

# ATAC-seq and RNA-seq data processing and analysis for Cazet and Juliano (2021)

---

The code in this repository has been divided into two sections based on the computing system for which the individual scripts were written. Scripts in the `cluster` folder were written to be executed on a remotely accessed computing cluster running Ubuntu that uses the slurm workload manager. Scripts in the `local` folder were written to be run on a Mac personal computer running macOS Catalina (10.15). `cluster` scripts perform raw read processing, mapping, and peak calling, while `local` scripts are used to identify significant differentials in read counts and generate plots.

A version of this repository is also available on [GitHub](#).

Descriptions for the files associated with this repository can be found in the `fileDescriptions.txt` file.

The file `fullRepo.zip` contains all the files associated with this repository organized within the correct directory structure.

## Publicly available resources

The following files are required and can be downloaded from other sources (links provided):

[the Hydra 2.0 genome](#) - required in both the `cluster/resources` and the `local/resources` folders.

[the Hydra 2.0 genome protein models](#) - required in the `local/wnt_Survey` folder. **File modification required:** The fasta headers need to be modified to consist only of the gene model ID. This can be done with the following command: `sed 's/^\.*\.g/>g/g' hydra2.0_genemodels.aa > hydra.augustus.fastp`

[the Hydra 2.0 genome gene models](#) - required in both the `cluster/resources` and `local/wnt_Survey` folders. **File modification required:** The fasta headers need to be modified to consist only of the gene model ID. This can be done with the following command: `sed 's/^\.*\.g/>g/g' hydra2.0_genemodels.nt > Dovetail_mRNAs.fa`

[refseq proteins for \*Nematostella vectensis\*](#) - required in the `local/wnt_Survey` folder.

[refseq proteins for \*Exaiptasia pallida\*](#) - required in the `local/wnt_Survey` folder.

[the swissprot database](#) - required in the `local/wnt_Survey` folder. **File modification required:** The analysis requires a file derived from the swissprot fasta file that contains only human sequences (`SP.human.fasta`). To generate that file, execute the following code in the directory containing the `uniprot_sprot.fasta` file:

```
sed -i 's/ .*//g' uniprot_sprot.fasta

awk \
'/^[>;]/ { if (seq) { print seq }; seq=""; print } /^[>;]/ { seq = seq $0 } END {
print seq }' \
uniprot_sprot.fasta > uniprot_sprot.fasta.tmp

grep -A 1 "HUMAN" uniprot_sprot.fasta.tmp | grep -v -- "^--$" > SP.human.fasta

rm uniprot_sprot.fasta.tmp
```

[picard tools v 2.17.8](#) - required in the `cluster/resources` folder.

[trimmomatic v 0.36](#) - required in the `cluster/resources` folder.

[the HOMER tool suite](#) - needs to be installed in the user's home directory.

[ucsc bedclip](#) - required in the `cluster/resources` folder

### Resources hosted on Dryad

These files were generated specifically for this analysis and are available at the version of this repository hosted on dryad. These files will also be provided by the authors upon request.

`BSgenome.Hymag.NIH.Hymag2_2.0.tar.gz` - Source code that allows for the installation of an R package containing the *Hydra* 2.0 genome reference. This package needs to be installed in R on the computer running the `local` analysis. To install the package, execute the following in R:

```
install.packages("path/to/BSgenome.Hymag.NIH.Hymag2_2.0.tar.gz", repos = NULL, type
="source")
```

`expanded_dovetail_SP_annot.csv` - Table that contains the best swissprot blast hit for each *Hydra* 2.0 genome gene model along with additional information associated with each swissprot hit. This file is required in the `local/resources` folder.

`Hydra_Seurat_whole_Genome_updated.rds` - R binary file containing a Seurat object of the *Hydra* single cell sequencing atlas data mapped to the *Hydra* 2.0 genome gene models. This file is required in the `local/resources` folder.

`hydra.augustus.gtf` - Genome coordinates that map the *Hydra* 2.0 genome gene models to the genome reference sequence. This file is required in the `local/resources` folder.

`Dovetail_Interproscan.csv` - Table that contains information on the domain composition of the *Hydra* 2.0 genome protein models. The table was generated using interProScan using default settings. Used by the `chromVar_Untreated_Analysis.R` script. This file is required in the `local/resources` folder.

# Cluster Analysis

---

## Setup

Before beginning the analysis, first download the 'cluster' folder from this repository. Raw data from this study as well as from Wenger et al. (2019) need to be downloaded to the `cluster` directory as gzipped fastq files. Raw data for this study can be found at the accession GSE152994. The accession numbers for the relevant data from Wenger et al. can be found in `cluster/resources/wenger_Accessions.txt`. Note that code within this section is written to be executed from within the `cluster` directory.

The analysis requires that the user have [bowtie2](#), [rsem](#), [samtools](#), and [bedtools](#), and [Python3](#) in their path.

## File naming conventions

The pipelines provided in this repository are designed to process gzipped fastq files whose names follow a specific structure. Names should be structured as follows:

`prefix_seqMethod_(Read#/Lane#).fastq.gz`

- `prefix` refers to the unique sample identifier. The prefix is structured: `time (Treatment/Source) Structure Rep`. Here are some examples:
  - The prefix `0iH1` refers to the first biological replicate of iCRT14-treated head regenerating tissue at 0 hours post-amputation
  - The prefix `3F2` refers to the second biological replicate of untreated foot regenerating tissue at 3 hours post-amputation.
  - The prefix `16wF1` refers to the first biological replicate of foot regenerating tissue at 16 hours post-amputation from the Wenger et al. data set.
- `seqMethod` refers to the library preparation methodology. This can either be `ATAC` for ATAC-seq or `RNA` for RNA-seq.
- `(Read#/Lane#)` is an optional field. `Read#` refers to the read number from paired-end sequencing data. `R1` is the forward read and `R2` is the reverse read. `Lane#` is used for samples that were split over multiple sequencing lanes. `L3` would refer to data generated on lane 3 of the sequencing machine.

Some examples of files named using this scheme would be:

- `12H1_ATAC_R1.fastq.gz` - This file contains the forward reads from a paired-end ATAC-seq library prepared from the first biological replicate of head regenerating tissue at 12 hours post-amputation.
- `8iF2_RNA_L3.fastq.gz` - This file contains reads from an RNA-seq library that was sequenced on Lane 3 and was prepared from the second biological replicate of iCRT14-treated foot regenerating tissue at eight hours post amputation.

Data files generated by this study will already be named using this scheme. Data from Wenger et al., need to be renamed to fit the specified format to be processed properly.

## Installing python packages

The `cluster` analysis requires a number of python packages (listed in the `cluster/resources/python_requirements.txt` file). To create a virtual environment in which to download the required packages and run the analysis execute the following commands:

```
python3 -m venv resources/venv
source resources/venv/bin/activate
pip -r resources/python_requirements.txt
deactivate
```

## Indexing reference sequences

Read mapping requires indexing the provided fasta files using either bowtie2 or rsem. If your computing cluster uses slurm, this can be done by executing the following command:

```
sbatch -p [partition name] resources/slurm_prepare_references.sh
```

On other linux systems, you can execute:

```
cd resources
./prep_references.sh
```

As written, the script requires 8 threads.

## RNA-seq data processing

The RNA read mapping pipeline first removes index sequences and low quality basecalls using trimmomatic, and then maps reads and quantifies transcript read counts using rsem. It will also generate fastqc reports from both the raw and trimmomatic-filtered data.

If your computing cluster uses slurm, you can execute:

```
sbatch -p [partition name] resources/slurm_RNA_pipeline_run.sh
```

For a system without slurm, you can run the pipeline on each individual sample using the `RNA_Mapping_Pipeline.sh` script like so:

```
./resources/RNA_Mapping_Pipeline.sh [insert prefix here]
```

For example, to process the first biological replicate of head regeneration at 12 hpa, you would execute:

```
./resources/RNA_Mapping_Pipeline.sh 12H1
```

As written, the script requires 16 threads.

After the pipeline has finished running for all samples, read count matrices need to be calculated for the data generated in this study and for the data from Wenger et al. If your cluster uses slurm you can execute:

```
sbatch -p [partition name] resources/slurm_generate_RNA_Matrix.sh
```

Alternatively, you execute the code within your terminal session by running:

```
./resources/generate_RNA_Matrix.sh
```

This final script will generate the read count matrices `RNA.counts.matrix` and `wRNA.counts.matrix`, which are used for the downstream differential gene expression analyses.

## ATAC-seq data processing

The ATAC-seq processing pipeline has three steps: 1) filtering reads and mapping them to the genome, 2) identifying biologically reproducible peaks within each treatment group, and 3) pooling peaklists to generate a consensus peakset.

### Generating mapped read files

The finalized set of reads used for peak calling and differential accessibility analysis should contain unique, unambiguously mapped, and non-mitochondrial reads. The pipeline filters adapter sequences and low quality basecalls using trimmomatic. It then maps the trimmed reads to both the whole genome as well as just the mitochondrial genome using bowtie2. Because of the highly fragmented nature of the current *Hydra* genome assembly, the cleanest way to remove mitochondrial reads is to filter all reads that mapped to the mitochondrial reference from the whole genome-mapped BAM file. We do this using the `FilterSamReads` function from Picard Tools. Improperly mapped read pairs and reads mapped with low confidence ( $\text{MAPQ} < 3$ ) are removed using samtools. Finally, PCR duplicates are marked and removed using the Picard Tools `MarkDuplicates` function and samtools. This pipeline will also generate fastqc reports before and after read trimming and will output statistics on PCR bottlenecking metrics.

To generate filtered BAM files for all replicates on a slurm-based cluster, execute the following command:

```
$ sbatch -p [partition name] resources/slurm_ATAC_Mapping_Pipeline.sh
```

Alternatively, you can run the pipeline on each individual sample by running:

```
./resources/ATAC_Mapping_Pipeline.sh [insert prefix here]
```

The script uses 16 threads and 50GB of memory.

### Peak calling

After read mapping and processing, biologically reproducible peaks need to be identified. Biological reproducibility is determined using the irreproducible discovery rate (IDR) framework established by the ENCODE consortium. We designate a peak as biologically reproducible if it passes a 0.1 IDR score threshold in at least three pairwise comparisons within one treatment group.

The peak calling script will also write an IDR report containing quality control metrics established by the ENCODE consortium. To calculate these metrics, the script produces pseudo-replicates, which are generated by pooling reads and then randomly distributing them across a designated number of replicates. This allows us to create replicates that emulate perfect reproducibility within a treatment group. We use these pseudo-replicates to determine the ratio between the number of reproducible peaks in a dataset that simulates perfect reproducibility and the real number of reproducible peaks found using true biological replicates. This metric is called the rescue ratio.

The script also generates self-pseudoreplicates by randomly splitting the reads from a single biological replicate into two files. The number of peaks passing the IDR threshold from a pairwise comparison of self-pseudo-replicates can then be compared across a treatment group. Ideally, IDR on self-pseudo-replicates should give peak counts that differ by less than 2 fold when comparing biological replicates within a single treatment group. The ratio comparing the number of peaks for self-pseudo-replicates across a treatment group is called the self-consistency ratio. Values for both the rescue ratio and the self-consistency ratio should be less than 2.

Finally, the peak calling pipeline will calculate the transcription start site (TSS) enrichment ratio, which measures the ratio in average read density near TSS relative to regions +/- 1000bp from TSS. A high TSS ratio indicates a strong signal to noise ratio.

Within the script, pooling and subsetting is done using samtools. Peak calling is done using MACS2. IDR analysis is done using the python idr package. TSS enrichment is calculated using DeepTools and R.

To run the peak calling pipeline on all treatment groups using slurm:

```
sbatch -p [partition name] resources/slurm_ATAC_Peak_Calling.sh
```

Alternatively, you can run the pipeline on each individual treatment group by running:

```
$ ./resources/ATAC_Peak_Pipeline.sh [treatment group prefix]
```

## Generating a consensus peakset

Finally, peaks that were identified as biologically reproducible in at least one treatment group are pooled. Going forward, we will be asking two distinct questions: how does injury affect chromatin accessibility, and how does iCRT14 affect the injury response at the level of chromatin accessibility. To address the former, we want to use a consensus peakset that only uses untreated samples so as to avoid including peaks that are only affected by iCRT14 and not by injury. To address the latter, we will need a consensus peakset generated from all treatment groups.

To generate consensus peaksets using slurm:

```
sbatch -p [partition name] resources/slurm_generate_ATAC_peak_consensus.sh
```

Alternatively, on a non-slurm system you can run:

```
$ ./resources/generate_consensus.sh
```

## Local Analysis

### Setup

The following section describes the `local` analysis, which was designed to be run on a personal computer running MacOS. In addition to the files included in the `local` folder of this repository, the code also requires a number of files produced by the `cluster` portion of the analysis.

For the RNA-seq analyses, add the `RNA.counts.matrix` and `wRNA.counts.matrix` files to the `local/resources` folder.

For the ATAC-seq analyses, add all finalized BAM files (named `[prefix]_final_shift.bam`) to the `/local/resources/Bams` folder, add all bigwig files (named `[prefix]_final_shift.bw`) to the `/local/resources/bigwigs` folder, and add the consensus peaksets (`untreated_consensus.bed` and `full_consensus.bed`) to the `/local/resources` folder.

Finally, we will use the uropa python package to annotate peaks based on proximity to genome gene models, so we will need to set up a virtual environment to be used later in the analysis. We will also need to install the pyfasta package to pull sequences from reference fasta files.

```
python3 -m venv resources/venv
source resources/venv/bin/activate
pip install uropa
pip install pyfasta
deactivate
```

## Identifying significant differentials and generating plots

Before we can identify differentially accessible regions of chromatin, we first need to quantify read counts in peaks across all samples. The `ATAC_Read_Counts.R` script uses the DiffBind package to calculate normalized read counts across all peaks in both the untreated and full consensus peak sets. It also runs uropa to annotate peaks based on the nearest genome gene model. To run the script, move to the `local` directory and execute:

```
Rscript ATAC_Read_Counts.R
```

We then use edgeR to identify differentially accessible peaks using a negative binomial generalized linear model for both the untreated samples and the full dataset. These scripts output the results both as a RData object and as individual results tables in a csv format.

```
Rscript ATAC_DGE.R  
Rscript iCRT_ATAC_DGE.R
```

Next we repeat the edgeR analyses using the RNA-seq read count matrices. There are three analyses, one for the untreated data, one for the full dataset, and a final analysis for the data from Wenger et al.

```
Rscript RNAseq_DGE.R  
Rscript iCRT_RNA_DGE.R  
Rscript Wenger_RNAseq_DGE.R
```

We then further analyze the DGE results to generate the RNA-seq and ATAC-seq plots used for the manuscript, including individual transcript/locus plots and global fold change comparison plots. In addition, the code integrates information from the Hydra scRNA-seq atlas to generate structural annotation plots.

```
Rscript RNA_Post_DGE_Analysis.R  
Rscript ATAC_Post_DGE_Analysis.R
```

We then identify transcription factor binding motifs associated with significant changes in chromatin accessibility using the chromVAR package. Because chromVAR requires peaks to have a uniform size, we need to re-calculate read counts using a resized peakset. The read counts are saved as RData objects to be used by scripts in the next step. In addition, a script is run to hierarchically cluster the transcription factor binding motifs that will be used in the downstream analysis using the output of the `compareMotifs.pl` script included in the HOMER tool suite. This clustering is used to reduce the redundancy of the chromVAR results, as many transcription factors have virtually identical binding motifs.



```
Rscript ChromVar_Counts.R  
Rscript motif_clustering.R
```

We then run the chromVAR analysis, focusing on motifs that change in accessibility in response to injury and how those motifs are affected by iCRT14. These scripts will generate the heatmap and individual accessibility plots used in the paper. The `chromVar_Untreated_Analysis.R` script will also perform an analysis to identify candidate regulators of injury-induced Wnt pathway component expression at 3hpa (presented in Figure 4). This part of the analysis integrates the chromVar results, the ATAC-seq and RNA-seq DGE results, and HOMER motif enrichment analyses.

```
Rscript chromVar_Untreated_Analysis.R  
Rscript chromVar_icrt_Analysis.R
```

Finally, because the cAMP response element was identified as a binding motif of interest in the chromVAR analysis, the following command can be used to generate a bed file containing putative bZIP binding sites genome-wide using HOMER.

```
./scan_for_CRE.sh
```

## Surveying Wnt Signaling Component Expression Patterns

The final part of the analysis generates plots of canonical Wnt signaling pathway component expression during regeneration. To identify candidate pathway components, the analysis uses Wnt pathway gene entries identified by KEGG to query the *Hydra* 2.0 genome reference. The KEGG database annotations for *Hydra* are supplemented using additional blast hits pulled using queries from KEGG annotations of other cnidarian genomes as well as the human Wnt pathway components.

Because this part of the analysis generates a fair number of intermediate files, it has been placed within its own folder: `wnt_Survey`. The commands below should be executed from within the `wnt_Survey` directory.

Blastable databases will need to be prepared for each reference used throughout this analysis executing the following commands:

```
makeblastdb -in SP.human.fasta -dbtype prot -out humanSP  
makeblastdb -in hydra.augustus.fastp -dbtype prot -out hvProt  
makeblastdb -in hydra2.0_genemodels.nt -dbtype nucl -out hvNuc1  
makeblastdb -in aepLRv2_transcript.fa -dbtype nucl -out lr  
makeblastdb -in GCF_000209225.1_ASM20922v1_protein.faa -dbtype prot -out nve  
makeblastdb -in GCF_001417965.1_Aiptasia_genome_1.1_protein.faa -dbtype prot -out  
epa
```

To generate the heatmap plots of Wnt component expression during regeneration, execute:

```
Rscript wntGenes.R
```