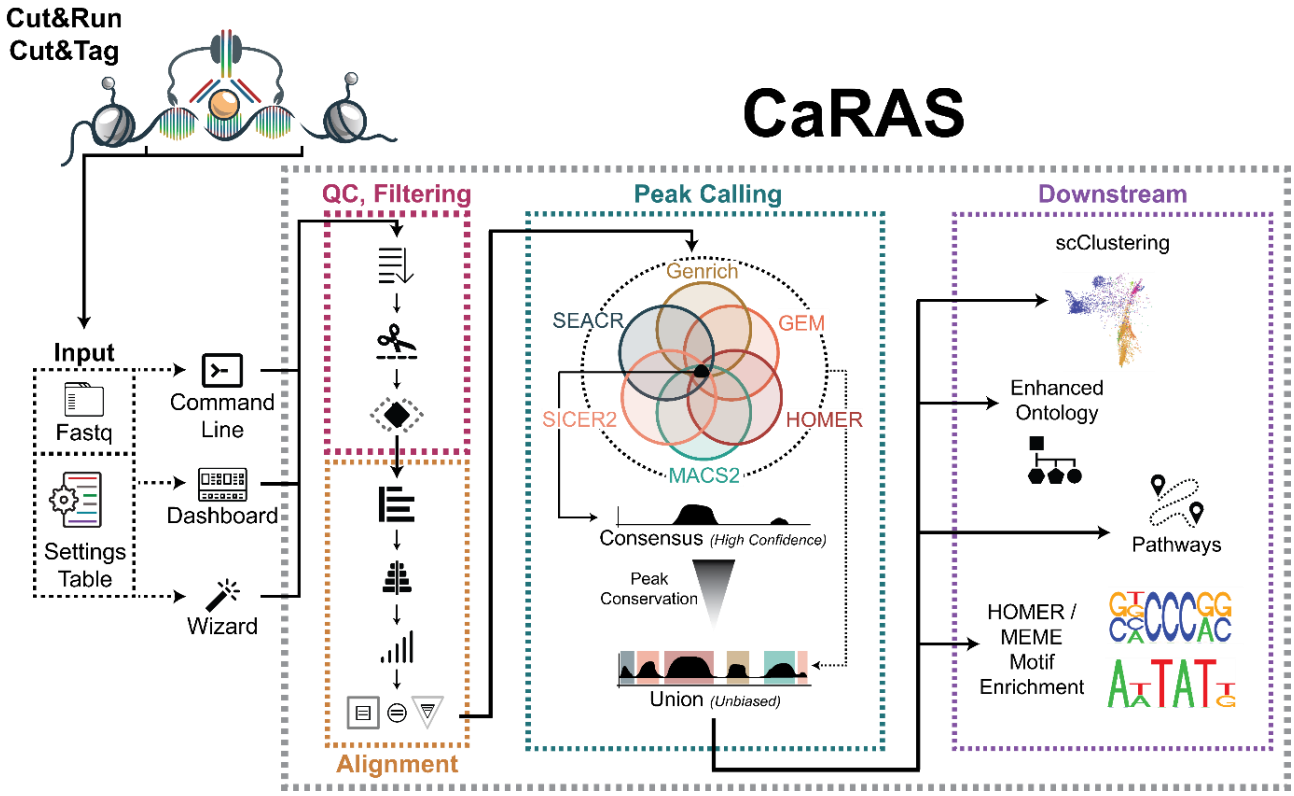
CaRAS Graphical Overview



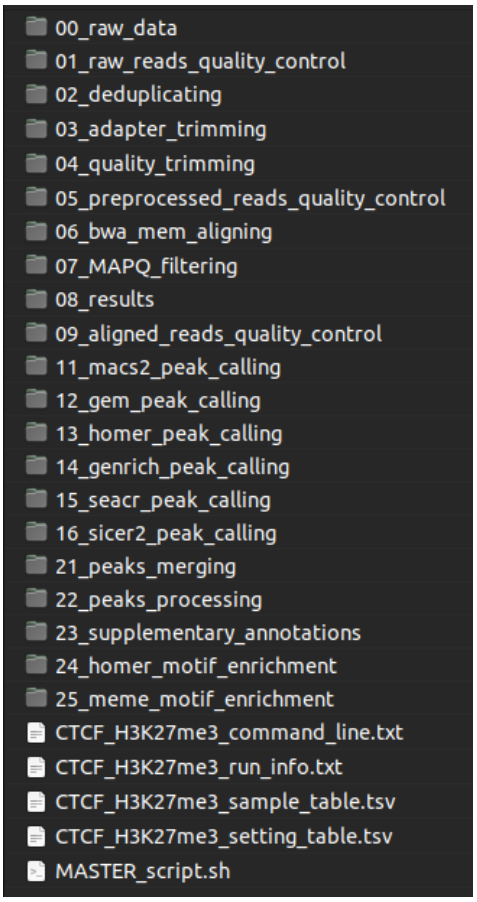
Detailed Explanation of Steps and Methodology Used

1. **Acquisition of raw sequencing files**. CaRAS can directly process the output files of sequencing instruments. Files may be in FASTQ, or compressed FASTQ (.gz) format. CaRAS can also process aligned reads in BAM format. Reads may be single or paired ends. Background control is recommended but ***not*** compulsory for CaRAS.

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 file (in case of paired end sequencing data). Afterwards, their formats and compression status is recognized and processed into gun-zipped FASTQ as necessary.

3. **Generation of multiple modular scripts**. Each process in CaRAS is executed from individual scripts generated. This was an intentional design decision as it allows for easy access for modifications of any step within the pipeline without having to drudge through the trenches of someone else's wall of codes.

   How does it look like? After this step is done, which is practically the end of your CaRAS 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 based on your --setname:

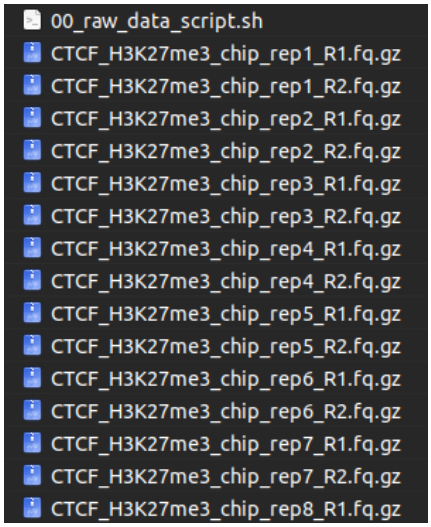


   Each of these folders are basically empty, and contains a script which is named based on the folder name (e.g., script **02_deduplicating.sh** inside folder **02_deduplicating**). Each of these scripts will be executed in numerical sequence when you run the pipeline. Aside from these scripts, there will 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 to sequentially run all the scripts within the aforementioned folders.

4. **Copying, compressing, and renaming of the raw sequencing reads**. In the very beginning, CaRAS makes (in the user-designated output folder) a copy of each unaligned sequence reads file (e.g., fastq), 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, every single reads file in your dataset will be 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.

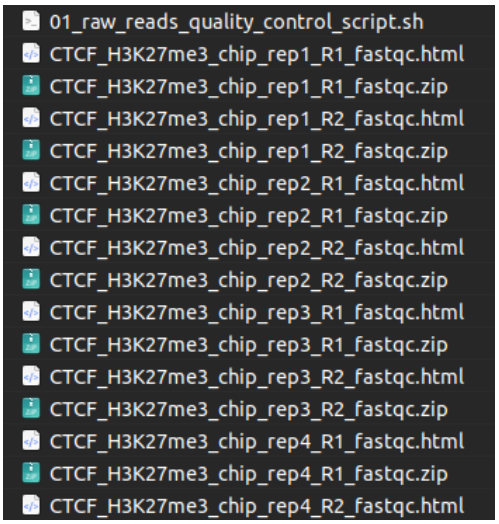


   In case of paired-end dataset, every sample has of two files: first read (R1) second read (R2), as opposed to only one in single-end dataset. All these **fq.gz** files will be immediately deleted at the end of 02_deduplicating_script.sh execution if --deltemp flag is used on CaRAS call. We won't explain how to open and read these **fq.gz** files, since if you need us to tell you that, you most probably will not be able to make anything out of the contents in there 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, its worthwhile to go through the multiple QC steps and track the quality of the data as its processed. 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 faily adequate job though for most datsets.

   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:

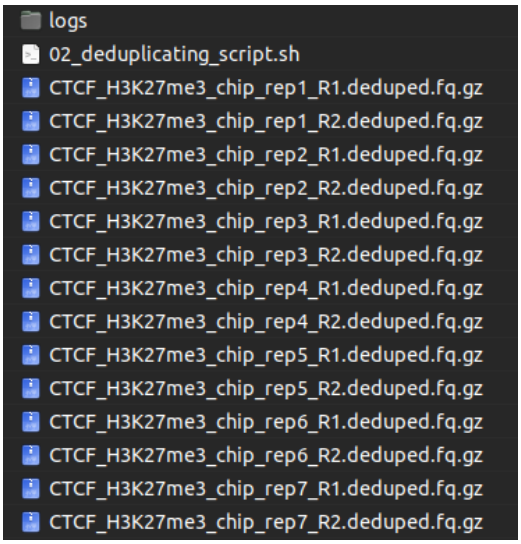


   In case of paired-end dataset, every sample has of two files: first read (R1) second read (R2), as opposed to only one in single-end dataset. The **.zip** files contains the individual components to be compiled for the report so you can ignore those. To read the reports, open the **.html** files. This file is a multitabular file in which you can evaluate the quality of your experiment and sequencing through the quality or your raw files. Comprehensive as it is, explaining the contents in detail would take a whole new guide by itself. Therefore, in case the reports do not spell everything out enough for you, check this out: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/.

6. **Deduplication of reads**. Performed by clumpify from BBMap package. Necessary command line argument is given to clumpify in order to remove optical duplicates and tile-edge duplicates from the reads file in addition to PCR duplicates. Optimization of file compression is also performed by clumpify during deduplication process, in order to minimize storage space and speed up reads file processing.

   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:

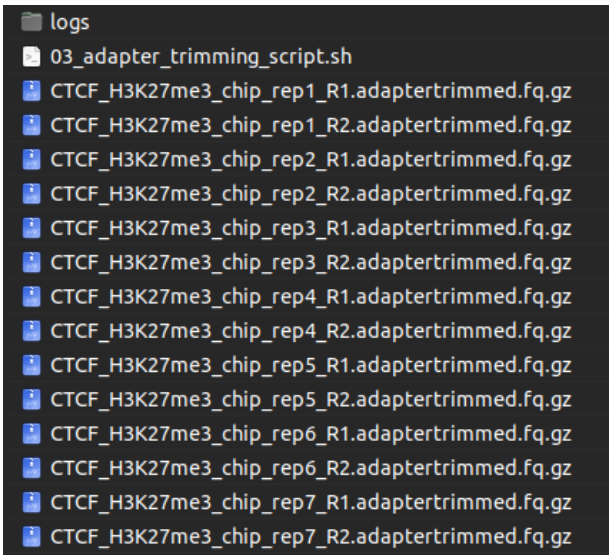


   In case of paired-end dataset, every sample has of two files: first read (R1) second read (R2), as opposed to only one in single-end dataset. There should be one deduplicated file for each processed raw reads file from folder **00_raw_data**. Paired-end files are processed in pairs by **clumpify**. All these **deduped.fq.gz** files will be deleted at the end of **02_deduplicating_script.sh** execution if --deltemp flag is used on CaRAS call.

7. **Adapter trimming of reads**. Performed by BBDuk from BBMap package. BBDuk scans every read for adapter sequence, based on the reference list adapters given in the command line argument. The standard BBDuk adapter sequence reference list 'adapter.fa' is be used as a default in the pipeline. Any sequencing adapter present in the reads is removed. Custom adapter sequence can be used whenever necessary or by modifying the adapter.fa file with your new sequences.

   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 CaRAS)*
   - **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:

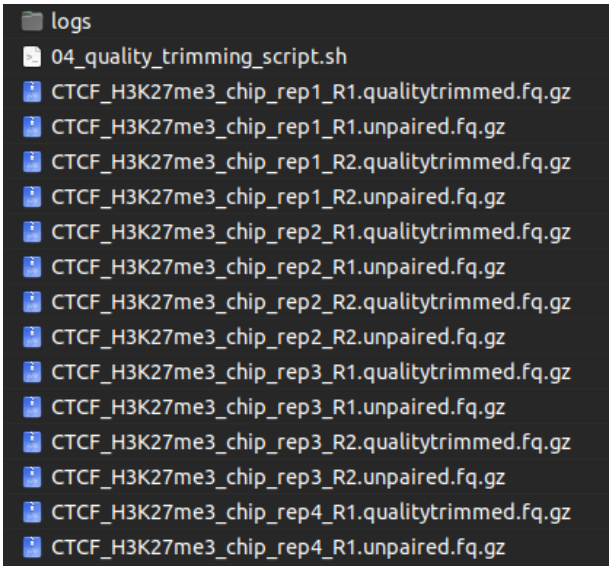


   In case of paired-end dataset, every sample has of two files: first read (R1) second read (R2), as opposed to only one in single-end dataset. There should be one adapter-trimmed file for each processed deduplicated reads file from folder **02_deduplicating**. Paired-end files are processed in pairs by **bbduk**. All these **adaptertrimmed.fq.gz** files will be deleted at the end of **03_adapter_trimming_script.sh** execution if --deltemp flag is used on CaRAS call.

8. **Quality trimming of reads**. Performed by trimmomatic. Trimmomatic scans every read trims low quality base calls from 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.

   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:



   In case of paired-end dataset, every sample has of two files: first read (R1) second read (R2), as opposed to only one in single-end dataset. 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 CaRAS 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:

In case of paired-end dataset, every sample has of two files: first read (R1) second read (R2), as opposed to only one in single-end dataset. 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. This file is a multitabular file in which you can evaluate how your preprocessings: deduplication, adapter trimming, and quality trimming, affected your sequencing reads. Comprehensive as it is, explaining the contents in detail would take a whole new guide by itself. Therefore, in case the reports do not spell everything out enough for you: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/.

10. **Reads alignment to target organism genome**. Performed by the mem algorithm in BWA aligner. Appropriate genome reference for the sample organism is given as a command line argument. The default genome reference is hg38. Precomputed genome references hg38, hg19, mm9, mm10, mm39, dm6, and sacCer3 are downloaded as part of the CaRAS installation process.

    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 CaRAS)*
    - **Process** : Align preprocessed reads to the designated reference genome
    - **Output** : 06_bwa_mem_aligning / [setname]_[chip/ctrl]_rep[#].aligned.bam

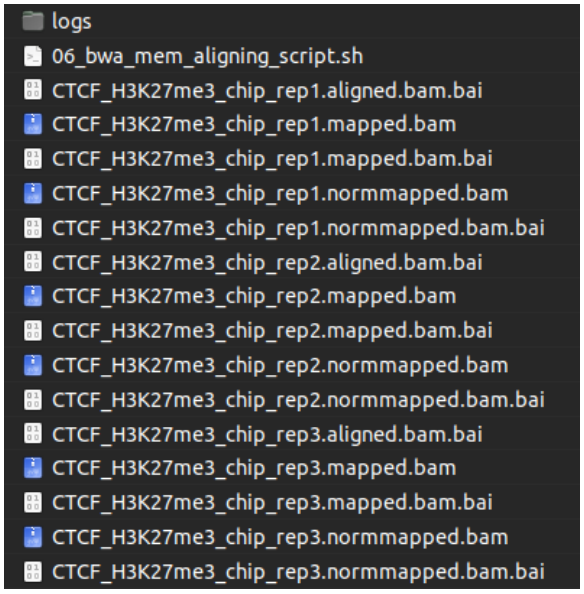
    How does it look like? *Detailed output preview of step 10 and 11 are combined below step 11*

11. **Reads alignment to read normalization organism genome**. Performed by the mem algorithm in BWA aligner. Appropriate genome reference for the spiked-in or carry-over DNA source organism is given as a command line argument. CaRAS does not perform read normalization as the correct read normalization organism genome has to be chosen carefully by users based on the CUT&RUN assay. Precomputed genome references sacCer3 and eColiK12 are downloaded as part of the CaRAS installation process.

    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 CaRAS)*
    - **Process** : Align preprocessed reads to designated read normalization organism genome
    - **Output** : 06_bwa_mem_aligning / [setname]_[chip/ctrl]_rep[#].normmapped.bam

How does it look like? After **06_bwa_mem_aligning_script.sh** had been executed, the folder: **06_bwa_mem_aligning** will contain files which are the reads aligned to the reference organism genome (marked by the extension: **.aligned.bam**), and to the normalization organism genome (marked by the extension: **.normapped.bam***), just like below:

* Only when CaRAS is instructed to normalize the aligned reads



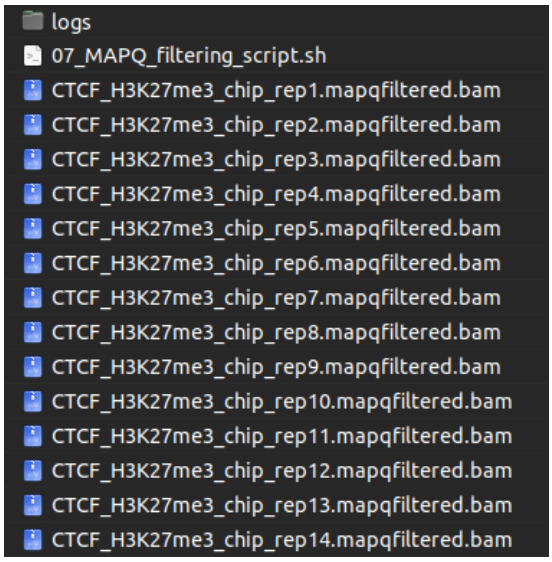
In case of paired-end sequencing, first reads (R1) and second reads (R2), which are in separated files prior to this alignment step, are now aligned into the same reference genome (and normalization organism genome*), and thus no longer separated in two different files. Paired-end files are processed in pairs by **bwa mem**. There should be one aligned reads file (**.aligned.bam**), or two* aligned reads files (**.aligned.bam** and **.normapped.bam***), for each processed single-end, or for every two processed paired-end quality-trimmed reads file from folder **04_quality_trimming**. For every **.bam** file, an index file (**.bam.bai**) is generated to assist with subsequent processes downstream. These files will be deleted at the end of **06_bwa_mem_aligning_script.sh** execution if --deltemp flag is used on CaRAS call.

* Only when CaRAS is instructed to normalize the aligned reads

12. **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.

    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:



    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 CaRAS call.

13. **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

    - **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*

    - **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

    - **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 13, 14, 15, 16, 17 are combined below step 17*

14. **Generation of aligned reads bedgraph files**. Performed by bamCoverage from the deeptools package. Generates bedgraph files to serve as inputs for peak calling by SEACR. The generated bedgraph files are normalized based on the number of reads successfully aligned to the read normalization organism genome.

    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 bedgraph coverage file for each individual bam file
    - **Output** : 08_results / [setname]_[chip/ctrl]_rep[#].bdg
      08_results / [setname]_[chip/ctrl]_merged.bdg

    How does it look like? *Detailed output preview of step 13, 14, 15, 16, 17 are combined below step 17*

15. **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, however a track hub needs to be generated – something CaRAS does not do at this stage.

    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? *Detailed output preview of step 13, 14, 15, 16, 17 are combined below step 17*

16. **Normalization of aligned reads files**. Performed by a custom script, with the help of samtools view and merge modules. Normalizes all aligned read files (.bam) by multiplying every single aligned read with each respective sample's scaling factor, determined based on the number of reads in that respective sample which are successfully aligned to the read normalization organism genome. Along with this, the same custom script also accordingly normalizes the bedgraph (**.bdg**) and BigWig (**.bw**) files of the same sample.

    Modular script used: **08_results_script.sh**
    - **Calls** : bamCoverage
    - **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].bam
      08_results / [setname]_[chip/ctrl]_merged.bam
      08_results / [setname]_[chip/ctrl]_rep[#].bdg
      08_results / [setname]_[chip/ctrl]_merged.bdg
      08_results / [setname]_[chip/ctrl]_rep[#].bw
      08_results / [setname]_[chip/ctrl]_merged.bw
    - **Process** : Normalize each individual .bam, .bdg, and .bw file
    - **Output** : 08_results / [setname]_[chip/ctrl]_rep[#].normalized.bam
      08_results / [setname]_[chip/ctrl]_merged.normalized.bam
      08_results / [setname]_[chip/ctrl]_rep[#].normalized.bdg
      08_results / [setname]_[chip/ctrl]_merged.normalized.bdg
      08_results / [setname]_[chip/ctrl]_rep[#].normalized.bw
      08_results / [setname]_[chip/ctrl]_merged.normalized.bw

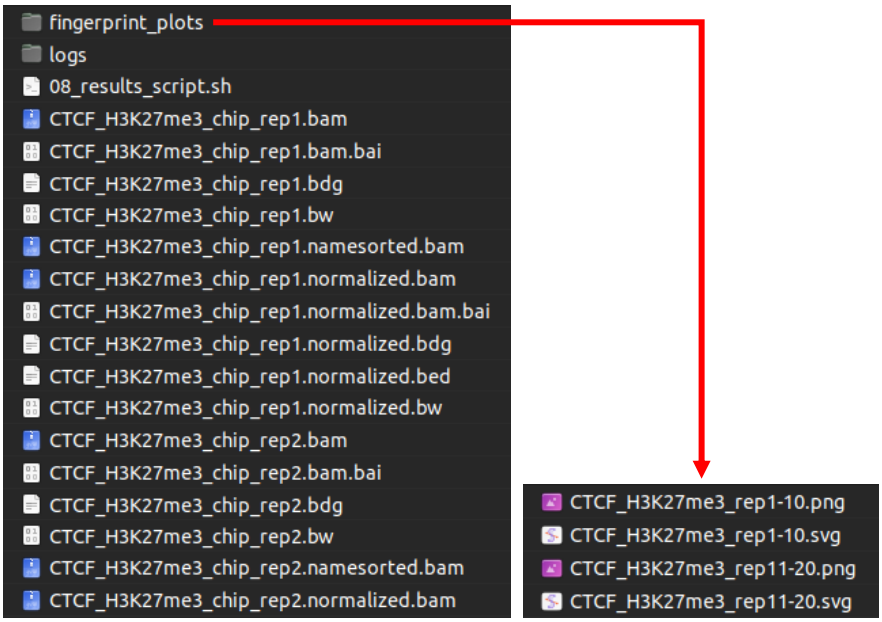
    How does it look like? *Detailed output preview of step 13, 14, 15, 16, 17 are combined below step 17*

17. **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 details. 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[#].normalized.bam
  08_results / [setname]_[chip/ctrl]_merged.normalized.bam
- **Process** : Generate fingerprint plots for all post-normalization bam files
- **Output** : 08_results / fingerprint_plots/[setname]_rep[#]-[#+9].[png/svg]
  08_results / fingerprint_plots/[setname]_merged.[png/svg]

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**), bedgraph files of all the sorted reads files (marked by the extension: **.bdg**), fingerprint plot files of all the sorted reads files (in its own folder: **fingerprint_plots**; marked by the extension: **.png** and **.svg**), and all the log files from all the program calls by **08_results_script.sh**.



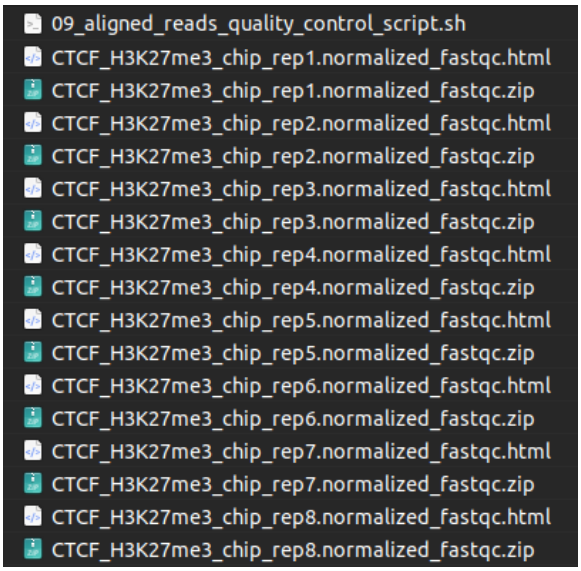
The sorted **.bam** files will be normalized if CaRAS is instructed to perform aligned reads normalization which results in **.normalized.bam**. Otherwise, the **.bam** file will simply be copy-pasted into **.normalized.bam** for filename consistency for subsequent processes. The important thing to note here: the fingerprint plots (**.svg** and **.png**), normalized BigWig (**.normalized.bw**) files, bedgraph (**.normalized.bdg**) files, **.namesorted.bam** files, and all their merged counterparts and associated index files are generated after the normalization, and thus, also commensurate to the normalization.

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

18. **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[#].normalized.bam
      08_results / [setname]_[chip/ctrl]_merged.normalized.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:



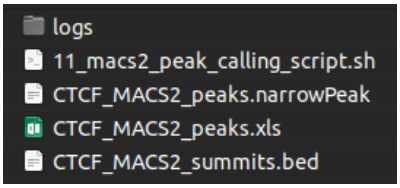
    The **.zip** files contains the individual components to be compiled for the report so you can ignore those. To read the reports, open the **.html** files. This file is a multitabular file in which you can evaluate how your alignment to reference genome and filtering based on mapping score affected your overall sequencing reads. Comprehensive as it is, explaining the contents in detail would take a whole new guide by itself. Therefore, in case the report is not enough for you: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/.

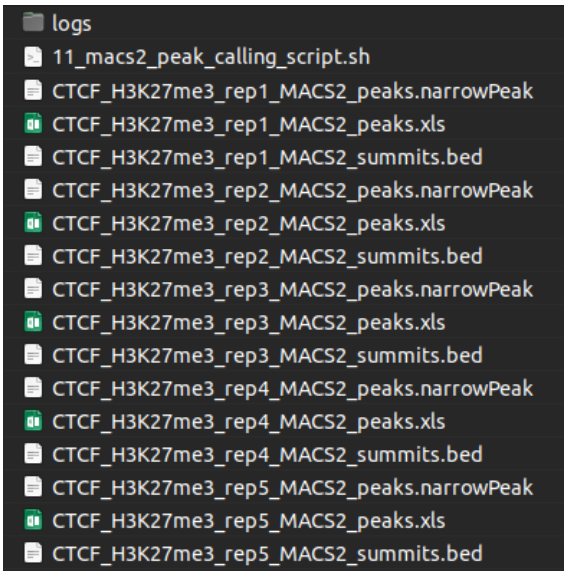
19. **Peak calling**. The same track of aligned reads is scanned for potential protein-DNA binding sites. The process returns a list of enriched regions in various formats.

Modular script used: **11_macs2_peak_calling_script.sh**

- **Calls** : macs2 callpeak
- **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].normalized.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:





In bulk analysis, samples are regarded as replicates of the same object and peaks are called based on all of them, resulting in one set of called peak list (left figure). In single-cell analysis, samples are regarded as different objects, which results in one set of peak list from every sample (right figure).

MACS2 generates three output files, which all are basically peak lists, but each of these has their own exclusive informations. The file that contains all necessary information relevant to analysis by CaRAS 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** is simply a list of peak summits coordinates and read depth at that respective 1 base coordinate.

The **.narrowPeak** file which CaRAS 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

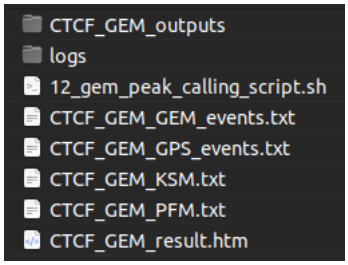
assigned scores 1-1000 based on signal value. Ideally the average signalValue per base spread is between 100-1000.

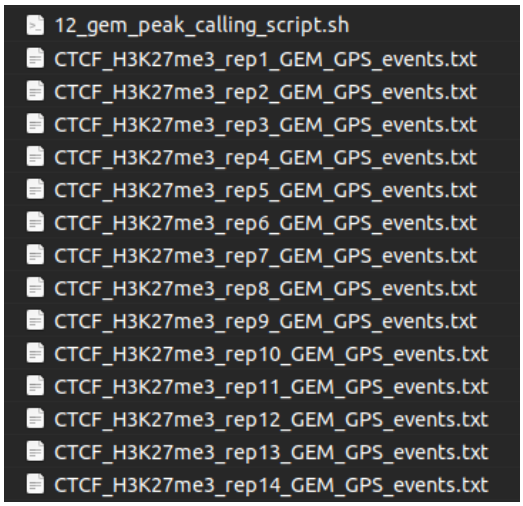
6. **strand** - +/- to denote strand or orientation (whenever applicable). Use "." if no orientation is assigned.
7. **signalValue** - Overall (usually, average) enrichment for the region.
8. **pValue** - Statistical significance (-log10). Use -1 if no pValue is assigned.
9. **qValue** - Statistical 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.

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

- **Calls** : gem
- **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].normalized.bam
- **Process** : Generate a list of called peaks
- **Output** : 12_gem_peak_calling / [setname]_GEM_GEM_events.txt

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:



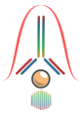


In bulk analysis, samples are regarded as replicates of the same object and peaks are called based on all of them, resulting in one set of called peak list (left figure). In single-cell analysis, samples are regarded as different objects, which results in one set of peak list from every sample (right figure).

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 CaRAS 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 informations. 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 CaRAS.

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 GEM documentations which link is provided at the end of this guide of you are interested. Lastly, **[setname]_GEM_result.htm** is a web-based comprehensive summary of all the binding events and motifs.



The increased accuracy provided by GEM motif enrichment analysis is not as crucial in single-cell analysis mode. Moreover, the motif enrichment analysis takes most of the processing time, which usage is untenable when numerous single cell samples are involved. Therefore, it will be bypassed and the analysis will carry on with **[setname]_GEM_GPS_events.txt** instead of **[setname]_GEM_GEM_events.txt**

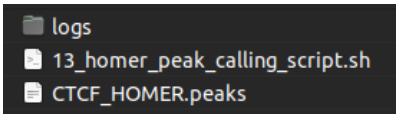
The _**GEM_events.txt** file which CaRAS utilizes, has the following columns:

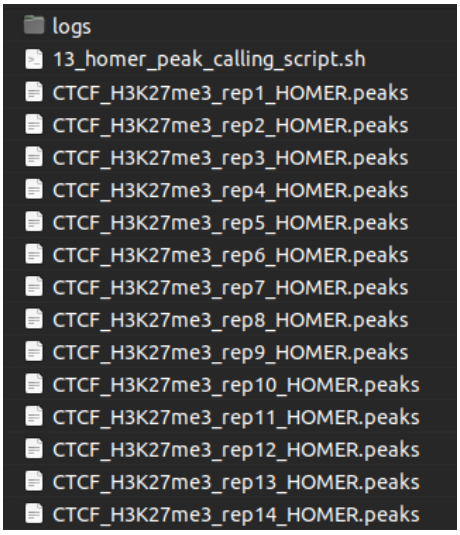
1. **Location -** The genome coordinate of this binding event
2. **IP binding strength -** The number of IP reads associated with the event
3. **Control binding strength -** the number of control reads in the corresponding region
4. **Fold -** Fold enrichment (IP/Control)
5. **Expected binding strength -** 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 Poission 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

Modular script used: **13_homer_peak_calling_script.sh**

- **Calls** : findPeaks
- **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].normalized.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:





In bulk analysis, samples are regarded as replicates of the same object and peaks are called based on all of them, resulting in one set of called peak list (left figure). In single-cell analysis, samples are regarded as different objects, which results in one set of peak list from every sample (right figure).

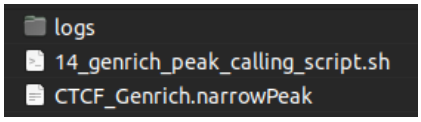
The HOMER.peaks which CARAS utilizes, has the following columns:

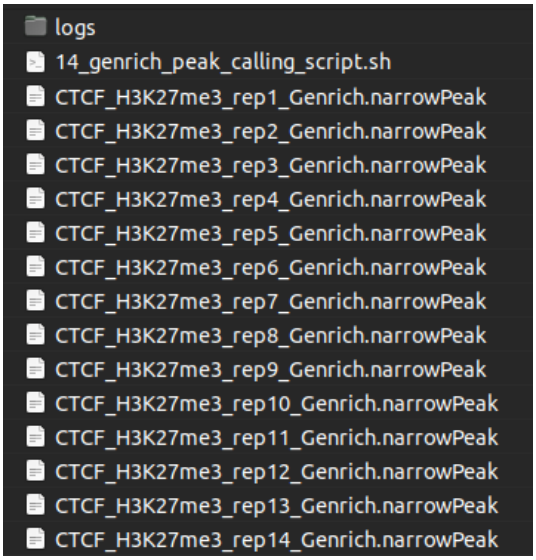
1. **PeakID -** Unique name for each peak
2. **chr -** Chromosome where peak is located
3. **start -** Starting position of peak
4. **end -** Ending position of peak
5. **Strand** (+/-)
6. **Normalized Tag Counts -** Number of tags found at the peak, normalized to 10 million total mapped tags (or defined by the user)
7. **Focus Ratio -** Fraction of tags found appropriately upstream and downstream of the peak center.
8. **Peak score -** Position adjusted reads from initial peak region reads per position may be limited)
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

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:





In bulk analysis, samples are regarded as replicates of the same object and peaks are called based on all of them, resulting in one set of called peak list (left figure). In single-cell analysis, samples are regarded as different objects, which results in one set of peak list from every sample (right figure).

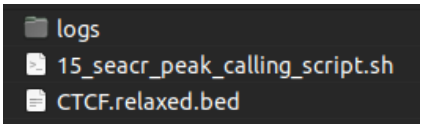
The **.narrowPeak** file which CaRAS utilizes, has the following columns:

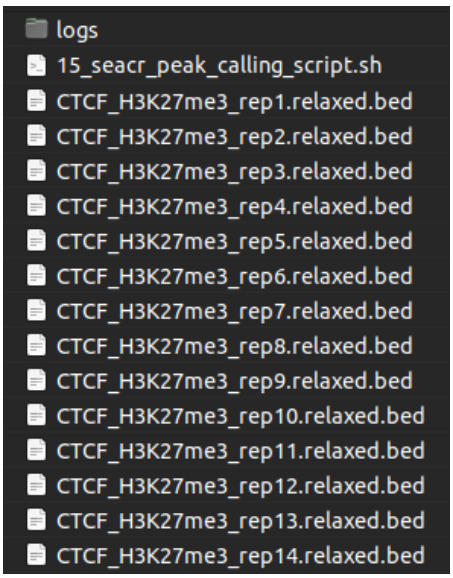
1. **chrom** - Name of the chromosome (or contig, scaffold, etc.).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base 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.

Modular script used: **15_seacr_peak_calling_script.sh**

- **Calls** : seacr
- **Input** : 08_results / [setname]_[chip/ctrl]_rep[#].normalized.bdg
- **Process** : Generate a list of called peaks
- **Output** : 15_seacr_peak_calling / [setname].[stringent/relaxed].bed

How does it look like? After **15_seacr_peak_calling_script.sh** had been executed, the folder: **15_seacr_peak_calling** will contain all these files, just like below:





In bulk analysis, samples are regarded as replicates of the same object and peaks are called based on all of them, resulting in one set of called peak list (left figure). In single-cell analysis, samples are regarded as different objects, which results in one set of peak list from every sample (right figure).

SEACR generates output file which naming is based on the peak calling mode used (with relaxed or stringent thresholding). The example shown here is when SEACR is used with the default settings (relaxed thresholding). Stringent thresholding is not needed due to the fact that the other five peak callers are already at work in tackling potential false positives.

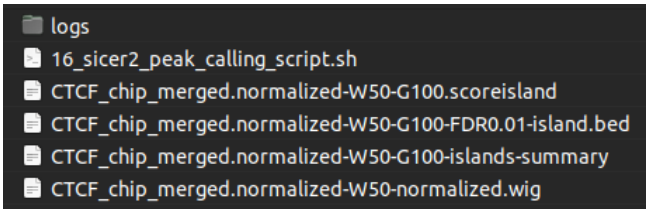
The **.[relaxed / stringent].bed** file which CaRAS utilizes, has the following columns:

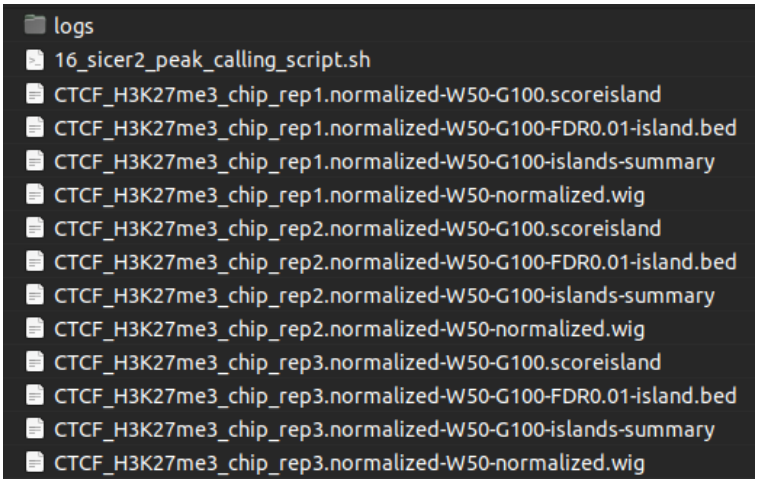
1. **Chromosome**
2. **Start coordinate**
3. **End coordinate**
4. **Total signal** contained within denoted coordinates
5. **Maximum bedgraph signal** attained at any base pair within denoted coordinates
6. **Furthest upstream and furthest downstream** bases within the denoted coordinates that are represented by the maximum bedgraph signal

Modular script used: **16_sicer2_peak_calling_script.sh**

- **Calls** : sicer
- **Input** : 08_results / [setname]_[chip/ctrl]_merged.normalized.bam
- **Process** : Generate a list of called peaks
- **Output** : 16_sicer2_peak_calling / [setname]-W*-G*-islands-summary
  (* depends on -w and -g flag arguments. Defaults are 50 and 100, respectively)

How does it look like? After **16_sicer2_peak_calling_script.sh** had been executed, the folder: **16_sicer2_peak_calling** will contain all these files, just like below:





In bulk analysis, samples are regarded as replicates of the same object and peaks are called based on all of them, resulting in one set of called peak list (top figure). In single-cell analysis, samples are regarded as different objects, which results in one set of peak list from every sample (bottom figure).

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.

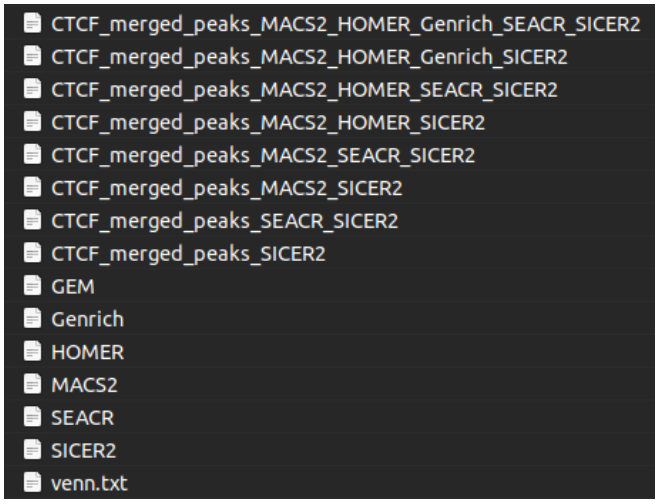
The file that contains all necessary information relevant to analysis by CaRAS is the one with **[setname]_chip_merged-W*-G*-islands-summary.**

20. **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. These multiple files are then concatenated together into a single regions list file.

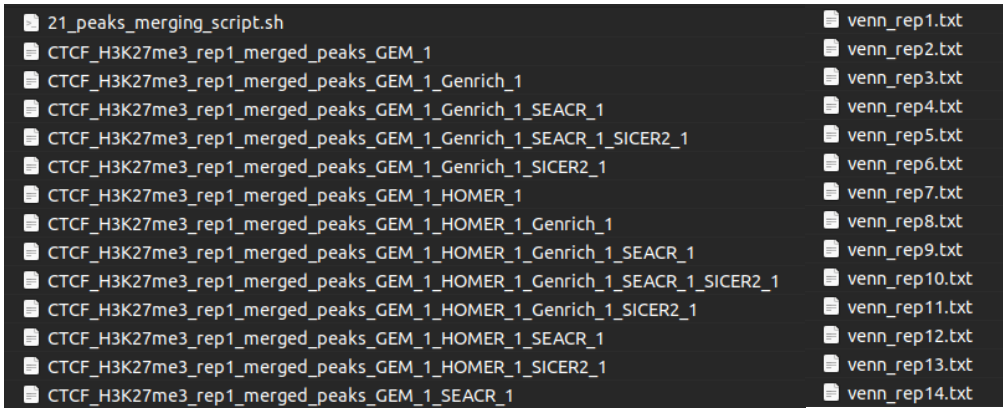
Modular script used: **21_peaks_merging_script.sh**

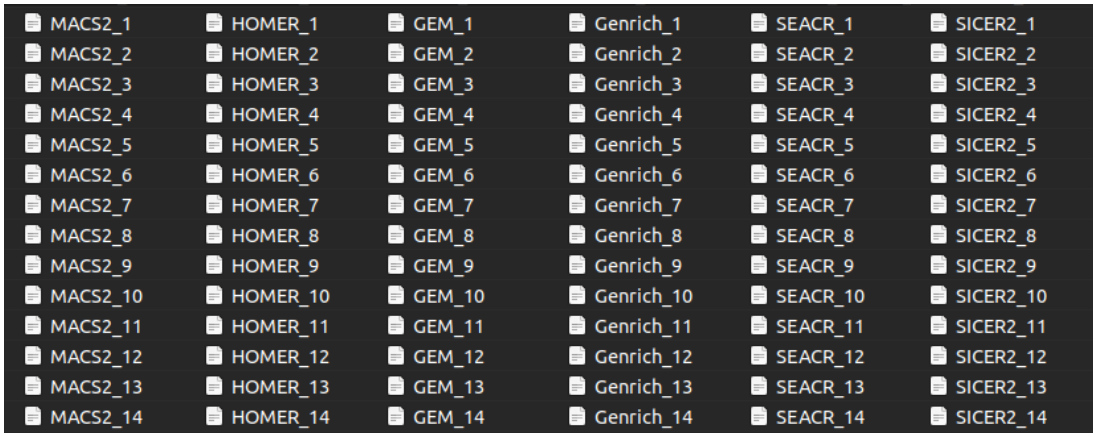
- **Calls** : mergePeaks
- **Input** : 11_macs2_peak_calling / [setname]_MACS2_peaks.narrowPeak
  12_gem_peak_calling / [setname]_GEM_GEM_events.txt
  13_homer_peak_calling / [setname]_HOMER.peaks
  14_genrich_peak_calling / [setname]_Genrich.narrowPeak
  15_seacr_peak_calling / [setname].[stringent/relaxed].bed
  16_sicer2_peak_calling / [setname]-W*-G*-islands-summary
- **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:



In bulk analysis, samples are regarded as replicates of the same object, generating one set of peak merging results (above figure). In single-cell analysis, samples are regarded as different objects, generating one set of peak merging results from every sample (below figures).





CaRAS takes the input files described above, then generates the four tab-separated-values files **MACS2**, **GEM**, **HOMER**, **Genrich**, **SEACR**, and **SICER2**, which are basically 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.

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 21, 22, 23, and 24 are combined below step 24*

21. **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 CaRAS), 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 of this, step 21, 22, 23, 24, 25, and 26 are combined below step 26*

22. **Fold enrichment calculations**. Performed by a custom script, with the help of samtools depth and view modules. For weighted peak center fold enrichment calculation, 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 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.py
    - **Input** : 22_peaks_processing / [setname]_all_peaks_annotated.tsv
    - **Process** : Calculate ChIP tag counts (read depth)
      Calculate weighted center fold change
      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

    How does it look like? *Detailed output preview of this, step 21, 22, 23, 24, 25, and 26 are combined below step 26*

23. **Peak feature extraction**. Performed by a custom script. Only for single-cell analysis. Gene ontology and pathway enrichment analysis are performed on each single cell sample, generating enriched terms ranked lists derived from various available databases. The similarity values between all possible combinations of two samples are calculated based on their distinct enriched terms ranked lists, using the rank-biased overlap (RBO) method, generating a distance matrix. These feature values provide distinctive signature of each sample, which will help unsupervised clustering algorithm to identify, measure, and group all the samples based on GO terms and pathways profile similarity.

    Modular script used: **22_peaks_processing_script.sh**
    - **Calls** : peak_feature_extractor.py
    - **Input** : 22_peaks_processing / [setname]_all_peaks_concatenated.tsv
    - **Process** : Extracts each sample's enriched GO terms and pathways profile for clustering

    How does it look like? *Detailed output preview of this, step 21, 22, 23, 24, 25, and 26 are combined below step 26*

24. **Sample clustering**. Performed by a custom script. Only for single-cell analysis. Based on the extracted features from the peak feature extraction step, this step projects the generated distance matrix onto a two-dimensional plane using the multidimensional scaling algorithm (MDS). After projection, samples are clustered into groups with spectral clustering. The 2D projection of the distance-based datapoints is visualized, color-labelled accordingly to the group the samples are assigned to, for users to examine whether they are appropriately segregated. When not explicitly provided by the user in the command line, CaRAS determines the optimal number of clusters by finding the eigenvalues and their associated eigen vectors followed by identifying the maximum gap which corresponds to the number of clusters by eigengap heuristic. These groups will be regarded as distinct cell populations and undergo identical but separate downstream analyses. CaRAS's cluster-based multiple analyses are aimed to provide more accurate and specific results for every different cell population, which is one of the main advantages of single-cell analysis over the typical bulk analysis for DNA-protein interactions.

    Modular script used: **22_peaks_processing_script.sh**
    - **Calls** : peak_feature_extractor.py
    - **Process** : Cluster each sample based on its GO terms and pathways profile
    - **Output** : 22_peaks_processing / [setname]_[peakset]_[GO/pathway database]_[group#]_ all_peaks _clustered.tsv
      22_peaks_processing / [setname]_[peakset]_[GO/pathway database]_Term_ Ranking_RBO_Distance_MDS_Spectral_Clustering.png

    How does it look like? *Detailed output preview of this, step 21, 22, 23, 24, 25, and 26 are combined below step 26*

25. **Irreproducibility rate (IDR) calculation.** Performed by a custom script plugged to IDR module. Only for bulk analysis. The IDR module compares two different peak sets and assigns to each listed peak in both peak sets an irreproducibility rate (IDR) value based on that peak's capacity to be recalled by the other peak set. IDR value of each peak listed in the full (union) peak list ([setname]_all_peaks_calculated.tsv) was obtained by pairing it against every individual peak caller sets, followed by calculating the pair-wise -logIDR values, then summing them all up, and finally converting it into a final IDR value. The final IDR value shows the chance of a finding (i.e., the peak) being unable to be reproduced by different peak calling algorithms. This step reprocesses the peak list [setname]_all_peaks_calculated.tsv, augment the table with relevant IDR values, and re-save it under the same file name. For more details about IDR calculation method, see the IDR module documentation.

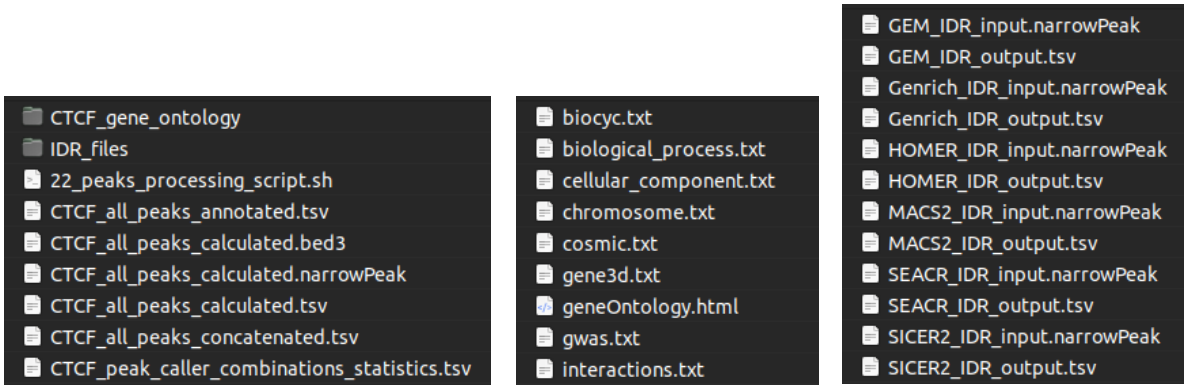
    Modular script used: **22_peaks_processing_script.sh**
    - **Calls** : IDR_integrator.py, IDR
    - **Input** : 22_peaks_processing / [setname]_all_peaks_calculated.tsv
    - **Process** : Calculate -logIDR of peaks between union peak set vs MACS2 peak set
      Calculate -logIDR of peaks between union peak set vs GEM peak set
      Calculate -logIDR of peaks between union peak set vs HOMER peak set
      Calculate -logIDR of peaks between union peak set vs Genrich peak set
      Calculate -logIDR of peaks between union peak set vs SEACR peak set
      Calculate -logIDR of peaks between union peak set vs SICER2 peak set
      Calculate IDR value of peaks from the sum of all six -logIDR values
    - **Output** : 22_peaks_processing / [setname]_all_peaks_calculated.tsv

    How does it look like? *Detailed output preview of this, step 21, 22, 23, 24, 25, and 26 are combined below step 26*

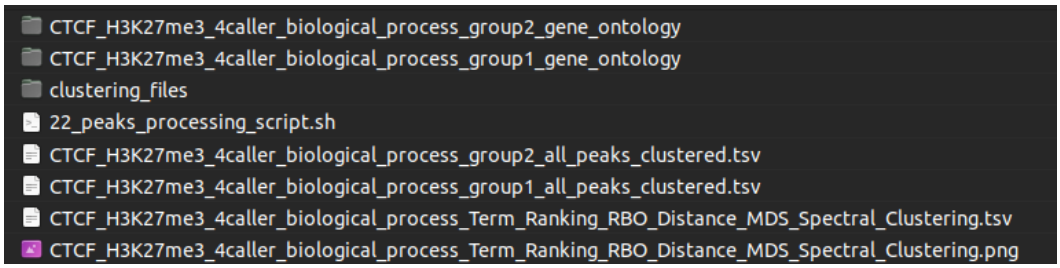
26. **Peak statistics summary**. Performed by a custom script designed for quality assessment of called peaks. Returns a summary text file containing information pertaining to the peak read depth, peak fold enrichment, known motif hits, and positive peak hits (based on known motif presence), in each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

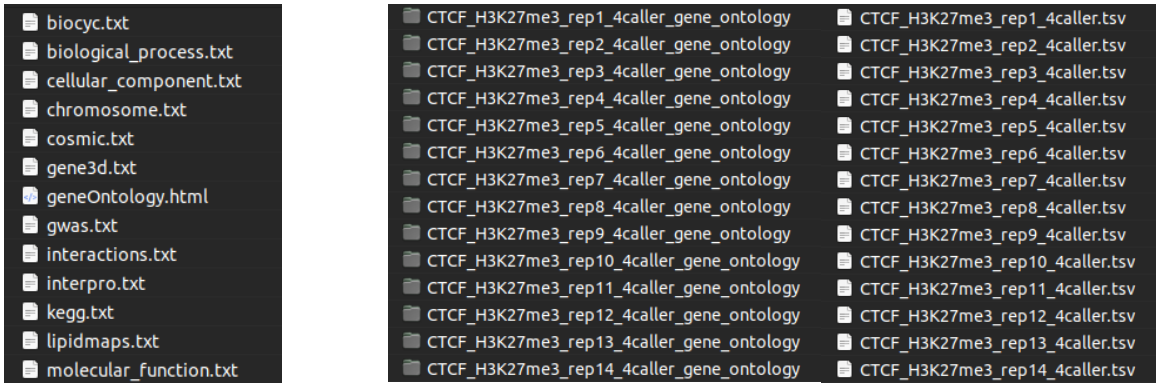
    Modular script used: **22_peaks_processing_script.sh**
    - **Calls** : peak_caller_stats_calculator.py
    - **Input** : 22_peaks_processing / [setname]_all_peaks_calculated.tsv
    - **Process** : Generate a separate summary table of key statistics in peak callers performance
    - **Output** : 22_peaks_processing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



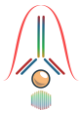
In bulk analysis, samples are regarded as replicates of the same object, generating one set of peak processing results (above figures). In single-cell analysis, samples are regarded as different objects, generating one set of peak processing results from every sample (below figures), on top of other clustering-replated files that lead to the different sample grouping.





In bulk analysis, the end of step 20 concatenates all peak caller combinations peak list files into one file: **[setname]_all_peaks_concatenated.tsv**, followed by annotation in step 21 that generates the file: **[setname]_all_peaks_annotated.tsv**, followed by fold change and IDR calculation in step 22 and 25 that generates the file: **[setname]_all_peaks_calculated.tsv**. Step 26 reads the resulting file **[setname]_all_peaks_calculated.tsv** and generates a statistics summary file **[setname]_peak_caller_combinations_statistics.tsv**.

In single-cell analysis, the end of step 20 concatenates all peak caller combinations peak list files ***from each sample*** into one file: **[setname]_[rep#]_all_peaks_concatenated.tsv**. The same procedure follows, generating the file: **[setname]_[rep#]_all_peaks_annotated.tsv**, followed by the file: **[setname]_[rep#]_all_peaks_calculated.tsv**. Afterwards, CaRAS single-cell analysis skips IDR calculation and proceeds with peak feature extraction (step 23), performing GO or pathway enrichment analysis on selected peak set of each sample (**[setname]_[rep#]_[#caller].tsv**) which generates results within a folder dedicated to each sample (**[setname]_[rep#]_[#caller]_gene_ontology**). This will then be followed by sample clustering (step 24) which segregates the numerous samples into a few groups. The file: **[setname]_[peakset]_[GO / pathway database]_Term_Ranking_RBO_Distance_MDS_Spectral_Clustering.png** provides a view of the clustering results of all samples. Meanwhile: **[setname]_[peakset]_[GO / pathway database]_Term_Ranking_RBO_Distance_MDS_**



Spectral_Clustering.tsv provides the information regarding which sample was clustered into which group. Peak lists of samples which belong to the same group are concatenated into: **[setname]_[peakset]_[GO / pathway database]_[group#]_all_peaks_clustered.tsv** for further downstream processing, starting immediately with GO or pathway enrichment analysis on the concatenated peak list, which results represent each whole group and saved in folder: **[setname]_[peakset]_[GO / pathway database]_[group#]_gene_ontology**. Later down the workflow, the concatenated-by-group peak lists are the ones processed in motif enrichment analyses, which results should reveal the potential DNA-binding motif of the target protein in their own respective sample group. Peak caller statistics for each and every sample will be too much details in single-cell analysis, therefore the step will be omitted in this case.

While HOMER annotatePeaks are 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 CaRAS 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.**

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.

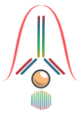
[setname]_all_peaks_concatenated.tsv already has information pertaining to:

- 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 are 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, 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. CaRAS 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 alternative ways 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.

To accomodate, at this point, the peak list now has the number of peak caller overlaps, which is simply 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.

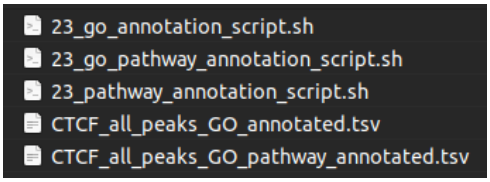
27. **(Optional) Downstream analysis: Gene ontology enrichment**. Each peak in the concatenated list is appended with all the gene ontology terms associated with its gene annotation. The gene ontology terms are derived from biological processes, molecular functions, and cellular compartments databases. This enables list filtering based on the gene ontology terms of the study's interest. Due to the high verbosity of the results, and thus large file sizes, this step is omitted from single-cell analysis.

    Modular script used: **23_go_annotation_script.sh**
    - **Calls** : go_annotator.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

28. **(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. Due to the high verbosity of the results, and thus large file sizes, this step is omitted from single-cell analysis.

    Modular script used: **23_pathway_annotation_script.sh**
    - **Calls** : pathway_annotator.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:

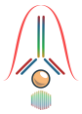


If **--goann** flag is used during CaRAS call, **23_go_annotation_script.sh** will be executed, and generates **[setname]_all_peaks_go_annotated.tsv** that contains the following gene ontology terms based on each peak's nearest gene:

- Biological Process
- Molecular Function
- Cellular Component

If **--pathann** flag is used during CaRAS 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 CaRAS 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.

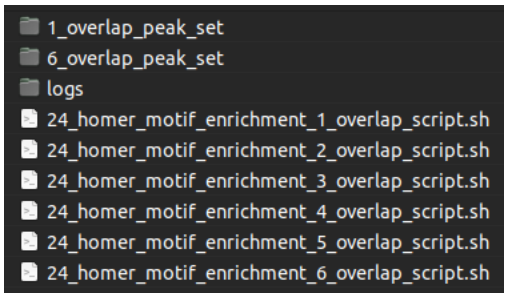
However, with respect to users who do not need such information and probably want their list to be much less verbose and smaller in size, this step is completely optional. The output files will not be generated, to save processing time should the user choose to omit this step. The scripts will still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

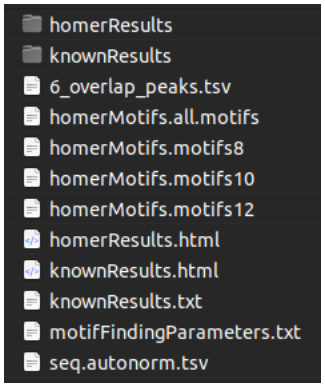
29. **(Optional) Downstream analysis: Motif enrichment analysis with HOMER**. Genomic sequences are extracted based on the coordinates of the peaks in consensus (six peak callers overlap), union (all called peaks), or both peak lists. HOMER performs analysis to identify specific DNA sequence motifs to which the experimented protein(s) have binding affinity towards. For the sake of processing speed, HOMER utilizes cumulative binomial distribution to calculate motif enrichment by default. However, by utilizing CaRAS custom setting table, user may choose to utilize cumulative hypergeometric distribution, which describes motif enrichment problem more accurately. Besides the typically performed calculations for de novo motifs discovery, HOMER also calculates the enrichment scores of the known motifs in HOMER motifs database.

Modular script used: **24_homer_motif_enrichment_[#]_overlap_script.sh**

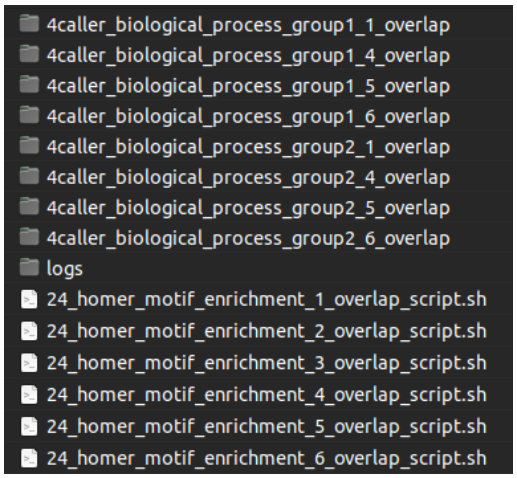
- **Calls** : findMotifsGenome.pl
- **Input** : 22_peaks_processing / [setname]_all_peaks_calculated.tsv
- **Process** : Performs motif enrichment analysis
- **Output** : 24_homer_motif_enrichment / … / homerResults.html
  24_homer_motif_enrichment / … / knownResults.html

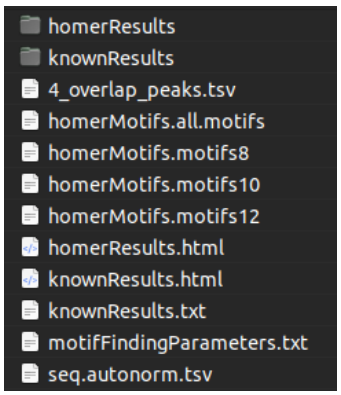
How does it look like? After **24_homer_motif_enrichment_[#]_overlap_script.sh** had been executed, the folder: **24_homer_motif_enrichment** will contain these folders and files, just like below:





In bulk analysis, the is only one peak list after peak processing, in which all samples were regarded as replicates within the same population or group. This will then generate one set of motif enrichment analysis result for each peak set (above figures). In single-cell analysis, samples were segregated into several groups, and peak list from every sample was combined into a single file according to the grouping. This will instead generate multiple sets of motif enrichment analysis result for each peak set, one set for each group (below figures).



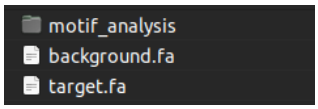


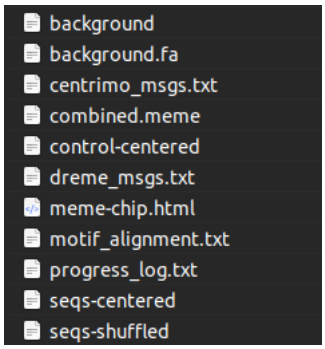
All six **24_homer_motif_enrichment_[#]_overlap_script.sh** scripts (where # = 1 - 6; left figure) will be generated but only selected ones will be executed (based on the argument given through the **--homer_motif** flag), and will generate **HOMER findMotifsGenome.pl** output folders and files as depicted by the right figure.

30. **(Optional) Downstream analysis: Motif enrichment analysis with MEME**. Genomic sequences are extracted based on the coordinates of the peaks in consensus (six peak callers overlap), union (all called peaks), or both peak lists. With or without control sequences extracted from random genomic sequences, MEME performs analysis to identify specific DNA sequence motifs to which the experimented protein(s) have binding affinity towards. By utilizing separate dedicated modules included in MEME suite, MEME-ChIP performs de novo motif discovery, motif enrichment analysis, motif location analysis and motif clustering in one go, providing a comprehensive picture of the DNA motifs that are enriched in the extracted sequences. MEME-ChIP performs two complementary types of de novo motif discovery: weight matrix–based discovery for high accuracy, and word-based discovery for high sensitivity.

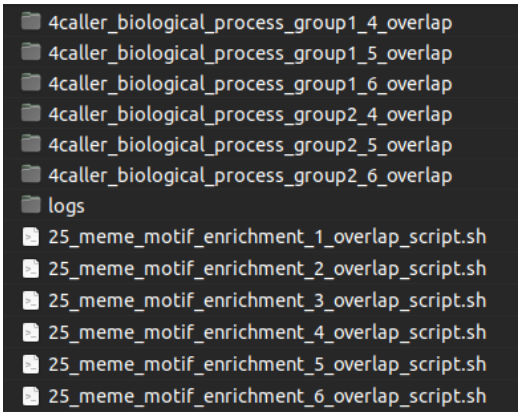
    Modular script used: **25_meme_motif_enrichment_[#]_overlap_script.sh**
    - **Calls** : meme-chip
    - **Input** : 22_peaks_processing / [setname]_all_peaks_calculated.tsv
    - **Process** : Performs motif enrichment analysis
    - **Output** : 25_meme_motif_enrichment / … / meme-chip.html

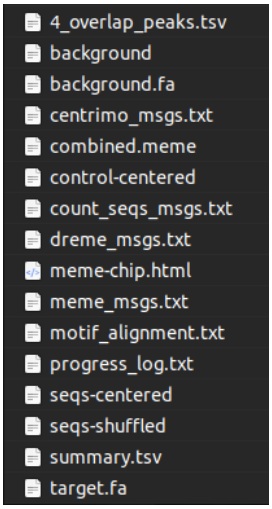
    How does it look like? After **25_meme_motif_enrichment_[#]_overlap_script.sh** had been executed, the folder: **25_meme_motif_enrichment** will contain these folders and files, just like below:





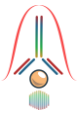
In bulk analysis, the is only one peak list after peak processing, in which all samples were regarded as replicates within the same population or group. This will then generate one set of motif enrichment analysis result for each peak set (above figures). In single-cell analysis, samples were segregated into several groups, and peak list from every sample was combined into a single file according to the grouping. This will instead generate multiple sets of motif enrichment analysis result for each peak set, one set for each group (below figures).





All six **25_meme_motif_enrichment_[#]_overlap_script.sh** scripts (where # = 1 - 6; left figure) will be generated but only selected ones will be executed (based on the argument given through the **--meme_motif** flag), and will generate **MEME-ChIP** output folders and files as depicted by the right figure.

When ChIP and control aligned reads (.bam) have the same number of replicates, CaRAS gives the option for merged (with **--fcmerge** flag) or pair-wise (without **--fcmerge** flag) fold change calculations. In pair-wise mode, peaks every ChIP vs control replicate have different weighted peak center coordinate, which directly affects the actual target and background sequences for meme-chip to perform enrichment analysis on. Therefore, CaRAS recognizes and processes multiple replicates separately (based on each weighted peak center coordinate), generating respective results for each replicate, stored in separate folders. On the contrary, whenever user choose to use **--fcmerge** flag, or when the number of ChIP and control samples are not the same, CaRAS will be forced to perform merged fold change calculation. In this situation every peak will have one weighted peak center coordinate and thus there will only be one replicate of meme-chip motif enrichment analysis results inside one single folder. ***Important note***: **--fcmerge** is irrelevant in single-cell analysis since every sample is going to be processed individually.



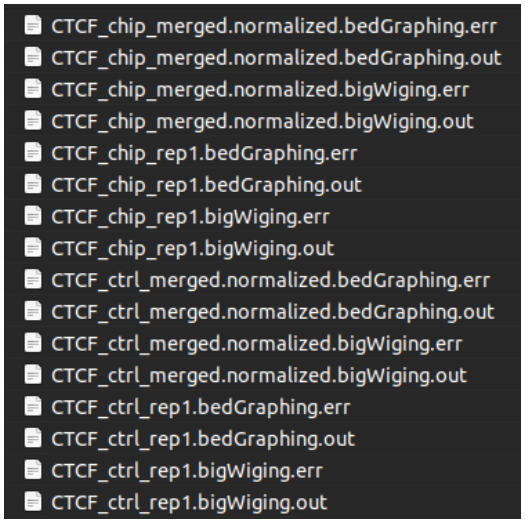
Log Files

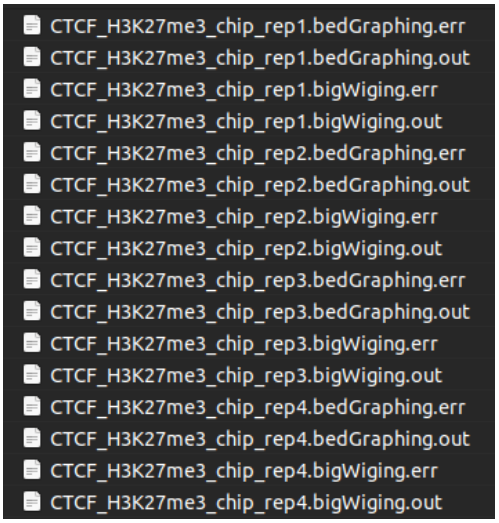
For most processes in every modular scripts, log files are recorded and saved in folder: /logs under their respective directories. There are two types of log files:

- 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

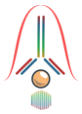
Sometimes, unlike what the file or channel name suggests, you might find errors reported in the **.out** files or something like normal program run progress report written in the **.err** files. That is just the way it is. Some programs do not follow the standard output / standard error convention, that's 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:





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/CaRAS/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 (unique 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 (chromosome)	
Column 3 (C)	Start (peak start coordinate)	
Column 4 (D)	End (peak end coordinate)	
Column 5 (E)	Strand (strand on which peak is found)	
Column 6 (F)	Peak Caller Combination	
Column 7 (G)	Peak Caller Overlaps	Pipeline script: 22_peaks_processing_script.sh Called script: fold_change_calculator.py IDR_integrator.py Called program: samtools depth samtools view IDR 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)	Peak Center	
Column 12 (L)	Number of Motifs	
Column 13 (M)	negLog10_IDR	
Column 14 (N)	IDR	
Column 15 (O)	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 16 (P)	Detailed Annotation	
Column 17 (Q)	Distance to TSS	
Column 18 (R)	Nearest PromoterID	
Column 19 (S)	Entrez ID	
Column 20 (T)	Nearest Unigene	
Column 21 (U)	Nearest Refseq	
Column 22 (V)	Nearest Ensembl	
Column 23 (W)	Gene Name	
Column 24 (X)	Gene Alias	
Column 25 (Y)	Gene Description	
Column 26 (Z)	Gene Type	
Column 27 (AA)	CpG%	
Column 28 (AB)	GC%	
Column 29 (AC)	Biological Process	Pipeline script: 23_go_annotation_script.sh Called script: GO_annotator.py Output file: [filename]_all_peaks_go_annotated.tsv Output folder: 23_supplementary_annotations
Column 30 (AD)	Molecular Function	
Column 31 (AE)	Cellular Component	
Column 32 (AF)	Interaction with Common Protein	Pipeline script: 23_pathway_annotation_script.sh Called script: pathway_annotator.py Output file: [filename]_all_peaks_pathway_annotated.tsv Output folder: 23_supplementary_annotations
Column 33 (AG)	Somatic Mutations (COSMIC)	
Column 34 (AH)	Pathway (KEGG)	
Column 35 (AI)	Pathway (BIOCYC)	
Column 36 (AJ)	Pathway (pathwayInteractionDB)	
Column 37 (AK)	Pathway (REACTOME)	
Column 38 (AL)	Pathway (SMPDB)	
Column 39 (AM)	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
Control Tag Count	Control read depth at weighted peak center
Fold Change	ChIP vs control fold change at weighted peak center
Peak Center	Genomic coordinate of the weighted peak center
Number of Motifs	Number of motif instances found in the peak region
negLog10_IDR	Sum of - $\log_{10}$ IDR obtained from peak IDR calculation between the union and each individual peak caller set
IDR	Peak irreproducibility rate converted from negLog10_IDR column value
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 basepairs
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.txt that are related to the nearest gene
Molecular Function	Terms from molecular_function.txt that are related to the nearest gene
Cellular Component	Terms from cellular component.txt that are related to the nearest gene
Interaction with Common Protein	Known interactions from interaction.txt with the nearest gene
Somatic Mutations (COSMIC)	Known cancer cases where mutation of the nearest gene are found
Pathway (KEGG)	Known pathways according to the KEGG database
Pathway (BIOCYC)	Known pathways according to the Biocyc database
Pathway (pathwayInteractionDB)	Known pathways from the pathwayInteractionDB database
Pathway (REACTOME)	Known pathways according to the REACTOME database
Pathway (SMPDB)	Known pathways according to the SMPDB database
Pathway (Wikipathways)	Known pathways according to the wikipathways database

Motif enrichment analysis results (by HOMER)

All the results are compiled and can be viewed by opening the file homerResults.html in an HTML file viewer such as your internet browser. This file gives you a formatted, organized view of the enriched de novo motifs and all the relevant information, as can be seen below. Additionally, more details can be accessed by simply clicking on the links in the table.

Homer *de novo* Motif Results

Known Motif Enrichment Results
Gene Ontology Enrichment Results
If Homer is having trouble matching a motif to a known motif, try copy/pasting the matrix file into STAMP
More information on motif finding results: HOMER | Description of Results | Tips
Total target sequences = 37301
Total background sequences = 35962
* - possible false positive

Rank	Motif	P-value	log P-pvalue	% of Targets	% of Background	STD(Bg STD)	Best Match/Details	Motif File
1		1e-12661	-2.915e+04	70.91%	15.19%	40.5bp (65.1bp)	Foxa2(Forkhead)/Liver-Foxa2-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
2		1e-578	-1.332e+03	27.14%	16.52%	54.0bp (65.5bp)	NF1-halfsite(CTF)/LNCaP-NF1-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
3		1e-384	-8.860e+02	17.77%	10.53%	53.9bp (62.1bp)	Unknown/Homeobox /Limb-p300-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
4		1e-164	-3.783e+02	3.17%	1.28%	52.2bp (62.9bp)	PH0048.1_Hoxa13 More Information | Similar Motifs Found	motif file (matrix)
5		1e-151	-3.485e+02	3.38%	1.47%	50.2bp (65.4bp)	NF-E2(bZIP)/K562-NFE2-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
6		1e-107	-2.485e+02	1.21%	0.35%	56.3bp (69.7bp)	CTCF(Zf)/CD4+-CTCF-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
7		1e-72	-1.671e+02	2.10%	1.02%	55.1bp (58.5bp)	MA0029.1_Evi1 More Information | Similar Motifs Found	motif file (matrix)

We can see these information listed below from the table:

Rank	**Motif Rank**
Motif	Motif position weight matrix logo
P-value	Final enrichment p-value
log P-pvalue	Log of p-value
% of Targets	Number of target sequences with motif/ total targets
% of Background	Number of background sequences with motif/ total background
STD(Bg STD)	Standard deviation of position in target and background sequences
Best Match/Details	Best match of de novo motif to motif database

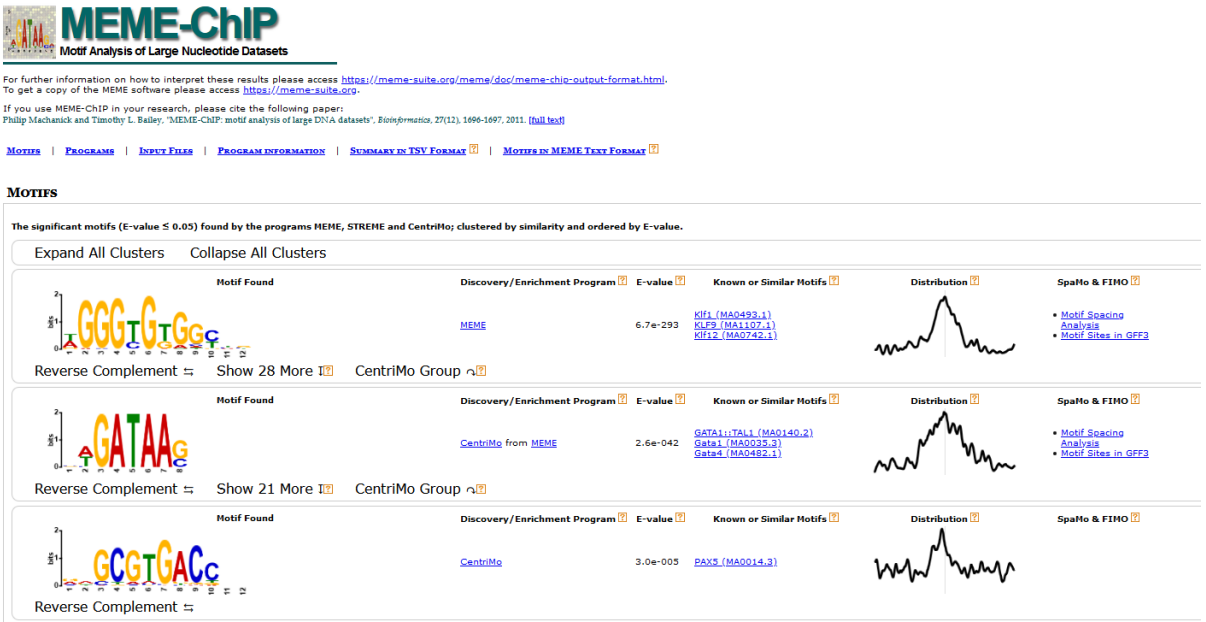
In addition to de novo motif enrichment, homer also performs motif enrichment analysis on the known binding motifs readily available within their database repertoire. The results for this analysis can be viewed in a similar way by opening the file knownResults.html, which contains similar information as its de novo counterpart.

More detailed information is available in http://homer.ucsd.edu/homer/ngs/peakMotifs.html

Motif enrichment analysis results (by MEME)

All the results are compiled and can be viewed by opening the file meme-chip.html in an HTML file viewer such as your internet browser. This file gives you a formatted, organized view of the enriched de novo motifs and all the relevant information. Additionally, more details can be accessed by simply clicking on the links in the table.

In datasets where there is an equal number of multiple-replicated ChIP and control samples, CaRAS will have MEME perform a pair-wise motif enrichment analysis. Therefore, in that case, there will be multiple replicates of motif enrichment results, each one looking like below.



While the file above gives a more graphical representation of the results, meme-chip also generates another file: summary.tsv, which contains the same information (see below) repackaged in table format suitable for subsequent processing, if needed.

MOTIF_INDEX	**The index of the motif in the "Motifs in MEME text format" file ('combined.meme') output by MEME-ChIP.**
MOTIF_SOURCE	The name of the program that found the de novo motif, or the name of the motif file containing the known motif.
MOTIF_ID	The name of the motif, which is unique in the motif database file.
ALT_ID	An alternate name for the motif, which may be provided in the motif database file.
CONSENSUS	The ID of the de novo motif, or a consensus sequence computed from the letter frequencies in the known motif (as described below).
WIDTH	The width of the motif.
SITES	The number of sites reported by the de novo program, or the number of "Total Matches" reported by CentriMo.
E-VALUE	The statistical significance of the motif.
E-VALUE_SOURCE	The program that reported the E-value.
MOST_SIMILAR_MOTIF	The known motif most similar to this motif according to Tomtom.
URL	A link to a description of the most similar motif, or to the known motif.

More detailed information is available in https://meme-suite.org/meme/doc/meme-chip.html

Miscellaneous Pipeline Outputs

Multiple peak callers statistics summary

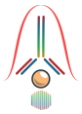
The table below shows the contents of *[filename]_peak_caller_combinations_statistics.tsv*.

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

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
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 order of samples were incorrect (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.

```
caras.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
```

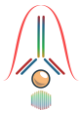
Sample Table

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 CaRAS 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

As in its predecessor, ChIP-AP, the *cornerstone* of CaRAS's functionality is the settings table. CaRAS, with the raw fq files and the settings table, is able to reproduce (near) identically any analysis that was performed (provided the same program versions are used). The 'near identically' statements is owing to the fact that reported alignments of multi-mappers may, in some cases, give ever so slightly different results. This ambiguity can be alleviated however by filtering out alignments with low MAPQ scores in the corresponding alignment filter step, post-alignment to ensure consistent results from every analysis run. The provision of the settings table therefore 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 CaRAS is run, a copy of the used settings table is saved as a table file: *[filename]_ setting_table.tsv* in the output save folder. If you have a custom settings table made and provided it as input, then CaRAS 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 settings table file in its default-setting state.

fastqc1	-q
clumpify	dedupe spany addcount qout=33 fixjunk
bbduk	ktrim=r k=21 mink=8 hdist=2 hdist2=1
trimmomatic	LEADING:20 SLIDINGWINDOW:4:20 TRAILING:20 MINLEN:20
fastqc2	-q
bwa_mem	
samtools_view	-q 20
plotfingerprint	
reads_normalizer	--norm properly_paired
fastqc3	-q
macs2_callpeak	
gem	-Xmx10G --k_min 8 --k_max 12
homer_findPeaks	-region
genrich	-v
seacr	
sicer2	-w 50 -g 100

homer_mergePeaks	
peak_feature_extractor	--filter 4 --top_rank 50 --database biological_process
homer_annotatePeaks	
fold_change_calculator	--normfactor user_value --chip_norm 1 --ctrl_norm 1
homer_findMotifsGenome	-size given -mask
meme_chip	-meme-nmotifs 25

Interpreting CaRAS Output

[Some parts adapted from CaRAS manual (https://github.com/JSuryatenggara/CaRAS)]. CaRAS does report a fair amount of stuff. If you ran it locally you have a swath of folders and you have no 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 we are going to walk you through that!

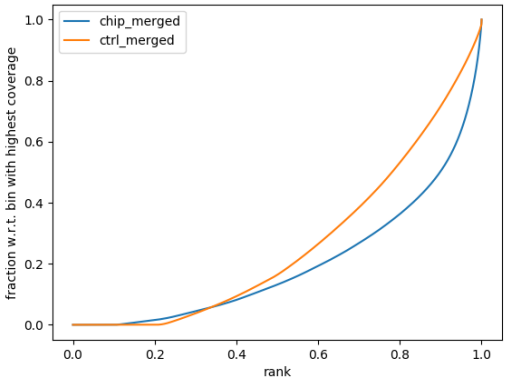
Did my analysis work?

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 your 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)

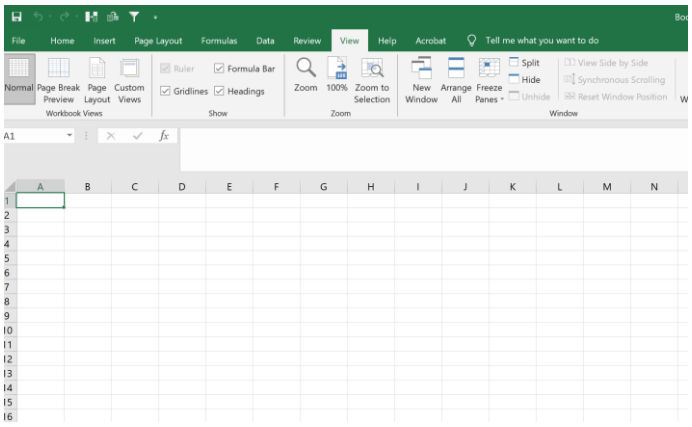
In the folder *"21_peaks_merging"* folder, you will find the *"venn.txt"* file. This will show you a textual Venn diagram of the overlap between the called peaks across all peak callers. To know your experiment worked well, you should see a full list with combinations of all peak callers and relatively 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 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.

MACS2	SICER2	HOMER	Genrich	Total	Name
			X	103	Genrich
		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

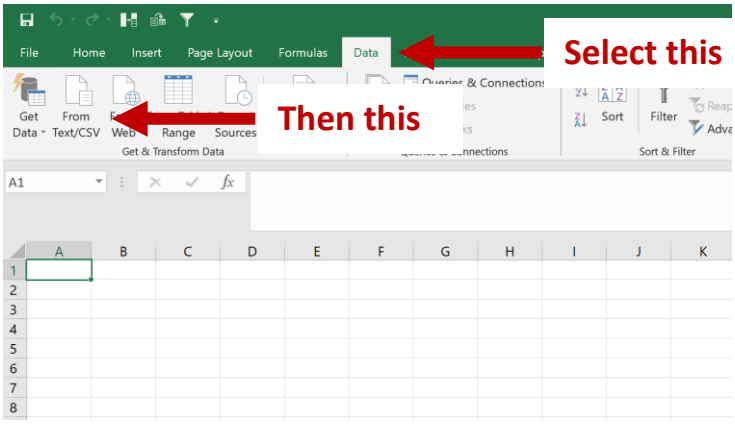
3 – What results files do I look at exactly?

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?)

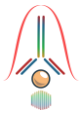
From this file, you can view all the results and data for you analysis. Refer to Setting Table & Default Parameters

As in its predecessor, ChIP-AP, the *cornerstone* of CaRAS's functionality is the settings table. CaRAS, with the raw fq files and the settings table, is able to reproduce (near) identically any analysis that was performed (provided the same program versions are used). The 'near identically' statements is owing to the fact that reported alignments of multi-mappers may, in some cases, give ever so slightly different results. This ambiguity can be alleviated however by filtering out alignments with low MAPQ scores in the corresponding alignment filter step, post-alignment to ensure consistent results from every analysis run. The provision of the settings table therefore 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 CaRAS is run, a copy of the used settings table is saved as a table file: *[filename]_ setting_table.tsv* in the output save folder. If you have a custom settings table made and provided it as input, then CaRAS 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 settings table file in its default-setting state.

fastqc1	-q
clumpify	dedupe spany addcount qout=33 fixjunk
bbduk	ktrim=r k=21 mink=8 hdist=2 hdist2=1
trimmomatic	LEADING:20 SLIDINGWINDOW:4:20 TRAILING:20 MINLEN:20
fastqc2	-q
bwa_mem	
samtools_view	-q 20
plotfingerprint	
reads_normalizer	--norm properly_paired
fastqc3	-q
macs2_callpeak	
gem	-Xmx10G --k_min 8 --k_max 12
homer_findPeaks	-region
genrich	-v
seacr	
sicer2	-w 50 -g 100
homer_mergePeaks	
peak_feature_extractor	--filter 4 --top_rank 50 --database biological_process
homer_annotatePeaks	
fold_change_calculator	--normfactor user_value --chip_norm 1 --ctrl_norm 1
homer_findMotifsGenome	-size given -mask
meme_chip	-meme-nmotifs 25

Interpreting CaRAS 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 CaRAS 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?

1 – check fingerprint plot and make sure it looks good

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

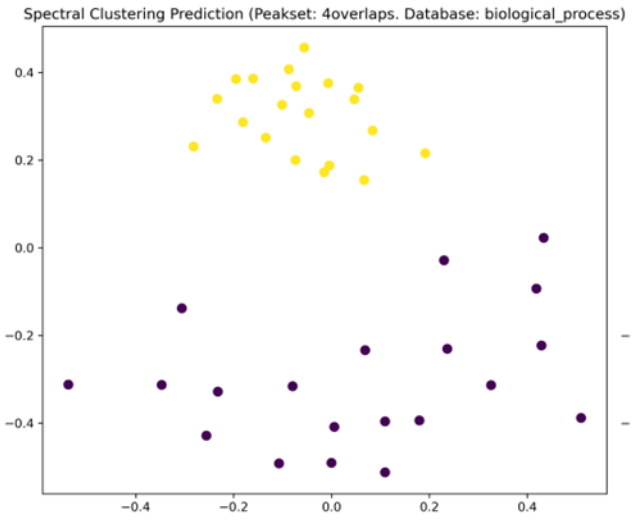
Together points 1 and 2 tell you your experiment worked!

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.tsv"* 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.

6 – What about the single-cell analysis results? What's the difference?

As mentioned before (see section peak feature extraction above), in single-cell analysis, the differences (distance values) between all sample's extracted features profile are to be projected into a 2D plane and clustered based on the projected their 2D positions to one another. After clustering, the peak lists of all samples which are clustered into a same group will be concatenated into one, namely [setname]_[peakset]_[GO/pathway database]_[group#]_ all_peaks _clustered.tsv. To ascertain that the samples are satisfactorily segregated as expected, user can view the MDS-projected 2D scatter plot, namely [setname]_[peakset]_[GO/pathway database]_Term_Ranking _RBO_Distance_MDS_Spectral_Clustering.png.



To the right is an example of MDS-projected 2D plot, showing a good segregation by spectral clustering of 2 different populations with 20 samples each. Besides the fact that spectral clustering works nicely on these samples 2D distribution, this also means that the algorithm of determining the optimal number by maximum eigengap value is accurate (because in this example, the optimal number of clusters was automatically estimated by CaRAS). In cases where the estimation is off (e.g., 3 clusters), users can explicitly state the optimal number of clusters, which is 2 in this case, via --clustnum flag in peak_feature_extractor.py and rerun the script. Beforehand, if the user already knows how many clusters to expect even without looking at the plot above first, users can explicitly state the expected number of clusters via --clustnum in the CaRAS command line.

Upon agreeable clustering results, a group-based gene ontology enrichment analysis is performed which results are stored in folder 22_peaks_processing, reflecting the collective cellular roles of the DNA-binding protein in each distinct population. Later down the workflow, motif enrichment analyses by HOMER and MEME are also performed in a group-based manner, resulting in the enrichment of DNA-binding motifs of the target protein in each distinct population.

Version Update Changelogs

Version 1.0

- **First version** to be released on GitHub.

Manuals and Citations

If you use CaRAS in your analysis, please cite the us and all the following programs

Programs	**References**
CaRAS v1.0	GitHub: https://github.com/JSuryatenggara/CaRAS
Python3 Linux 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 CaRAS. 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 v0.12.1	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 v3.5.2 (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 v2.2.7.1	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 v1.0.3	Guide: https://zanglab.github.io/SICER2/ GitHub: https://github.com/bioinf/SICER2 Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2732366/
SEACR v.1.3	GitHub: https://github.com/FredHutch/SEACR Citation: https://epigeneticsandchromatin.biomedcentral.com/articles/10.1186/s13072-019-0287-4
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 v0.6.1	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/
IDR v2.0.4.2	GitHub: https://github.com/nboley/idr Citation: https://projecteuclid.org/journals/annals-of-applied-statistics/volume-5/issue-3/Measuring-reproducibility-of-high-throughput-experiments/10.1214/11-AOAS466.full
HOMER findMotifsGenome v4.11 (HOMER)	Guide: http://homer.ucsd.edu/homer/ngs/peakMotifs.html Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2898526/
MEME meme-chip V5.0.5 (MEME)	Guide: https://meme-suite.org/meme/doc/meme-chip.html Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2703892/