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 greenscreens 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 [22]

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 [20]
- Picard 2.26.0: identify duplicate reads [25]
  - 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 [16]
  - Compiler: gcc version 8.3.1 20170224 (experimental) (GCC)
- MACS2 2.2.7.1 [42]
- BEDTools v2.27.1 [34]
- kentUtils v419 [14]
- deeptools v3.5.1 [26]

1.2 Docker Workspace

To make this tutorial easier it is recommended to install git [5] (https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) and Docker [22] (https://docs.docker.com/get-docker/). Git is simply a version control system which allows the user to obtain the latest scripts from the greenscreen github repository (https://github.com/sklasfeld/GreenscreenProject). Docker is a software to ensure these scripts are run in the same computational environment in which these scripts were also generated. This environment is set in a docker image which users can upload on their own computers.

To manage docker images as a non-root user (and avoid using *sudo* for every docker command) see https://docs.docker.com/engine/install/linux-postinstall/. An image of the greenscreen environment called 'sklasfeld/greenscreen' is on dockerhub. To pull the image use the following command:

```
docker pull sklasfeld/greenscreen:latest
```

To run and generate the interactive environment (also called a docker "container") using this image type the following into the command line (replace the text in brackets with the correct information):

```
$ docker run --name [CONTAINER_NAME] -i -t sklasfeld/greenscreen bash
```

[CONTAINER_NAME] can be set to whatever name the user wants to name the new docker container.

The interactive environment will open to working directory '/home/'. Files generated within the local computer environment or the Docker container environment can 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 from the local environment 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 reload the interactive container with the following commads:

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

To run this tutorial, clone the github repository in the docker environment.

```
git clone https://github.com/sklasfeld/GreenscreenProject.git
```

Set the working directory from within the repository. All commands need to be run within this working directory, unless specified otherwise.

```
cd GreenscreenProject/
```

1.3 greenscreen generation

To run the entire greenscreen generation pipeline in one command use the demo script:

```
$ python3 scripts/generate_gsRegions_demo.py
```

The demo directory contains three demo input fastq files which are run through the pipeline to generate a greenscreen for chromosome two in Arabidopsis. For computational efficiency these fastq files have very low sequencing depth. The demo should produce greenscreen regions in a file located at ‘demo/data/macs2_out/inputControls/qval10/merge500bp_20inputs.txt‘

1.3.1 Import the raw input samples' reads

The raw sequences of the twenty inputs from Table S1 are retrieved using NCBI SRA Tools fastq-dump command (`scripts/import_raw_fasta_inputs.sh`). Run these commands in the docker container from the '/home/GreensceenProject/' directory:

```
$ bash scripts/import_raw_fasta_inputs.sh
```

Note that if you get an error about enabling remote access you may need to set sra-tools to interactive mode using the following command and typing "E" to enable access, "S" to save the environment, and "X" to exit.

```
/usr/src/sratoolkit.2.11.2-ubuntu64/bin/vdb-config --interactive
```

Downloaded fastq files are compressed, moved into the 'fastq/raw' directory and renamed according to their respective ID from Table S1 (`scripts/organize_raw_fasta_input.sh`).

```
$ bash scripts/organize_raw_fasta_input.sh
```

Note that InputJ contains three run files; these reads are concatenated and the individual run files are deleted.

1.3.2 Clean the input reads

Presence of adaptors and overall sequencing quality are checked using FASTQC (`scripts/fastqc_raw_inputs.sh`).

```
$ bash scripts/fastqc_raw_inputs.sh
```

The FASTQC outputs in Inputs A,J,K,M,O,R,S showed 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`).

```
$ bash scripts/trimmomatic_inputs.sh
```

1.3.3 Mapping the input reads

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 [7]. 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 the genome index files are generated 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. Upon completion, samtools is used to sort the reads, compress the reads into BAM files (samtools sort) and generate index files for the reads (samtools index). Next, reads 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 are filtered out. Afterwards, duplicates are marked with PICARD (`scripts/mapped_inputs.sh`).

```
$ bash scripts/mapped_inputs.sh
```

1.3.4 Input Quality Control

A representative result of a successful ChIP-seq experiment is when mapped reads contain a strand bias signature. More precisely, a strand-bias signature is when mapped reads form two distinct clusters, separated about a fragment size apart and directed towards the opposite peak (Watson and Crick strand-enriched) [17]. 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 in which successfully immunoprecipitated samples show highest enrichment at the experiment's fragment size whereas input samples show enrichment 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 immunoprecipitated sample data sets tend to be enriched for peaks with strand-bias (RelCC>.8), whereas inputs have little or no such bias [17]. 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 files from MAC2. Other columns can further describe the experiment (ie. replicate number, treatment, genotype).

- **Genome Annotation**: a file containing features (ie. gene, transcript ID) 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.

Make sure the chromosome names are the same between your Genome Annotation (GFF), Genome (FASTA) and Chromosome sizes files. For example, Chromosome 1 can be labeled as either "1", "chr1", "Chr1", etc.

A sample sheet 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/GreensceenProject/meta/ArabidopsisGenome. This gff file is written using unicode percent encodings. These encodings can be translated into UNIX format 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 calculated 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 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`. Once we have developed the list of greenscreen regions, the reads that overlay greenscreen regions can be masked out and the remaining reads can be applied to the ChIPQC library using the same script as above. After masking the metrics should show cleaner input samples.

1.3.5 Visualize Mapped Input-Seq Sequences

To visualize the inputs genome-wide, the mapped read depths across the genome were normalized by each sample's total sequencing depth using bedtools genome-

CoverageBed function and then converted into bigwig format using kentUtils bedGraphToBigWig function in `scripts/bamToBigwig_input.sh` [14].

```
$ bash scripts/bamToBigwig_input.sh
```

Bigwig files can be visualized using genome browsers, such as IGV.

1.3.6 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. Read sizes, which can be found in the ChIPQC Report of each input, were manually added to a comma-delimited table where each input ID is in the first column and the respective sample read length in the second column (`meta/input_readsizes.csv`). Using this text file as a template, peaks are called on each input sample using (`scripts/macs2_callpeaks_inputs.sh`).

```
$ bash scripts/macs2_callpeaks_inputs.sh
```

Greenscreen parameters were set with respect to the number of inputs used (20) and the size of the Arabidopisis genome. MACS2 outputs peaks were 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} (QVAL=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 5kb are merged (MERGE_DISTANCE=5000). Given 20 inputs, regions that are called in less than 10 distinct samples are dropped to get the final greenscreen regions (DISTINCT_NINPUTS=10). The following script was run using the respective parameters: `scripts/generate_20input_greenscreenBed.sh`.

```
$ bash scripts/generate_20input_greenscreenBed.sh [QVAL]
↪  [MERGE_DISTANCE] [DISTINCT_NINPUTS]
```

After running this command your file at `data/macs2_out/inputControls/qval10/gs_merge5000bp_call10_20inputs.txt` should match the provided greenscreen regions file at `meta/arabidopsis_greenscreen_20inputs.bed`.

1.4 Calling ChIP-Seq peaks

1.4.1 Import the raw ChIP-seq samples' reads

The raw sequences of the ChIP-Seq samples [45, 12, 8, 23, 29, 30] and their respective MACS2 controls (ie. inputs

or mock) are retrieved using wget and NCBI SRA Tools (`scripts/import_raw_fasta_publishedChIPsAndControls.sh`).

```
$ bash 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 using the format: "[ChIP Factor]_[mutation if applicable]_[lab]_[publication Year]" (`organize_raw_fasta_publishedChIPsAndControls.sh`). For example, LFY ChIP from Jin *et al* 2021 which originated from the Wagner lab is labelled "LFY_W_2021".

```
$ bash scripts/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`).

```
$ bash scripts/fastqc_raw_publishedChIPsAndControls.sh
```

1.4.2 Clean the ChIP-seq reads

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 greenscreen (`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 assess the quality of the trimmed reads.

```
$ bash scripts/trimmomatic_publishedChIPsAndControls.sh
```

1.4.3 Mapping the ChIP-seq 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 filtered 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. The marked duplicates can be fully removed with samtools, but we choose instead to handle the duplicates as a parameter for MACS2.

```
$ bash scripts/mapped_publishedChIPsAndControls.sh
```

1.4.4 ChIP-Seq Quality Control

The ChIPQC library is used to quantify the strand bias in each of the immunoprecipitated samples and respective controls. As reads within artificial signals can overpower true ChIP signals, all reads overlapping with greenscreen regions are masked with the script `scripts/ChIPQC_gsMaskReads_publishedChIPsAndControls.sh` which requires a samplesheet with paths to masked mapped reads, `meta/gsMaskReads_publishedChIPsAndControls_sampleSheet.csv`.

```
$ bash scripts/ChIPQC_gsMaskReads_publishedChIPsAndControls.sh
```

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.

1.4.5 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 [37]. `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".

```
$ bash scripts/downSampleBam_publishedChIPsAndControls.sh
```

To 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 in this tutorial, if a publication contained both a mock and input control that the MACS2 control was set to the input control. Despite evidence that some duplicates are necessary for calling peaks in immunoprecipitation studies, MACS2 only retains a single read per location by default. However, with the setting "keep-dup auto", MACS2 keeps duplicates based on the expected maximum tags at the same location. Furthermore, single-end reads are extended to reach the set fragment length with the –extsize option

scripts/macs2_callpeaks_publishedChIPs.sh. Then, using MACS2 narrowPeak output, an AWK statement is called to filter out peaks with summit q-value $> 10^{-10}$ and bedtools intersect is used to remove regions which overlap greenscreen regions.

```
$ bash scripts/macs2_callpeaks_publishedChIPs.sh
```

1.4.6 Visualize Mapped ChIP-Seq Sequences

As opposed to the input samples which, to be visualized, only requires normalization of the signal based on the sequencing depth, ChIP immunoprecipitated samples require an additional step to see the signal where the protein of interest binds. As explained earlier, reads from immunoprecipitated samples form strand-biased clusters around protein binding sites. 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.

To generate bigwig files for IGV using the script, scripts/bamToBigWig_publishedChIPsAndControls_replicates.sh, we first need a table of the fragment size of each sample, meta/chip_readsize_fragsize.csv. The fragment sizes were set using the output from ChIPQC as a reference.

```
$ bash scripts/bamToBigWig_publishedChIPsAndControls_replicates.sh
```

To visualize the signal after down-sampling 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.

```
$ bash scripts/bamToBigWig_publishedChIPsAndControls_pooled.sh
```

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. Round one annotates summits found within a gene (intragenic) and within a 4kb distance upstream of a gene. Round two annotates orphan peaks within a 10kb distance from a gene. 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/GreensceenProject/scripts

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

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

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 are available to compare bigwig signals in 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 (a score of 1 indicates that the user-expected clusters are identical to those found by the unsupervised algorithm). 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] Chacon, S., & Straub, B. (2014). Pro git. Apress.

[6] 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.

[7] 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.

[8] Collani, S., Neumann, M., Yant, L., & Schmid, M. (2019). FT modulates genome-wide DNA-binding of the bZIP transcription factor FD. Plant physiology, 180(1), 367-380.

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

[10] Goretti, D., Silvestre, M., Collani, S., Langenecker, T., Méndez, C., Madueno, F., & Schmid, M. (2020). TERMINAL FLOWER1 Functions as a Mobile Transcriptional Cofactor in the Shoot Apical Meristem. Plant Physiology, 182(4), 2081-2095.

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

[12] Jin, R., Klasfeld, S., Zhu, Y., Garcia, M.F., Xiao, J., Han, S.K., Konkol, A. and Wagner, D. (2021). LEAFY is a pioneer transcription factor and licenses cell reprogramming to floral fate. Nature Communications, 12(1), pp.1-14.

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

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

[15] 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

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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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).

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

[23] Moyroud, E., Minguet, E.G., Ott, F., Yant, L., Posé, D., Monniaux, M., Blanchet, S., Bastien, O., Thévenon, E., Weigel, D. and Schmid, M., 2011. Prediction of regulatory interactions from genome sequences using a biophysical model for the Arabidopsis LEAFY transcription factor. The Plant Cell, 23(4), pp.1293-1306.

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

[25] 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/

[26] 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.

[27] 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.

[28] 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.

[29] Romera-Branchat, M., Severing, E., Pocard, C., Ohr, H., Vincent, C., Née, G., Martinez-Gallegos, R., Jang, S., Lalaguna, F.A., Madrigal, P. and Coupland, G., 2020. Functional divergence of the Arabidopsis florigen-interacting bZIP transcription factors FD and FDP. Cell reports, 31(9), p.107717.

[30] Sayou, C., Nanao, M.H., Jamin, M., Posé, D., Thévenon, E., Grégoire, L., Tichtinsky, G., Denay, G., Ott, F., Llobet, M.P. and Schmid, M. (2016). A SAM oligomerization domain shapes the genomic binding landscape of the LEAFY transcription factor. Nature communications, 7(1), 1-12.

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

[32] 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

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

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

[35] 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.

[36] 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.

[37] 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.

[38] 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)

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

[40] 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

[41] Winter, C. M., Austin, R. S., Blanvillain-Baufume, S., Reback, M. A., Monniaux, M., Wu, M. F., Sang Y., Yamaguchi, A., Yamaguchi, N., Parker, J.E., Parcy. F., Jensen. S.T., Li, H., & Wagner, D. (2011). LEAFY target genes reveal floral regulatory logic, cis motifs, and a link to biotic stimulus response. Developmental cell, 20(4), 430-443.

[42] 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.

[43] 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.

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

[45] Zhu, Y., Klasfeld, S., Jeong, C. W., Jin, R., Goto, K., Yamaguchi, N., & Wagner, D. (2020). TERMINAL FLOWER 1-FD complex target genes and competition with FLOWERING LOCUS T. Nature communications, 11(1), 1-12.