MATT'S PIPELINE DOCUMENTATION

MATTHEW KNOWLES

Contents

1. Introduction

This document contains documentation on the programs and scripts that have been used in the undertaking of the BPSI project. Broadly speaking, scripts are split into two main sections, pre-analysis, and analysis. The pre-analysis scripts are mainly python scripts that are used to call and automate third-party programs such as BWA and MACS. The analysis scripts are R scripts that make use of libraries such as DiffBind and VULCAN to perform the analysis. These scripts are ran through a higher level Python script to make life a lot easier.

2. Pre-Analysis

Before any analysis can be done, we need to prepare the data to put it in a form which is readable by the R libraries we want to use later on. This section details these scripts, in order of usage.

2.1. **aligner.py.**

- **Purpose:** Takes fastq files and aligns them into BAM files
- **Requires:** Python, BWA

Description
aligner.py is a Python script that makes use of the os library to run the BWA 'mem' algorithm from the user's local bwa directory. It requires a directory full of fastq files which is looped through, and for each file, the fastq file is aligned to the hs38 reference genome, and placed into a user-specified directory of BAM files.

2.2. **sort_index.py.**

- **Purpose:** Sort and index aligned BAM files
- **Requires:** Python, samtools

Description
sort_index.py again makes use of the os library in Python to call the samtools function *sort, index*. For each BAM file, the sorted BAM has an "S" appended onto the end to distinguish it from unsorted samples, and then this sorted sample is indexed, creating a .bam.bai file for each sample.

Date: 21 August 2020.

2.3. **peakcalls.py.**

- **Purpose:** Call peaks using MACS
- **Requires:** Python, MACS, controlfile

Description *peakcalls.py* is a Python file which calls the MACS2 *callpeak* algorithm to call the peaks of sorted BAM files. A controlfile, which is a txt file containing pairs of samples, one being the primary sample, one being the corresponding control sample. The script iterates through this file, and for each pair, runs MACS2 using the appropriate sample and control file.

3. Analysis

Now that we have our data in the form needed. We can start doing some analysis. Again, this section is written in order of usage.

3.1. **run_diffbind.R.**

- **Purpose:** Perform differential binding analysis on a dataset
- **Requires:** R, DiffBind, samplesheet

Description $run\_diffbind.R$ takes a samplesheet written by the user, and then uses the DiffBind package to run differential binding analysis. This is great for a lot of things, but most importantly it gives us a counts file.

3.2. **testdb.R.**

- **Purpose:** Test DiffBind count object is of the correct dimension
- **Requires:** R, counts object from $run\_diffbind.R$

Description Very simple file to check that the number of samples and the number of counts are the same, otherwise VULCAN won't work

3.3. **run_vulcan.R.**

- **Purpose:** Run VULCAN on the dataset
- **Requires:** R, VULCAN, counts object from $run\_diffbind.R$

Description $run\_vulcan.R$ is a script which takes the count object, and performs analysis with the VULCAN package. In addition it will perform GSEA analysis.

3.4. **analysis_pipeline.py.**

- **Purpose:** Automate the running of analysis scripts
- **Requires:** Python, LaTeX

Description This is a high level Python script which runs through the other scripts in this section automatically, outputting a PDF of the results compiled in LaTeX when it is finished. It will terminate if the $testdb.R$ file returns an error.