then reads are then indexed. Next, reads are filtered out that are bowtie2 labeled as unmapped, not a primary alignment, failed the platform/vendor quality checks, mapped to a chromosome outside the nucleosome, or does not have MAPQ$\geq$30. Duplicates are marked with PICARD (`scripts/mapped_inputs.sh`).

1.3.2 Input Quality Control

A key result of sequencing reads immunoprecipitated by a ChIP-seq antibody is a strand bias signature, when mapped reads form two distinct clusters about a fragment size apart and are enriched with reads biased by reads directed towards the opposite peak [13]. Enrichment of specific distances between strand clusters is calculated using a Pearson linear correlation between the opposing Watson and Crick strands after shifting the Watson strand by k base pairs. Enrichment of these correlations is visualized using a strand bias correlation plot. For ChIP-seq samples, these plots should show the highest enrichment at the experiment's fragment size value. On the other hand, reads from input controls did not originate from immunoprecipitated DNA, and therefore should only a strong peak at the read size.

Strand bias correlation plots and relative strand-bias cross correlation (RelCC) are calculated for each sample using the ChIPQC library. RelCC

quanitifies the enrichment of a peak around the fragment size (strand bias signal) in strand bias correlation plots divided by enrichment at the read length (no strand bias signal). High-quality ChIP-seq data sets tend to be enriched for peaks with strand-bias (RelCC>.8), whereas inputs have little or no such bias [13]. A custom script (`scripts/ChIPQC.R`) applies this library.

To run the ChIPQC.R script we need the following:

- **Sample sheet**: a file containing a csv table describing each sample. Required column headers include: 'SampleID' and 'bamReads'. The 'SampleID' column contains IDs for each sample. The 'bamReads' column contains paths to the mapped reads in bam format with respect to the row 'SampleID'. Later, to analyze ChIP-Seq samples, one can add a column header 'Peaks' for a column which should contain paths to narrowPeak file containing the peaks called by MACS2 with respect to the row 'SampleID'. Other columns can futher describe the experient (ie. replicate number, treatment, genotype).
- **Genome Annotation**: a file containing features in the genome in GFF format
- **Chromosome sizes**: a file counting a tab-delimited 2-column table with no header. The first column contains the chomosome names and the second column contains the respective chromosome sizes in basepairs.

A sample sheet to check the ChIP quality of the Input-seq experiments is `meta/noMaskReads_Inputs_sampleSheet.csv`. Genome annotations can be found online. The *Arabidopsis* genome annotation `meta/ArabidopsisGenome/Araport11_GFF3_genes_transposons.201606.gff.gz` was found at: https://datacommons.cyverse.org/browse/iplant/home/araport/public_data/Araport11_Release_201606/annotation. It is transferred to the docker container using the "docker cp" command to the docker container path: /home/meta/ArabidopsisGenome. This gff file is written using unicode percent encodings. These encodings can be translated using the following script `scripts/translateUniCodeFile.py`.

```
$ chmod +x scripts/translateUniCodeFile.py
$ gunzip
↪  meta/ArabidopsisGenome/Araport11\_GFF3\_genes\_transposons.201606.gff.gz
$ ./scripts/translateUniCodeFile.py \
  meta/ArabidopsisGenome/Araport11_GFF3\_genes_transposons.201606.gff
   ↪  \

   ↪  meta/ArabidopsisGenome/Araport11_GFF3_genes_transposons.UPDATED.201606.gff
```

Arabidopsis chromosome sizes were found using the genome assembly fasta file, `meta/ArabidopsisGenome/TAIR10_Chr.all.fasta.gz`. The text

file, `meta/ArabidopsisGenome/TAIR10_chr_count.txt`, is generated by decompressing the genome assembly file using the *gunzip* command and then running an awk command in `scripts/measureContigLengthFromFasta.sh`:

```
$ bash scripts/measureContigLengthFromFasta.sh \
    meta/ArabidopsisGenome/TAIR10_Chr.all.fasta \
    > meta/ArabidopsisGenome/TAIR10_chr_count.txt
```

For more details on the custom ChIPQC script, the following command can be used:

```
$ Rscript scripts/ChIPQC.R --help
```

To run this custom script on each of the ChIP-seq samples the *Arabidopsis* genome annotation file is decompressed using *gunzip* and the following command is run on the custom R script:

```
$ Rscript scripts/ChIPQC.R \
    --indivReports -g Araport11 \
    -c Chr1 Chr2 Chr3 Chr4 Chr5 \
    -a
    ↪  meta/ArabidopsisGenome/Araport11_GFF3_genes_transposons.201606.gff
    ↪  \
    -s meta/ArabidopsisGenome/TAIR10_chr_count.txt \
    meta/noMaskReads_Inputs_sampleSheet.csv \
    data/ChIPQCreport/20inputs_noMask
```

Output of ChIPQC will appear in the path `data/ChIPQCreport/20inputs_noMask`. Metrics should be skewed due to reads in artificial signal regions. Once we have developed the list of greenscreen regions, the reads that overlay green screen regions can be masked out and the remaining reads can be applied to the ChIPQC library. After masking the metrics should show cleaner input samples.

1.3.3 Visualize Mapped Input-Seq Sequences

To visualize the inputs genome-wide the read signal was normalized by each sample's total sequencing depth using BEDTOOLS genomeCoverageBed function. BEDTOOLS exports a file in BEDGRAPH format which is compressed into BIGWIG format using kentUtils bedGraphToBigWig function in `scripts/bamToBigwig_input.sh` [10].

1.3.4 Call Greenscreen Regions With Inputs

To call regions of significant artificial signal within each input, the read sizes of each sample must be set into MACS2. A text file, `meta/input_readsizes.csv`, contains a comma-delimited table where each input ID is in the first column and the respective sample read length in the second column. Using this text file as a template, peaks are called on each input sample using (`scripts/macs2_callpeaks_inputs.sh`).

MACS2 outputs peaks are then filtered by removing all peaks in Chloroplast and Mitochondia (since we are only interested in nuclear chromosome) and all peaks that do not have an average base pair q-value $<= 10^{-10}$. After peaks have been filtered for each input experiment, the peaks from all of the input experiments are pooled. Then, any regions that overlap or are within the distance of 1kb are merged. Note that information about the distinct inputs that made up a called region are retained. Given 20 inputs, regions that are called in less than 10 distinct samples are dropped to get the final greenscreen regions. The parameters for the final greenscreen are set using the following script: `scripts/generate_20input_greenscreenBed.sh`.

```
$ bash scripts/generate_20input_greenscreenBed.sh 10 5000 10
```

1.4 Calling ChIP-Seq peaks

1.4.1 Mapping the input samples

The raw sequences of the ChIP-Seq samples and the respective MACS2 controls in Table S3 are retrieved using wget and NCBI SRA Tools (`scripts/import_raw_fasta_publishedChIPsAndControls.sh`). To compress and organize the files within the working space, fastq files are compressed, if not already, moved into the 'fastq/raw' directory, and renamed with reference to Table S3 (`organize_raw_fasta_publishedChIPsAndControls.sh`). For the next steps the code uses a text file containing all the sample ID names, `meta/chip_controls_sampleIDs.list`. Before trimming the reads, adapters and overlap sequence quality are checked using FASTQC (`fastqc_raw_publishedChIPsAndControls.sh`).

Each of the samples were sequenced using HiSeq machines, therefore Trimmomatic is applied to each sample given the same TruSeq3 adapters that were used previously for the input samples to generate the green screen (`scripts/trimmomatic_publishedChIPsAndControls.sh`). Low quality or N bases from the read ends are trimmed, and reads less than 36bp long are removed. FASTQC is then applied to quanitify the quality of the trimmed reads.

Reads from each replicate are mapped to the genome with Bowtie2 and processed the same way as was done for the input samples used to generate the greenscreen (`scripts/mapped_publishedChIPsAndControls.sh`). The default parameters of Bowtie2 are used, and samtools commands are used to fil-

tered reads out that are unmapped, not a primary alignments, failed the platform/vendor quality checks, mapped to a chromosome outside the nucleosome, or do not have MAPQ≥30. Duplicates are marked with PICARD. The final analysis includes the marked duplicates, however, they can be removed with samtools.

1.4.2 ChIP-Seq Quality Control

The ChIPQC library is used to quantify the strand bias in each ChIP-Seq experiment and control. However, reads within artificial signals can overpower true ChIP signals. To remove many of the reads in artificial signals, all reads that overlap greenscreen regions are masked and then the ChIPQC library is run using the script `scripts/ChIPQC_gsMaskReads_publishedChIPsAndControls.sh` which requires a samplesheet with paths to masked mapped reads `meta/gsMaskReads_publishedChIPsAndControls_sampleSheet.csv`.

ChIPQC reports a strand cross-correlation plot and estimated fragment size for each experimental replicate. The largest estimated fragment size reported in a samples replicates were used to extend reads to improve signal visualization and call peaks.

1.4.3 Call ChIP-Seq peaks

To call peaks using multiple replicates or use multiple MACS2 controls one must first control for non-uniform read depth in replicates. MACS2 does not provide a function to normalize multiple BAM files. Therefore, before importing multiple BAM files into MACS2, replicates should be randomly down-sampled to match the replicate with the lowest read depth [30]. `scripts/downSampleBam_publishedChIPsAndControls.sh` generates downsampled BAM-files for the ChIP-seq and control samples to use for pooling in MACS2 in directory "mapped/chip/downsample".

To batch run MACS2 on either each ChIP replicate or the pooled replicates, a meta-table is created to match the ChIP sample names to the number of replicates, fragment size, the MACS2 control sample name, and the number of replicates in the MACS2 control `meta/chip_controls_fragsize_nreps.csv`. Note that if a publication contained both a mock and input control that the MACS2 control was set to the input control. MACS2 keeps duplicates based on the expected maximum tags at the same location with the setting "keep-dup auto", extends the single-end reads to the set fragment length, and calls peaks `scripts/macs2_callpeaks_publishedChIPs.sh`. Then, using MACS2 narrowPeak output, peaks are removed if they overlap greenscreen regions or have summit q-value $> 10^{-10}$.

1.4.4 Visualize Mapped ChIP-Seq Sequences

To visualize next generation sequencing normally one must simply normalize the signal based on the sequencing depth. However, for ChIP-Seq one must

consider an extra step to see signal where the protein of interest binds.

Due to the technical pipeline of ChIP-seq, reads form strand-biased clusters around sites where the protein of interest was immunoprecipitated. Therefore, to visualize the binding sites of a protein of interest, one must normalize the samples and also expand mapped reads towards their 3' end.

First we need a table of the fragment size of each sample, `meta/chip_readsize_fragsize.csv`. Then we can run the script `scripts/bamToBigWig_publishedChIPsAndControls_replicates.sh` to generate bigwig files for IGV. To visualize the signal after downsampling and pooling the replicates, the downsampled replicates are processed the same way into bedgraph format, then, using bedtools and the awk function, the average signal is reported in bigwig format (`scripts/bamToBigWig_publishedChIPsAndControls_pooled.sh`). Note that read extension is not needed to visualize the published ChIP-Seq controls.

1.5 ChIP-Seq Analysis

1.5.1 Annotate LFY ChIP-Seq Peaks

LFY ChIP-Seq peak summits are annotated to genes in two rounds. In round one, summits are first matched to genes with which the summits are intragenic. Of the non-intragenic summits, matches are made between the nearest genes within 3kb of which summits are upstream to the gene. Round 2 involves annotating orphan peak summits, ones that were not already annotated in round 1, to the nearest LFY dependent genes within 10kb of called summits. Differential expression output from four RNA-Seq experiments are found in `meta/lfy_rna_diffExp`.

The Araport11 GFF file contains several details about each of the feature groups in the annotated genome. The annotations for the gene feature group are expanded into a tab-delimited table format using `scripts/gff2annTable.py`.

```
$ python3 scripts/gff2annTable.py
↪  meta/ArabidopsisGenome/Araport11_GFF3_genes_transposons.201606.gff
↪  meta/ArabidopsisGenome/Araport11_GFF3_genesAttributes.tsv -f gene
```

Given the annotation table, an awk command is used to generate a bed file limited to non-hypothetical protein-coding genes and miRNA.

```
$ awk 'BEGIN{OFS="\t"} \
    (NR>1 && $14!~"hypothetical" && \
    ($7=="protein_coding" || $7=="mirna")){ \
    print $2,$3,$4,$1, $5,$6}' \
    meta/ArabidopsisGenome/Araport11_GFF3_genesAttributes.tsv > \
   meta/ArabidopsisGenome/Araport11_GFF3_true_protein_miRNA_genes.bed
```

To organize all of the annotation information, a custom script was generated at https://github.com/sklasfeld/ChIP_Annotation. This repository is cloned in the docker container with the following commands:

```
# change working directory to the scripts subdirectory
$ cd /home/scripts

# clone the ChIP_Annotation repo
$ git clone https://github.com/sklasfeld/ChIP_Annotation.git

# reset the working directory
$ cd /home/
```

To use this annotation script to annotate the LFY peak summits, a BED file is needed that contains the locations of the summits. In addition, to include extra information about the peaks that are only included in NARROWPEAK format (such as the summit q-value), a NARROWPEAK file is also created with the locations of the summits (`scripts/lfyPooledPeaks2Summits.sh`). The annotation code is run with the following batch script: (scripts/annotate_LFY_W_2021.sh). This script generates the following files in the output directory "data/annotations/LFY_Jin_2021/":

- LFY_W_2021_genewise_ann.csv - gene-centric comma-delimited annotation table
- LFY_W_2021_genewise_ann.tsv - gene-centric tab-delimited annotation table
- LFY_W_2021_peakwise_ann.csv - ChIP summit-centric comma-delimited annotation table
- LFY_W_2021_peakwise_ann.csv - ChIP summit-centric tab-delimited annotation table
- LFY_W_2021_counts.txt - ChIP-Seq annotation meta-data
- LFY_W_2021_convergent_upstream_peaks.txt - tab-delimited table containing distance information about LFY summits upstream of genes which are equally upstream to two different genes. The LFY summit BED-formatted information is in columns 1-6. The gene BED-formatted information is in columns 7-12. The distance between the two features is in column 13.
- LFY_W_2021_unassigned.bed - BED formatted file containing summits of peaks that remain unannoted to genes
- LFY_W_2021_unassigned.narrowPeak - BED formatted file containing summits of peaks that remain unannoted to genes

1.5.2 Compare ChIP-Seq samples from different publications

Custom scripts were created to measure bigwig signals user-set regions, calculate pairwise Pearson correlation values between the samples, perform hierarchical clustering given the correlation values, and return rand-index values based on user-set clustering expectations. To run these scripts we need a bed file that specifies the regions of interest and a table that lists the bigwig files with their respective sample names. From each ChIP-seq experiment, peaks are concatenated and merged to create a bed file of the regions of interest (`scripts/merge_chip_peaks.sh`). The file with the respective bigwig files is generated:

```
$ awk -F"," 'BEGIN{OFS=","; \
        print "sample_name,mapping_file"} \
        {for(i=1;i<=$2;i++){print
↪  $1"_R"i,"data/bigwigs/chip/individual_replicates/"$1"_R"i".bw"}}'
↪  \
        meta/chip_controls_fragsize_nreps.csv > \
        meta/chip_trueRep_bigwigs.csv
```

Given the regions of interest and the respective signals we develop a matrix using `scripts/coverage_bed_matrix.py`.

```
$ python3 scripts/coverage_bed_matrix.py \
    meta/chip_trueRep_bigwigs.csv \
    data/macs2_out/chipPeaks/gsMask_qval10/ChIPseq_Peaks.merged.bed \
    -o data/plotCorrelation \
    -m coverage_matrix_trueRep_peaks_merged.csv
```

The custom script `scripts/readCorrelationPlot.py` calculates pairwise Pearson correlation values, performs unsupervised hierarchical clustering, and displays the results in a heatmap and dendrogram respectively. Rather than label the samples using text, the code provides an option to use a shape and/or color to label the samples by setting a table which match the sample names to specific matplotlib colors and shapes (see `meta/chip_trueReps_colorshapeLabels.csv`). To calculate rand-index values between the unsupervised clusters to expected clusters a table can also be set by matching sample names to cluster labels as in `meta/chip_trueReps_expectedCluster.csv`. To generate the plot and calculate the rand-index value, run the following:

```
$ python3 scripts/readCorrelationPlot.py \
    data/plotCorrelation/coverage_matrix_trueRep_peaks_merged.csv \
    data/plotCorrelation/trueRep_peaks_merged_heatmap.png \
```

```
-lm ward --plot_numbers -k 2 -ri \
-sl meta/chip_trueReps_colorshapeLabels.csv \
-cf meta/chip_trueReps_expectedCluster.csv
```

References

[1] Andrews S. (2010). FastQC: a quality control tool for high throughput sequence data. Available online at: http://www.bioinformatics.babraham.ac.uk/projects/fastqc

[2] Auguie B. (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

[3] Bolger, A. M., Lohse, M., and Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics, 30(15), 2114-2120.

[4] Carroll, T. S., Liang, Z., Salama, R., Stark, R., and de Santiago, I. (2014). Impact of artifact removal on ChIP quality metrics in ChIP-seq and ChIP-exo data. Frontiers in genetics, 5, 75.

[5] Chen, D., & Kaufmann, K. (2017). Integration of Genome-Wide TF Binding and Gene Expression Data to Characterize Gene Regulatory Networks in Plant Development. In Plant Gene Regulatory Networks (pp. 239-269). Humana Press, New York, NY.

[6] Cheng, C. Y., Krishnakumar, V., Chan, A. P., Thibaud-Nissen, F., Schobel, S., and Town, C. D. (2017). Araport11: a complete reannotation of the Arabidopsis thaliana reference genome. The Plant Journal, 89(4), 789-804.

[7] Davis, Maintainer Trevor L. "Package 'argparse'." (2015).

[8] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(3), 90.

[9] Jones, E., Oliphant, T., and Peterson, P. (2001). SciPy: Open source scientific tools for Python.

[10] Kent, J. (2018). KentUtils (Version 419) [Source Code]. http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/.(2021)

[11] Kharchenko PK, Tolstorukov MY, Park PJ, Design and analysis of ChIP-seq experiments for DNA-binding proteins Nat Biotechnol. 2008 Dec;26(12):1351-9

[12] Langmead, B., and Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. Nature methods, 9(4), 357.

[13] Landt, S.G., Marinov, G.K., Kundaje, A., Kheradpour, P., Pauli, F., Batzoglou, S., Bernstein, B.E., Bickel, P., Brown, J.B., Cayting, P. and Chen, Y. (2012). ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. Genome research, 22(9), 1813-1831.

[14] Lawrence, M., Huber, W., Pages, H., Aboyoun, P., Carlson, M., Gentleman, R., Morgan, M.T. and Carey, V.J. (2013). Software for computing and annotating genomic ranges. PLoS computational biology, 9(8), e1003118.

[15] Lekhonkhobe, T. (2017). mwaskom/seaborn: v0. 8.1 (September 2017).

[16] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G. and Durbin, R. (2009). The sequence alignment/map format and SAMtools. Bioinformatics, 25(16), 2078-2079.

[17] McKinney, W. (2010, June). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51-56).

[18] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.

[19] Oliphant, T. E. (2006). A guide to NumPy (Vol. 1, p. 85). USA: Trelgol Publishing.

[20] Picard: a set of command line tools (in Java) for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF. Retrieved from: http://broadinstitute.github.io/picard/

[21] Ramírez, Fidel, Devon P. Ryan, Björn Grüning, Vivek Bhardwaj, Fabian Kilpert, Andreas S. Richter, Steffen Heyne, Friederike Dündar, and Thomas Manke. deepTools2: A next Generation Web Server for Deep-Sequencing Data Analysis. Nucleic Acids Research (2016). doi:10.1093/nar/gkw257.

[22] Robinson, J. T., Thorvaldsdóttir, H., Wenger, A. M., Zehir, A., and Mesirov, J. P. (2017). Variant review with the integrative genomics viewer. Cancer research, 77(21), e31-e34.

[23] Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., Lander, E. S., Getz, G., and Mesirov, J. P. (2011). Integrative genomics viewer. Nature biotechnology, 29(1), 24.

[24] Seabold, Skipper, and Josef Perktold. “Statsmodels: Econometric and statistical modeling with python.” Proceedings of the 9th Python in Science Conference. 2010.

[25] Shirley MD, Ma Z, Pedersen BS, Wheelan SJ. (2015) Efficient "pythonic" access to FASTA files using pyfaidx. PeerJ PrePrints 3:e1196 https://dx.doi.org/10.7287/peerj.preprints.970v1

[26] SignalSmooth. SciPy Cookbook; 2017 [accessed 2019 Aug 30]. https://scipy-cookbook.readthedocs.io/items/SignalSmooth.html.

[27] Quinlan, A. R., and Hall, I. M. (2010). BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics, 26(6), 841-842.

[28] Thorvaldsdóttir, H., Robinson, J. T., and Mesirov, J. P. (2013). Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. Briefings in bioinformatics, 14(2), 178-192.

[29] Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. Computing in Science & Engineering, 13(2), 22.

[30] Wang, H., Li, S., Li, Y.A., Xu, Y., Wang, Y., Zhang, R., Sun, W., Chen, Q., Wang, X.J., Li, C. and Zhao, J. (2019). MED25 connects enhancer–promoter looping and MYC2-dependent activation of jasmonate signalling. Nature plants, 5(6), 616-625.

[31] Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Lukauskas, S., Gemperline, D.C., Augspurger, T., Halchenko, Y., Cole, J.B., Warmenhoven, J. and de Ruiter, J. (2017). mwaskom/seaborn: v0. 8.1 (September 2017)

[32] Wickham, H. (2016). ggplot2: elegant graphics for data analysis. Springer.

[33] Wickham, H., François, R., Henry, L., and Müller, K. (2019). dplyr: A Grammar of Data Manipulation. R package version 0.8.3. https://CRAN.R-project.org/package=dplyr

[34] Zhang, Y., Liu, T., Meyer, C.A., Eeckhoute, J., Johnson, D.S., Bernstein, B.E., Nusbaum, C., Myers, R.M., Brown, M., Li, W., Liu, X.S.. Model-based analysis of ChIP-Seq (MACS). Genome biology. 2008 Nov;9(9):1-9.

[35] Zhu, L. J., Gazin, C., Lawson, N. D., Pagès, H., Lin, S. M., Lapointe, D. S., and Green, M. R. (2010). ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. BMC bioinformatics, 11(1), 237.

[36] Zhu, L. J. (2013). Integrative analysis of ChIP-chip and ChIP-seq dataset. In Tiling Arrays (pp. 105-124). Humana Press, Totowa, NJ.