EMBL single-cell Omics Course

Hyobin Jeong

May 12, 2019

Contents

1 Introduction 2
2 Download pipeline for the preprocessing of Strand-seq data . . 2
3 Add sequencing data to the pipeline 5
4 Execute the pipeline 5
5 Explore the output folders and plots 6
6 Session Info 7

1 Introduction

In this practical we will learn how to perform pre-processing of Strand-seq data after the sequencing. We will start from the fastq file to align it to the human reference genome (hg38) and then make a chromosome overview plot based on Snakemake pipeline.

2 Download pipeline for the preprocessing of Strand-seq data

After the sequencing, we will get paired-end Strand-seq fastq files. As a pre-processing, we need to align the reads to the human reference genome (hg38), remove PCR duplicate, count reads for each of the bins in fixed windows. This will require multiple softwares and R script, so that workflow management tool such as Snakemake is useful to proceed convenient and reproducible processing. Snakemake is a python-based workflow management tool and it can run workflows consisting of multiple inter-dependent tasks. These can be computed either locally or on the cluster. In this session, we will download pipeline from the git repository, explore the overall script, and run the script to analyze Strand-seq data from RPE-1 C7.

```
## Download and unzip the Snakemake pipeline
git clone https://github.com/jeongdo801/Strand_seq_practical_computational.git \
Practical_C7

cd Practical_C7
tar -xvzf practical.tar.gz

## Explore the overall script of Snakemake pipeline
cd practical
less Snakefile
```

The Snakemake script begins with the settings. We can change the sample name here according to the data we will analyze. The pipeline will expect raw files will be in a subfolder 'fastq' and will have name with the following format.

```
##############################################################################
 # SETTINGS                                                                  #
 #                                                                           #
 # Set the sample name                                                       #
 #                                                                           #
SAMPLE_NAME = "testSample"
 #                                                                           #
 #                                                                           #
 # By default, the pipeline will expect file to be in a subfolder called     #
 # 'fastq' and to be names *_1_sequence.txt.gz *_2_sequence.txt.gz           #
 #                                                                           #
FASTQ, = glob_wildcards("fastq/{cell}_1_sequence.txt.gz")
 #                                                                           #
 #                                                                           #
abbreviate_names = False
 #                                                                           #
 ##############################################################################
```

Let's change the sample name testSample into C7data and save it.

```
vi Snakefile
:wq
```

In the Part1, Snakemake script will perform read mapping using bwa.

```
 #
 # PART I
 # Read mapping
 #

def get_time_for_map_reads(wc, input):
    fsize = os.stat(input[1]).st_size
    return max(1,int(round(fsize/400000000)))

rule map_reads:
    input:
        "fastq/{cell}_1_sequence.txt.gz", "fastq/{cell}_2_sequence.txt.gz"
    output:
        temp("bam/{cell}.bam")
    params:
        name = lambda wc: NAMES[wc.cell],
        genome = config["genome"]
    threads:
        4
    resources:
        time = get_time_for_map_reads
    message:
        "Start mapping cell {input} using {threads} threads"
    shell:
        """
        module load BWA/0.7.15-foss-2016b SAMtools/1.3.1-foss-2016b
        bwa mem -t {threads} \
            -R '@RG\tID:{params.name}\tSM:{SAMPLE_NAME}\tLB:{wildcards.cell}' \
            {params.genome} \
            {input} \
        | samtools view -bT {params.genome} - \
        > {output}
        """

rule sort_bam:
    input:
        "bam/{cell}.bam"
    output:
        temp("bam/{cell}.sort.bam")
    threads:
        2
    shell:
        """
        module load SAMtools/1.3.1-foss-2016b
        samtools sort -@ {threads} -O BAM -o {output} {input}
```

```
        """

rule markdups:
    input:
        "bam/{cell}.sort.bam"
    output:
        "bam/{cell}.sort.mdup.bam"
    threads:
        2
    shell:
        """
        module load biobambam2/2.0.76-foss-2016b
        bammarkduplicates markthreads={threads} I={input} O={output} index=1 rmdup=0
        """

rule index:
    input:
        "bam/{cell}.sort.mdup.bam"
    output:
        "bam/{cell}.sort.mdup.bam.bai"
    shell:
        """
        module load SAMtools/1.3.1-foss-2016b
        samtools index {input}
        """
```

In the Part2, Snakemake script will perform read counting and make overview plot and chromosome plot.

```
#
# PART II
# MosaiCatcher counts & plots
#

rule mosaic_count:
    input:
        bam = expand("bam/{cell}.sort.mdup.bam", cell = FASTQ),
        bai = expand("bam/{cell}.sort.mdup.bam.bai", cell = FASTQ)
    output:
        counts = "counts/" + SAMPLE_NAME + ".{window}.txt.gz",
        info   = "counts/" + SAMPLE_NAME + ".{window}.info"
    params:
        mosaic = config["mosaic"],
        exclude = config["exclude"]
    shell:
        """
        {params.mosaic} count \
            -o {output.counts} \
            -i {output.info} \
            -x {params.exclude} \
            -w {wildcards.window} \
```

```
                {input.bam}
            """

    rule plot_mosaic_counts:
        input:
            counts = "counts/" + SAMPLE_NAME + ".{window}.txt.gz",
            info   = "counts/" + SAMPLE_NAME + ".{window}.info"
        output:
            "plot_overview/" + SAMPLE_NAME + ".{window}.pdf"
        params:
            qc_plot = config["qc_plot"]
        shell:
            """
            module load R-bundle-Bioconductor/3.5-foss-2016b-R-3.4.0
            Rscript {params.qc_plot} {input.counts} {input.info} {output}
            """

    rule plot_chromosomes:
        input:
            counts = "counts/{SAMPLE_NAME}.{window}.txt.gz"
        output:
            dynamic("plot_chromosomes/{SAMPLE_NAME}/window_{window}.{chrom}.pdf")
        params:
            sv_plot = config["sv_plot"]
        shell:
            """
            module load R-bundle-Bioconductor/3.5-foss-2016b-R-3.4.0
            Rscript {params.sv_plot} {input} plot_chromosomes/{SAMPLE_NAME}/window_{wildcards.window}
            """
```

3 Add sequencing data to the pipeline

After downloading the pipeline, we need to tell the pipeline which data to use by copying/linking files into a fastq subdirectory. The paired-end raw data of RPE-1 C7 can be downloaded using the following command.

```
## Download the raw data of 10 cells from RPE-1 C7
wget -O fastq.tar.gz \
https://www.dropbox.com/s/ehoptyan8yany3g/fastq.tar.gz?dl=0

tar -xvzf fastq.tar.gz
```

4 Execute the pipeline

Edit the Snakemake file to specify the sample name and other variables if necessary. Using snakemake -n you can see which jobs are going to be computed. Snakemake script can be launched locally or on the cluster.

```
## Check which jobs are going to be computed
snakemake -n

## Compute locally using 4 cores
snakemake -j 4
```

5 Explore the output folders and plots

After the function has finished, you will find the four different folders **bam**, **counts**, **plot chromosomes**, and **plot overview** containing all produced files and plots.

- **bam** This folder contains the bam files and index files after the bwa alignment. During the snakemake pipeline PCR duplicates were maked with SAM Flag but not removed from the bam files. Those bam files will be an input of the downstream analysis such as BreakpointR, StrandPhaseR, or MosaiCatcher.
- **counts** This folder has the count matrix (text files) of Strand-seq reads on the fixed sized bins. The numbers in the file name (20000, 50000, 100000, 200000, 500000) indicates the size (bp) or the window. The count matrix will show the number of Crick and Watson reads separately. Bins have a fixed width, starting from position 0 up the end of the chromosome. Mapped reads were assigned to bins based on their start position and filtered according to the following criteria for read counting: non-primary and supplementary alignments are excluded; alignments with the QC failure flag are excluded; PCR duplicates are excluded; reads with mapping quality $<= 10$ are excluded. In case of paired-end data only the first read of each pair was used to avoid double-counting.This folder also give info matrix which contains the parameters p and r of the negative binomial (NB) distribution which is important for the SV classification in the downstream analysis. During parameter estimation, bins were excluded from the estimation process if their mean coverage across all cells was very low or if they showed a highly abnormal fraction across cells.
- **plot overview** This folder will show you five pdf files with different sizes of genomic windows. The first page of the pdf file shows the statistics of the analysis result such as the distribution of total number of reads per cell, duplication rate, or excluded bins per chromosomes. Afterwards, every pages show the overview of alignment result of each of the single-cells. The depth of Crick reads are depicted in the green color in the right side, and the depth of Watson reads are depicted in the orange color in the left side of each chromosome lines.
- **plot chromosomes** Under this folder, there will be subfolder with the name of sample you specified in the config file before running the pipeline, and it will contain pdf files per window size per chromosomes.

6 Session Info

```
toLatex(sessionInfo())
```

- R version 3.5.1 (2018-07-02), `x86_64-apple-darwin15.6.0`
- Locale: `C`
- Running under: `macOS High Sierra 10.13.3`
- Matrix products: default
- BLAS: `/System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libBL`
- LAPACK: `/Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.26.0, GenomeInfoDb 1.16.0, GenomicRanges 1.32.6, IRanges 2.14.11, S4Vectors 0.18.3, breakpointR 1.0.0, breakpointRdata 1.0.0, cowplot 0.9.3, ggplot2 3.0.0, knitr 1.20
- Loaded via a namespace (and not attached): Biobase 2.40.0, BiocParallel 1.14.2, BiocStyle 2.8.2, Biostrings 2.50.2, DT 0.4, DelayedArray 0.6.5, DelayedMatrixStats 1.2.0, FNN 1.1.2.1, GenomeInfoDbData 1.1.0, GenomicAlignments 1.16.0, MOFAtools 0.99.0, Matrix 1.2-14, MultiAssayExperiment 1.6.0, R6 2.2.2, RColorBrewer 1.1-2, RCurl 1.95-4.11, Rcpp 0.12.18, Rhdf5lib 1.2.1, Rsamtools 1.34.0, SingleCellExperiment 1.2.0, SummarizedExperiment 1.10.1, XVector 0.20.0, assertthat 0.2.0, backports 1.1.2, beeswarm 0.2.3, bindr 0.1.1, bindrcpp 0.2.2, bitops 1.0-6, codetools 0.2-15, colorspace 1.3-2, compiler 3.5.1, corrplot 0.84, crayon 1.3.4, data.table 1.11.4, digest 0.6.16, doParallel 1.0.11, dplyr 0.7.6, dynamicTreeCut 1.63-1, edgeR 3.22.3, evaluate 0.11, foreach 1.4.4, ggbeeswarm 0.6.0, ggrepel 0.8.0, glue 1.3.0, grid 3.5.1, gridExtra 2.3, gtable 0.2.0, gtools 3.8.1, highr 0.7, htmltools 0.3.6, htmlwidgets 1.2, httpuv 1.4.5, igraph 1.2.2, iterators 1.0.10, jsonlite 1.5, later 0.7.4, lattice 0.20-35, lazyeval 0.2.1, limma 3.36.3, locfit 1.5-9.1, magrittr 1.5, matrixStats 0.54.0, mime 0.5, munsell 0.5.0, pheatmap 1.0.10, pillar 1.3.0, pkgconfig 2.0.2, plyr 1.8.4, promises 1.0.1, purrr 0.2.5, reshape2 1.4.3, reticulate 1.10, rhdf5 2.24.0, rjson 0.2.20, rlang 0.3.1, rmarkdown 1.10, rprojroot 1.3-2, rstudioapi 0.7, scales 1.0.0, scater 1.8.4, scran 1.8.4, shiny 1.1.0, shinydashboard 0.7.0, statmod 1.4.30, stringi 1.2.4, stringr 1.3.1, tibble 1.4.2, tidyr 0.8.1, tidyselect 0.2.4, tools 3.5.1, tximport 1.8.0, vipor 0.4.5, viridis 0.5.1, viridisLite 0.3.0, withr 2.1.2, xtable 1.8-3, yaml 2.2.0, zlibbioc 1.26.0