SPIDR Analysis Documentation

Darvesh Gorhe, Ahmed Abdou, Erica Wolin

May 8, 2023

Contents

1 Introduction

This document will outline the analysis pipeline for the SPIDR project. The pipeline is divided into three main sections: preprocessing, post-pipeline, and downstream analysis. The preprocessing section will outline the steps taken to go from fastq files to bed files. The post-pipeline section will outline the steps taken to prepare the data from bed files to bedgraph files. The downstream analysis will outline the steps taken to analyze the data from bedgraph files to the final results.

The preprocessing and the postprocessing pipeline use Snakemake to orchestrate several steps in an efficient way using `sbatch` jobs. These `sbatch` jobs are ways to run individual sections of the pipeline on high-performance computing clusters.

2 Preprocessing Pipeline

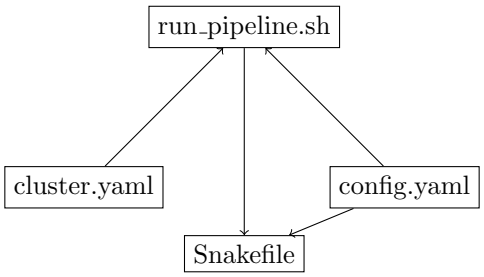

Figure 1: Overview of preprocessing configuration.

Above we see an overview of the primary files involved in configuration of the preprocessing pipeline. `run_pipeline.sh` is the shell file that actually gets called to run the pipeline. `cluster.yaml` is the file that configures the `sbatch` jobs. It sets things like the account ID so jobs actually get queued, the memory per job, the number of cpus per job, the maximum time a job should run, etc. `config.yaml` is the file that configures the pipeline itself. This has parameters like the filepaths for indicies for the bowtie2 and STAR aligners. `Snakefile` is the file that actually runs the pipeline and contains all the rules. The rules are the individual steps that are run in the pipeline.

Rules are chained together automatically based on the name of input and output file specified within each rule. Snakemake will create a directed acyclic graph (DAG) upon launching the pipeline. A DAG is a data structure which tells Snakemake the dependencies between rules so Snakemake can determine how to optimally schedule rules on a system. Snakemake will then run the rules in the correct order and in parallel when possible. Below is an example of a DAG for the preprocessing

pipeline with `num_chunks`= 2. As the number of chunks becomes larger, the middle of the graph becomes correspondingly wider.

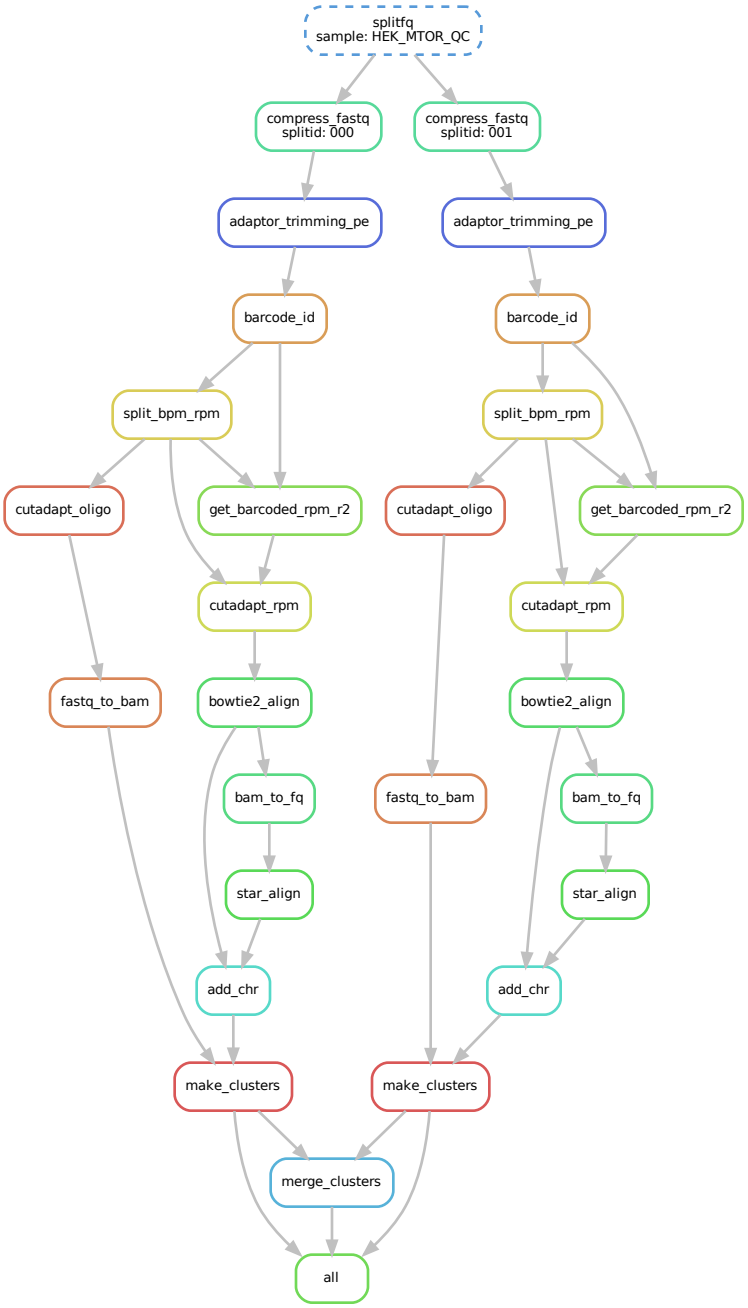

Figure 2: Example of directed acyclic graph (DAG) for preprocessing pipeline with `num_chunks`= 2

Each color corresponds to a different rule. To see the contents of each rule you can view the `Snakefile`. The arrows represent the flow of data and the dependencies between rules. A rule will not run until all of its dependencies have successfully completed.

3 Postprocessing Pipeline

I'm not sure how the post-processing pipeline works to be completely honest, so I'm not comfortable writing documentation for it.

4 Downstream Analysis (Ahmed)

Not sure where Ahmed's code and figures are located.

5 Downstream Analysis (Darvesh)

Below is a comparison of the ENCODE data to the outputs of SPIDR. Specifically, we compare the transcript proportion as given by the supplementary data of ENCODE along with the ENCODE raw data processed through the same pipeline as the SPIDR data.

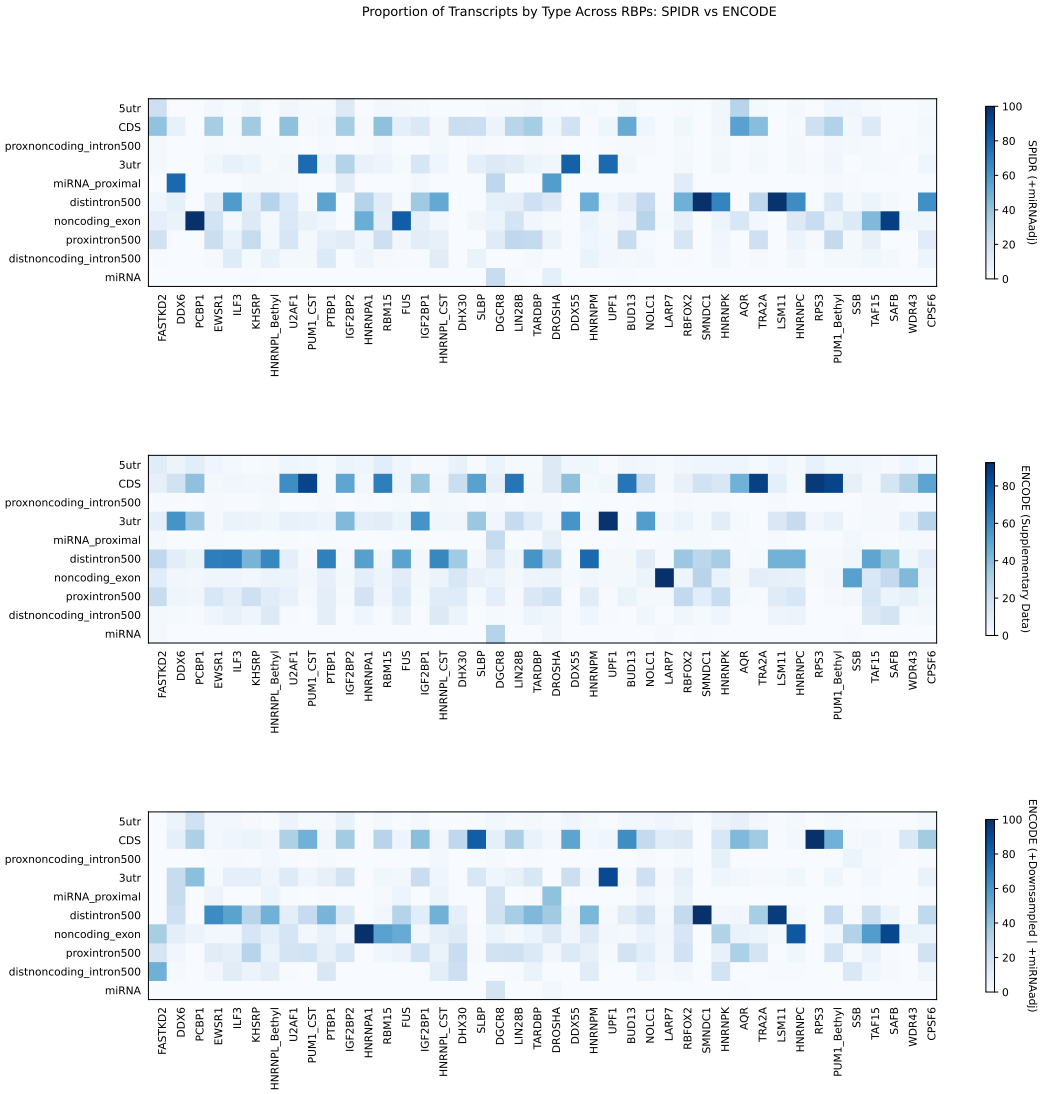

Figure 3: Comparing ENCODE to SPIDR

We also compared the distribution of imputed transcript proportions versus the true transcript proportions. Essentially, we wanted to ensure that our observed transcript proportions were significantly different from those that we would expect by chance. Indeed, this was the case.

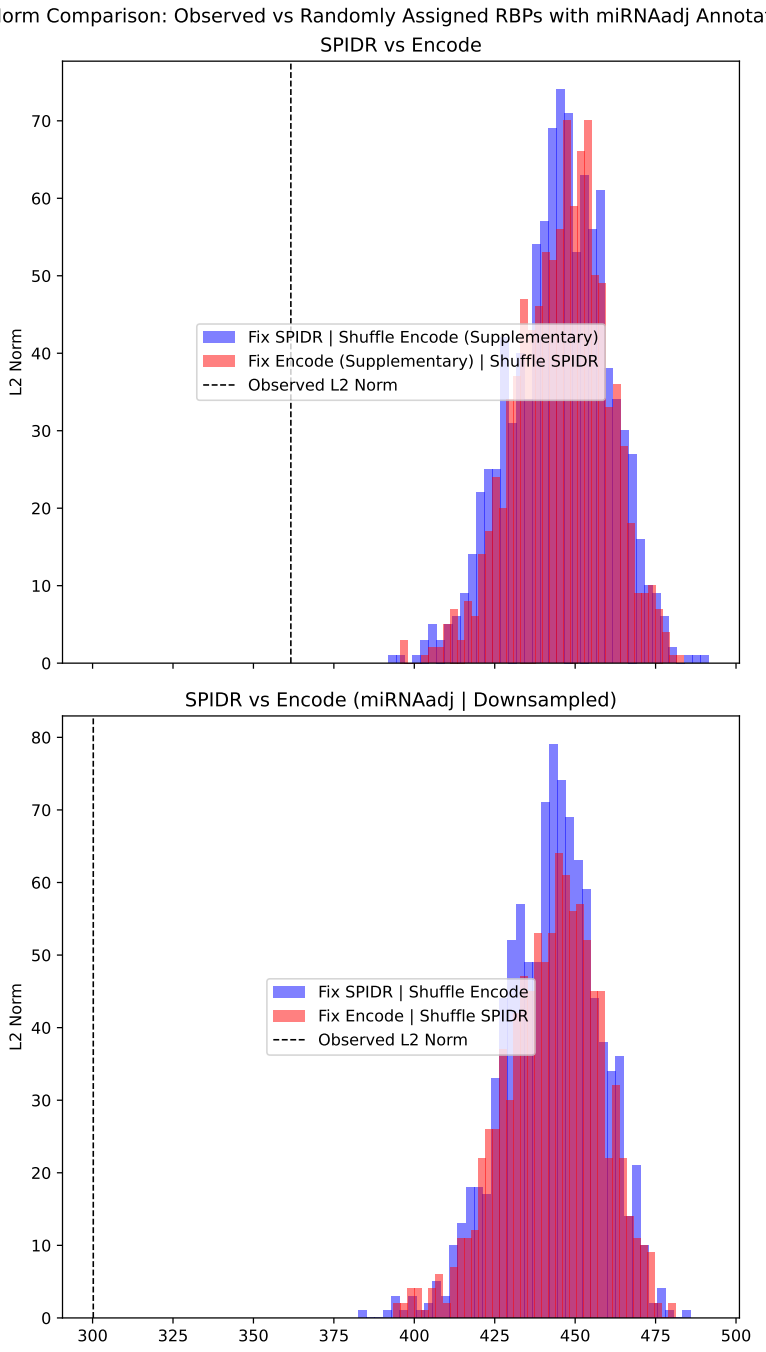

Figure 4: Comparing ENCODE to SPIDR