The ChIP-Seq Distillery

User Manual

May 2015

Victor Hanson-Smith

victorhansonsmith@gmail.com

# Introduction

Chromatin immunoprecipitation followed by DNA sequencing, commonly known as ChIP-seq, is a popular experimental technique to determine genomic locations that are enriched for binding by a particular DNA-binding protein. A common experimental approach is to perform ChIP-seq in different conditions – perhaps in different environmental conditions, or different species – and then compare the genomic location bound by a DNA-binding protein of interest.

ChIP-seq methods have become well established for mapping sequenced reads to their reference genome [cite XX], and for finding “peaks” with significantly enriched binding [cite XX]. However, it remains less established how to compare ChIP-Seq binding data between replicate experiments, between different conditions, and between different species.

The ChIP-Seq Distillery is a software pipeline that automates the multi-step protocol of aligning reads to a genome, calling peaks, and then comparing those peaks between replicates or species. The Distillery combines several popular software into a single unified pipeline.

Table of Contents

Introduction 2

Download and Install 4

Usage 5

Output Files 6

Troubleshooting 7

# Download and Install

The ChIP-Seq Distillery source code is available here: https://github.com/vhsvhs/SeqTools

## Not Designed for Laptops

The Distillery is designed to run on a cluster or supercomputer running Linux, Unix, or OSX. Computer analysis of ChIP-Seq data can run for many hours and sometimes days. It requires large amounts of computer memory, and this analysis may be inappropriate for your laptop computer.

## Prerequisites

The Distillery is written in **Python** and **R**. It makes use of functions in three commonly-used Python libraries: **scipy**, **numpy**, and **matplotlib**. You will need to install these libraries separately, but they may already be installed.

* SciPy: <http://www.scipy.org>
* NumPy: <http://www.numpy.org>
* Matplotlib: <http://matplotlib.org>

You will also need to install the software Bowtie 2 for aligning reads to genomes, and MACS2 for calling peaks.

* Bowtie 2: <http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>
* MACS2: <https://github.com/taoliu/MACS/>

## Install

The Distillery itself does not need to be installed, per se, but you will need to download its source code as follows:

1. Open a terminal on your computer, and navigate to the folder where you would like to install the ChIP-Seq Distillery.

cd ~/Applications

1. Clone the GitHub repository, containing source code, onto your computer.

git clone https://github.com/vhsvhs/SeqTools

1. At this point, you can launch the distillery by running the python script located at PATH/after\_illumina/run\_distill.py, where PATH is the folder in which you cloned the GitHub repository.

Note for UCSF Johnson Lab: The software is already installed on the lab server. It can be invoked as a Python script at the following filepath:

/common/REPOSITORY/SeqTools/after\_illumina/run\_distill.py

See the section titled “Example” for more details.

# Usage

The ChIP-Seq Distillery (CSD) is a Python script that is run from the command-line. CSD may require several hours to finish, depending on the size of your ChIP-seq project. During its run, CSD will print useful updates to the terminal.

## Command:

python run\_distill.py

## Required Parameters:

--annopath <PATH> or --configpath <PATH>

--datadir <PATH>

--outdir <PATH>

--project\_name <NAME>

--genome\_list <PATH>

--*pillarspath*

## Optional Parameters:

--dbpath <PATH>

--use\_mpi 1

--*mpinp*

--mismatch\_thresh 1000000

--chrom\_filter <KEYWORD>

--jump <INTEGER>

--stop <INTEGER>

*--bowtie2*

--*samtools*

*--macs2*

*--gcb*

--*seqtoolsdir*

--*eliminate\_multialign*

*--restrict\_to\_sample*

*--restrict\_to\_strain*

*--practice\_mode*

--minqval

# Input Files

1. Annotation file
2. Pillars File
3. DB file (optional, will create new if no DB)

## The Annotation File

The annotation file describes your FASTQ reads. This file associates reads with known genomes, replicate IDs, and media conditions. The annotation file indicates which reads are from a untagged control experiment, and which reads are the tagged experimental observations.

The annotation file should contain a text table, described as follows. The first line should be a header row with the following columns

* sample\_tube\_name
* library\_name
* index
* file\_name
* strain
* species
  + Current Options: cdub, calb, ctrop
  + Hybrid species should be species1\_species2, e.g. “cdub\_calb”
* tr
* tag – “YES” or “NO”
* media
* condition
* replicate – an integer number
* comment

# We need to build a better

# The Analysis Pipeline

1. Map Reads to Genomes (Bowtie)

1.1. Check Bowtie Output

2. Extract Reads from Bowtie Output

2.1. Get Hybrid Pairs

2.3. Find Hybrid Unique Reads

2.31. Pring Reads Histograms

2.4. Write Filtered SAM files

3. Write Sorted BAM files

3.1. Check BAMS

3.2. BAM to BedGraph

3.3. Check BedGraph files

4. Run Peak Calling (MACS2)

4.1 Check Peaks

5. Calculate FE

5.1. Check FE

6. BED to WIG

6.2. Check WIG

7. Setup Visualizations

8. Create Visualizations

i. Build Visualization Database

ii. Import Pillars

iii. Resolve Alias Gene IDs

iv. Import Summits (BED) and FE (BDG)

v. Correlate Replicates, Groups, and Species

vi. Plot Replicates, Groups, and Species

**Notes from Meeting May 6th, 2015:**

Parameters to add to run\_distill.py

--mismatch\_threshold X (for post-Bowtie)

--eliminate\_multialign – this is now DEFAULT turned on. (skips those reads with multiple alignment positions, based on the XS flag in SAM)

--min\_qval X (for MACS2)

--peak\_style <any intergenic, peak intergenic, any exonic>

--rep\_comp\_style <INTERSECTION/UNION>

1. BOWTIE - Read mismatch threshold
   1. Eugenio: You can only play with the mismatches in the seed of the alignment, not the entire alignment.
   2. Eugenio: Eliminate reads that map to multiple locations
      1. Do this use “–q 1” in samtools view
   3. Eugenio: remove PCR artifacts, i.e. reads that are identical
      1. MACS2 does this, but the duplicates appear in the WIG file
      2. “samtools rmdup” will do this.
         1. These will create “square” edges in the peaks
2. MACS2 – shiftsize –effective genome size
   1. Let’s compute genome size from the genome list, rather than hardcoding the genome size.
   2. Let’s do a run where we double the genome size.
   3. –slocal versus –llocal
   4. Let’s build a false-positive discovery rate estimate into the pipeline.
      1. Control versus control?
      2. Mean and median Q-value for control vs. control.
      3. Let’s compare the distributions of Q-values for control versus control, and the distribution of Q-values for experimental versus control. Then do a statistical test to see if the distributions are significantly different.
   5. Examine #3 Call peaks with given -log10qvalue cutoff: 1.30 ...
3. Peaks to Genes
   1. Add an option –peak\_style for rejecting/keeping peaks.
   2. Use all the intergenic region? Versus use just the 1KB upstream from a gene.
   3. Add a histogram of peak distance to nearest gene.
4. Correlating Replicates
   1. Maybe add a criterion to evaluate overlap of peaks between species.
5. Matt: add an option to try different correlation steps
6. Eugenio: fix the PNG visualztion to color yellow last.
7. Matt: think about a way to compare between different transcription factors

# ARGUMENTS LIST:

# name = Scer\_PpNdt802

# format = AUTO

# ChIP-seq file = ['Scerevisiae\_PpNdt802.sort.bam']

# control file = ['Scerevisiae\_untagYPD2.sort.bam']

# effective genome size = 1.43e+07

# band width = 300

# model fold = [5, 50]

# qvalue cutoff = 5.00e-02

# Larger dataset will be scaled towards smaller dataset.

# Range for calculating regional lambda is: 1000 bps and 10000 bps

# Broad region calling is off

# MACS will save fragment pileup signal per million reads

Datasets:

Isabel’s NDT80 is an overexpression dataset, Eugenio’s.

Rather than doing all-by-all combinations, vary one at at time.

**Genome size – double it, and try halving it.**

Matt: use wor4 – use one where it worked well, and one where it worked less well. – Is there information in the less-well experiment that we’re missing?

Sandy: Maybe the regular expressed Ndt80 data contains some cryptic information that’s in the overexpressed Ndt80.

# Output Files

Asdfadsf

# Troubleshooting

asdfasdf