CUT-**a**nd-**R**UN **A**nalysis **S**uite

Inherently Simple CUT&RUN Analysis

Developed by:

Jeremiah Suryatenggara

Mahmoud Bassal

Table of Contents

Introduction

CaRAS (**C**UT-**a**nd-**R**UN **A**nalysis **S**uite) (https://github.com/JSuryatenggara/CaRAS) was conceived as an all-in-one analysis pipeline specifically tailored and optimized for CUT&RUN/Tag. CaRAS is a unified interface and wrapper tailored for the new generation of ChIP-Seq: CUT&RUN and CUT&Tag analysis (for the sake of conciseness, both techniques will be referred to as simply CnR). CaRAS builds on the same programming framework as ChIP-AP (https://github.com/JSuryatenggara/ChIP-AP), sharing a number of similar methodologies and working characteristics but diverges in that it tailors specific analysis steps specifically for CnR analyses that are yet unpublished for CnR aimed workflows.

CaRAS is the first analysis pipeline and framework which **generates normalized bam (nBAM) files,** which are aligned bam files that are normalized to the spiked-in or leftover DNA, which is critical in CnR analysis. Facilitated by the normalization of aligned reads, ChIP-Seq peak callers were able to be optimized to work properly for CnR peak detection. This leads to the next key feature which is the utilization of **five modified ChIP-Seq peak callers in combination with a CUT&RUN dedicated peak caller (SEACR)**, giving a total of 6 peak callers to establish consensus with. CaRAS is also equipped with **single cell analysis capabilities** (beta), where single cell samples are clustered based on their enriched gene ontology terms (GO) or pathways, and downstream analyses will be performed on each cluster to enable capturing the states / phases / differentiation of the cells.

Just like ChIP-AP, our **C**ut-**a**nd-**R**UN **A**nalysis **S**uite **(CaRAS)** utilizes multiple peak callers, which enables filtering in/out results from every peak caller used in this pipeline (MACS2, GEM, HOMER, Genrich, SEACR, and SICER2) to:

1. Selectively obtain high confidence peaks based on overlaps by different peak callers
2. Avoid the caveats of depending on a single peak caller in datasets with different peak characteristics and signal-to-noise ratios
3. Freely choose the “sweet spot” between the consensus peaks set (intersection of detected peaks from all peak callers) and the total peaks set (the union of all detected peaks), that answers one’s biological questions which may be supported by additional evidence from other experiments. This can be done from the output without re-processing data.

Also, just like ChIP-AP, CaRAS is a fully automated ChIP-seq data processing and analysis pipeline with these general features:

1. For input, it takes unaligned sequencing reads in FASTQ format (extension: .fastq / .fq / .fastq.gz / .fq.gz) or previously aligned reads in BAM format (extension: .bam).
2. CaRAS is capable of processing and analyzing multiple sample replicates.
3. CaRAS is a complete, integrated workflow which performs all analysis steps (QC, cleanup, alignment, peak-calling, pathway analysis) and outputs a final integrated peak set.
4. The output of CaRAS is a detected peaks list, annotated with information on associated gene names, IDs, functions, gene ontology, and related pathways.

System requirements

OS – Linux (Ubuntu-variants tested), MacOS (10.15 or later), Windows 10 (v1903 or later)

CPU – (minimum) Quad-Core Intel/AMD CPU, (recommended) Octa-Core Intel/AMD CPU.

RAM – (minimum) 8Gb, (recommended) 16Gb+

Storage (SSD/HDD) – Installation alone takes ~60Gb. Roughly, an additional 30-100Gb space is required for processing and analyzing samples.

Quick start – For Command line User Only

CaRAS is capable of handling multiple sample replicates in a single bulk analysis run. It is also capable of handling an imbalanced number of sample replicates (i.e., 3 ChIP, 2 Controls). It does so by merging each corresponding sample type following alignment and these merged files will be used for downstream processing, yielding one set of analysis results.

In single-cell analysis mode, multiple samples are considered as unique samples instead of replicates of one another. These samples will later be clustered into several groups based on their enriched GO terms or pathways profile similarity. Each group will be considered as unique cell population and undergo its own downstream processing, yielding at the end of the run multiple sets of analysis results.

For peak calling, peaks are called based on ChIP signal over control.

Example: To run *bulk analysis* of *single-end unaligned reads* with default settings (sample organism human hg38, no reads normalization):

```
caras.py    --analysis bulk \
            --mode single \
            --chipR1 [chip fastq replicate1] [chip fastq replicate2] … \
            --ctrlR1 [control fastq replicate1] [control fastq replicate2] … \
            --genome [path to genome folder] \
            --output [path to output save folder] \
            --setname [dataset name]
```

Example: To run *single-cell analysis* of *paired-end unaligned reads* with default settings (sample organism mm10, reads normalization based on reads mapped to E. coli K12):

```
caras.py    --analysis single_cell \
            --mode paired \
            --chipR1 [chip fastq replicate1, first read] [chip fastq replicate2, first read] … \
            --chipR2 [chip fastq replicate1, second read] [chip fastq replicate2, second read] … \
            --ctrlR1 [control fastq replicate1, first read] [control fastq replicate2, first read] … \
            --ctrlR2 [control fastq replicate1, second read] [control fastq replicate2, second read] … \
            --genome [path to output save folder] \
            --ref mm10 \
            --norm_ref eColiK12 \
            --output [path to output save folder] \
            --setname [dataset name]
```

Example: To run *bulk analysis* of *single/paired-end aligned reads* with default settings:

```
caras.py    --analysis bulk \
            --mode single / paired \
            --chipR1 [chip bam replicate1] [chip bam replicate2] … \
            --ctrlR1 [control bam replicate1] [control bam replicate2] … \
            --genome [path to genome folder] \
            --output [path to output save folder] \
            --setname [dataset name]
```

Installing CaRAS on local machine or laptop

CaRAS will run on Linux (Ubuntu-variants tested).

1 - Before setting up CaRAS, Anaconda 3 ***MUST*** be installed first. Simply search “anaconda 3” in your search engine of choice and download and install the correct setup for your operating system (OS).

2 - CaRAS can be downloaded from our GitHub: https://github.com/JSuryatenggara/CaRAS.

3 - After downloading CaRAS, move the downloaded CaRAS package folder to a directory of your choosing. This will be the installation directory for CaRAS.

4 - To install, run: **caras_install.py** from the command line inside the unpacked folder.

Usage notes and Command Line Flags / Parameters

Required Arguments

--analysis	*bulk / single_cell*	Bulk or single-cell analysis type. Bulk analysis is the typical protocol for NGS-based protein-DNA genome-wide binding study, where every sample typically comes from numerous cells and will be regarded as a biological replicate. As such, bulk analysis will generate one set of results. On the other hand, single-cell analysis is the advanced protocol applicable only for a few select cutting edge experiments, where every sample comes from one single cell and will be regarded as potentially unique until clustered based on enriched GO terms or pathways profile similarity. As such, single-cell analysis will generate multiple sets of results, depending on how many distinct cell populations (clusters) are recognized in all samples (i.e., n clusters = n sets).
--mode	*single / paired*	Single-end or paired-end sequencing reads. If a paired-end run, files will typically be labelled ending in *_R1 and *_R2 before the file extension. If these labels aren't present, then, likely, you have single-ended sequencing data and should select the "single" option.
--genome	*[directory]*	Your genome folder directory. Requires full path, ***not*** relative path. This is the folder where the pre-computed genome alignment and processing files are saved. These genomes can be downloaded from (https://www.dropbox.com/s/ioqd3hwdahh9xon/genomes.zip) or you can compute your own (in our GitHub guides directory, look for "How to Generate Your Own CaRAS Genome Files").
--output	*[directory]*	Your desired output folder. Required full path, ***not*** relative path.
--setname	*[text]*	The prefix to label output and intermediate files (no space allowed). CaRAS will rename all processed data files to have this prefix.

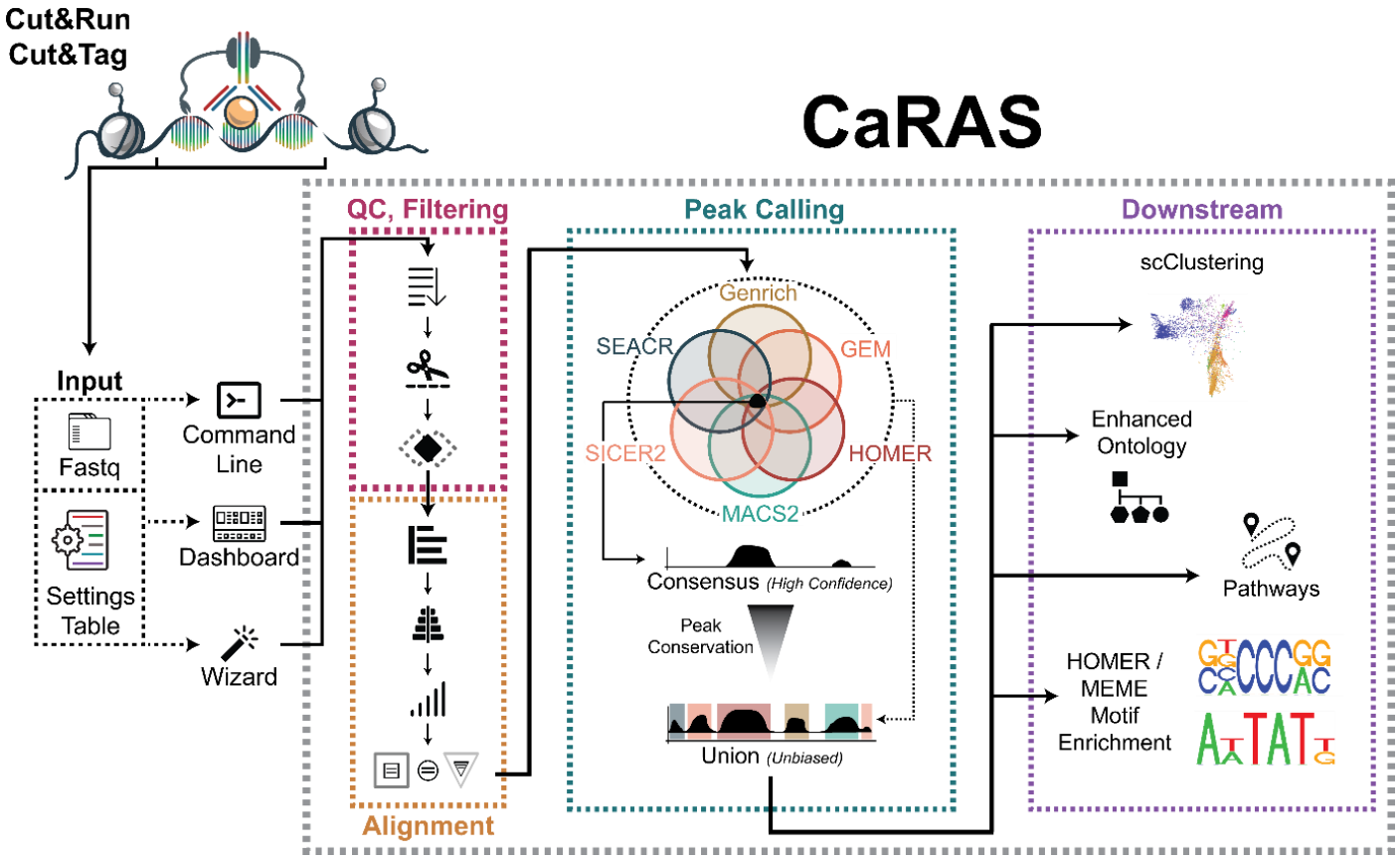
Optional Arguments

--chipR1	*[repl1 repl2 …]*	Your ChIP datasets: ordered by replicate, separated by space. Best to include full path rather than relative paths.
--chipR2	*[repl1 repl2 …]*	**[Paired-end Only]** Your ChIP datasets second read: ordered by replicate, separated by space. Best to include full path rather than relative paths.
--ctrlR1	*[repl1 repl2 …]*	Your control datasets: ordered by replicate, separated by space. Best to include full path rather than relative paths.
--ctrlR2	*[repl1 repl2 …]*	**[Paired-end Only]** Your control datasets second read: ordered by replicate, separated by space. Best to include full path rather than relative paths.

--sample_table	*[file]*	Rather than including the input sample file names and paths on the command line, one can make a sample table containing the same information and input this instead. The sample-table is a 4*n-sized table (with n = number of replicates) containing the absolute paths to each of your ChIP and control replicates (See below for more information regarding this file and its layout). When this option is used, this table will disregard any assigned values to **--chipR1, --chipR2, --ctrlR1,** and **--ctrlR2**.
--custom_setting_table	*[file]*	**! For Advanced Users ONLY!** The settings-table allows you to fine-tune the behaviour of every program as part of CaRAS. *Hence, can be disastrous if you mess up!* If you are unsure even a little about what you're doing, then please stick to default settings – this goes even for bioinformaticians. This .tsv file is a 2*n-sized table (with n = number of replicates) containing custom arguments for every program as part of CaRAS. The default settings table is provided in the genome folder. You can *COPY* this file and make changes as necessary. To use your customs settings table, provide full path to updated .tsv file. See below for more information regarding this file and its layout.
--ref	*hg19 / hg38 / mm9 / mm10 / mm39 / dm6 / sacCer3*	Your sample genome reference build. Default is hg38 (human). The genomes listed to the left are provided and pre-calculated. To add your own custom genome to CaRAS, look for "How to Generate Your Own CaRAS Genome Files" in our GitHub guides directory.
--norm_ref	*eColiK12 / sacCer3*	Your read normalizer organism genome reference build. Choose according to the organism which DNA is spiked-in or carried-over to the samples during the experiment. CaRAS will attempt to align the reads to the chosen genome and use the outcome as the basis for inter-samples read normalization. **Important:** 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. To add your own custom genome to CaRAS, look for "How to Generate Your Own CaRAS Genome Files" in our GitHub guides directory.
--motif	*[file]*	Your predicted/known motif file, in HOMER matrix format (.motif). If provided, once peaks are called, HOMER motif discovery will be run on the total called peak set for this motif.
--fcmerge		This flag will force fold change analysis to be computed based on merged replicates instead of on each replicate separately. This flag will be automatically activated when the number of ChIP samples is unequal to control samples.
--clustnum	*[integer]*	Expected number of clusters to be formed by all single-cell samples. When not provided, CaRAS will attempt to calculate the optimal number of clusters by itself (auto). Only relevant for single-cell analysis and will be ignored in bulk analysis. Use this if you preemptively know or want your cells to be divided into a certain number of groups. CaRAS will then generate the best threshold to divide between these groups based on their mapped 2D distribution based on GO terms or pathways profile similarity

--goann		This flag will instruct to annotate peaks with all relevant GO terms as provided by HOMER.
--pathann		This flag will instruct to annotate peaks with all relevant pathway and interaction enrichment terms as provided by HOMER.
--deltemp		This flag will instruct to delete large intermediary files right after they are not going to be used for further processes (e.g., intermediary fq files). This option will save a significant amount of space by the end of your analysis, so recommended.
--thread	*[integer]*	Maximum number of processes to use. Default is half the maximum available on your system so as to not choke it during the run. If running on a laptop or low-thread count CPU, best to push this up to maximum number threads available -1 – but this **will** significantly slow your laptop if attempting other tasks while CaRAS is running.
--run		Use to immediately run the suite by running the master script. This is the big red button. Use it at your own risk. When not used, the generated master script (MASTER_script.sh) in the output folder can be run manually by user.
--homer_motif	*1 / 2 / 3 / 4 / 5 / 6*	This flag will instruct HOMER (findMotifsGenome.pl) to perform motif enrichment analysis at the end of the pipeline run. The analysis will be performed on selected peaks called by 1/2/3/4/5/6 peak callers. 1 being the same as the union peak set (all called peaks), and 6 being the same as the consensus peak set (intersection of all peak caller sets). Inputting multiple arguments (space-separated) instructs HOMER to perform multiple analysis runs on selected sets.
--meme_motif	*1 / 2 / 3 / 4 / 5 / 6*	This flag will instruct MEME suite (meme-chip) to perform motif enrichment analysis at the end of the pipeline run. The analysis will be performed on selected peaks called by 1/2/3/4/5/6 peak callers. 1 being the same as the union peak set (all called peaks), and 6 being the same as the consensus peak set (intersection of all peak caller sets). Inputting multiple arguments (space-separated) instructs MEME suite to perform multiple analysis runs on selected sets.

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.

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

5. **Raw sequencing reads quality assessment**. Performed by FastQC. Reads quality assessment is performed to check for duplicates, adapter sequences, base call scores, etc. Assessment results are saved as reports for user viewing. If the final results are not as expected, it's worthwhile to go through the multiple QC steps and track the quality of the data as its processed. If the default QC steps aren't cleaning up the data as desired, you may need to modify some parameters to be more / less stringent with cleanup.

   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

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

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

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

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

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 designated target organism reference genome
    - Output : 06_bwa_mem_aligning / [setname]_[chip/ctrl]_rep[#].aligned.bam

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[#].normaligned.bam

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

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

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[#].normalized.bdg
      08_results / [setname]_[chip/ctrl]_merged.normalized.bdg

15. **Generation of aligned reads visualization track**. 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 generated bigwig files are normalized based on the number of reads successfully aligned to the read normalization organism genome. The read 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[#].normalized.bw
      08_results / [setname]_[chip/ctrl]_merged.normalized.bw

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.

Modular script used: **08_results_script.sh**

- Calls : bamCoverage
- Input : 08_results / [setname]_[chip/ctrl]_rep[#].bam
- Process : Normalize each individual bam file
- Output : 08_results / [setname]_[chip/ctrl]_rep[#].normalized.bam

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[#].bam
    - Process : Generate fingerprint plots for all bam files
    - Output : 08_results / fingerprint_plots/[setname]_rep[#]-[#].[png/svg]
      08_results / fingerprint_plots/[setname]_merged.[png/svg]

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[#].bam
      08_results / [setname]_[chip/ctrl]_merged.bam
    - Process : Generate raw reads quality assessment reports
    - Output : 08_results / [setname]_[chip/ctrl]_rep[#]_fastqc.html
      08_results / [setname]_[chip/ctrl]_merged_fastqc.html

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

    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

    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

    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

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

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 200 and 600, respectively)

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 six 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)

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

21. **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

22. **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

23. **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 : peak_feature_extractor.py
    - 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

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

    For average fold enrichment calculation in cases of broad peak type datasets, samtools view simply sums up the number of reads in the whole peak region, then calculates the fold enrichment values, corrected based on 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 (narrow peak only)
  Calculate average fold change (broad peak only)
  Calculate number of peak callers overlaps
  Calculate number of user-provided (via --motif flag) motif instances found
- Output : 22_peaks_processing / [setname]_all_peaks_calculated.tsv
  (Ready to view if user does not wish for gene ontology or pathway annotations)

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
      (Ready to view, if user does not wish for gene ontology or pathway annotations)

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 six 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

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

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.

    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

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_consensus_script.sh**
    24_homer_motif_enrichment_union_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

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_consensus_script.sh**
25_meme_motif_enrichment_union_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

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)	

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

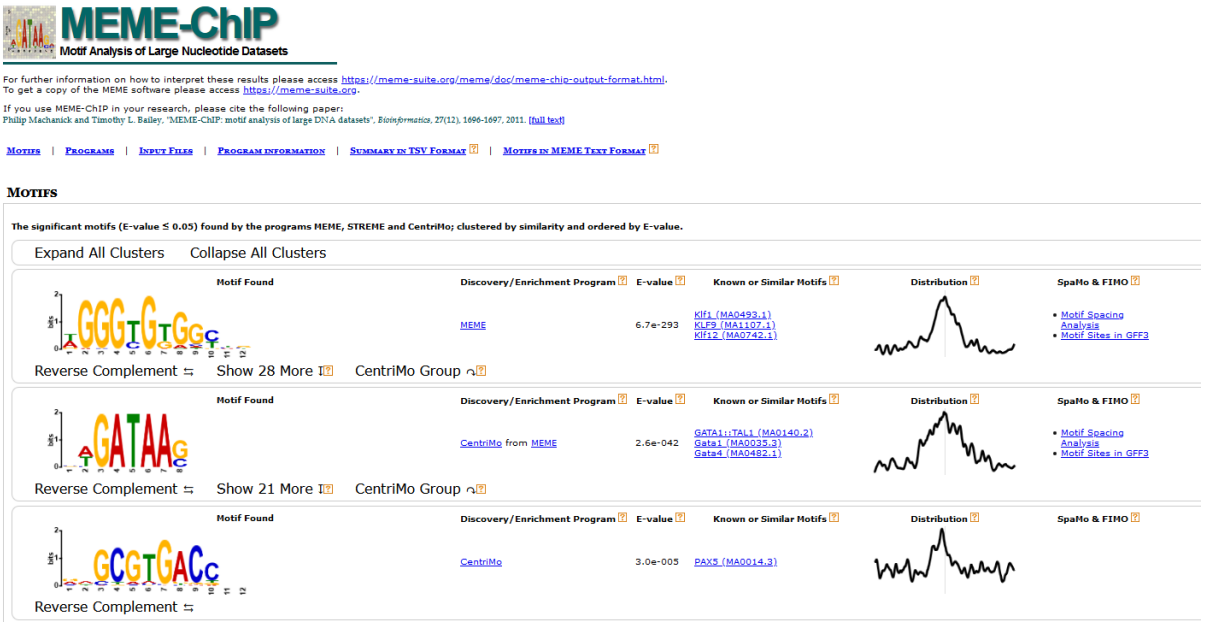
In addition to de novo motif enrichment, homer also performs motif enrichment analysis on the known binding motifs readily available within their database repertoire. The results for this analysis can be viewed in a similar way by opening the file knownResults.html, which contains similar information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

- 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 incorrectly entered (e.g., 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 directory]/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 replicates 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
sicer2	-w 50 -g 100
homer_findPeaks	-region
genrich	-v
seacr	
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

As can see, certain flags and values for some programs have been preset as per our testing and opinions. Below is a listing of the values listed above in the default-settings with an explanation as to why we set these values as they are.

Program	Flag / Argument	Explanation
clumpify	dedupe spany addcount qout=33 fixjunk	These parameters are added since they instruct clumpify to remove optical duplicates (dedupe spany) and count the reads input. Clumpify will automatically optimize compression of output gz files with no additional parameters. qout=33 will make sure clumpify output is in phred33 format, which is required for the subsequent processes. fixjunk

		prompts clumpify to attempt repairs on broken or incompatible sequences to avoid crashes during subsequent processes.
bbduk	ktrim=r k=21 mink=8 hdist=2 hdist2=1	The flag ktrim=r will instruct bbduk to operate in right-trimming mode for matched adaptor kmers (i.e., trim from the right / 3' end of read). The flags k=21 and mink=8 set the full-length k-mer to 21 bases, which can gradually shorten down to 8 bases when scanning read ends. The flags hdist=2 and hdist2=1 instructs bbduk to use a hamming distance of 2 when processing full-length k-mers, and 1 when processing shorter -kmers.
Trimmomatic	LEADING:20 SLIDINGWINDOW:4:20 TRAILING:20 MINLEN:20	These 4 flags and parameters instruct Trimmomatic (T) on how to trim reads. The LEADING:20 flag directs T to remove bases at the 5' start of the read with PHRED scores <20. Likewise, TRAILING:20 instructs T to do the same but for the 3' end of the read. The SLIDINGWINDOW:4:20 parameter instructs T to use a sliding window of 4bp, and to scan the entire read and when the average PHRED score drops below 20, then T will trim the remainder of the read at that point. Finally, the MINLEN:20 flag instructs T to drop all trimmed reads with a length <20bp as we believe shorter reads will likely map more ambiguously to the genome and not be informative.
samtools view	-q 20	This option instructs samtools to remove all aligned reads with a MAPQ score < 20. This is seen as an optional but recommended step to remove reads with less favorable mapping scores. For more information on MAPQ scores, search online as that discussion is beyond the scope of this explanation.
reads_normalizer	--norm properly_paired	This option instructs reads_normalizer.py to only consider reads successfully mapped to the read normalizer organism genome which are properly paired (only relevant in paired-end reads sample).
GEM	-Xmx10G --k_min 8 --k_max 12	As GEM is java jar package, the amount of ram required to run needs to be specified. For this, the -Xmx10G flag is used which allocates 10Gb of ram for GEM to run. This can (and probably) should be increased if your system has the ram to accommodate it, to avoid any potential issues with out of memory errors and crashes. The –k_min and –k_max flags instruct GEM to search for kmers between 8 and 12bp. Depending on your dataset you may need to change this, but for defaults these values will suffice.

Genrich	--adjustp	This flag is ***NOT*** part of Genrich's default behavior. We noted aberrant peak calling with low coverage datasets in our dataset and confirmed this behavior with the developer in private communications. Through our testing, we derived equations that allow us to curtail Genrich's aberrant behavior in such scenarios. This adjustment is performed and derived by us and is not attributed to Genrich and/or its developer(s).
peak_feature_extractor	--filter 4 --top_rank 50 --database biological_process	--filter 4 instructs peak_feature_extractor.py to use homer gene ontology enrichment analysis only on peaks that is called by at least 4 out of 6 peak callers. Then, --top_rank 50 instructs it to calculate differences of the top 50 enriched terms or pathways between the ranked lists of every sample. --database biological_process instructs it to use the ranked terms list from biological process gene ontology database.
FC Calculator	--normfactor uniquely_mapped	This flag instructs the fold-change calculator script to only consider uniquely-mapped reads in the fold-change calculation. While multi-mapped reads might be important to consider in some scenario's, we deemed it more accurate to calculate FC based on uniquely-mapped reads.
HOMER **findMotifsGenome**	-size given -mask	These instruct findMotifsGenome to perform motif enrichment analysis based on the original size of the sequences based on the peak widths, with the genomic repeat sequences masked (discounted from contributing to the motif enrichment).
meme-chip	-meme-nmotifs 25	This instructs meme-chip to find up to 25 top enriched motifs. Any more than rank 25 typically has too low enrichment values to be considered as a true binding event.

Some flags for programs, such as -BAMPE in MACS2, are "hard-coded" into the pipeline and cannot be modified. For this example of -BAMPE in MACS2, this is "hard-coded" because this flag is essential for running peak calling in paired-end datasets. Parameters and flags like this that must be set are "hard-coded" and hidden and cannot be changed unless by choosing the appropriate narrow/broad run modes.

The next pages are a listing of all the "hard-coded" settings hidden in CaRAS's code with a brief explanation of why they are used (full details can be found in the official documentation for each program).

Program Subcommand	Flag	Definition	Explanation
fastqc1	-t	Number of threads	Determined by CaRAS --thread
	-o	Output directory	Locked to *[output directory] / 01_raw_data*
clumpify	in=	Sequencing data input	Locked by CaRAS file naming and pathing
	out=	Sequencing data output	Locked by CaRAS file naming and pathing
	in2=	Sequencing data input	Locked by CaRAS file naming and pathing
	out2=	Sequencing data output	Locked by CaRAS file naming and pathing
bbduk	in=	Sequencing data input	Locked by CaRAS file naming and pathing
	out=	Sequencing data output	Locked by CaRAS file naming and pathing
	in2=	Sequencing data input	Locked by CaRAS file naming and pathing
	out2=	Sequencing data output	Locked by CaRAS file naming and pathing
	ref=	Path to adapter sequences file	Locked to *[path to genome folder] / bbmap / adapters.fa*
trimmomatic	SE	Flag to use single-end reads mode	Determined by CaRAS --mode
	PE	Flag to use paired-end reads mode	Determined by CaRAS --mode
	-threads	Number of threads	Determined by CaRAS --thread
fastqc2	-t	Number of threads	Determined by CaRAS --thread
	-o	Output directory	Locked by CaRAS file naming and pathing
bwa_mem	-t	Number of threads	Determined by CaRAS --thread
Samtools view	-@	Number of threads	Determined by CaRAS --thread
	-h	Flag to include SAM file header	Locked to use -h
	-b	Flag to produce result in BAM format	Locked to use -b
reads_normalizer	--thread	Number of threads	Determined by CaRAS --thread
	--bam	Sequencing data input/output	Locked by CaRAS file naming and pathing
	--make_bdg	Flag to generate normalized bedgraph	Locked by CaRAS file naming and pathing

	--make_bw	Flag to generate normalized bigwig	Locked by CaRAS file naming and pathing
	--make_bam	Flag to generate normalized bam	Locked by CaRAS file naming and pathing
	--log_dir	Log output directory	Locked by CaRAS file naming and pathing
plotfingerprint	-p	Number of threads	Determined by CaRAS --thread
	-b	Sequencing data input/output	Locked by CaRAS file naming and pathing
	-l	Sample input/output labels	Locked by CaRAS file naming and pathing
	-o	Output directory	Locked by CaRAS file naming and pathing
fastqc3	-t	Number of threads	Determined by CaRAS --thread
	-o	Output directory	Locked by CaRAS file naming and pathing
macs2 callpeak	-f	Input format	Locked to SAM format
	-t	Sequencing data input/output	Locked by CaRAS file naming and pathing
	-c	Sequencing data input/output	Locked by CaRAS file naming and pathing
	-g	Sample genome size	Determined by CaRAS internal genome size data
	--name	Output prefix	Locked by CaRAS file naming and pathing
	--outdir	Output directory	Locked by CaRAS file naming and pathing
	--broad	Flag to use broad peak calling mode	Determined by CaRAS --peak
gem	--expt	Sequencing data input/output	Locked by CaRAS file naming and pathing
	--ctrl	Sequencing data input/output	Locked by CaRAS file naming and pathing
	--t	Number of threads	Determined by CaRAS --thread
	--d	Path to file containing sample reads distribution	Locked to *[path to genome folder] / GEM / Read_Distribution_default.txt*
	--g	Path to file containing sample genome chromosome sizes	Locked to *[path to genome folder] / GEM / [sample genome reference build].chrom.sizes*
	--genome	Path to file containing sample whole genome FASTA sequence	Locked to *[path to genome folder] / GEM / [sample genome reference build]_Chr_FASTA*
	--s	Sample genome size	Determined by CaRAS internal genome size data
	--f	Input format	Locked to BAM format
	--out	Output directory	Locked by CaRAS file naming and pathing

sicer2	-cpu	Number of threads	Determined by CaRAS --thread
	-t	Sequencing data input/output	Locked by CaRAS file naming and pathing
	-c	Sequencing data input/output	Locked by CaRAS file naming and pathing
	-s	Sample genome reference build	Determined by CaRAS --ref
HOMER findPeaks	-style	Sample peak type	Determined by CaRAS --peak
	-gsize	Sample genome size	Determined by CaRAS internal genome size data
	-o	Output file name	Locked by CaRAS file naming and pathing
	-i	Input directory	Locked by CaRAS file naming and pathing
genrich	--mode	Sample read mode	Determined by CaRAS --mode
	-y	Flag to keep unpaired reads	Determined by CaRAS --mode
	-t	Sequencing data input/output	Locked by CaRAS file naming and pathing
	-o	Output file name	Locked by CaRAS file naming and pathing
	-c	Sequencing data input/output	Locked by CaRAS file naming and pathing
HOMER mergePeaks	-prefix	Output merged peaks files prefix	Locked to [setname]_merged_peaks
	-matrix	Output data matrix files prefix	Locked to matrix; useful but not mandatory for subsequent processes for now
	-venn	Output venn diagram file name	Locked to venn.txt; useful but not mandatory for subsequent processes for now
peak_feature_extractor	--thread	Number of threads	Determined by CaRAS --thread
	--input_tsv	Merged peak lists from all callers	Locked by CaRAS file naming and pathing
	--output	Output directory	Locked by CaRAS file naming and pathing
	--setname	Dataset name	Locked by CaRAS file naming and pathing
	--ref	Sample organism genome reference	Determined by CaRAS --ref
	--repnum	Number of samples	Determined by the number single-cell samples
HOMER annotatePeaks	-m	Path to file containing known motifs (.motif)	Determined by CaRAS --motif
	-nmotifs	Flag to report the number of motifs per peak	Mandatory for subsequent processes

	-matrix	Output data matrix files prefix	Locked to *[output directory] / 22_peaks_processing*
	-go	Output directory for HOMER genome ontology analyses	Mandatory for subsequent processes
fold_change_calculator	--thread	Number of threads	Determined by CaRAS –thread
	--input_tsv	Path to input peak list	Locked to *[output directory] / 22_peaks_processing / [setname]_all_peaks_annotated.tsv*
	--output_tsv	Path to output peak list	Locked to *[output directory] / 22_peaks_processing / [setname]_all_peaks_calculated.tsv*
	--chip_bam	Sequencing data input/output	Locked by CaRAS file naming and pathing
	--ctrl_bam	Sequencing data input/output	Locked by CaRAS file naming and pathing
	--peak	Sample peak type	Locked to ‘narrow’ for CUT&RUN and CUT&Tag applications
HOMER findMotifsGenome	-p	Number of threads	Determined by CaRAS –thread
	-dumpFasta	Flag to include fasta files of the analyzed sequences in the output	Locked on; useful but not mandatory for subsequent processes for now
MEME meme-chip	-oc	Path to output folder	Locked by CaRAS file naming and pathing
	-neg	Path to background sequences fasta	Locked by CaRAS file naming and pathing

Effective Genome Sizes

All effective genome sizes were calculated based on the number of non-N characters in the genome's FASTA file:

hg19_effective_genome_size = '2861327131'

hg38_effective_genome_size = '2937639113'

mm9_effective_genome_size = '2558509480'

mm10_effective_genome_size = '2647521431'

mm39_effective_genome_size = '2649921816'

dm6_effective_genome_size = '137057575'

sacCer3_effective_genome_size = '12071326'

Aligned reads (BAM) normalization formula

For normalization purpose, it has been shown that simply adding the same low amount of spiked-in DNA fragments from a different species to each sample would suffice as a valid basis of normalization and allows accurate quantification of protein occupancy (Skene & Henikoff, 2017). In a later publication on an improved CUT&RUN method, it was shown that the bacterial DNA fragments originating from the purification of pA-MNase complex that was carried over throughout the procedures would suffice to serve the same purpose (Meers, Bryson, Henikoff, & Henikoff, 2019). To calculate the normalization factor for each sample, the reads from the sequencing run need to be realigned onto the spiked-in or the carry-over DNA's respective genomes and have the number of successfully aligned reads counted. Once the number of aligned reads is determined, it is therefore possible to perform normalization based on the number of these aligned non-sample reads as they ought to be equal across all samples and replicates. CARAS performs this alignment with subsequent normalization with formula as described below:

$$batch\ multiplier = \frac{1}{\frac{highest\ number\ of\ reference\ reads}{2nd\ highest\ number\ of\ reference\ reads} - 1}$$

$$normalization\ factor = \frac{batch\ multiplier}{number\ of\ reads}$$

$$normalized\ number\ of\ reads = original\ number\ of\ reads \times normalization\ factor$$

Given multiple samples involved in a single batch of analysis, the batch multiplier is determined by the largest and the second largest number of mapped spiked-in or carry-over DNA (reference) reads. The normalization factor for each sample is then calculated by dividing the batch multiplier with the corresponding sample's number of reads. Reads normalization will then be carried out by simply multiplying the original number of reads in each sample with its corresponding normalization factor.

Once the normalization factors are calculated per sample/replicate, it is possible to perform the normalization. To date in the literature, the method for normalizing read counts is to quantify the number of reads per base position throughout the genome in a bedgraph file,

and then multiply the number of quantified reads by the normalization factor, i.e., convert aligned BAM file to bedgraph file. The issue though, is that ChIP-Seq peak callers do not call peaks from normalized begraphs, they instead require aligned BAM files. This raised the issue in that to date, there is no published method for normalizing aligned BAM files. To circumvent this issue and to correctly process CnR datasets using traditional ChIP-Seq peak callers, we developed a novel approach for generating normalized BAM (nBAM) files.

The concept of normalization is relatively straight forward, one simply needs to multiply up the number of reads present by the pre-determined normalization factor. While this is a simple operation when working with bedgraphs, when working with aligned reads (.bam) files, this requires “multiplying” each read by the appropriate normalization factor. In other words, if the normalization factor is 5, and only one read is at genomic position X, then we effectively need to have 5 reads at position X instead of 1. So, our approach multiplies each read in the BAM file by the required normalization factor.

In certain datasets with low read depth (e.g., single cell sequencing datasets), the number of mapped spiked-in or carry-over DNA often fluctuate excessively between samples, resulting in a too large of a normalization factor. Due to our direct multiplication of read instances in aligned reads (.bam) files, very large normalization factor may generate a massively oversized files, which will take too much processing resource and time while providing little to no benefit for the analysis quality. Therefore, to avoid this, we applied an upper limit of 10 to the batch multiplier.

Additionally, this approach requires caution in its implementation as in that duplicate reads will appear as “PCR duplicate artefacts” to processing tools. As such, to ensure satisfactory normalization behaviour, each peak caller must be tuned to ignore “PCR duplicates” when processing nBAMs so as to not reverse the normalization of read presence.

ChIP vs Control fold change calculation

ChIP weighted peak center coordinate is determined by the median coordinate of all reads in the peak region. Fold change was then calculated as read depth in ChIP sample divided by non-zero read depth of control sample at the weighted peak center coordinate.

To correct for read depth discrepancies due to imbalanced number of reads between ChIP and control samples, CaRAS uses one of three available normalization factors depending on the command line flag specified:

- Based on only uniquely mapped reads in ChIP vs control bam files (filtered using samtools view -F256 and counted using samtools view -c)
- Based on only successfully mapped reads in ChIP vs control bam files (filtered using samtools view -F4 and counted using samtools view -c)
- (Default) Based on user-determined value.

Advanced users may choose to change this by changing the argument field for fold_change_calculator in the settings table as follows:

- --normfactor uniquely_mapped
    - Based on uniquely mapped reads
- --normfactor mapped
    - Based on all mapped reads
- --normfactor user_value --chip_norm [x] --ctrl_norm [y]

- change to user-determined normalization factor, where x / y is the user-intended ratio of the number of ChIP reads to the number of control reads

The feature for read depth normalization (based on total number of reads) before fold change value is calculated was added to compensate for read depth discrepancy between paired ChIP and control samples with same replicate number. This feature was mainly developed for ChIP-AP, where read normalization was not carried out prior, and thus the feature was needed to estimate users a more accurate fold change value of each peak. In CaRAS, read normalization was already performed across all ChIP and control samples (altogether, not separately) directly after read alignment to target organism genome. This normalization method (based on the number of reads successfully mapped to the read normalizer organism genome) is regarded as the standard, and more proper normalization method for CUT&RUN and CUT&Tag. Therefore, CaRAS by default will not perform any read further read depth normalization prior to fold change calculation, as reflected in the default settings table argument for fold_change_calculator: "--normfactor user_value --chip_norm 1 --ctrl_norm 1". Users may still opt to perform read depth normalization based on total number of reads prior to fold change calculation by modifying the settings table.

Genrich p-value threshold adjustment formula

Based on our testing, Genrich tends to misbehave when processing datasets with low read depth. It starts to call low enrichment regions as peaks, leaving users with an impossibly large number of called peaks. We contacted the developer of Genrich and confirmed such behavior with them. Through our testing, we derived equations that allow us to curtail Genrich's aberrant behavior in such scenarios. This adjustment is performed and derived by us and is not attributed to Genrich and/or its developer(s).

To avoid the afore mentioned peak calling depth issue, we curtail such behavior by setting up an auto-adjusting p value threshold that responsively raises the default limit of peak's minimum p value as the read depth gets lower using following equations Q1, Q2, Q3.

Q1

$$genrich\_negative\_log\_p\_threshold = 2^{\lfloor((-0.016 * mapped\_read\_count\_average) + 0.5) * \log_2(mapped\_read\_count\_average)\rfloor}$$

Q2

$$adjusted\_genrich\_negative\_log\_p\_threshold = 0.5 * genrich\_negative\_\log\_p\_threshold$$

Q3

$$genrich\_p\_threshold = 10^{-adjusted\_genrich\_neg\_\log\_p\_threshold}$$

Irreproducibility Rate (IDR) Calculation

The irreproducible discovery rate (IDR) calculations for all peaks are integrated into a ChIP AP run. As the IDR suite calculate peak reproducibility rate between two replicates (i.e., peak sets) only, we chose as "replicates" each individual peak caller set and the union peak set. The output full peak list (union peak set) is ranked by number of detecting peak callers and fold change of sample signal over input, whereas the individual peak caller sets are ranked by peak score (or fold change if peak score is not available).

The reproducibility of all peaks in the output full peak list are calculated based on their detectability by different individual peak callers and ranked accordingly. These six -log IDR values are copied into the output full peak list, providing six -log IDR values for every peak, which are then summed and converted into a final IDR value.

$$negLog_{10}IDR = negLog_{10}IDR\_MACS2\_vs\_union + negLog_{10}IDR\_GEM\_vs\_union + negLog_{10}IDR\_HOMER\_vs\_union + negLog_{10}IDR\_Genrich\_vs\_union + negLog_{10}IDR\_SEACR\_vs\_union + negLog_{10}IDR\_SICER2\_vs\_union$$

As reproducibility is not significantly relevant for single-cell analysis with its duplicitous number of cell samples, on top of the time-consuming process were this step is to be applied to every single sample, the IDR calculation step is only applicable to bulk analysis.

Peak Feature Extraction

With CaRAS, every peak is given a gene annotation based on the closest gene to its location in the genome. Since every gene would have one or more GO terms or pathways associated to it, enriched terms or pathways can be derived from a list of called peaks, inferring the regulatory effects of the target DNA-binding protein on every single cell sample in the experiment. Therefore, a ranked list of enriched terms or pathways can be used as a profile to compare sample to sample and group them based on how similar their enriched GO terms or pathways are. With this, one can ideally segregate multiple groups of diverse single cells based on their cell types, activation states, or differentiation phases. Each of these groups are then regarded as one entity and should yield a set of results which represents every single cell within that group. Ultimately, CaRAS will follow-up with regular downstream analyses on every group, rather than on every single cell.

To go into the details of how the samples are grouped based on extracted features, CaRAS is employing the methods of rank-biased overlap (RBO) to calculate the distances between each sample's features (ranked list of enriched terms or pathways) to one another. From these pair-wise distances, an *n*n* distance matrix is generated, where *n* is equal to the number of analyzed samples. From here, multidimensional scaling (MDS) method is applied to the distance matrix, transforming the inter-sample distances into a 2D plane where the clusters can later be discernably observed. To determine the optimal number of clusters, CaRAS calculates eigenvalues and their associated eigen vectors from the transformed distance matrix. This is then followed by, with eigengap heuristic, the identification of maximum gap value which corresponds to the number of clusters. Unless provided a pre-determined expected number of resulting clusters, CaRAS will then use the number of clusters with the maximum eigengap in segregating the samples during spectral clustering. Lastly, the spectral

clustering is carried out on based on the transformed distance matrix, resulting in grouped samples based on their extracted features, which are the enriched GO terms and pathways.

By default, as recommended in default_settings_table.tsv, CaRAS will use HOMER to perform the gene ontology and pathway analysis only on the subset of peaks which at least called by 4 peak callers. This is done to increase the accuracy of the ranked GO terms or pathways list by taking only the sufficiently confident peaks. CaRAS will also by default only take the top 50 GO terms or pathways for profiling each single cell sample and compare it to one another. As the ranked list profile similarity is done by the RBO (rank-biased overlap) method, this is done with a rationale of removing poorly enriched GO terms or pathways, which has a high proclivity towards randomness in ranked order, and thus will negatively affect the estimation accuracy of profile similarity between each single cell sample. Lastly, CaRAS will by default choose to extract the biological_process GO terms ranked list as profile to represent each single cell sample. Within our limited number of acquired and publicly available datasets, these parameters above are the combination which separate different groups of cells with the clearest segregation and greatest accuracy.

Interpreting CaRAS Output

[Some parts adapted from ChIP-AP manual (https://github.com/JSuryatenggara/ChIP-AP)]. 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!

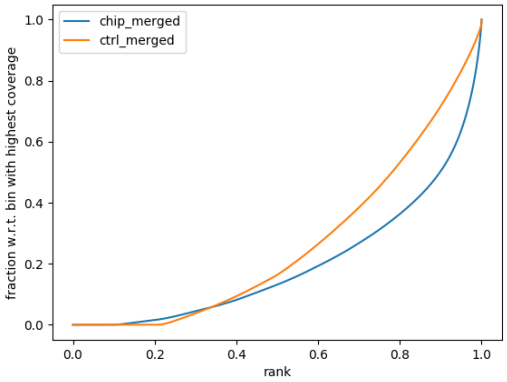
Did my analysis work?

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) –

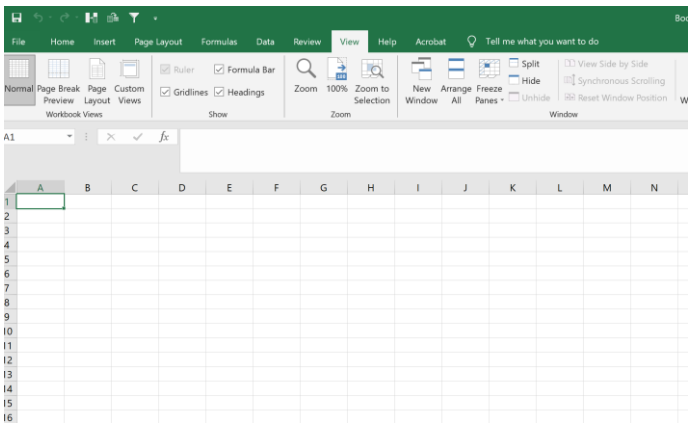
MACS2	SICER2	HOMER	Genrich	Total	Name
			X	103	Genrich
		X		2151	HOMER
		X	X	12	HOMER|Genrich
	X			14499	SICER2
	X		X	328	SICER2|Genrich
	X	X		10346	SICER2|HOMER
	X	X	X	687	SICER2|HOMER|Genrich
X				522	MACS2
X			X	606	MACS2|Genrich
X		X		78	MACS2|HOMER
X		X	X	44	MACS2|HOMER|Genrich
X	X			1115	MACS2|SICER2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

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.

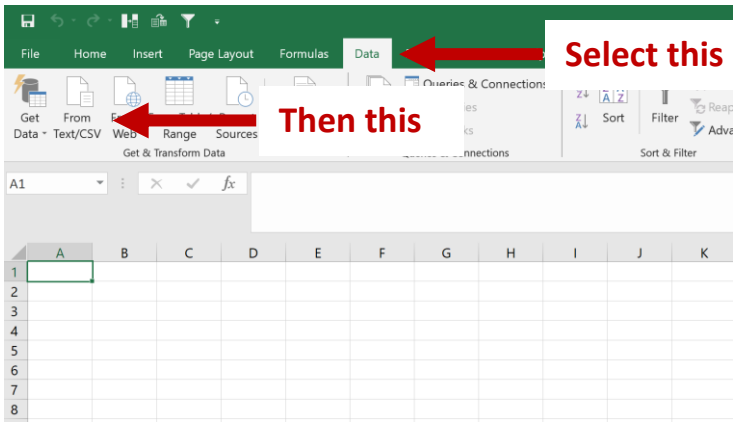
3 – What results files do I look at exactly?

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

Open a new blank workbook in excel



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



In the dialog box that opens up, find and open the peaks files "xxxx_all_peaks_calculated.tsv." Follow all the prompts and keep pressing "Next" / "Proceed" till the end and the file opens. Opening the peak file this way circumvents an issue that Excel constantly makes which is it will interpret some gene names such as OCT1 as a date when it's not. So, by following the afore mentioned steps, excel will not do this stupid conversion and instead, when you save the file as an xlsx, it will ensure that

this issue doesn't happen (seen it in sooooo many publications it's not funny – just import data this way please people?)

From this file, you can view all the results and data for you analysis. Refer to Interpreting 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

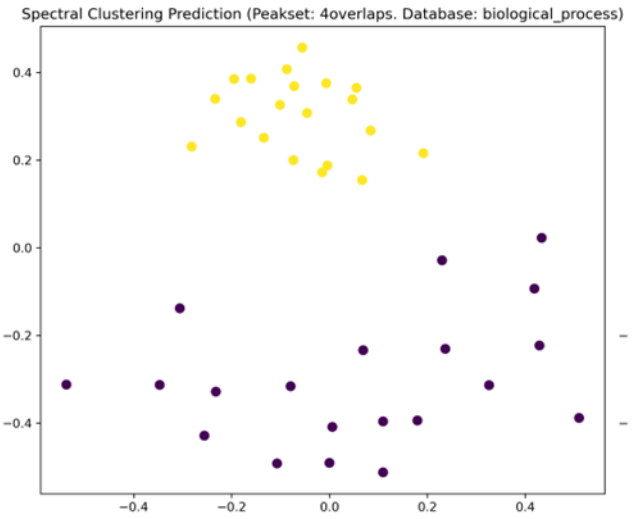
Together points 1 and 2 tell you your experiment worked!

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/