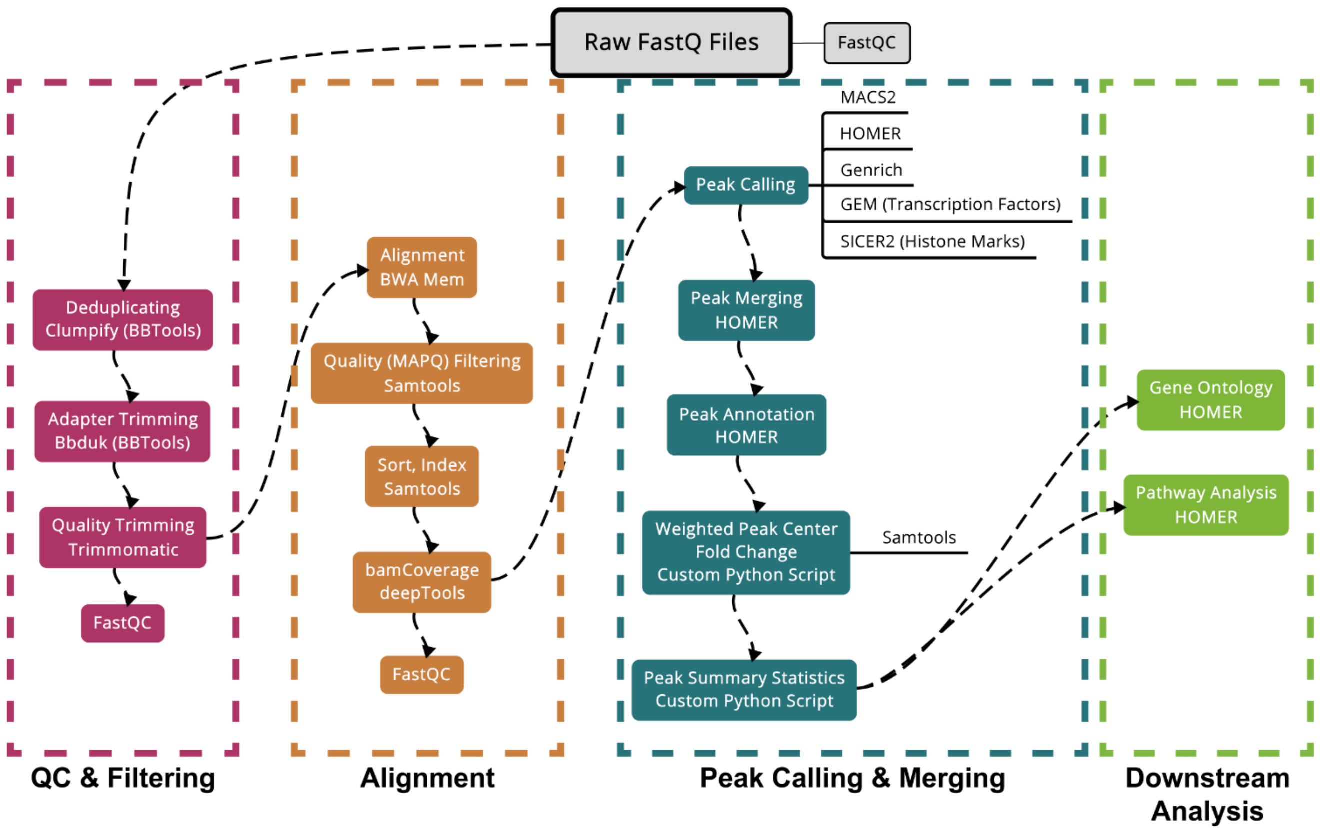
ChIP-AP Graphical Overview



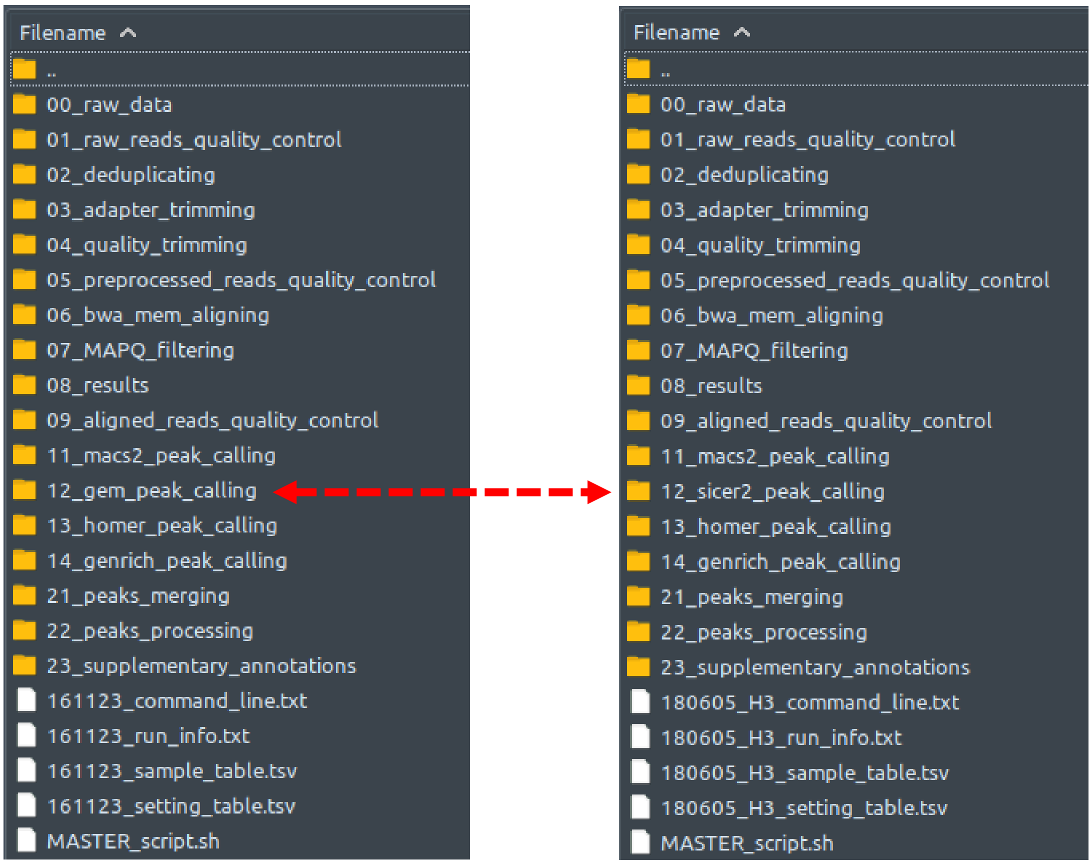
Detailed Explanation of Steps and Methodology Used

1. **Acquisition of raw sequencing files**. ChIP-AP can directly process demultiplexed FASTQ, or compressed FASTQ (.gz) format sequencing files. ChIP-AP can also process aligned reads in BAM format (which bypasses QC & Filtering and Alignment stages of the pipeline and proceeds directly to Peak Calling & Merging). Reads may be single or paired ends. Background control is compulsory, no unmatched samples allowed here. For background, Input control is the commonly accepted standard and not IgG controls.

2. **Sample recognition and registration**. Performed by the main script. Each input sample is registered into the system and given a new name according to their sample category (ChIP or background control), replicate number, and whether it's the first or second read (in case of paired end sequencing data). Afterwards, their format and compression status is recognized and processed into gun-zipped FASTQ as necessary.

3. **Generation of multiple modular scripts**. Each stage of analysis in ChIP-AP is executed from persistent individually generated scripts. This was an intentional design decision as it allows for easy access for debugging and modifications of any step within the pipeline without hunting through the master ChIP-AP scripts; simply re-run the troublesome step to figure out what's going on or swap it out entirely if you really want. You can thank us later if you have to modify and tailor something later and don't have to drudge through the trenches of someone else's code. Chocolate treats always welcome!

   How does it look like? After this step is done (which is the end of your ChIP-AP processes if you don't use the "*--run*" flag to run the pipeline immediately), you can see within your designated output directory a single folder named as per your "*--setname*" input, with contents as shown below regardless of single-end or paired end-mode:



```
Filename ^
..
00_raw_data
01_raw_reads_quality_control
02_deduplicating
03_adapter_trimming
04_quality_trimming
05_preprocessed_reads_quality_control
06_bwa_mem_aligning
07_MAPQ_filtering
08_results
09_aligned_reads_quality_control
11_macs2_peak_calling
12_gem_peak_calling
13_homer_peak_calling
14_genrich_peak_calling
21_peaks_merging
22_peaks_processing
23_supplementary_annotations
161123_command_line.txt
161123_run_info.txt
161123_sample_table.tsv
161123_setting_table.tsv
MASTER_script.sh
```

```
Filename ^
..
00_raw_data
01_raw_reads_quality_control
02_deduplicating
03_adapter_trimming
04_quality_trimming
05_preprocessed_reads_quality_control
06_bwa_mem_aligning
07_MAPQ_filtering
08_results
09_aligned_reads_quality_control
11_macs2_peak_calling
12_sicer2_peak_calling
13_homer_peak_calling
14_genrich_peak_calling
21_peaks_merging
22_peaks_processing
23_supplementary_annotations
180605_H3_command_line.txt
180605_H3_run_info.txt
180605_H3_sample_table.tsv
180605_H3_setting_table.tsv
MASTER_script.sh
```

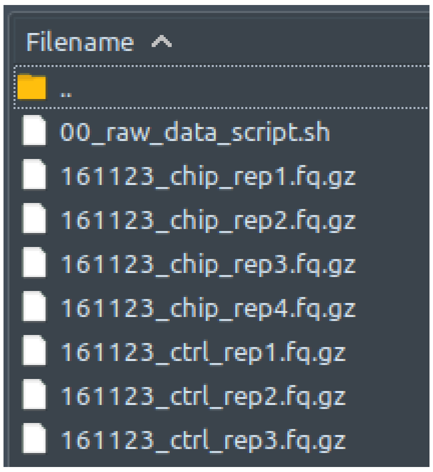
   The left figure shows a dataset with narrow peak type, while the right figure shows a dataset with broad peak type. Note that the peak calling module – folder number 12 – is interchangeable between GEM for narrow peak type and SICER2 for broad peaks. This is because of the significant difference between these 2 peak callers in handling the respective datasets. GEM does a great job for TF's and SICER2 for broad peaks.

   Each of these folders are basically empty, and contains a script which is named based on the folder name (e.g., script **02_deduplicating.sh** inside folder **02_deduplicating**). Each of these scripts will be executed in numerical sequence when you run the pipeline. Aside from these scripts, there will be several miscellaneous text files which contain essential information of your pipeline run (See "**Miscellaneous Pipeline Output"** section below for details). Lastly, there is your big red button: the **MASTER_script.sh** that you can simply call (if you did not use the "*--run*" flag when calling ChIP-AP) to sequentially run all the scripts within the aforementioned folders.

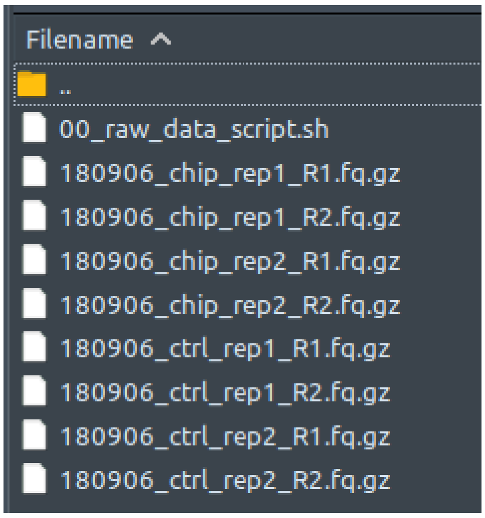
4. **Copying, compressing, and renaming of the raw sequencing reads**. In the very beginning, ChIP-AP makes (in the user-designated output folder) a copy of each unaligned sequence reads file, compresses them into a gunzipped file (if not already), and renames them with the prepared new name from step "**2. Sample recognition and registration**". If the given inputs are aligned reads (bam files), the pipeline starts at step "**12. Sorting and indexing of aligned reads files**" (see below) and the copying and renaming are taken over by **08_results_script.sh** where the original bam files are directly sorted and the pipeline proceeds normally from there.

Modular script used: **00_raw_data_script.sh**
➢ **Operation**: cp, gzip, mv (Bash) ➢ **Input** : [origin directory] / [original ChIP/control filename] ➢ **Process** : Copy, compress, and rename raw reads files ➢ **Output** : [output directory] / 00_raw_data / [setname]_[chip/ctrl]_rep[#]_R[1/2].fq.gz

How does it look like? After **00_raw_data_script.sh** had been executed, all reads files in your dataset will have been copied into this folder: **00_raw_data**, compressed into **fq.gz**, and renamed into something like the preview below, regardless of your initial filenames, fastq formatting or extensions.



```
Filename ^
..
00_raw_data_script.sh
161123_chip_rep1.fq.gz
161123_chip_rep2.fq.gz
161123_chip_rep3.fq.gz
161123_chip_rep4.fq.gz
161123_ctrl_rep1.fq.gz
161123_ctrl_rep2.fq.gz
161123_ctrl_rep3.fq.gz
```



```
Filename ^
..
00_raw_data_script.sh
180906_chip_rep1_R1.fq.gz
180906_chip_rep1_R2.fq.gz
180906_chip_rep2_R1.fq.gz
180906_chip_rep2_R2.fq.gz
180906_ctrl_rep1_R1.fq.gz
180906_ctrl_rep1_R2.fq.gz
180906_ctrl_rep2_R1.fq.gz
180906_ctrl_rep2_R2.fq.gz
```

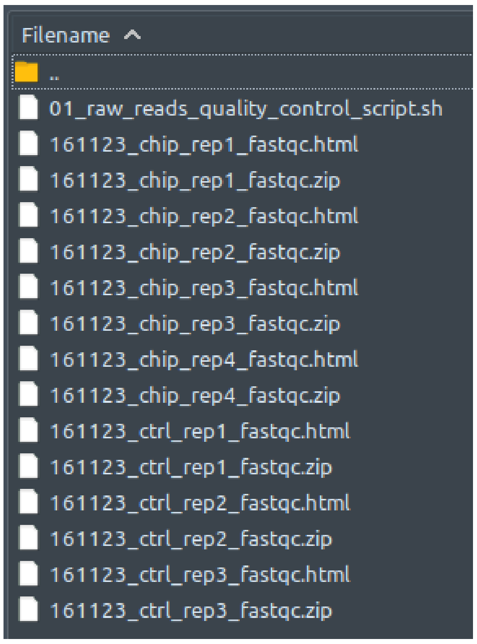
The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples, in which every sample consists of two files: the first read (R1), and the second read (R2).

All these **fq.gz** files will be immediately deleted at the end of **02_deduplicating_script.sh** execution if "*--deltemp*" flag is used when calling ChIP-AP. We won't explain how to open and read these **fq.gz** files, since if you need us to tell you that, you most probably won't be able to understand the contents of the files anyway.

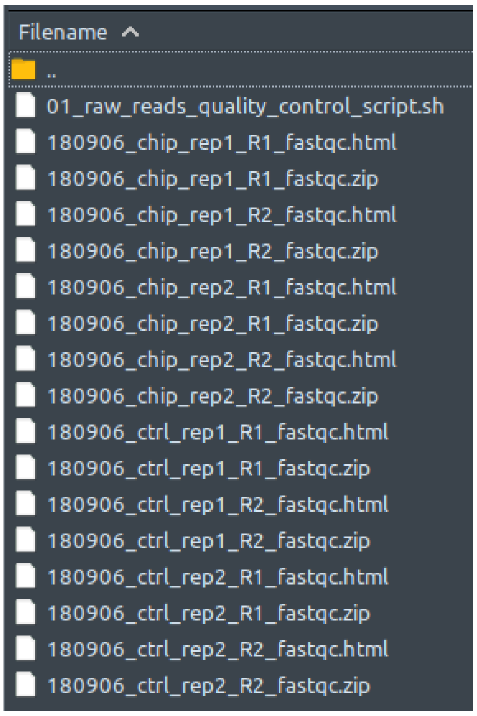
5. **Raw sequencing reads quality assessment**. Performed by FastQC. Reads quality assessment is performed to check for duplicates, adapter sequences, base call scores, etc. Assessment results are saved as reports for user viewing. If the final results are not as expected, it's worthwhile to go through the multiple QC steps and track the quality of the data as its processed from this folder onwards. If the default QC steps aren't cleaning up the data adequately, you may need to modify some parameters to be more/less stringent with cleanup. From our testing, our default values seem to do a fairly adequate job though for most datasets. They even work pretty well for RNA-Seq and RIP-Seq datasets too!

Modular script used: **01_raw_reads_quality_control_script.sh**
➢ **Calls** : fastqc ➢ **Input** : 00_raw_data / [setname]_[chip/ctrl]_rep[#]_R[1/2].fq.gz ➢ **Process** : Generate raw reads quality assessment reports ➢ **Output** : 01_raw_reads_quality_control / [setname]_[chip/ctrl]_rep[#]_R[1/2]_fastqc.html

How does it look like? After **01_raw_reads_quality_control_script.sh** had been executed, the folder: **01_raw_reads_quality_control** will contain all these quality assessment reports for every raw reads file in folder **00_raw_data**, just like below:



```
Filename ^
..
01_raw_reads_quality_control_script.sh
161123_chip_rep1_fastqc.html
161123_chip_rep1_fastqc.zip
161123_chip_rep2_fastqc.html
161123_chip_rep2_fastqc.zip
161123_chip_rep3_fastqc.html
161123_chip_rep3_fastqc.zip
161123_chip_rep4_fastqc.html
161123_chip_rep4_fastqc.zip
161123_ctrl_rep1_fastqc.html
161123_ctrl_rep1_fastqc.zip
161123_ctrl_rep2_fastqc.html
161123_ctrl_rep2_fastqc.zip
161123_ctrl_rep3_fastqc.html
161123_ctrl_rep3_fastqc.zip
```



```
Filename ^
..
01_raw_reads_quality_control_script.sh
180906_chip_rep1_R1_fastqc.html
180906_chip_rep1_R1_fastqc.zip
180906_chip_rep1_R2_fastqc.html
180906_chip_rep1_R2_fastqc.zip
180906_chip_rep2_R1_fastqc.html
180906_chip_rep2_R1_fastqc.zip
180906_chip_rep2_R2_fastqc.html
180906_chip_rep2_R2_fastqc.zip
180906_ctrl_rep1_R1_fastqc.html
180906_ctrl_rep1_R1_fastqc.zip
180906_ctrl_rep1_R2_fastqc.html
180906_ctrl_rep1_R2_fastqc.zip
180906_ctrl_rep2_R1_fastqc.html
180906_ctrl_rep2_R1_fastqc.zip
180906_ctrl_rep2_R2_fastqc.html
180906_ctrl_rep2_R2_fastqc.zip
```

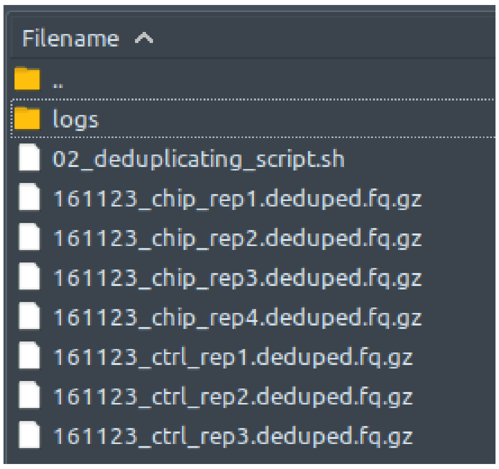
The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples, in which every sample consists of two files: the first read (R1), and the second read (R2).

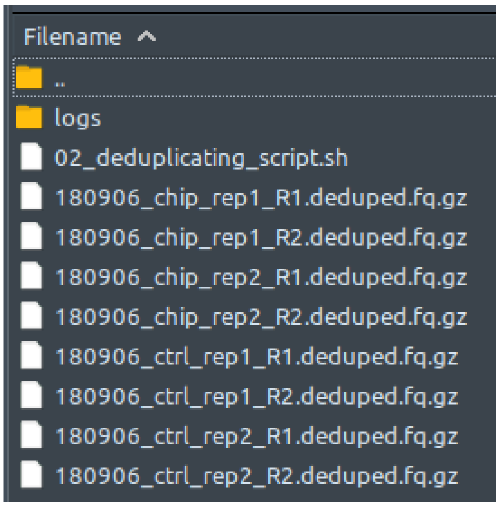
The **.zip** files contains the individual components to be compiled for the report so you can ignore those. To read the reports, open the **.html** files in your web-browser of choice (Safari, Edge, Firefox). This file is a multitabular file in which you can evaluate the quality of your experiment, sequencing, etc. Comprehensive as it is, explaining the contents in detail would take a whole new guide by itself. Therefore, its best to check out the documentation of the developers for all the details:
https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/.

6. **Deduplication of reads**. Performed by clumpify from BBMap package. Clumpify is used to remove optical duplicates and tile-edge duplicates from the reads file in addition to PCR duplicates. This step is performed in preference to a PCR deduplication step downstream. Optimization of file compression is also performed by clumpify during deduplication process, in order to minimize storage space and speed up reads file processing. Clumpify will also ensure your processed files are output in PHRED33 format so as to not break any down-stream program processing. Also, if there are ill-formatted reads (if using publicly available dataset) then Clumpify will also attempt to fix these reads.

Modular script used: **02_deduplicating_script.sh**	
➢ **Calls**	: clumpify.sh
➢ **Input**	: 00_raw_data / [setname]_[chip/ctrl]_rep[#]_R[1/2].fq.gz
➢ **Process**	: Remove PCR duplicates, optical duplicates, and tile-edge duplicates
➢ **Output**	: 02_deduplicating / [setname]_[chip/ctrl]_rep[#]_R[1/2].deduped.fq.gz

How does it look like? After **02_deduplicating_script.sh** had been executed, the folder: **02_deduplicating** will contain all these deduplicated reads files (marked by the extension: **.deduped.fq.gz**), just like below:


```
Filename ^
..
logs
02_deduplicating_script.sh
161123_chip_rep1.deduped.fq.gz
161123_chip_rep2.deduped.fq.gz
161123_chip_rep3.deduped.fq.gz
161123_chip_rep4.deduped.fq.gz
161123_ctrl_rep1.deduped.fq.gz
161123_ctrl_rep2.deduped.fq.gz
161123_ctrl_rep3.deduped.fq.gz
```


```
Filename ^
..
logs
02_deduplicating_script.sh
180906_chip_rep1_R1.deduped.fq.gz
180906_chip_rep1_R2.deduped.fq.gz
180906_chip_rep2_R1.deduped.fq.gz
180906_chip_rep2_R2.deduped.fq.gz
180906_ctrl_rep1_R1.deduped.fq.gz
180906_ctrl_rep1_R2.deduped.fq.gz
180906_ctrl_rep2_R1.deduped.fq.gz
180906_ctrl_rep2_R2.deduped.fq.gz
```

The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples, in which every sample consists of two files: the first read (R1), and the second read (R2).

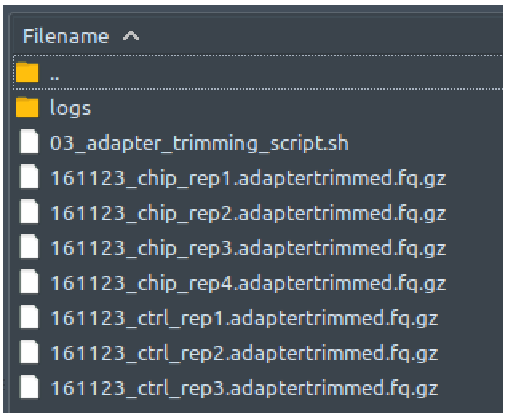
There should be one deduplicated file for each processed raw reads file from folder **00_raw_data**. Paired-end files are processed in pairs by clumpify. All these **deduped.fq.gz** files will be deleted at the end of **02_deduplicating_script.sh** execution if "*—deltemp*" flag is used on ChIP-AP call.

7. **Adapter trimming of reads**. Performed by BBDuk from BBMap package. BBDuk scans every read for adapter sequences in its reference list of adapter sequences. The standard BBDuk adapter sequence reference list 'adapter.fa' is used as the default in the pipeline. Any sequencing adapter present in the reads is removed. Custom adapter sequence can be used whenever necessary by modifying the adapter.fa file and adding your new sequences. Just remember to make a backup of the original yeah? We warned you!

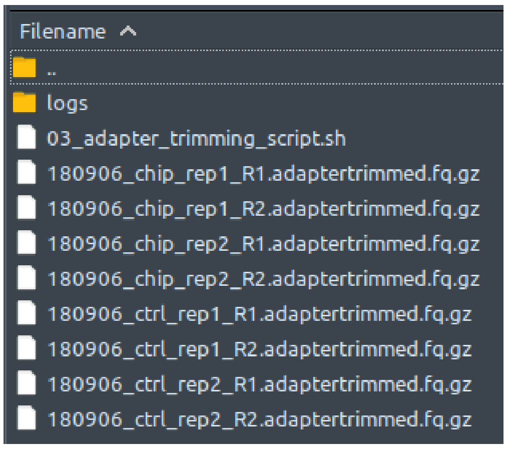
Modular script used: **03_adapter_trimming_script.sh**

- **Calls** : bbduk.sh
- **Input** : 02_deduplicating / [setname]_[chip/ctrl]_rep[#]_R[1/2].deduped.fq.gz
  [path to genome folder] / bbmap / adapters.fa *(file provided by ChIP-AP)*
- **Process** : Trim away adapter sequences based on given sequences in file 'adapters.fa'
- **Output** : 03_adapter_trimming / [setname]_[chip/ctrl]_rep[#]_R[1/2].adaptertrimmed.fq.gz

How does it look like? After **03_adapter_trimming_script.sh** had been executed, the folder: **03_adapter_trimming** will contain all these adapter-trimmed reads files (marked by the extension**: .adaptertrimmed.fq.gz**), just like below:



```
Filename ^
..
logs
03_adapter_trimming_script.sh
161123_chip_rep1.adaptertrimmed.fq.gz
161123_chip_rep2.adaptertrimmed.fq.gz
161123_chip_rep3.adaptertrimmed.fq.gz
161123_chip_rep4.adaptertrimmed.fq.gz
161123_ctrl_rep1.adaptertrimmed.fq.gz
161123_ctrl_rep2.adaptertrimmed.fq.gz
161123_ctrl_rep3.adaptertrimmed.fq.gz
```



```
Filename ^
..
logs
03_adapter_trimming_script.sh
180906_chip_rep1_R1.adaptertrimmed.fq.gz
180906_chip_rep1_R2.adaptertrimmed.fq.gz
180906_chip_rep2_R1.adaptertrimmed.fq.gz
180906_chip_rep2_R2.adaptertrimmed.fq.gz
180906_ctrl_rep1_R1.adaptertrimmed.fq.gz
180906_ctrl_rep1_R2.adaptertrimmed.fq.gz
180906_ctrl_rep2_R1.adaptertrimmed.fq.gz
180906_ctrl_rep2_R2.adaptertrimmed.fq.gz
```

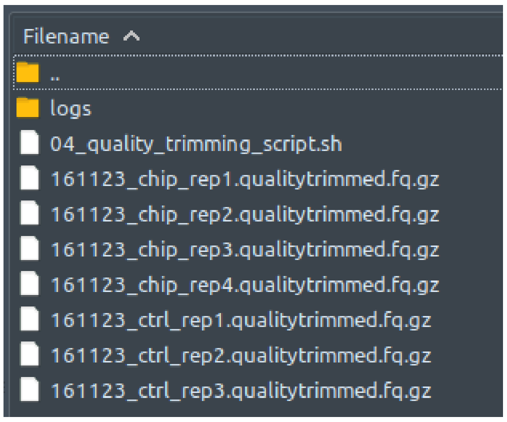
The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples, in which every sample consists of two files: the first read (R1), and the second read (R2).

There should be one adapter-trimmed file for each processed deduplicated reads file from folder **02_deduplicating**. Paired-end files are processed in pairs by bbduk. All these **adaptertrimmed.fq.gz** files will be automatically deleted at the end of **03_adapter_trimming_script.sh** execution if "--deltemp" flag is used on ChIP-AP call.

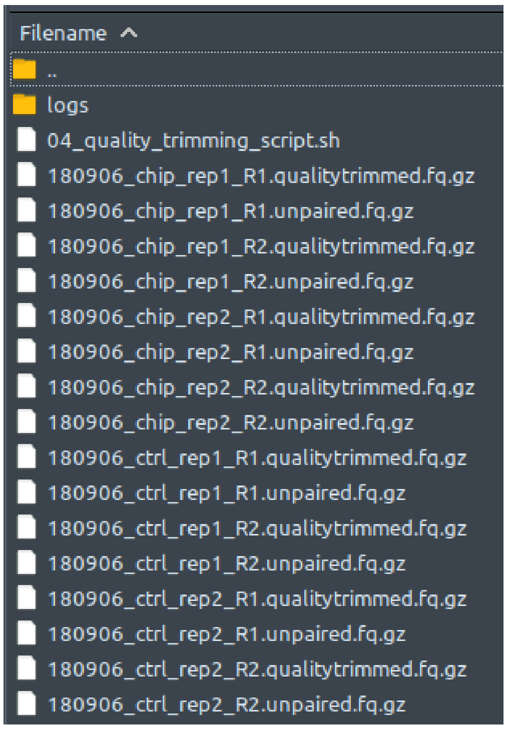
8. **Quality trimming of reads**. Performed by Trimmomatic. Trimmomatic scans every read and trims low quality base calls from the ends of the reads. Additionally, it scans with a moving window along the read and cuts the remainder of the read when the average quality of base calls within the scanning window drops below the set threshold. Finally, it discards the entirety of a read of it gets too short post-trimming for alignment to reference genome, minimizing the chance of reads being multi-mapped to multiple genomic locations. Check out the settings table for the parameters used. Some value changes won't make a big difference to the output, some will make a huge difference, so careful what you set this to.

Modular script used: **04_quality_trimming_script.sh**
➢ **Calls** : trimmomatic
➢ **Input** : 03_adapter_trimming / [setname]_[chip/ctrl]_rep[#]_R[1/2].adaptertrimmed.fq.gz
➢ **Process** : Remove reads with low PHRED (base calling) score
➢ **Output** : 04_quality_trimming / [setname]_[chip/ctrl]_rep[#]_R[1/2].qualitytrimmed.fq.gz

How does it look like? After **04_quality_trimming_script.sh** had been executed, the folder: **04_quality_trimming** will contain all these quality-trimmed reads files (marked by the extension: **.qualitytrimmed.fq.gz**), just like below:



```
Filename ^
..
logs
04_quality_trimming_script.sh
161123_chip_rep1.qualitytrimmed.fq.gz
161123_chip_rep2.qualitytrimmed.fq.gz
161123_chip_rep3.qualitytrimmed.fq.gz
161123_chip_rep4.qualitytrimmed.fq.gz
161123_ctrl_rep1.qualitytrimmed.fq.gz
161123_ctrl_rep2.qualitytrimmed.fq.gz
161123_ctrl_rep3.qualitytrimmed.fq.gz
```



```
Filename ^
..
logs
04_quality_trimming_script.sh
180906_chip_rep1_R1.qualitytrimmed.fq.gz
180906_chip_rep1_R1.unpaired.fq.gz
180906_chip_rep1_R2.qualitytrimmed.fq.gz
180906_chip_rep1_R2.unpaired.fq.gz
180906_chip_rep2_R1.qualitytrimmed.fq.gz
180906_chip_rep2_R1.unpaired.fq.gz
180906_chip_rep2_R2.qualitytrimmed.fq.gz
180906_chip_rep2_R2.unpaired.fq.gz
180906_ctrl_rep1_R1.qualitytrimmed.fq.gz
180906_ctrl_rep1_R1.unpaired.fq.gz
180906_ctrl_rep1_R2.qualitytrimmed.fq.gz
180906_ctrl_rep1_R2.unpaired.fq.gz
180906_ctrl_rep2_R1.qualitytrimmed.fq.gz
180906_ctrl_rep2_R1.unpaired.fq.gz
180906_ctrl_rep2_R2.qualitytrimmed.fq.gz
180906_ctrl_rep2_R2.unpaired.fq.gz
```

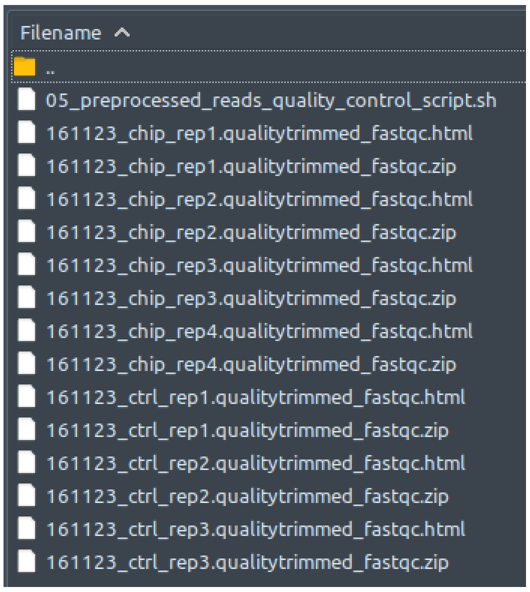
The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples, in which every sample consists of two files: the first read (R1), and the second read (R2).

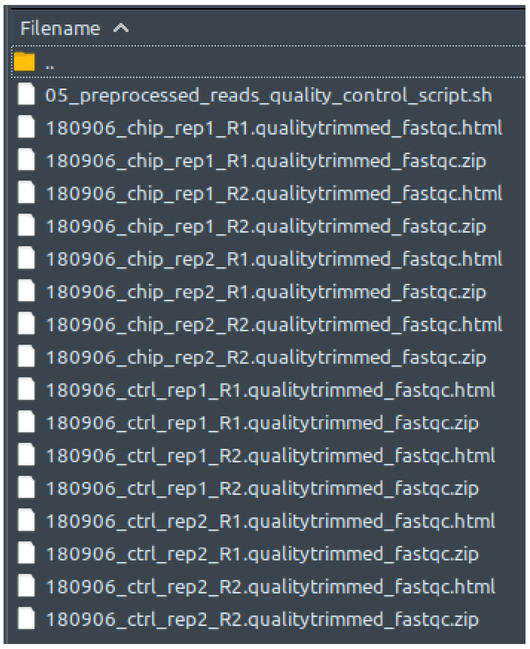
There should be one quality-trimmed file for each processed adapter-trimmed reads file from folder **03_adapter_trimming**. Paired-end files are processed in pairs by trimmomatic. Unpaired reads are separated (saved into **unpaired.fq.gz** files) from the paired reads (saved into **qualitytrimmed.fq.gz** files). Only the paired reads (extension: **.qualitytrimmed.fq.gz**) are processed further in the pipeline. All these **qualitytrimmed.fq.gz** and **unpaired.fq.gz** files will be immediately deleted at the end of **04_quality_trimming_script.sh** execution if "--*deltemp*" flag is used on ChIP-AP call.

9. **Pre-processed reads quality assessment**. Performed by FastQC. Quality assessment is performed to check for the efficiency of cleanup. Results are saved as reports.

Modular script used: **05_preprocessed_reads_quality_control_script.sh**	
➢ **Calls**	: fastqc
➢ **Input**	: 04_quality_trimming / [setname]_[chip/ctrl]_rep[#]_R[1/2].qualitytrimmed.fq.gz
➢ **Process**	: Generate preprocessed reads quality assessment reports
➢ **Output**	: 05_preprocessed_reads_quality_control / [setname]_[chip/ctrl]_rep[#]_R[1/2]_fastqc.html

How does it look like? After **05_preprocessed_reads_quality_control_script.sh** had been executed, the folder: **05_preprocessed_reads_quality_control** will contain all these quality assessment reports for every **qualitytrimmed.fq.gz** file in folder **04_quality_trimming**, just like below:


```
Filename ^
..
05_preprocessed_reads_quality_control_script.sh
161123_chip_rep1.qualitytrimmed_fastqc.html
161123_chip_rep1.qualitytrimmed_fastqc.zip
161123_chip_rep2.qualitytrimmed_fastqc.html
161123_chip_rep2.qualitytrimmed_fastqc.zip
161123_chip_rep3.qualitytrimmed_fastqc.html
161123_chip_rep3.qualitytrimmed_fastqc.zip
161123_chip_rep4.qualitytrimmed_fastqc.html
161123_chip_rep4.qualitytrimmed_fastqc.zip
161123_ctrl_rep1.qualitytrimmed_fastqc.html
161123_ctrl_rep1.qualitytrimmed_fastqc.zip
161123_ctrl_rep2.qualitytrimmed_fastqc.html
161123_ctrl_rep2.qualitytrimmed_fastqc.zip
161123_ctrl_rep3.qualitytrimmed_fastqc.html
161123_ctrl_rep3.qualitytrimmed_fastqc.zip
```


```
Filename ^
..
05_preprocessed_reads_quality_control_script.sh
180906_chip_rep1_R1.qualitytrimmed_fastqc.html
180906_chip_rep1_R1.qualitytrimmed_fastqc.zip
180906_chip_rep1_R2.qualitytrimmed_fastqc.html
180906_chip_rep1_R2.qualitytrimmed_fastqc.zip
180906_chip_rep2_R1.qualitytrimmed_fastqc.html
180906_chip_rep2_R1.qualitytrimmed_fastqc.zip
180906_chip_rep2_R2.qualitytrimmed_fastqc.html
180906_chip_rep2_R2.qualitytrimmed_fastqc.zip
180906_ctrl_rep1_R1.qualitytrimmed_fastqc.html
180906_ctrl_rep1_R1.qualitytrimmed_fastqc.zip
180906_ctrl_rep1_R2.qualitytrimmed_fastqc.html
180906_ctrl_rep1_R2.qualitytrimmed_fastqc.zip
180906_ctrl_rep2_R1.qualitytrimmed_fastqc.html
180906_ctrl_rep2_R1.qualitytrimmed_fastqc.zip
180906_ctrl_rep2_R2.qualitytrimmed_fastqc.html
180906_ctrl_rep2_R2.qualitytrimmed_fastqc.zip
```

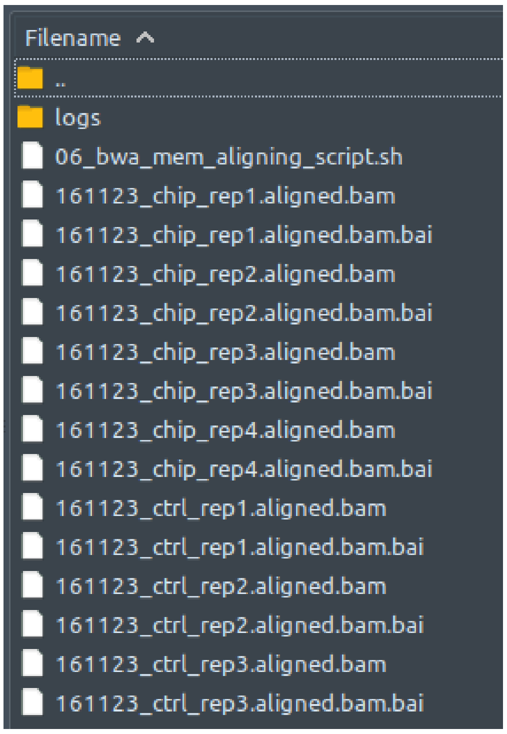
The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples, in which every sample consists of two files: the first read (R1), and the second read (R2).

The **.zip** files contains the individual components to be compiled for the report so you can ignore those. To read the reports, open the **.html** files in your web-browser of choice (Safari, Edge, Firefox). This file is a multitabular file in which you can evaluate the quality of your experiment, sequencing, etc. Comprehensive as it is, explaining the contents in detail would take a whole new guide by itself. Therefore, its best to check out the documentation of the developers for all the details:
https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/.

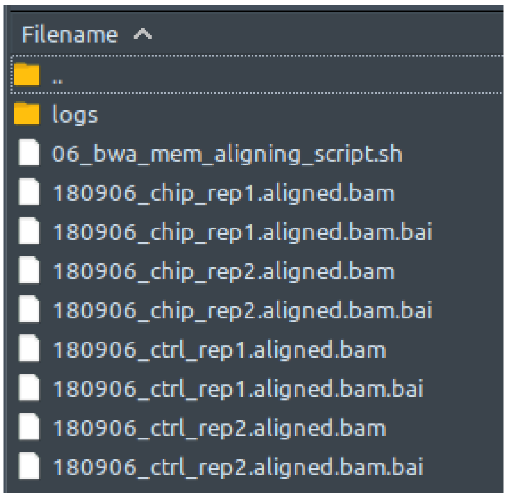
10. **Reads alignment to reference genome**. Performed by the mem algorithm in BWA aligner. The appropriate genome reference for the sample is given as a command line argument. The default genome reference is hg38. Precomputed genome references for hg38, hg19, mm9, mm10, dm6, and sacCer3 are downloaded as part of the ChIP-AP installation process. Tutorials for custom/different genome references can be found on the ChIP-AP GitHub (coming soon!). BWA is used in preference to other aligners such as Bowtie2 as in benchmarking papers (Thankaswamy-Kosalai et al., 2017), we were more satisfied with the results of BWA, hence its inclusion in this pipeline.

Modular script used: **06_bwa_mem_aligning_script.sh**
➢ **Calls** : bwa mem ➢ **Input** : 04_quality_trimming / [setname]_[chip/ctrl]_rep[#]_R[1/2].qualitytrimmed.fq.gz [path to genome folder] / bwa / *(reference genome provided by ChIP-AP)* ➢ **Process** : Align preprocessed reads to the designated reference genome ➢ **Output** : 06_bwa_mem_aligning / [setname]_[chip/ctrl]_rep[#].aligned.bam

How does it look like? After **06_bwa_mem_aligning_script.sh** had been executed, the folder: **06_bwa_mem_aligning** will contain all these aligned reads files (marked by the extension**: .aligned.bam**), just like below:



```
Filename ^
..
logs
06_bwa_mem_aligning_script.sh
161123_chip_rep1.aligned.bam
161123_chip_rep1.aligned.bam.bai
161123_chip_rep2.aligned.bam
161123_chip_rep2.aligned.bam.bai
161123_chip_rep3.aligned.bam
161123_chip_rep3.aligned.bam.bai
161123_chip_rep4.aligned.bam
161123_chip_rep4.aligned.bam.bai
161123_ctrl_rep1.aligned.bam
161123_ctrl_rep1.aligned.bam.bai
161123_ctrl_rep2.aligned.bam
161123_ctrl_rep2.aligned.bam.bai
161123_ctrl_rep3.aligned.bam
161123_ctrl_rep3.aligned.bam.bai
```



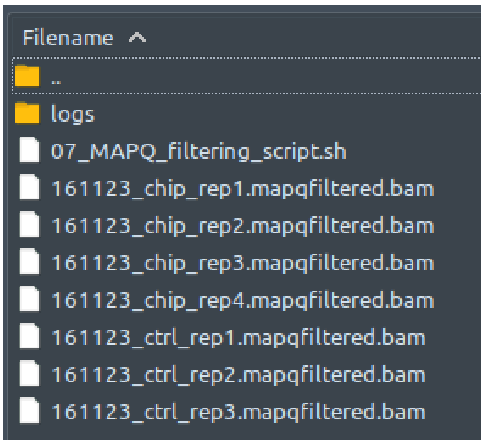
```
Filename ^
..
logs
06_bwa_mem_aligning_script.sh
180906_chip_rep1.aligned.bam
180906_chip_rep1.aligned.bam.bai
180906_chip_rep2.aligned.bam
180906_chip_rep2.aligned.bam.bai
180906_ctrl_rep1.aligned.bam
180906_ctrl_rep1.aligned.bam.bai
180906_ctrl_rep2.aligned.bam
180906_ctrl_rep2.aligned.bam.bai
```

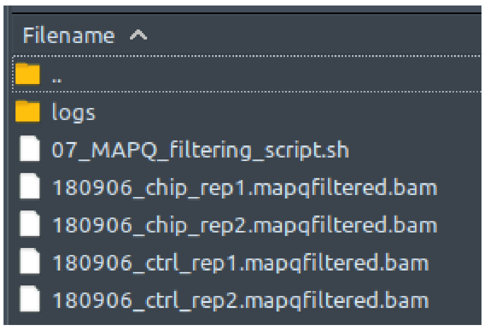
The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples. Note that right here the first reads (R1), and the second reads (R2) had both been aligned into the same reference genome, and thus no longer separated in two different files. Paired-end files are processed in pairs by bwa mem. There should be one aligned reads file here for each processed single-end, or for every two processed paired-end quality-trimmed reads file from folder **04_quality_trimming**. All these **aligned.bam** files will be deleted at the end of **06_bwa_mem_aligning_script.sh** execution if "--*deltemp*" flag is used on ChIP-AP call.

11. **Alignment score quality filtering**. Performed by samtools view. This filter (if set) will remove all reads with alignment score (MAPQ) below a user defined threshold. Reads with suboptimal fit into the genome and/or reads with multiple ambiguous mapped locations can easily be excluded from the reads file using this filter step also. To disable MAPQ filtering, simply remove all flags from the settings table for this step. As for what is an appropriate filter to set? There are many blog posts by bioinformaticians talking about MAPQ inconsistency and what the scores mean, so that discussion is a long one to have. Basically, we have 2 takes on the matter, if you want to include everything, then remove the parameters for this step from the settings table. If you want to be relatively stringent then use the default MAPQ filter of 20.

Modular script used: **07_MAPQ_filtering_script.sh**	
➢ **Calls**	: samtools view
➢ **Input**	: 06_bwa_mem_aligning / [setname]_[chip/ctrl]_rep[#].aligned.bam
➢ **Process**	: Remove reads with low MAPQ (alignment) score
➢ **Output**	: 07_MAPQ_filtering / [setname]_[chip/ctrl]_rep[#].mapqfiltered.bam

How does it look like? After **07_MAPQ_filtering_script.sh** had been executed, the folder: **07_MAPQ_filtering** will contain all these MAPQ-filtered reads files (marked by the extension**: .mapqfiltered.bam**), just like below:


```
Filename ^
..
logs
07_MAPQ_filtering_script.sh
161123_chip_rep1.mapqfiltered.bam
161123_chip_rep2.mapqfiltered.bam
161123_chip_rep3.mapqfiltered.bam
161123_chip_rep4.mapqfiltered.bam
161123_ctrl_rep1.mapqfiltered.bam
161123_ctrl_rep2.mapqfiltered.bam
161123_ctrl_rep3.mapqfiltered.bam
```


```
Filename ^
..
logs
07_MAPQ_filtering_script.sh
180906_chip_rep1.mapqfiltered.bam
180906_chip_rep2.mapqfiltered.bam
180906_ctrl_rep1.mapqfiltered.bam
180906_ctrl_rep2.mapqfiltered.bam
```

The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples. Again, note that right here the first reads (R1), and the second reads (R2) had both been aligned into the same reference genome by bwa mem in above, and thus no longer separated in two different files.

There should be one MAPQ-filtered reads file here for each processed aligned reads file from folder **06_bwa_mem_aligning**. All these **mapqfiltered.bam** files will be deleted at the end of **07_MAPQ_filtering_script.sh** execution if "--*deltemp*" flag is used on ChIP-AP call.

12. **Sorting and indexing of aligned reads files**. Performed by samtools sort and samtools index, which do nothing to the aligned reads files other than sorting and indexing, priming the aligned reads files for further processing.

Modular script used: **08_results_script.sh**
➢ **Calls** : samtools sort
➢ **Input** : 07_MAPQ_filtering / [setname]_[chip/ctrl]_rep[#].mapqfiltered.bam
➢ **Process** : Sort all bam files based on coordinate
➢ **Output** : 08_results / [setname]_[chip/ctrl]_rep[#].bam

Modular script used: **08_results_script.sh**
➢ **Calls** : samtools merge
➢ **Input** : 08_results / [setname]_chip_rep[#].bam 08_results / [setname]_ctrl_rep[#].bam
➢ **Process** : Merge all sorted ChIP bam files and all sorted control bam files.
➢ **Output** : 08_results / [setname]_chip_merged.bam 08_results / [setname]_ctrl_merged.bam
➢ *Condition: --fcmerge flag is used OR unequal number of ChIP and control samples OR peak type is broad*

Modular script used: **08_results_script.sh**
➢ **Calls** : samtools index
➢ **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].bam 08_results / [setname]_[chip/ctrl]_merged.bam
➢ **Process** : Make indices for all coordinate-sorted bam files
➢ **Output** : 08_results / [setname]_[chip/ctrl]_rep[#].bam.bai 08_results / [setname]_[chip/ctrl]_merged.bam.bai

Modular script used: **08_results_script.sh**
➢ **Calls** : samtools sort -n
➢ **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].bam
➢ **Process** : Sort all bam files based on read name
➢ **Output** : 08_results / [setname]_[chip/ctrl]_rep[#]_namesorted.bam

How does it look like? *Detailed output preview of step 12, 13, and 14 are combined below step 14*

13. **ChIP pulldown efficiency assessment**. Performed by plotFingerprint from the deeptools package, which generates fingerprint plots. These serve as a quality control figure that shows DNA pulldown efficiency of the ChIP experiment. Refer to the appropriate documentation for full details but in short – the input should be as close to the 1:1 diagonal as possible and the better enrichment seen in your sample, the more its curve will bend towards the bottom right. You want (ideally) a large gap between the chip and the control samples. PNG files are provided for easy viewing, SVG files provided if you want to make HQ versions later for publication.

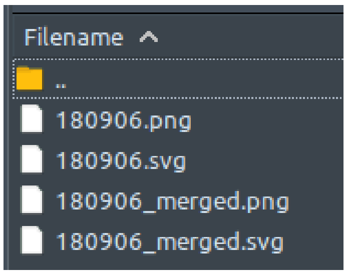
Modular script used: **08_results_script.sh**	
➢ **Calls**	: plotFingerprint
➢ **Input**	: 08_results / [setname]_[chip/ctrl]_rep[#].bam
➢ **Process**	: Generate fingerprint plots for all bam files
➢ **Output**	: 08_results / fingerprint_plots/[setname].[png/svg]
	08_results / fingerprint_plots/[setname]_merged.[png/svg]

How does it look like? *Detailed output preview of step 12, 13, and 14 are combined below step 14*

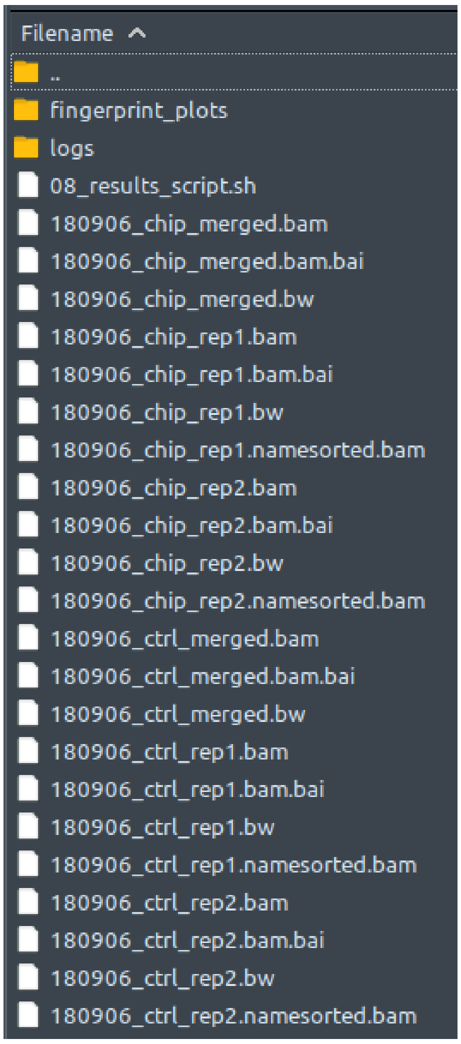
14. **Visualization track generation of aligned reads files**. Performed by bamCoverage from the deeptools package. Generates bigwig files for quick and simple visualization of reads distribution along the referenced genome using local tools such as IGV. The Coverage tracks can be uploaded to genome browsers such as UCSC or a track hub needs to be generated and uploaded to a publicly accessible server. This is not something ChIP-AP does at this stage (and frankly we don't want to do this unless there's a huge demand from the public to automate this). If you're running ChIP-AP on your computer then just view it locally. But I want to share the results with colleagues and just give them the UCSC link…. Oh, look a cricket!!!

Modular script used: **08_results_script.sh**	
➢ **Calls**	: bamCoverage
➢ **Input**	: 08_results / [setname]_[chip/ctrl]_rep[#].bam
	08_results / [setname]_[chip/ctrl]_merged.bam
➢ **Process**	: Generate BigWig coverage file for each individual bam file
➢ **Output**	: 08_results / [setname]_[chip/ctrl]_rep[#].bw
	08_results / [setname]_[chip/ctrl]_merged.bw

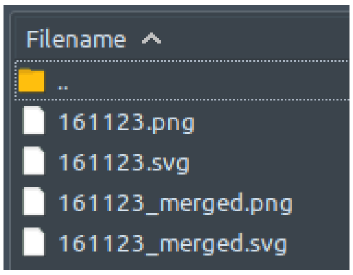
How does it look like? After **08_results_script.sh** had been executed, the folder: **08_results** will contain all these sorted reads files (marked by the extension: **.bam**), a couple merged sorted reads files* (marked by the extension: **_merged.bam**), indices to all the sorted reads files (marked by the extension: **.bam.bai**), the same sorted reads files re-sorted by name (marked by the extension: **namesorted.bam**), bigwig files of all the sorted reads files (marked by the extension: **.bw**), fingerprint plot files of all the sorted reads files (in its own folder: **fingerprint_plots**), and all the log files from all the program calls by **08_results_script.sh**.



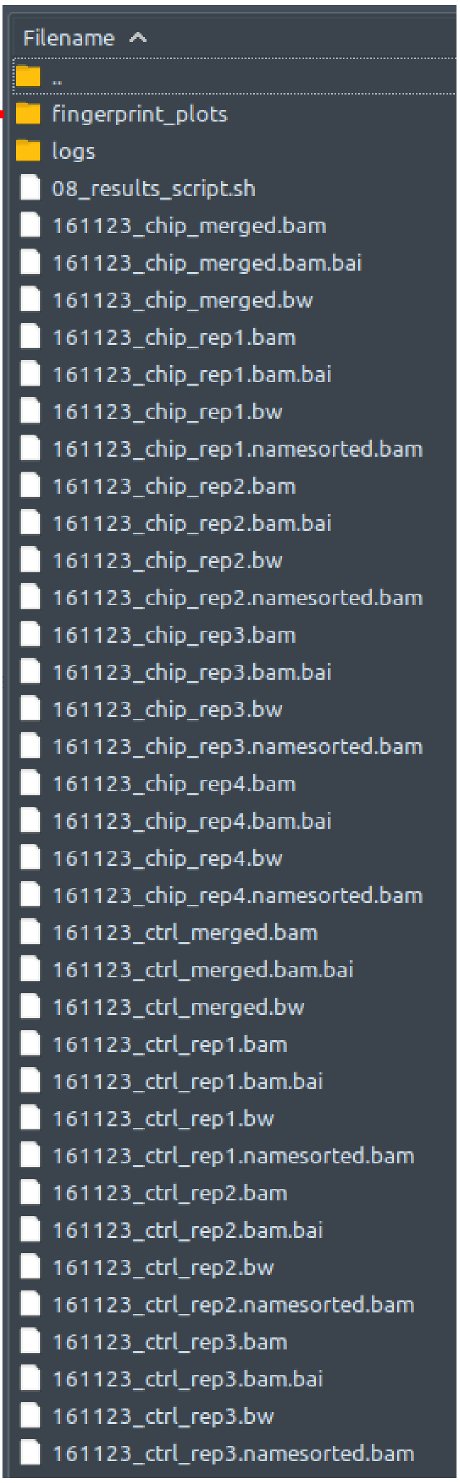
```
Filename ^
..
180906.png
180906.svg
180906_merged.png
180906_merged.svg
```



```
Filename ^
..
fingerprint_plots
logs
08_results_script.sh
180906_chip_merged.bam
180906_chip_merged.bam.bai
180906_chip_merged.bw
180906_chip_rep1.bam
180906_chip_rep1.bam.bai
180906_chip_rep1.bw
180906_chip_rep1.namesorted.bam
180906_chip_rep2.bam
180906_chip_rep2.bam.bai
180906_chip_rep2.bw
180906_chip_rep2.namesorted.bam
180906_ctrl_merged.bam
180906_ctrl_merged.bam.bai
180906_ctrl_merged.bw
180906_ctrl_rep1.bam
180906_ctrl_rep1.bam.bai
180906_ctrl_rep1.bw
180906_ctrl_rep1.namesorted.bam
180906_ctrl_rep2.bam
180906_ctrl_rep2.bam.bai
180906_ctrl_rep2.bw
180906_ctrl_rep2.namesorted.bam
```



```
Filename ^
..
161123.png
161123.svg
161123_merged.png
161123_merged.svg
```



```
Filename ^
..
fingerprint_plots
logs
08_results_script.sh
161123_chip_merged.bam
161123_chip_merged.bam.bai
161123_chip_merged.bw
161123_chip_rep1.bam
161123_chip_rep1.bam.bai
161123_chip_rep1.bw
161123_chip_rep1.namesorted.bam
161123_chip_rep2.bam
161123_chip_rep2.bam.bai
161123_chip_rep2.bw
161123_chip_rep2.namesorted.bam
161123_chip_rep3.bam
161123_chip_rep3.bam.bai
161123_chip_rep3.bw
161123_chip_rep3.namesorted.bam
161123_chip_rep4.bam
161123_chip_rep4.bam.bai
161123_chip_rep4.bw
161123_chip_rep4.namesorted.bam
161123_ctrl_merged.bam
161123_ctrl_merged.bam.bai
161123_ctrl_merged.bw
161123_ctrl_rep1.bam
161123_ctrl_rep1.bam.bai
161123_ctrl_rep1.bw
161123_ctrl_rep1.namesorted.bam
161123_ctrl_rep2.bam
161123_ctrl_rep2.bam.bai
161123_ctrl_rep2.bw
161123_ctrl_rep2.namesorted.bam
161123_ctrl_rep3.bam
161123_ctrl_rep3.bam.bai
161123_ctrl_rep3.bw
161123_ctrl_rep3.namesorted.bam
```

The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples. There should be one sorted reads file here for each processed MAPQ-filtered reads file from folder **07_MAPQ_filtering**; a merged ChIP and a merged control reads files*; one index file for each sorted reads file in this folder; one name-sorted reads file for each sorted reads file in this folder**; one bigwig file for each sorted reads file in this folder; two fingerprint plot files in the **fingerprint_plots** folder (in extension: **.png** and **.svg**); and another two fingerprint plot files in the **fingerprint_plots** folder (in extension: **.png** and **.svg**)***.

To view and analyze the read distribution of your peaks, load the BigWig files (**.bw**) into the program: **IGV** (more details down below). To view and evaluate your ChIP experiment DNA pulldown efficiency, open the **.png** or **.svg** using any supporting image viewer program (more details down below).

* Only when needed by the pipeline

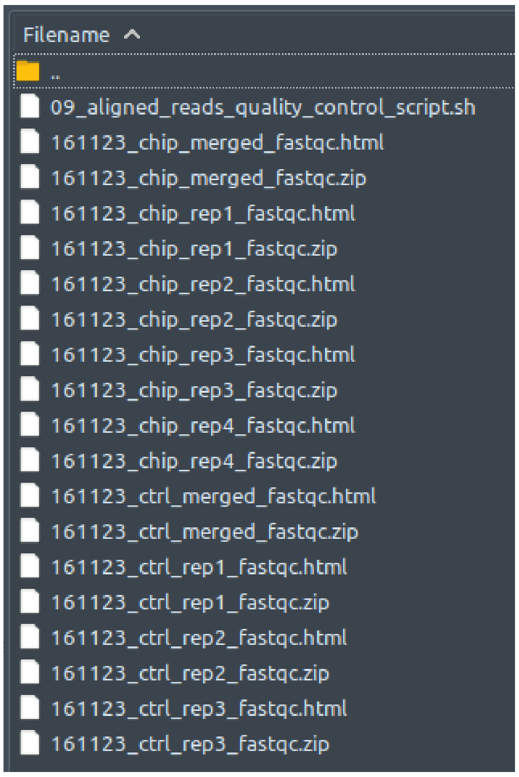
** Except the merged ones

*** Only when merged sorted reads files are present

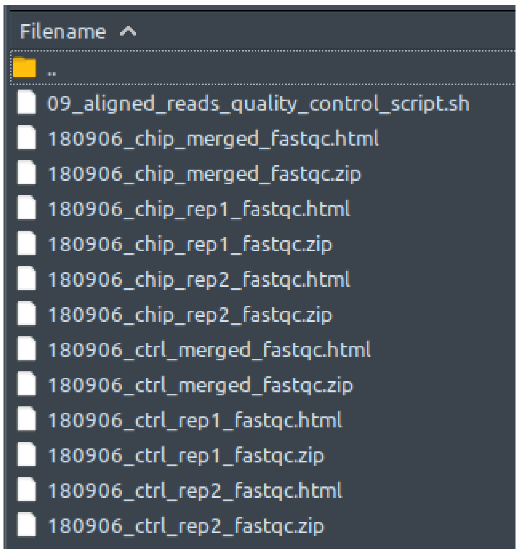
15. **Aligned reads quality assessment**. Processed by FastQC. Quality assessment is performed to check for the alignment efficiency, such as how many reads failed to be mapped. Assessment results are saved as reports.

Modular script used: **09_aligned_reads_quality_control_script.sh**	
➤ **Calls**	: fastqc
➤ **Input**	: 08_results / [setname]_[chip/ctrl]_rep[#].bam 08_results / [setname]_[chip/ctrl]_merged.bam
➤ **Process**	: Generate raw reads quality assessment reports
➤ **Output**	: 08_results / [setname]_[chip/ctrl]_rep[#]_fastqc.html 08_results / [setname]_[chip/ctrl]_merged_fastqc.html

How does it look like? After **09_aligned_reads_quality_control_script.sh** had been executed, the folder: **09_aligned_reads_quality_control** will contain all these quality assessment reports for every **.bam** file in folder **08_results**, just like below:



```
Filename ^
..
09_aligned_reads_quality_control_script.sh
161123_chip_merged_fastqc.html
161123_chip_merged_fastqc.zip
161123_chip_rep1_fastqc.html
161123_chip_rep1_fastqc.zip
161123_chip_rep2_fastqc.html
161123_chip_rep2_fastqc.zip
161123_chip_rep3_fastqc.html
161123_chip_rep3_fastqc.zip
161123_chip_rep4_fastqc.html
161123_chip_rep4_fastqc.zip
161123_ctrl_merged_fastqc.html
161123_ctrl_merged_fastqc.zip
161123_ctrl_rep1_fastqc.html
161123_ctrl_rep1_fastqc.zip
161123_ctrl_rep2_fastqc.html
161123_ctrl_rep2_fastqc.zip
161123_ctrl_rep3_fastqc.html
161123_ctrl_rep3_fastqc.zip
```



```
Filename ^
..
09_aligned_reads_quality_control_script.sh
180906_chip_merged_fastqc.html
180906_chip_merged_fastqc.zip
180906_chip_rep1_fastqc.html
180906_chip_rep1_fastqc.zip
180906_chip_rep2_fastqc.html
180906_chip_rep2_fastqc.zip
180906_ctrl_merged_fastqc.html
180906_ctrl_merged_fastqc.zip
180906_ctrl_rep1_fastqc.html
180906_ctrl_rep1_fastqc.zip
180906_ctrl_rep2_fastqc.html
180906_ctrl_rep2_fastqc.zip
```

The left figure shows a single-end dataset with four ChIP samples and three control samples. The right figure shows a paired-end dataset with two ChIP samples and two control samples.

The **.zip** files contains the individual components to be compiled for the report so you can ignore those. To read the reports, open the **.html** files in your web-browser of choice (Safari, Edge, Firefox). This file is a multitabular file in which you can evaluate the quality of your experiment, sequencing, etc. Comprehensive as it is, explaining the contents in detail would take a whole new guide by itself. Therefore, its best to check out the documentation of the developers for all the details:
https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/.

16. **Peak calling**.

<u>For Transcription Factors -</u> Performed by MACS2 (default setting), GEM, HOMER (factor setting), and Genrich for transcription factor proteins of interest.

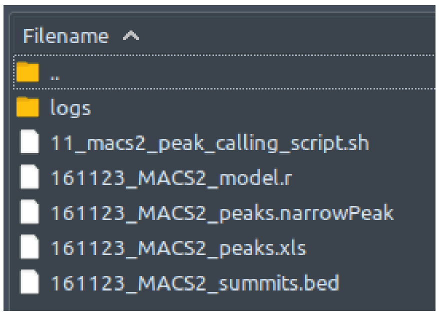
<u>For Broad Peaks (Histone Marks) -</u> Performed by MACS2 (broad setting), SICER2, HOMER (broad setting), and Genrich for histone modifier protein of interest.

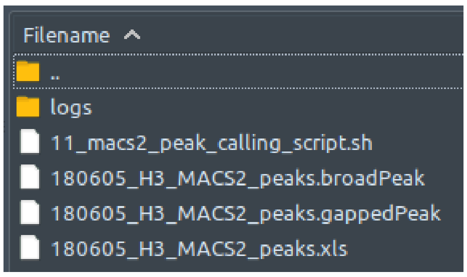
The same track of aligned reads is scanned for potential protein-DNA binding sites. The process returns a list of enriched regions in various formats.

MACS

Modular script used: **11_macs2_peak_calling_script.sh**
➢ **Calls** : macs2 callpeak ➢ **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].bam ➢ **Process** : Generate a list of called peaks ➢ **Output** : 11_macs2_peak_calling / [setname]_MACS2_peaks.narrowPeak

How does it look like? After **11_macs2_peak_calling_script.sh** had been executed, the folder: **11_macs2_peak_calling** will contain all these files, just like below:


```
Filename ^
..
logs
11_macs2_peak_calling_script.sh
161123_MACS2_model.r
161123_MACS2_peaks.narrowPeak
161123_MACS2_peaks.xls
161123_MACS2_summits.bed
```


```
Filename ^
..
logs
11_macs2_peak_calling_script.sh
180605_H3_MACS2_peaks.broadPeak
180605_H3_MACS2_peaks.gappedPeak
180605_H3_MACS2_peaks.xls
```

For some reason, MACS2 does not generate the statistical model of the dataset's background noise (**.r**) in paired-end mode (right figure).

MACS2 generates three output files, which all are basically peak lists, but each of these has their own exclusive information. In datasets with narrow peak type (left figure) the file that contains all necessary information relevant to analysis by ChIP-AP is the one with **.narrowPeak** extension. The **peaks.xls** file has the pileup value and peak length information, which tells us about the overall coverage of the corresponding peak region. The **summits.bed** basically are just lists of only the peak summits coordinates and read depth at the respective 1 base coordinate.

On the other hand, in datasets with broad peak type (right figure) the file that contains all necessary information relevant to analysis by ChIP-AP is the one with **.broadPeak** extension. The **.gappedPeak** file is basically a variant of **.broadPeak**, which is dedicated for peaks with both narrow and broad characteristics mixed in together, and thus has values that describes where and how deep are the "thick" and "thin" regions along each peak. As in the case of narrow peak type, the **peaks.xls** file has the pileup value and peak length information, which tells us about the overall coverage of the corresponding peak region. No **summits.bed** file generated in this case because of the absence of such "summit" in broad peaks.

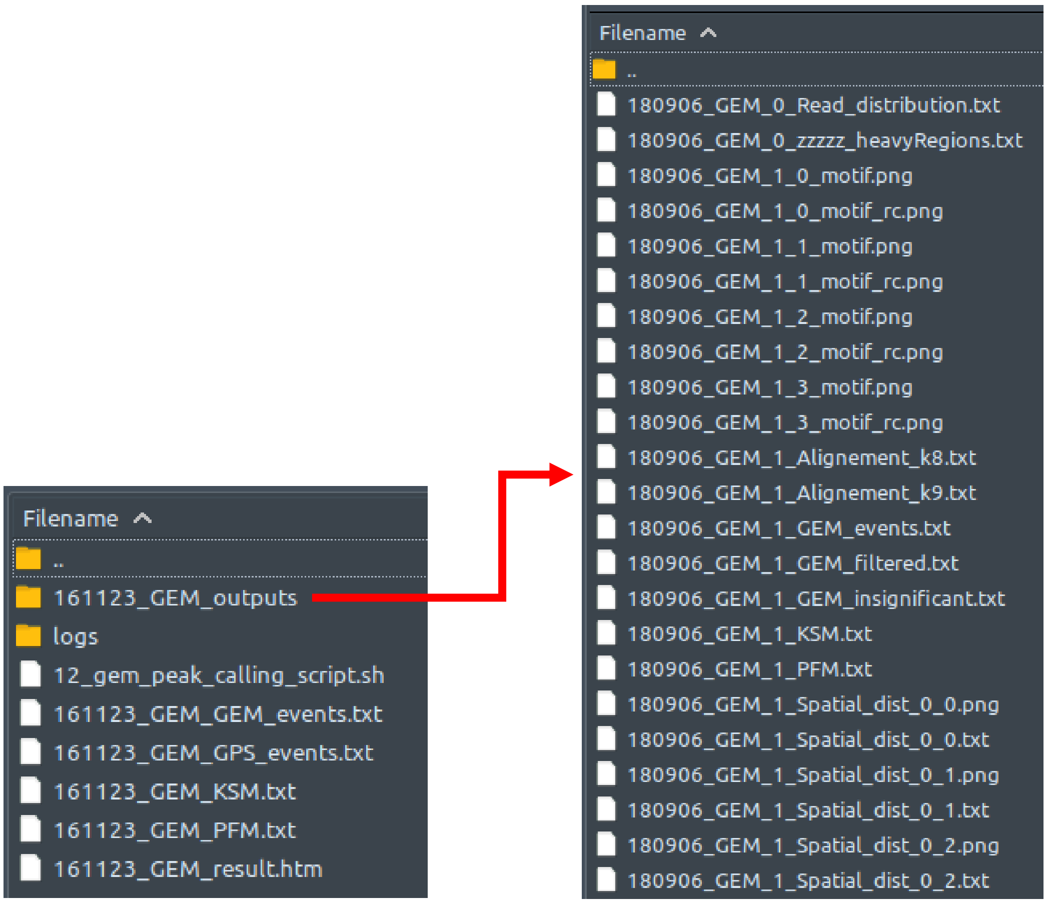
The **.narrowPeak** file which ChIP-AP utilizes, has the following columns:

1. **chrom** - Name of the chromosome (or contig, scaffold, etc.).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.
4. **name** - Name given to a region. Use "." if no name is assigned.
5. **score** - Indicates how dark the peak will be displayed in the browser (0-1000). If all scores were "'0"' when the data were submitted to the DCC, the DCC assigned scores 1-1000 based on signal value. Ideally the average signalValue per base spread is between 100-1000.
6. **strand** - +/- to denote strand or orientation (whenever applicable). Use "." if no orientation is assigned.
7. **signalValue** - Overall (usually, average) enrichment for the region.
8. **pValue** - Statistical significance (-log10). Use -1 if no pValue is assigned.
9. **qValue** - Statistical significance using false discovery rate (-log10). Use -1 if no qValue is assigned.
10. **peak** - Point-source called for this peak; 0-based offset from chromStart. Use -1 if no point-source called.

GEM

Modular script used: **12_gem_peak_calling_script.sh**
➢ **Calls** : gem ➢ **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].bam ➢ **Process** : Generate a list of called peaks ➢ **Output** : 12_gem_peak_calling / [setname]_GEM_GEM_events.txt ➢ *Condition: Peak type is narrow. GEM is replaced by SICER2 for broad peak type.*

How does it look like? After **12_gem_peak_calling_script.sh** had been executed, the folder: **12_gem_peak_calling** will contain all these files, just like below:



```
Filename ^
..
161123_GEM_outputs
logs
12_gem_peak_calling_script.sh
161123_GEM_GEM_events.txt
161123_GEM_GPS_events.txt
161123_GEM_KSM.txt
161123_GEM_PFM.txt
161123_GEM_result.htm
```

```
Filename ^
..
180906_GEM_0_Read_distribution.txt
180906_GEM_0_zzzzz_heavyRegions.txt
180906_GEM_1_0_motif.png
180906_GEM_1_0_motif_rc.png
180906_GEM_1_1_motif.png
180906_GEM_1_1_motif_rc.png
180906_GEM_1_2_motif.png
180906_GEM_1_2_motif_rc.png
180906_GEM_1_3_motif.png
180906_GEM_1_3_motif_rc.png
180906_GEM_1_Alignement_k8.txt
180906_GEM_1_Alignement_k9.txt
180906_GEM_1_GEM_events.txt
180906_GEM_1_GEM_filtered.txt
180906_GEM_1_GEM_insignificant.txt
180906_GEM_1_KSM.txt
180906_GEM_1_PFM.txt
180906_GEM_1_Spatial_dist_0_0.png
180906_GEM_1_Spatial_dist_0_0.txt
180906_GEM_1_Spatial_dist_0_1.png
180906_GEM_1_Spatial_dist_0_1.txt
180906_GEM_1_Spatial_dist_0_2.png
180906_GEM_1_Spatial_dist_0_2.txt
```

GEM outputs both the binding event files and the motif files. Because of the read distribution re-estimation, GEM outputs event prediction and read distribution files for multiple rounds. All of these are in saved in folder **[setname]_GEM_outputs**. However, as long as ChIP-AP is concerned, we are only interested in GEM's final output files, which are saved outside the said folder. Each of these has their own exclusive information. GEM is actually a suite consisting of multiple modules performing their specific tasks. The GPS module is the one that detects peaks based on reads distribution (similar to most peak callers). The resulting peak list from solely running this GPS module can be viewed in file **[setname]_GEM_GPS_events.txt**.

However, GEM is also equipped with motif enrichment analysis module that helps improve true peaks detection. This GPS peak list is then processed further and modified based on the motif enrichment analysis, resulting in the final peak list in file **[setname]_GEM_GEM_events.txt**, which is the one utilized by ChIP-AP.

GEM also generates two secondary output files **[setname]_GEM_KSM.txt** and **[setname]_GEM_PFM.txt**, which are more of motif enrichment results rather than peak lists, and thus will not be discussed further here. Do check the GEM documentation link provided at the end of this guide if you are interested. Lastly, **[setname]_GEM_result.htm** is a web-based comprehensive summary of all the binding events and motifs.

The **_GEM_events.txt** file which ChIP-AP utilizes, has the following columns:

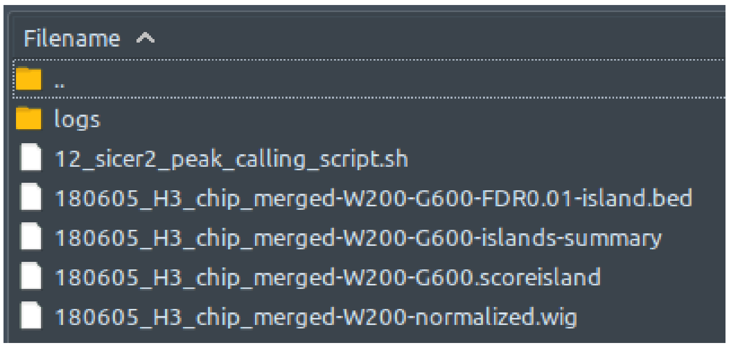
1. **Location -** The genome coordinate of this binding event
2. **IP binding strength -** The number of IP reads associated with the event
3. **Control binding strength -** the number of control reads in the corresponding region
4. **Fold -** Fold enrichment (IP/Control)
5. **Expected binding strength -** The number of IP read counts expected in the binding region given its local context (defined by parameter W2 or W3), this is used as the Lambda parameter for the Poisson test
6. **Q_-lg10 -** -log10(q-value), the q-value after multiple-testing correction, using the larger p-value of Binomial test and Poisson test
7. **P_-lg10 -** -log10(p-value), the p-value is computed from the Binomial test given the IP and Control read counts (when there are control data)
8. **P_poiss -** -log10(p-value), the p-value is computed from the Poisson test given the IP and Expected read counts (without considering control data)
9. **IPvsEMP -** Shape deviation, the KL divergence of the IP reads from the empirical read distribution (log10(KL)), this is used to filter predicted events given the --sd cutoff (default=-0.40).
10. **Noise -** The fraction of the event read count estimated to be noise
11. **KmerGroup -** The group of the k-mers associated with this binding event, only the most significant k-mer is shown, the n/n values are the total number of sequence hits of the k-mer group in the positive and negative training sequences (by default total 5000 of each), respectively
12. **KG_hgp -** log10(hypergeometric p-value), the significance of enrichment of this k-mer group in the positive vs negative training sequences (by default total 5000 of each), it is the hypergeometric p-value computed using the pos/neg hit counts and total counts
13. **Strand -** The sequence strand that contains the k-mer group match, the orientation of the motif is determined during the GEM motif discovery, '*' represents that no k-mer is found to associated with this event

SICER2

Modular script used: **12_sicer2_peak_calling_script.sh**

- **Calls** : sicer
- **Input** : 08_results / [setname]_[chip/ctrl]_merged.bam
- **Process** : Generate a list of called peaks
- **Output** : 12_sicer2_peak_calling / [setname]_chip_merged-W*-G*-islands-summary
  (* depends on -w and -g flag arguments. Defaults are 200 and 600, respectively)
- *Condition:Peak type is broad. SICER2 is replaced by GEM for narrow peak type.*

How does it look like? After **12_sicer2_peak_calling_script.sh** had been executed, the folder: **12_sicer2_peak_calling** will contain all these files, just like below:



```
Filename
..
logs
12_sicer2_peak_calling_script.sh
180605_H3_chip_merged-W200-G600-FDR0.01-island.bed
180605_H3_chip_merged-W200-G600-islands-summary
180605_H3_chip_merged-W200-G600.scoreisland
180605_H3_chip_merged-W200-normalized.wig
```

SICER2 generates multiple output files as follows:

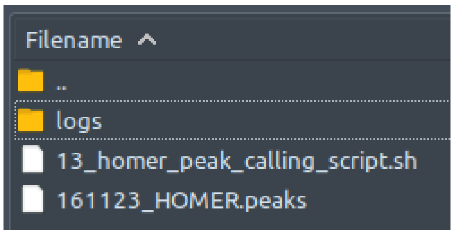
- **[setname]_chip_merged-W*-G*-.scoreisland**: delineation of significant islands controlled by E- value of 1000. It is in “chrom start end score” format.
- **[setname]_chip_merged-W*-normalized.wig**: wig file that can be used to visualize the windows generated by SICER2. Read count is normalized by library size per million.
- **[setname]_chip_merged-W*-G*-islands-summary**: summary of all candidate islands with their statistical significance. It is a tab-separated-values file that has the following columns format: **chrom, start, end, ChIP_island_read_count, CONTROL_island_read_count, p_value, fold_change, FDR_threshold.**
- **[setname]_chip_merged-W*-G*-FDR*-island.bed**: delineation of significant islands filtered by false discovery rate (FDR). It has the following format: chrom, start, end, read-count.

The file that contains all necessary information relevant to analysis by ChIP-AP is the one with **[setname]_chip_merged-W*-G*-islands-summary.**

HOMER

Modular script used: **13_homer_peak_calling_script.sh**
➢ **Calls** : findPeaks ➢ **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].bam ➢ **Process** : Generate a list of called peaks ➢ **Output** : 13_homer_peak_calling / [setname]_HOMER.peaks

How does it look like? After **13_homer_peak_calling_script.sh** had been executed, the folder: **13_homer_peak_calling** will contain all these files, just like below:



```
Filename ^
..
logs
13_homer_peak_calling_script.sh
161123_HOMER.peaks
```

There is no output files difference between single-end and paired-end mode.

Even though HOMER has different modes for narrow peak type (using flag argument -style factor) and broad peak type (using flag argument -style histone), the output files stay the same and so do the contents of these files, except for column 7 (see below)

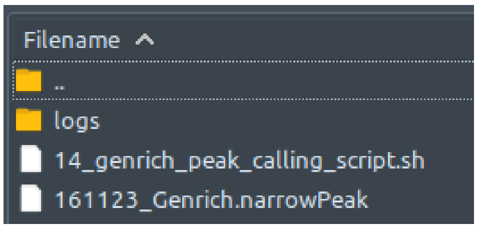
The **HOMER.peaks** file which CHIP-AP utilizes, has the following columns:

1. **PeakID -** Unique name for each peak
2. **chr -** Chromosome where peak is located
3. **start -** Starting position of peak
4. **end -** Ending position of peak
5. **Strand** (+/-)
6. **Normalized Tag Counts -** Number of tags found at the peak, normalized to 10 million total mapped tags (or defined by the user)
7. **Focus Ratio -** Fraction of tags found appropriately upstream and downstream of the peak center. (when sample peak type = narrow), **<u>OR</u>**
   Region Size - Length of enriched region (when sample peak type = broad)
8. **Peak score** (read HOMER annotatePeaks documentation for details)
9. **Total Tags -** Peak depth in the ChIP sample (normalized to control)
10. **Control Tags -** Peak depth in the control sample
11. **Fold Change vs Control -** Peak depth fold change of ChIP compared to control
12. **p-value vs Control -** Statistical significance of ChIP peak compared to control
13. **Fold Change vs Local -** Peak depth fold change of ChIP compared to its surrounding regions
14. **p-value vs Local -** Statistical significance of ChIP peak compared to its surrounding regions
15. **Clonal Fold Change -** Statistical significance of ChIP peak considering the abundance of read fragment clones, or duplicates

Genrich

Modular script used: **14_genrich_peak_calling_script.sh**
➢ **Calls** : Genrich ➢ **Input** : 08_results / [setname]_[chip/ctrl]_rep[#]_namesorted.bam ➢ **Process** : Generate a list of called peaks ➢ **Output** : 14_genrich_peak_calling / [setname]_Genrich.narrowPeak

How does it look like? After **14_genrich_peak_calling_script.sh** had executed, the folder: **14_genrich_peak_calling** will contain all these files, just like below:



```
Filename ^
..
logs
14_genrich_peak_calling_script.sh
161123_Genrich.narrowPeak
```

There is no output files difference between single-end and paired-end mode.

The **.narrowPeak** file which ChIP-AP utilizes, has the following columns:

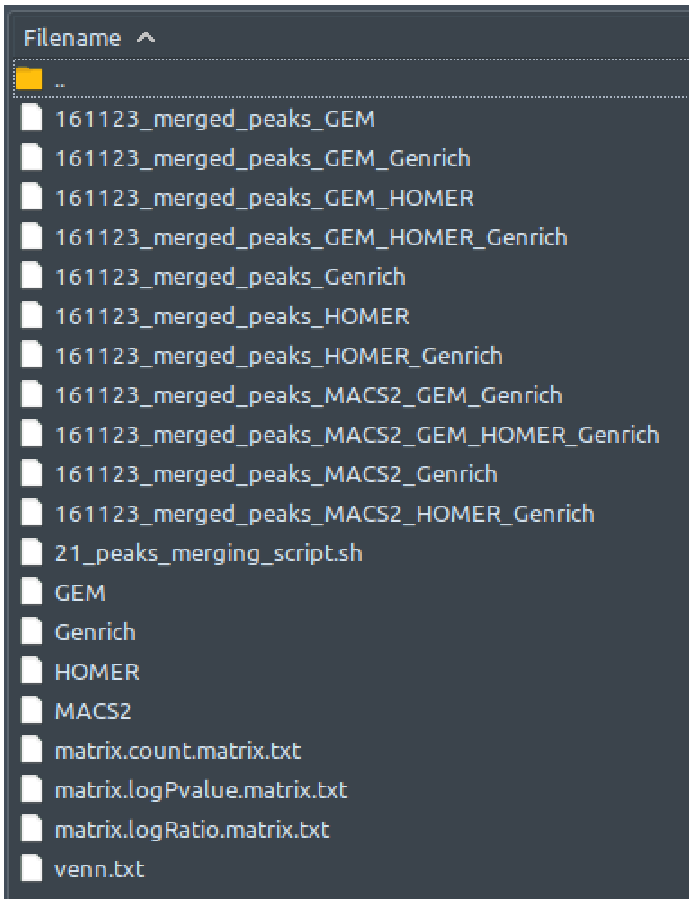
1. **chrom** - Name of the chromosome (or contig, scaffold, etc.).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.
4. **name** - Name given to a region. Use "." if no name is assigned.
5. **score** - Indicates how dark the peak will be displayed in the browser (0-1000). If all scores were "'0"' when the data were submitted to the DCC, the DCC assigned scores 1-1000 based on signal value. Ideally the average signalValue per base spread is between 100-1000.
6. **strand** - +/- to denote strand or orientation (whenever applicable). Use "." if no orientation is assigned.
7. **signalValue** - Overall (usually, average) enrichment for the region.
8. **pValue** - Statistical significance (-log10). Use -1 if no pValue is assigned.
9. **qValue** - Statistical significance using false discovery rate (-log10). Use -1 if no qValue is assigned.
10. **peak** - Point-source called for this peak; 0-based offset from chromStart. Use -1 if no point-source called.

17. **Peaks merging**. Performed by a custom script and HOMER's mergePeaks. The custom script reformats necessary peak caller outputs into HOMER region list format. mergePeaks looks for overlaps between the regions in the four peak caller outputs and lists the merged regions in multiple files based on the peak caller(s) that calls them.

Modular script used: **21_peaks_merging_script.sh**

- **Calls** : mergePeaks
- **Input** : 11_macs2_peak_calling / [setname]_MACS2_peaks.narrowPeak
  12_gem_peak_calling / [setname]_GEM_GEM_events.txt (narrow peak only)
  12_sicer2_peak_calling / [setname]-W*-G*-islands-summary (broad peak only)
  13_homer_peak_calling / [setname]_HOMER.peaks
  14_genrich_peak_calling / [setname]_Genrich.narrowPeak
- **Process** : Generate multiple lists of merged peak coordinates based on peak callers
- **Output** : 21_peaks_merging / [setname]_merged_peaks*
  * is the combination of peak callers where the listed peaks in are found in
  (e.g., [setname] _merged_peaks_MACS2_Genrich)

How does it look like? After **21_peaks_merging_script.sh** had executed, the folder: **21_peaks_merging** will contain all these files, just like below:


```
Filename ^
..
161123_merged_peaks_GEM
161123_merged_peaks_GEM_Genrich
161123_merged_peaks_GEM_HOMER
161123_merged_peaks_GEM_HOMER_Genrich
161123_merged_peaks_Genrich
161123_merged_peaks_HOMER
161123_merged_peaks_HOMER_Genrich
161123_merged_peaks_MACS2_GEM_Genrich
161123_merged_peaks_MACS2_GEM_HOMER_Genrich
161123_merged_peaks_MACS2_Genrich
161123_merged_peaks_MACS2_HOMER_Genrich
21_peaks_merging_script.sh
GEM
Genrich
HOMER
MACS2
matrix.count.matrix.txt
matrix.logPvalue.matrix.txt
matrix.logRatio.matrix.txt
venn.txt
```

There is no output files difference between single-end and paired-end mode. Also, you will find SICER2 instead of GEM when running broad peak type datasets.

ChIP-AP takes the input files described above, then generates the four tab-separated-values files **MACS2**, **GEM** or **SICER2**, **HOMER**, and **Genrich**, which are lists of peaks detected by their respective peak caller.

With **MACS2**, **GEM** or **SICER2**, **HOMER**, and **Genrich** as the input peak list, HOMER mergePeaks generates separate files based on overlapping peaks for each set of peaks: **21_peaks_merging/[setname]_merged_peaks***, where * is the combination of peak callers where the listed peaks in are found in. Files with certain peak callers combinations that contains zero peak will be non-existent in this folder, so don't be alarmed if, for example, you cannot find **[setname]_merged_peaks_MACS2_GEM** file. That simply means that there is no peak that is detected ONLY by MACS2 and GEM, just like what we can see from the example above.

The following three **matrix.txt** files (below) are the secondary outputs of **HOMER mergePeaks**. These files contains the statistics about the pairwise overlap of peaks between the four callers peak sets, which could provide additional information for you despite them not being of any use for further processes down the line. The explanations below are taken directly from HOMER documentation - no further explanation is given by HOMER, so we cannot give a clearer explanation:

- **matrix.logPvalue.matrix.txt**: natural log p-values for overlap using the hypergeometric distribution, positive values signify divergence
- **matrix.logRatio.matrix.txt**: natural log of the ratio of observed overlapping peaks to the expected number of overlapping peaks
- **matrix.count.matrix.txt**: raw counts of overlapping peaks

Finally, there is the file **venn.txt**. This contains the numbers needed for you to create a Venn diagram depicting the peak overlaps between the four peak caller sets. Some custom scripts are available online which are able to directly take this **venn.txt** file as an input and generates a Venn diagram image as a result.

The resulting multiple **21_peaks_merging/[setname]_merged_peaks*** files are then concatenated together into a single list file **[setname]_all_peaks_concatenated.tsv** by **22_peaks_processing_script.sh** (the subsequent script in the pipeline), and saved in folder **22_peaks_processing**.

Modular script used: **22_peaks_processing_script.sh**

- **Operation**: cat (Bash)
- **Input** : 21_peaks_merging / [setname]_merged_peaks*
- **Process** : Generate concatenated list of peak coordinates
- **Output** : 22_peaks_processing / [setname]_all_peaks_concatenated.tsv

How does it look like? *Detailed output preview of this, step 18, 19, and 20 are combined below step 20*

18. **Peaks annotation**. Performed by annotatePeaks from HOMER package. Each region in the concatenated list is annotated based on its genomic location for the genome specified. The process returns the same list of regions, with each entry row appended with various information pertaining to the gene name, database IDs, category, and instances of motif (if HOMER known motif matrix file is provided to ChIP-AP), etc.

Modular script used: **22_peaks_processing_script.sh**

➢ **Calls**	: annotatePeaks
➢ **Input**	: 22_peaks_processing/[setname]_all_peaks_concatenated.tsv
➢ **Process**	: Append gene annotations to the list of peak coordinates
➢ **Output**	: 22_peaks_processing/[setname]_all_peaks_annotated.tsv

How does it look like? *Detailed output preview step 18, 19, and 20 are combined below step 20*

19. **Fold enrichment calculations**. Performed by a custom script, with the help of samtools depth and view modules. For weighted peak center fold enrichment calculations in cases of narrow peak type datasets, the custom script sends out the reformatted genomic regions as command line arguments for multi-threaded samtools depth runs. Samtools depth returns a list of read depths at each base within the region and saves them in a temporary file. The script then reads the temporary files and determine the weighted peak centers and returns the read depth values along with the base locations. The custom script sends out the weighted peak center base locations as command line arguments for multi-threaded samtools view runs. Samtools view returns the read depth values at the given base locations. The custom script then calculates the fold enrichment values, corrected based on the ChIP-to-control normalization factor.

    For average fold enrichment calculation in cases of broad peak type datasets, samtools view simply sums up the number of reads in the whole peak region, then calculates the fold enrichment values, corrected based on the ChIP-to-control normalization factor.

    In addition, the custom-made script also makes some reformatting and provides additional information necessary for downstream analysis.

Modular script used: **22_peaks_processing_script.sh**

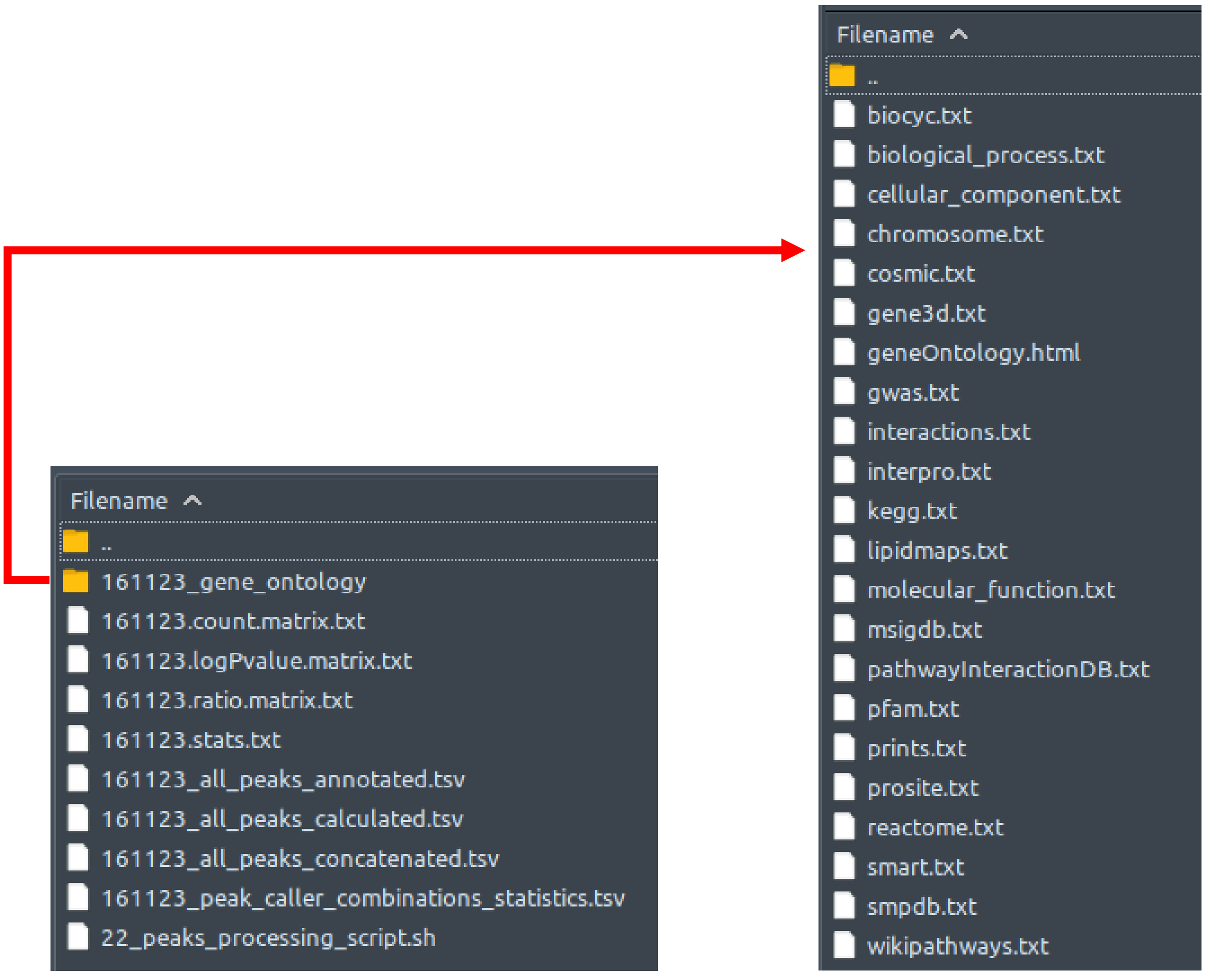
➢ **Calls**	: fold_change_calculator_suite_xx.py
➢ **Input**	: 22_peaks_processing / [setname]_all_peaks_annotated.tsv
➢ **Process**	: Calculate ChIP tag counts (read depth) Calculate weighted center fold change (narrow peak only) Calculate average fold change (broad peak only) Calculate number of peak callers overlaps Calculate number of user-provided (via --motif flag) motif instances found
➢ **Output**	: 22_peaks_processing / [setname]_all_peaks_calculated.tsv *(Ready to view if user does not wish for gene ontology or pathway annotations)*

How does it look like? *Detailed output preview step 18, 19, and 20 are combined below step 20*

20. **Peak statistics summary**. Performed by a custom script designed for quality assessment of called peaks. Returns a summary text file containing information pertaining to the peak read depth, peak fold enrichment, known motif hits, and positive peak hits (based on known motif presence if a HOMER formatted motif file was included when calling ChIP-AP), in each peak set along the continuum between single peak callers and the absolute consensus of all four peak callers.

Modular script used: **22_peaks_processing_script.sh**	
➢ **Calls**	: peak_caller_stats_suite.py
➢ **Input**	: 22_peaks_processing / [setname]_all_peaks_calculated.tsv
➢ **Process**	: Generate a separate summary table of key statistics in peak callers performance
➢ **Output**	: 22_peaks_processing / [setname]_peak_caller_combinations_statistics.tsv

How does it look like? After **22_peaks_processing_script.sh** had executed, the folder: **22_peaks_processing** will contain all these files, just like below:



```
Filename ^
..
161123_gene_ontology
161123.count.matrix.txt
161123.logPvalue.matrix.txt
161123.ratio.matrix.txt
161123.stats.txt
161123_all_peaks_annotated.tsv
161123_all_peaks_calculated.tsv
161123_all_peaks_concatenated.tsv
161123_peak_caller_combinations_statistics.tsv
22_peaks_processing_script.sh
```

```
Filename ^
..
biocyc.txt
biological_process.txt
cellular_component.txt
chromosome.txt
cosmic.txt
gene3d.txt
geneOntology.html
gwas.txt
interactions.txt
interpro.txt
kegg.txt
lipidmaps.txt
molecular_function.txt
msigdb.txt
pathwayInteractionDB.txt
pfam.txt
prints.txt
prosite.txt
reactome.txt
smart.txt
smpdb.txt
wikipathways.txt
```

The last segment of step 17 concatenates all the peak caller combinations peak list files into one file: **[setname]_all_peaks_concatenated.tsv**, followed by annotation in step 18 that generates the file: **[setname]_all_peaks_annotated.tsv**, followed by fold change calculation etc. in step 19 that generates the file: **[setname]_all_peaks_calculated.tsv**. Step 20 reads the resulting file **[setname]_all_peaks_calculated.tsv** and generates a statistics summary file **[setname]_peak_caller_combinations_statistics.tsv**.

While HOMER annotatePeaks is working on annotating our concatenated peak list, it also performs a gene ontology enrichment analysis which generates several tab-separated-values text files as depicted in the right figure. Each of these files contains ranked list of enriched terms coming from specific genome ontology or pathway database. As ChIP-AP will only further append **[setname]_peak_caller_combinations_statistics.tsv** with terms from certain databases (optional; activated by **--goann** and/or **--pathann** flag), not all files generated here will be used further in the pipeline. The files used are **biological_process.txt, molecular_function.txt, cellular_component.txt, interactions.txt, cosmic.txt, kegg.txt, biocyc.txt, pathwayInteractionDB.txt, reactome.txt, smpdb.txt, and wikipathways.txt.**

The following three **matrix.txt** and a **stats.txt** files below are the secondary outputs of HOMER annotatePeaks. These files contain the statistics about the co-occurrence of motif instances (provided with **--motif** flag argument) in the peak sets, which could provide some additional information to you despite not being any use for further processes down the pipeline. These explanations below are taken directly from HOMER documentation - no further explanation is given by HOMER, so we cannot give a clearer explanation:

- **setname.count.matrix.txt** - number of peaks with motif co-occurrence
- **setname.ratio.matrix.txt** - ratio of observed vs. expected co-occurrence
- **setname.logPvalue.matrix.txt** - co-occurrence enrichment
- **setname.stats.txt** - table of pair-wise motif co-occurrence statistics

At this point, the results are actually ready to for your to view and analyze as they already have the essential information typically needed for ChIP-seq analysis. More details are described in the section below: "**Main Pipeline Output - Final Analysis Table"**. Here is a quick summary of what these are and why are they relevant to your analysis.

[setname]_all_peaks_concatenated.tsv already has information pertaining to:

- Peak ID
- Chr
- Start
- End
- Strand
- Peak Caller Combination

So, if you basically only need to know where the peaks are, and which peak caller managed to detect particular peaks, this will suffice. For example: if you want to overlap the list detected peaks with your list of genomic coordinates (e.g., of genome-wide motif instance locations, or genome-wide histone marker locations, or regions of interests obtained from different experiment, or peak list you obtained by using your favorite peak caller etc.).

[setname]_all_peaks_annotated.tsv has these following information in addition to what is already in **[setname]_all_peaks_concatenated.tsv:**

- Annotation
- Detailed Annotation
- Distance to TSS
- Nearest PromoterID
- Entrez ID
- Nearest Unigene
- Nearest Refseq
- Nearest Ensembl

- Gene Name
- Gene Alias
- Gene Description
- Gene Type
- CpG%
- GC%

At this point, the peak list is finally something biologically relevant, as each peak is now appended with the information that can be used to infer role and functionality at cellular or organism level, based on the nearest gene from the peak coordinate. As the protein used in ChIP pulldown experiments are typically transcription factor or histone modifier, a binding event in the close vicinity to a gene suggests regulation of gene expression. For instance, if you analyze this together with RNAseq differentially expressed genes data, then you might find which genes or which pathways your protein of interest is upregulating or downregulating.

As these peaks are now also equipped by the multiple databases' ID of their nearest genes, the user can now connect this peak list with another list which entries are identified by a specific unique ID (e.g., ChIP-AP supplementary annotations relies on individual peak's Entrez ID in order to connect to HOMER genome ontology databases and subsequently add genome ontology and pathway terms into every peak in the list).

[setname]_all_peaks_calculated.tsv has these following information in addition to what is already in **[setname]_all_peaks_annotated.tsv:**

- Peak Caller Overlaps
- ChIP Tag Count
- Control Tag Count
- Fold Change
- Number of Motifs

At this point, you have more power to evaluate and select the peaks you want in your final set. In this file you now have the actual read depth of your peaks, and also the fold change value where you can see the enrichment of reads at the potential binding site compared to the control sample with no pulldown. Along with that, each peak also have the number of known DNA binding motif found in the sample. **Note** that this value will only appear if you provided the **.motif** file using the "--motif" flag argument. These values provided here are the most basic properties of peaks pertaining to their confidence, and are the most standard ways of filtering and ranking of peaks in the list. These values are provided here as an alternative way to apply some thresholds in order to select your peak set for further analysis, in addition to the more powerful main method provided by this pipeline: multiple peak caller overlaps.

At this point, the peak list now has the number of peak caller overlaps, which is the number of peak callers that detected this peak. Although user can already filter in or out their peak list based on the peak caller names provided by the column "Peak Caller Combination" in the file **[setname]_all_peaks_concatenated.tsv**, this value is here to give user a more convenient way of filtering their peaks based on how many peak callers "agree" with a particular detected peak, regardless of which peak callers detected it.

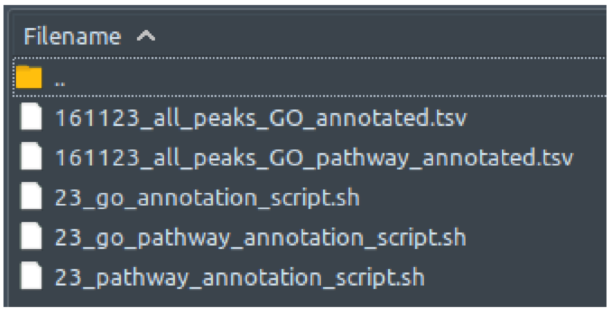
21. **(Optional) Downstream analysis: Gene ontology enrichment**. Each peak in the concatenated list is appended with all the gene ontology terms associated with its gene annotation. The gene ontology terms are derived from biological processes, molecular functions, and cellular compartments databases. This enables list filtering based on the gene ontology terms of the study's interest

Modular script used: **23_go_annotation_script.sh**
➢ **Calls** : go_annotate_suite_xx.py ➢ **Input** : 22_peaks_processing / [setname]_all_peaks_calculated.tsv ➢ **Process** : Append related gene ontology terms to the list of peaks ➢ **Output** : 23_supplementary_annotations / [setname]_all_peaks_go_annotated.tsv

22. **(Optional) Downstream analysis: Pathway enrichment**. Each peak in the concatenated list is appended with all the related biological pathways associated with its gene annotation. The biological pathway terms are derived from KEGG, SMPDB, Biocyc, Reactome, Wikipathways, and pathwayInteractionDB databases. This enables list filtering based on the biological pathways of the study's interest. Additionally, this analysis also adds other terms pertaining to known interactions with common proteins and known gene mutations found in malignant cases, derived from common protein interaction and COSMIC databases, respectively.

Modular script used: **23_pathway_annotation_script.sh**
➢ **Calls** : pathway_annotate_suite_xx.py ➢ **Input** : 22_peaks_processing / [setname]_all_peaks_calculated.tsv ➢ **Process** : Append known pathways and interactions to the list of peaks ➢ **Output** : 23_supplementary_annotations / [setname]_all_peaks_pathway_annotated.tsv

How does it look like? After **23_go_annotation_script.sh** had executed, the folder: **23_supplementary_annotations** will contain all these files, just like below:


Filename ^
..
161123_all_peaks_GO_annotated.tsv
161123_all_peaks_GO_pathway_annotated.tsv
23_go_annotation_script.sh
23_go_pathway_annotation_script.sh
23_pathway_annotation_script.sh

There is no output files difference between single-end and paired-end mode.

If "--goann" flag is used during ChIP-AP call, **23_go_annotation_script.sh** will be executed, and generates **[setname]_all_peaks_go_annotated.tsv** that contains the following gene ontology terms based on each peak's nearest gene:

- Biological Process
- Molecular Function
- Cellular Component

If “--pathann” flag is used during ChIP-AP call, **23_pathway_annotation_script.sh** will be executed, and generates **[setname]_all_peaks_pathway_annotated.tsv** that contains the following known pathways terms based on each peak’s nearest gene:

- Interaction with Common Protein
- Somatic Mutations (COSMIC)
- Pathway (KEGG)
- Pathway (BIOCYC)
- Pathway (pathwayInteractionDB)

If both “--goann” flag and “--pathann” flag are used during ChIP-AP call, both **23_go_annotation_script.sh** and **23_pathway_annotation_script.sh** will be executed, and generates **[setname]_all_peaks_go_pathway_annotated.tsv** that contains all the information that are gained by running the two scripts one after another:

- Biological Process
- Molecular Function
- Cellular Component
- Interaction with Common Protein
- Somatic Mutations (COSMIC)
- Pathway (KEGG)
- Pathway (BIOCYC)
- Pathway (pathwayInteractionDB)

These columns contains **ALL** the related terms from their respective gene ontology or pathway databases, which are comma-separated between terms. That said, depending on how developed the databases are, and how much is known about the gene, the amount of information can range between none to overwhelming.

The information is very useful for filtering the peak list based on the protein of interest’s potential roles in specific biological activities or pathways, based on the interest of the study. This can also be very handy, because contrary to the conventional way of only looking at the gene ontology enrichment analysis result and manually checking back-and-forth if specific genes of interest are actually related to specific terms of interest, we have it already linked to each peak in the list.

However, with respect to users who do not need such information and probably want their list to be much less verbose and smaller in size, this step is completely optional. The output files will not be generated, to save processing time should the user choose to omit this step (by not using the respective flags or not ticking the boxes in the GUI). The scripts will still be generated by ChIP-AP, though, just not executed. You can simply run the script should you change your mind and decide to have these supplementary annotations.

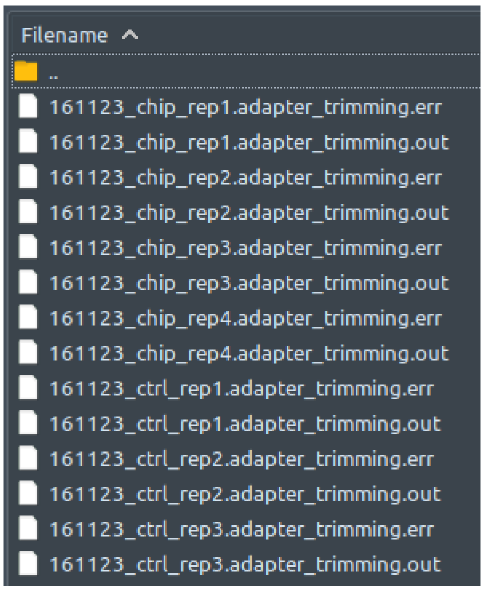
Log Files

For most processes in every modular script, log files are recorded and saved in folder: /logs under their respective directories. There are two types of log files:

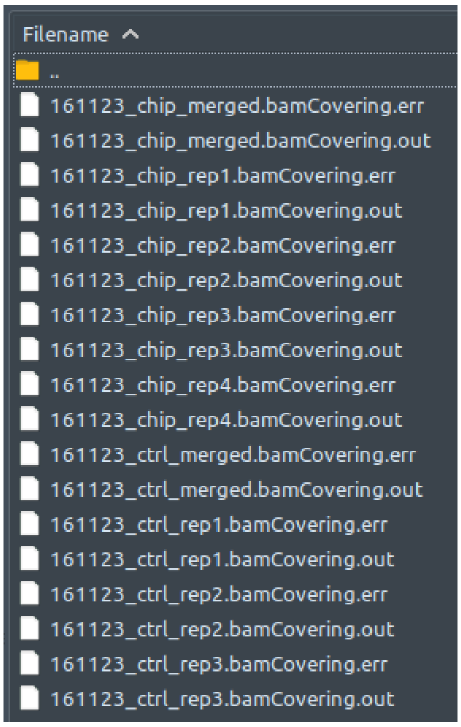
- Log files with **.out** extension: captures whatever the program writes out through channel 1>, a.k.a. the standard output
- Log files with **.err** extension: captures whatever the program writes out through channel 2>, a.k.a. the standard error

Sometimes, unlike what the file or channel name suggests, you might find errors reported in the **.out** files or something like normal program run progress report written in the **.err** files. That is just the way it is. Some programs do not follow the standard output / standard error convention, that's why. So, if your pipeline crashed at a certain process and the **.err** log file does not show anything wrong, the error message might be in the **.out** file instead!

The example contents inside folder /logs can be seen below:



```
Filename ^
..
161123_chip_rep1.adapter_trimming.err
161123_chip_rep1.adapter_trimming.out
161123_chip_rep2.adapter_trimming.err
161123_chip_rep2.adapter_trimming.out
161123_chip_rep3.adapter_trimming.err
161123_chip_rep3.adapter_trimming.out
161123_chip_rep4.adapter_trimming.err
161123_chip_rep4.adapter_trimming.out
161123_ctrl_rep1.adapter_trimming.err
161123_ctrl_rep1.adapter_trimming.out
161123_ctrl_rep2.adapter_trimming.err
161123_ctrl_rep2.adapter_trimming.out
161123_ctrl_rep3.adapter_trimming.err
161123_ctrl_rep3.adapter_trimming.out
```



```
Filename ^
..
161123_chip_merged.bamCovering.err
161123_chip_merged.bamCovering.out
161123_chip_rep1.bamCovering.err
161123_chip_rep1.bamCovering.out
161123_chip_rep2.bamCovering.err
161123_chip_rep2.bamCovering.out
161123_chip_rep3.bamCovering.err
161123_chip_rep3.bamCovering.out
161123_chip_rep4.bamCovering.err
161123_chip_rep4.bamCovering.out
161123_ctrl_merged.bamCovering.err
161123_ctrl_merged.bamCovering.out
161123_ctrl_rep1.bamCovering.err
161123_ctrl_rep1.bamCovering.out
161123_ctrl_rep2.bamCovering.err
161123_ctrl_rep2.bamCovering.out
161123_ctrl_rep3.bamCovering.err
161123_ctrl_rep3.bamCovering.out
```

All these outputs are recorded for your convenience in troubleshooting and error reporting. Do open and read these files to see what's happening with your program. After you spot the potential problem, you can either post the logged error message in our GitHub - issues (https://github.com/JSuryatenggara/ChIP-AP/issues), or go hit Google search if you think the problem is simple and quick enough to figure out yourself.

Main Pipeline Output

Final Analysis Table (including supplementary annotations)

The table below shows the contents of *[filename]_all_peaks_go_pathway_annotated.tsv*. Smaller sized and less verbose variants of this table are saved in the output folder with suffixes: concatenated, annotated, and calculated (see Source in the table below)

Column #	Peak Attribute	Source
Column 1 (A)	Peak ID	Pipeline script: 22_peaks_ processing _script.sh Called program: cat (Bash) Output file: [filename]_all_peaks_concatenated.tsv Output folder: 22_peaks_processing
Column 2 (B)	Chr	
Column 3 (C)	Start	
Column 4 (D)	End	
Column 5 (E)	Strand	
Column 6 (F)	Peak Caller Combination	
Column 7 (G)	Peak Caller Overlaps	Pipeline script: 22_peaks_processing_script.sh Called script: fold_change_calculator_suite_xx.py Output file: [filename]_all_peaks_calculated.tsv Output folder: 22_peaks_processing
Column 8 (H)	ChIP Tag Count	
Column 9 (I)	Control Tag Count	
Column 10 (J)	Fold Change	
Column 11 (K)	Number of Motifs	
Column 12 (L)	Annotation	Pipeline script: 22_peaks_processing_script.sh Called program: HOMER annotatePeaks Output file: [filename]_all_peaks_annotated.tsv Output folder: 22_peaks_processing
Column 13 (M)	Detailed Annotation	
Column 14 (N)	Distance to TSS	
Column 15 (O)	Nearest PromoterID	
Column 16 (P)	Entrez ID	
Column 17 (Q)	Nearest Unigene	
Column 18 (R)	Nearest Refseq	
Column 19 (S)	Nearest Ensembl	
Column 20 (T)	Gene Name	
Column 21 (U)	Gene Alias	
Column 22 (V)	Gene Description	
Column 23 (W)	Gene Type	
Column 24 (X)	CpG%	
Column 25 (Y)	GC%	
Column 26 (Z)	Biological Process	Pipeline script: 23_go_annotation_script.sh Called script: GO_annotate_suite_xx.py Output file: [filename]_all_peaks_go_annotated.tsv Output folder: 23_supplementary_annotations
Column 27 (AA)	Molecular Function	
Column 28 (AB)	Cellular Component	
Column 29 (AC)	Interaction with Common Protein	Pipeline script: 23_pathway_annotation_script.sh Called script: pathway_annotate_suite_xx.py Output file: [filename]_all_peaks_pathway_annotated.tsv Output folder: 23_supplementary_annotations
Column 30 (AD)	Somatic Mutations (COSMIC)	
Column 31 (AE)	Pathway (KEGG)	
Column 32 (AF)	Pathway (BIOCYC)	
Column 33 (AG)	Pathway (pathwayInteractionDB)	
Column 34 (AH)	Pathway (REACTOME)	
Column 35 (AI)	Pathway (SMPDB)	
Column 36 (AJ)	Pathway (Wikipathways)	

Peak Attribute	**Description**
Peak ID	Given unique peak ID
Chr	Chromosome where the peak is located
Start	Starting coordinate of the peak region
End	End coordinate of the peak region
Strand	DNA strand (positive/negative) where the peak is located
Peak Caller Combination	Name of peak callers that detected this peak
Peak Caller Overlaps	Number of peak callers that detected this peak
ChIP Tag Count	ChIP Read depth at weighted peak center (narrow peak) **OR** average read depth of the whole peak region (broad peak)
Control Tag Count	Control read depth at weighted peak center (narrow peak) **OR** average read depth of the whole peak region (broad peak)
Fold Change	ChIP vs control fold change in read depth
Number of Motifs	Number of motif instances found in the peak region
Annotation	Type of annotated region (Exon, Intron, Promoter-TSS)
Detailed Annotation	Detailed, longer version of Annotation in previous column
Distance to TSS	Distance from the peak to the nearest annotated Transcription Start Site (TSS) in base pairs
Nearest PromoterID	PromoterID of the nearest annotated promoter region
Entrez ID	Entrez ID of the nearest annotated gene
Nearest Unigene	Unigene ID of the nearest annotated gene
Nearest Refseq	Refseq ID of the nearest annotated gene
Nearest Ensembl	Ensembl ID of the nearest annotated gene
Gene Name	Name of the nearest annotated gene (abbreviated)
Gene Alias	Alternate names of the nearest annotated gene
Gene Description	Name of the nearest annotated gene (full)
Gene Type	e.g. protein-coding / non-coding / pseudogene / etc
CpG%	The proportion of known methylation spots within this peak region
GC%	The proportion of GC contents within this peak region
Biological Process	Terms from “biological process” GO that are related to the nearest gene
Molecular Function	Terms from “molecular function” GO that are related to the nearest gene
Cellular Component	Terms from “cellular component” GO that are related to the nearest gene
Interaction with Common Protein	Known interactions between common proteins and the expressed proteins of the nearest gene
Somatic Mutations (COSMIC)	Known cancer cases where mutations are found in the nearest gene
Pathway (KEGG)	Known pathways that are related to the nearest gene according to the KEGG database
Pathway (BIOCYC)	Known pathways that are related to the nearest gene according to the Biocyc database
Pathway (pathwayInteractionDB)	Known pathways that are related to the nearest gene according to the pathwayInteractionDB database
Pathway (REACTOME)	Known pathways that are related to the nearest gene according to the REACTOME database
Pathway (SMPDB)	Known pathways that are related to the nearest gene according to the SMPDB database
Pathway (Wikipathways)	Known pathways that are related to the nearest gene according to the Wikipathways database

Miscellaneous Pipeline Outputs

Multiple peak callers statistics summary

The table below shows the contents of *[filename]_peak_caller_combinations_statistics.tsv*.

Column 1 (A)	**Peak Callers Combination**
Column 2 (B)	Exclusive Peak Count
Column 3 (C)	Exclusive Positive Peak Count
Column 4 (D)	Exclusive Motif Count
Column 5 (E)	Exclusive Positive Peak Hit Rate
Column 6 (F)	Exclusive Motif Hit Rate
Column 7 (G)	Exclusive ChIP Peak Read Depth
Column 8 (H)	Exclusive ChIP Peak Fold Change
Column 9 (I)	Inclusive Peak Count
Column 10 (J)	Inclusive Positive Peak Count
Column 11 (K)	Inclusive Motif Count
Column 12 (L)	Inclusive Positive Peak Hit Rate
Column 13 (M)	Inclusive Motif Hit Rate
Column 14 (N)	Inclusive ChIP Peak Read Depth
Column 15 (O)	Inclusive ChIP Peak Fold Change

Columns	Description
Peak Callers Combination	Name of peak callers that detected this peaks subset
Peak Count	Number of peaks
Positive Peak Count	Number of peaks containing known binding motif (--motif)
Motif Count	Total number of binding motif instances
Positive Peak Hit Rate	Positive Peak Count divided by Peak Count
Motif Hit Rate	Motif Count divided by Peak Count
ChIP Peak Read Depth	ChIP Read depth at weighted peak center (narrow peak) OR average read depth of the whole peak region (broad peak)
ChIP Peak Fold Change	ChIP vs control fold change in read depth

- Exclusive: Only counts for a specific peak caller combination (e.g., Exclusive peak count of MACS2 only counts for peaks that is exclusively called by MACS2 alone).
- Inclusive: Counts for other peak caller combinations containing the same peak callers (e.g., Inclusive peak count of MACS2|GEM also counts for all other peaks in MACS2|GEM|HOMER, MACS2|GEM|Genrich, and MACS2|GEM|HOMER|Genrich).

Pipeline Run Info

This file summarizes the assignment of the files (IP sample or control, read 1 or 2; replicate number) and the file name conversion for every unaligned or aligned sequencing reads to be processed. Each line tells the user what the original files have been renamed into. Check this file if you suspect the samples were incorrectly entered (i.e., swapped chip with control)

```
Chromatin IP dataset replicate 1, 1st read : Original filename = a.fastq --> New filename = setname_chip_rep1_R1.fq.gz
Chromatin IP dataset replicate 2, 1st read : Original filename = b.fastq --> New filename = setname _chip_rep2_R1.fq.gz
Chromatin IP dataset replicate 1, 2nd read : Original filename = c.fastq --> New filename = setname _chip_rep1_R2.fq.gz
Chromatin IP dataset replicate 2, 2nd read : Original filename = d.fastq --> New filename = setname _chip_rep2_R2.fq.gz
Control dataset replicate 1, 1st read : Original filename = e.fastq --> New filename = setname _ctrl_rep1_R1.fq.gz
Control dataset replicate 2, 1st read : Original filename = f.fastq --> New filename = setname _ctrl_rep2_R1.fq.gz
Control dataset replicate 1, 2nd read : Original filename = g.fastq --> New filename = setname _ctrl_rep1_R2.fq.gz
Control dataset replicate 2, 2nd read : Original filename = h.fastq --> New filename = setname _ctrl_rep2_R2.fq.gz
```

Pipeline Run Command

Contains the input command line that was used to call the pipeline in a text file: *[filename]_command_line.txt* in the output save folder. This is useful for documentation of the run, and for re-running of the pipeline after a run failure or some tweaking if need be.

```
[chipap directory]/chipap_xx.py --mode paired --ref [genome_build] --genome [path_to_computed_genome_folders]
--output [full_path_to_output_save_folder] --setname [dataset name] --sample_table [path_to_sample_table_file]
--custom_setting_table [path_to_setting_table_file].tsv --motif [path_to_known_motif_file]
--fcmerge --goann --pathann --deltemp --thread [#_of_threads_to_use] --run
```

Sample Table

Contains the full path of each input ChIP and control sample in the pipeline run in a tab-separated value file: *[filename]_sample_table.tsv* in the output save folder in ChIP-AP sample table format. This is useful for documentation of the run, and for re-running of the pipeline after a run failure or some tweaking if need be. Below is an example of sample table file content (header included), given paired-end samples with two ChIP and two control replicates.

chip_read_1	**chip_read_2**	**ctrl_read_1**	**ctrl_read_2**
... /a.fastq	... /c.fastq	... /e.fastq	... /g.fastq
... /b.fastq	... /d.fastq	... /f.fastq	... /h.fastq

If your sample is single-ended, then the sample table can simply be formatted as follows.

chip_read_1	**ctrl_read_1**
... /a.fastq	... /e.fastq
... /b.fastq	... /f.fastq

Setting Table & Default Parameters

A *cornerstone* of ChIP-AP's functionality is the settings table. ChIP-AP, with the raw fq files and the settings table, is able to reproduce any analysis (provided the same program version numbers are used). The provision of the settings table ensures reproducibility of any analysis with minimal effort and bypasses the usually sparse and significantly under-detailed methods sections of publications. Science is supposed to be reproducible, yet bioinformatics analysis are typically black-boxes which are irreproducible. This 1 file, changes that!

The structure of the settings table is simple. It is a 2-column tab-separated value file with the names of the programs on the 1st column, and the necessary flags required or changed in the 2nd column. If making your own custom table, then the 1st column below must be copied as-is and not changed. These 2 columns together, list the flags and argument values for each program used in the pipeline.

When ChIP-AP is run, a copy of the used settings table is saved as a tab-separated value file: *[filename]_ setting_table.tsv* in the output save folder. If you have a custom settings table made and provided it as input, then ChIP-AP will make a 2nd copy of this table in the same output save folder. This decision is made as it is useful documentation of the run performed. This file is also useful for re-running of the pipeline after run failure or some tweaking if necessary. If submitting an issue request on GitHub, you ***must*** provide us your settings table used as well as all other requested information. See GitHub for details regarding this.

We consider the dissemination of the information of this file as vital and essential along with results obtained. The table can be included as a supplemental table in a manuscript or can be included as a processed data file when submitting data to GEO – either way, the information of this file must be presented when publishing data.

Below is an example of setting table file in its default-setting state:

fastqc1	-q
clumpify	dedupe spany addcount qout=33 fixjunk
bbduk	ktrim=l hdist=2
trimmomatic	LEADING:20 SLIDINGWINDOW:4:20 TRAILING:20 MINLEN:20
fastqc2	-q
bwa_mem	
samtools_view	-q 20
plotfingerprint	
fastqc3	-q
macs2_callpeak	
gem	-Xmx10G --k_min 8 --k_max 12
sicer2	
homer_findPeaks	
genrich	--adjustp -v
homer_mergePeaks	
homer_annotatePeaks	
fold_change_calculator	--normfactor uniquely_mapped

Interpreting ChIP-AP Output

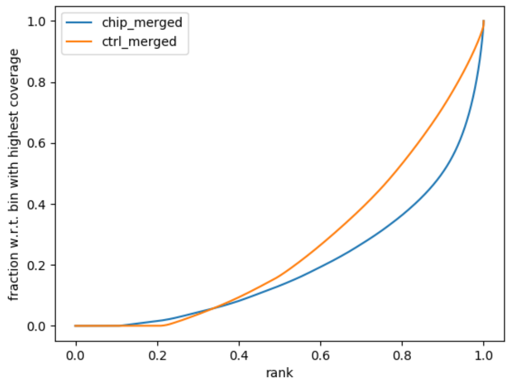
Ok so ChIP-AP does report a fair amount of stuff. If you ran it locally you have a swath of folders and you have nooooo clue what to look for and it's all confusing. We get that. The reality though it's very simple to know what to look for to know your experimental run worked and in this section were going to walk you through that!

Did my analysis work?

There are a couple of things to look for to answer this question. 1, the fingerprint plot and 2, the Venn diagram of the merged peaks. Let's begin…

1 – The fingerprint plot

The fingerprint plot tells us how well the enrichment of your samples worked. It is generated by the function from the deeptools package and is generated after the alignment files. As such, the plots are found in the *"08_results"* folder and are labelled "fingerpring_xxxxx.png/svg." The PNG files allow you to view them in any image viewer, the SVG files are for opening in Adobe Illustrator or Inkscape to make HQ publication figures later if you need.

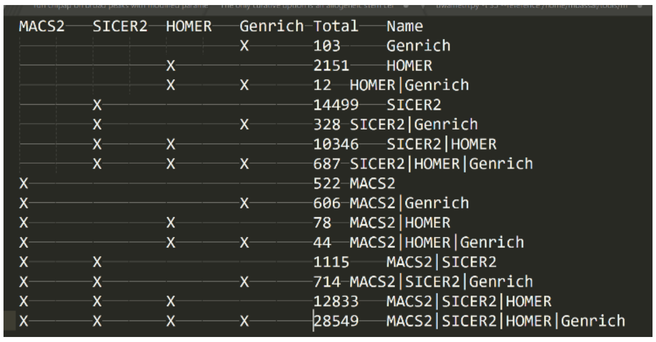


To interpret the fingerprint plot, (more information can be found on the deeptools documentation site), but put simply (image to the right), the input control should be a diagonal line as close as possible toward the 1:1 diagonal. Your ChIP sample should have a bend/kink towards the bottom right corner. The greater the separation between the input and the chip sample, the greater the enrichment you will see in the final result (i.e., lots of peaks). If the lines are overlapping, then you will see little enrichment and your experiment didn't work that well. If you're sample lines are switched – then you probably switched the sample names and we recommend doing the right thing and repeating the experiment and not simply switch sample names for the sake of a publication.

In this example, there is reasonable enrichment in our chip samples. And so, we are confident we can see enrichment.

2 – The Venn Diagram (well Venn Text)

In the folder *"21_peaks_merging"* folder, you will find the *"venn.txt"* file. This will show you a textual Venn diagram of the overlap between the called peaks across all peak callers. To know your experiment worked well, you should see a full list with combinations of all peak callers and relatively large numbers for the consensus peak sets (i.e., peaks called by multiple peak callers) – this is the ideal case. However, from our experience, there will almost always be 1 maybe 2 peak callers that don't like a dataset for some reason and so you may find a peak caller performed poorly but the others performed admirably. This is still a good and valid result. If you look at this file and only see small number of peaks and little overlap, and only 1 peak caller seems to have



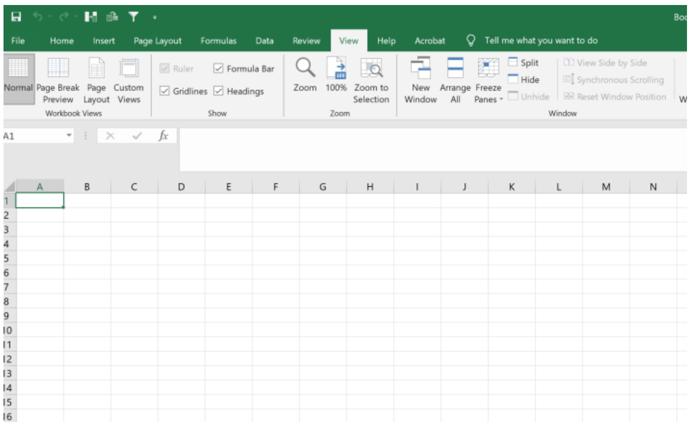
```
MACS2   SICER2  HOMER   Genrich Total   Name
                        X       103     Genrich
                HOMER
                X               2151    HOMER
                X       X       12  HOMER|Genrich
        X                       14499   SICER2
        X               X       328 SICER2|Genrich
        X       X               10346   SICER2|HOMER
        X       X       X       687 SICER2|HOMER|Genrich
X                               522 MACS2
X                       X       606 MACS2|Genrich
X               X               78  MACS2|HOMER
X               X       X       44  MACS2|HOMER|Genrich
X       X                       1115    MACS2|SICER2
X       X               X       714 MACS2|SICER2|Genrich
X       X       X               12833   MACS2|SICER2|HOMER
X       X       X       X       28549   MACS2|SICER2|HOMER|Genrich
```

dominated peak calling, then likely your experiment didn't work that great. Just because only 1 peak caller performed well though, doesn't mean the experiment is a write-off and a failure. It can still be valid and so doing some manual validations on the top fold-change differential peaks by chip-PCR might give you an indication whether there is salvageable data or not. Also, if you have other confirmatory experimental evidence then even 1 peak caller getting results is fine. This is why we implemented multiple peak callers because there are many instances where the signal:noise just creates a mess for most peak callers but generally 1 will be the super-hero of the day in such a situation.

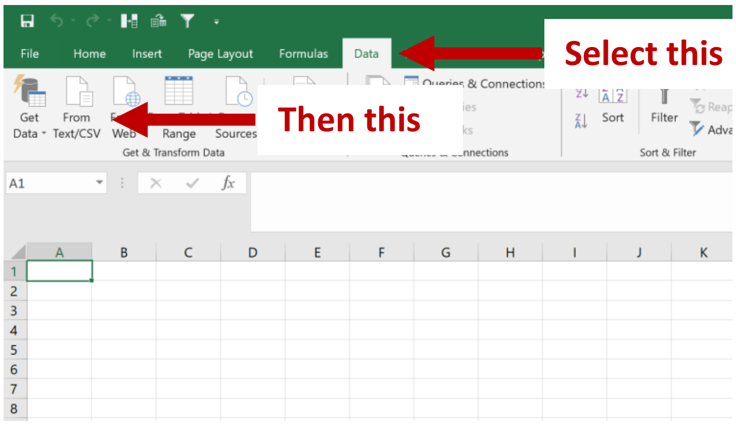
3 – What results files do I look at exactly?

Valid question. In the folder "22_peak_processing," open the "xxxx_all_peaks_calculated.tsv" file in excel and you're good to go. Now to open it there is a little step to do…

Open a new blank workbook in excel



In the ribbon at the top, go to "Data", then select "From Text/CSV"



In the dialog box that opens up, find and open the peaks files "xxxx_all_peaks_calculated.tsv." Follow all the prompts and keep pressing "Next" / "Proceed" till the end and the file opens. Opening the peak file this way circumvents an issue that Excel constantly makes which is it will interpret some gene names such as OCT1 as a date, when it's not. So, by following the afore mentioned steps, excel will not do this stupid conversion and instead, when you save the file as an xlsx, it will ensure that this issue doesn't happen (seen it in sooooo many publications it's not funny – just import data this way please people?)

From this file, you can view all the results and data for you analysis. Refer to Interpreting ChIP-AP Output for the definition of what each column means.

4 – How do I view my alignments and data?

People typically want to view their results on UCSC or other genome browsers. As we don't have a web-server to host such coverage files (and making accessible UCSC hub is a real pain and we don't want to implement that), the onus is on you to view them locally on your machine. All laptops, whether then can run ChIP-AP or not can run IGV Downloads | Integrative Genomics Viewer (broadinstitute.org) and view the coverage and bam files. The coverage and bam files can be located in the *"08_results"* folder.

Download IGV, install it (super easy) and then load the coverage and bam files needed. Make sure you load the right genome build however! That's critical. From section Main Pipeline Output, you can copy columns B,C,D straight into IGV and it will take you to the peak section.

5 – In Short, what's relevant?

Easy answers

1 – check fingerprint plot and make sure it looks good

2 – check venn.txt file and make sure you get good spread of peaks

Together points 1 and 2 tell you your experiment worked!

3 – Your final peak file is in *"22_peak_processing"* open the *"xxxx_all_peaks_calculated.tsv"* – This is the file you need to upload to GEO as your processed data file for your analysis and the only file you need to work with when looking through your data.

4 – Also as part of your submission to GEO or as a supplemental table in your manuscript, you MUST include the settings table named *"default_settings_table.txt"* located in the root analysis directory. This provided with the raw fq files, which must be uploaded to GEO, will ensure complete reproducibility of the analysis performed.

5 – Manuscript details for M&M. A statement such as the following should suffice.

For processing our ChIP-Seq analysis, we utilized ChIP-AP (REF). Raw fq files are uploaded to GEO with accession number XXXXX, and the custom settings table utilized for analysis can be found on GEO as a processed settings file and also in supplemental table XX in our manuscript. Full details of ChIP-AP and its function can be found in its corresponding manuscript (REF).

References and Citations

If you use ChIP-AP in your analysis, please cite the us and all the following programs

Programs	References
ChIP-AP v4.1	Guide: https://github.com/JSuryatenggara/ChIP-AP/wiki/ChIP-AP-Guide GitHub: https://github.com/JSuryatenggara/ChIP-AP Citation: (coming soon…)
Python3 Linux 3.7.x/3.8.x macOS 3.7.x	We have noted in our testing that there is a change in python 3.8 on macOS in how multi-threading is handled which breaks ChIP-AP. As such, for macOS installs you must ensure that ptyhon3.7.x is installed. If using our installation guides, the provided yml files will ensure all the correct dependencies and requirements are met automatically.
FastQC v0.11.9	Guide: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ GitHub: https://github.com/s-andrews/FastQC
Clumpify v38.18 (BBmap)	Introduction: https://www.biostars.org/p/225338/ Guide: https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/clumpify-guide/ GitHub: https://github.com/BioInfoTools/BBMap/blob/master/sh/clumpify.sh Citation: https://www.osti.gov/biblio/1241166-bbmap-fast-accurate-splice-aware-aligner
BBDuk v38.18 (BBmap)	Introduction: http://seqanswers.com/forums/showthread.php?t=42776 Guide: https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/ GitHub: https://github.com/BioInfoTools/BBMap/blob/master/sh/bbduk.sh Citation: https://www.osti.gov/biblio/1241166-bbmap-fast-accurate-splice-aware-aligner
Trimmomatic v0.39	Guide: http://www.usadellab.org/cms/?page=trimmomatic Downloadable manual page: http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf GitHub: https://github.com/timflutre/trimmomatic Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4103590/
bwa v0.7.17	Guide: http://bio-bwa.sourceforge.net/bwa.shtml GitHub: https://github.com/lh3/bwa Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2705234/
samtools view v1.9 (samtools)	Guide: http://www.htslib.org/doc/samtools-view.html GitHub: https://github.com/samtools/samtools Citation: https://pubmed.ncbi.nlm.nih.gov/19505943/
deeptools plotFingerprint v3.5.0 (deepTools)	Guide: https://deeptools.readthedocs.io/en/develop/content/tools/plotFingerprint.html Citation: https://academic.oup.com/nar/article/44/W1/W160/2499308?login=true
MACS2 v2.2.6	Guide: https://hbctraining.github.io/Intro-to-ChIPseq/lessons/05_peak_calling_macs.html Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2732366/ GitHub: https://github.com/macs3-project/MACS/wiki
GEM v2.7	Guide: https://groups.csail.mit.edu/cgs/gem/ GitHub: https://github.com/gifford-lab/GEM Citation: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002638
SICER2 v1.0.2	Guide: https://zanglab.github.io/SICER2/ GitHub: https://github.com/bioinf/SICER2 Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2732366/

HOMER findPeaks v4.11 (HOMER)	Guide: http://homer.ucsd.edu/homer/ngs/peaks.html Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2898526/
Genrich v0.6	Guide: https://informatics.fas.harvard.edu/atac-seq-guidelines.html GitHub: https://github.com/jsh58/Genrich
Homer mergePeaks v4.11 (HOMER)	Guide: http://homer.ucsd.edu/homer/ngs/mergePeaks.html Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2898526/
HOMER annotatePeaks v4.11 (HOMER)	Guide: http://homer.ucsd.edu/homer/ngs/annotation.html Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2898526/