Time Series Memories (always in construction)

María José Olmo Uceda

2022-04-19

Contents

Chapter 1

Introduction

Analysis of gene expression data in time series of viral infections. Tracking of the selected experiments and the processes followed from data download to DEG.

One the differently expressed genes have been selected we will beging the study of those genes that can serve as early warnings. The first method we use use is based on **Dynamical Network Biomarkers**.

1.1 General workflow

General information:

- From begining to matrix counts: chapter 3
- DEG analysis: chapter 4

Each project analyzed will have its own chapter

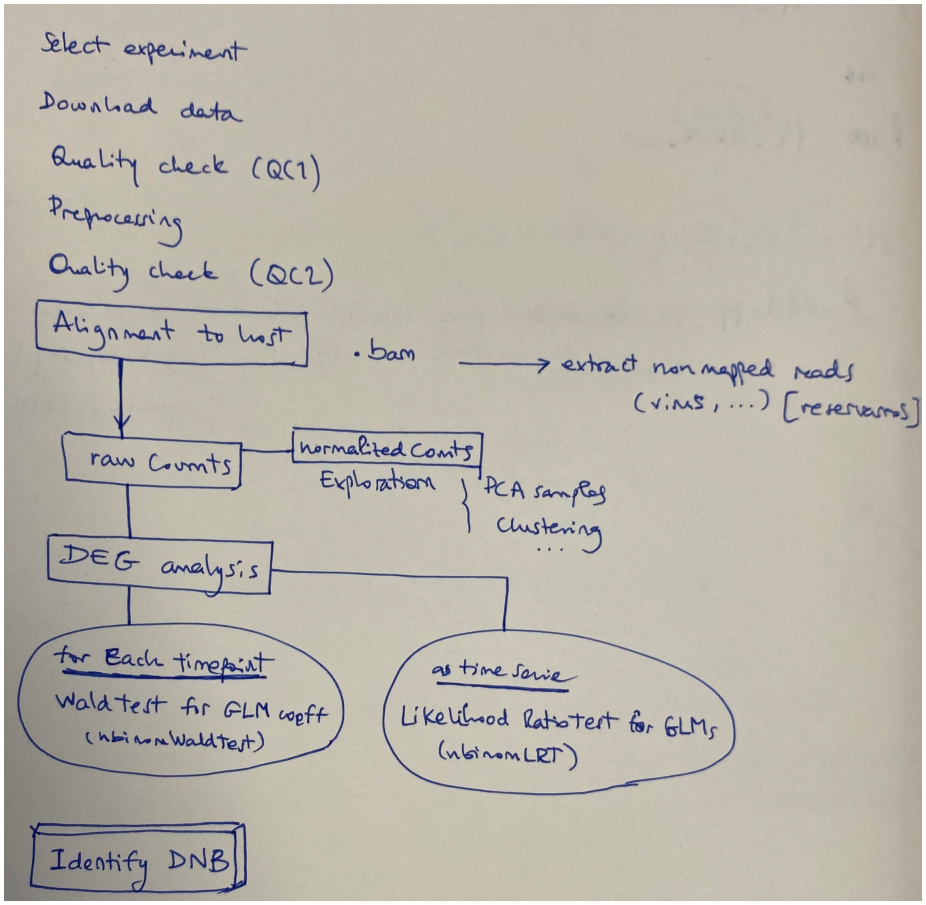

Figure 1.1: General process

Chapter 2

Experiments selected

Previous conditions:

- Viral infection time series data available
- At least 4 time points
- Control non infected for each time point
- Preferably RNA-seq but we will study the microarray data previously selected by Juan Carlos and Santiago.

I will start by the RNA-seq data, and adding the datasets following the chronology of their analysis.

2.1 Public data

2.1.1 Host: Human

- PRJNA636173 (April 2022)

2.2 Laboratory data

2.2.1 Host: C. elegans

2.2.1.1 Project:

- *Experimental work*: Victoria G. Castiglioni,

Chapter 3

General steps until have the count matrix

In this chapter we resume all the steps needed to construct the raw count matrix. Although most of the experiments we are using the data have the constructed matrix available we are going to start from the fastq files to unify the process.

3.1 Download fastq from NCBI

We are using the SRAtoolkit, concretly the `fasterq-dump` tool. For the moment we are downloading the fastqs in the '/storage/evsysvir/TimeSeries' directory (all the group could access) in garnatxa.

Each one of the experiments will have its own directory inside TimeSeries.

Script used:

```
#!/bin/bash
#SBATCH --job-name=downloadSRA
#SBATCH --partition=short
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=32
#SBATCH --mem=10gb
#SBATCH --time=1-00:00:00
#SBATCH --output=downloadSRA%j.log
<<downloadSRA.sh
Download with SRAtoolkit the fastqs asociated with the SRR identificators
present in the file passed as first argument.
    The file is in the format:
```

```
SRR1
SRR2
...
The easy way to create the file is with the SRA Run Selector from NCBI

2022/02/10
MJ
downloadSRA.sh

SRAFILE=$1

while read run
do
    fasterq-dump --progress $run
done < $SRAFILE
```

As is noted in the code, the easy way of create the SRA access file is with the SRA Run Selector. I have written the code to use the this file as first argument, so we only will need to put the script in the right path and use the correspondent SRA access file. It is a moderate time and resources consuming so we execute through SLURM: `sbatch downloadSRA.sh srr_acc_list.txt`

3.2 Quality Control 1

The fastq quality was checked with `fastqc` tool and the reports were stored in the QC1 folder `srun fastqc *.fastq -o QC1` and posterior recopilation of the analysis with `multiqc .`

3.3 Cleaning the fastq

The cleaning was performed with bbduk.sh with the following parameters:
- `ref=adapters.fa` removing contaminants (adapters present in adapters.fa)
- `ktrim=r` → Trim to the right reads to remove bases matching reference kmers.
- `k=21` → kmer length used to find contaminants
- `mink=11` → Look for shorter kmers at read tips down o this length when ktrimming or masking.
- `qtrim=r` → Trim read ends (right end only) to remove bases with quality below `trimq`. Performed AFTER looking for kmers.
- `trimq=10` → Regions with average quality below this will be trimmed.
- `maq=5` → (or `minavgquality`) Reads with average quality (after trimming) below this will be discarded

- `minlength=$minlength` → It will depend on the project. F.e., in PRJNA636173 the mean length is 50, so we need to be less restrictives than usually. cd ## Quality Control 2

The fastq quality was checked with `fastqc` tool and the reports were stored in the QC1 folder `srun fastqc clean/*_clean.fastq -o QC2` and posterior recopilation of the analysis with `multiqc .`

3.4 Map the clean fastq vs the reference genome

This step will be performed with the HOST and with the VIRUS as reference. The reference used will be specify in each experiment.

In overall, we will align with **STAR**, taking advantage of the possibility of have the binary output sorted by coordinate.

3.5 Quality Control 3

QC of the alignments will be performed with `samstats`.

3.6 Mark duplicates

My current opinion about optical/PCR duplicates is that we should remove them before perform the DEA, but there are not a golden standard so we are going to mantain the both branches: a count matrix of all the reads and a count matrix without duplicates (*_nodup*).

We will use the MarkDuplicates option from th GATK4 toolkit, writing the unique and marked as duplicate reads in the same alignment file: {}_dedup.bam.

3.7 Removing duplicates

A copy without duplicates will be written in the correspondent nodup directory. This process will be performed with `samtools view -hbF0x400`.

3.8 From alignments to count matrix data

As far as I am not really sure that we don't should deduplicate in transcriptomics, indeed, I tend to think that we should, I am going to obtain the count matrix from both types of alignments.

In general, we will use R launched in garnatxa.

Example of script 'alignments2counts.R':

```
#alignments2counts.R
################################################################################
# GENERATE COUNT DATA MATRIX
# ALINEAMIENTOS SIN LOS DUPLICADOS
################################################################################
## 2022/03/07
## MJ

library("Rsamtools")
library("GenomicAlignments")
library("GenomicFeatures")
library("BiocParallel")

# Información sobre las muestras
#csvfile <- "/Users/mariajoseolmo/Documents/2021/gradualTransitions/PlantTranscriptome/
#sampleTable <- read.csv(csvfile, header = 1, sep="\t")

# alineamientos
bamfiles_dir <- "/storage/evsysvir/TimeSeries/PRJNA636173/alignments/nodup"

bamfiles <- list.files(path = bamfiles_dir, full.names = TRUE)

# Comprobamos que existen los ficheros
file.exists(bamfiles)

alignments <- BamFileList(bamfiles,
                          yieldSize = 2000000)


# Construyendo el gene model desde el GTF
gtffile <- "/storage/evsysvir/TimeSeries/references/genome_hg19_index/gencode.v39lift37
(txdb <- makeTxDbFromGFF(gtffile,
                         format = "gtf",
                         circ_seqs = character()))

## exones por gen
(ebg <- exonsBy(txdb, by="gene"))

register(MulticoreParam(6))

## Read count
se <- summarizeOverlaps(features = ebg,
```

```
                        reads = alignments,
                        mode = "Union",
                        singleEnd=FALSE,
                        ignore.strand = TRUE,
                        fragments=TRUE)

save(se, file="rawGeneCounts_PRJNA636173_nodup.rda")
```

And the sh script for launch the process with slurm:

```
#!/bin/bash
#lanch_matCounts_nodup.sh
#SBATCH --job-name=matCounts_nodup
#SBATCH --partition=long
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=32
#SBATCH --mem=60gb
#SBATCH --time=7-00:00:00
#SBATCH --output=matCounts_nodup_%j.log

# Cargamos el módulo de R
module load R/3.6.1

# Tiempo 0
start=`date +%s`

# Ejecutamos el .R
R < alignments2counts.R --no-save

echo "SE generado y guardado"

# Tiempo total de ejecución
end=`date +%s`
runtime=$((end-start))
echo "Total time: $runtime s"
```

Chapter 4

DEG analysis

The main purpose of this step is to reduce the number of total genes to be considered for the next steps of the process. We assume that the genes acting as early warnings will be in the selected differentially expressed genes.

To simplify, we only have two groups: infected and control (not infected). We can do the selection of these most variant genes with two methods:

1) Selecting at each time point the genes that are differently expressed: **Wald Test for GLM coeffincients**. Next, working with the join set, i.e., all the genes that have been selected as DE in any timepoint.

2) Select the genes that act deferentially across the timepoints: **Likelihood Ratio Test (LRT) for GLMs**.

The idea is that most of them will overlap, later on I may decide to use only one of the methods, the data will tell.

Chapter 5

PRJNA636173

- **Title:** Experimental and natural evidence of **SARS-CoV-2** infection-induced activation of type I interferon responses (**human**)
- **Paper:** [Banerjee et al., 2021]
- **NCBI project link:** https://www.ncbi.nlm.nih.gov/bioproject/PRJNA636173
- **Overall design:** We infected triplicated human lung epithelial cells (Calu-3) at a multiplicity of infection (MOI) for SARS-CoV-2 of 2, with comparison to triplicated uninfected controls. One hour post infection, the inoculum was removed and the clock was set to zero. We extracted and sequenced poly-A enriched RNA at 0, 1, 2, 3, 6 and 12 hours post infection (hpi) using an Illumina HiSeq 2500 with 2 x 50 bp chemistry to a minimum of 21.9 million clusters per replicate. Paired-end sequencing reads were mapped to the human reference transcriptome (GRCh37.67)

5.1 Time series aspects

- Host type: human lung epithelial cells (Calu-3)
- Sample points: 0, 1, 2, 3, 6 and 12 hours post infection (hpi)
- Groups: infected & uninfected
- Biological replicates: 3
- Total samples: 36

5.2 RNA-seq considerations

- Sequence mean length: 50 bp
- Illumina HiSeq 2500
- Paired-end

5.3 Preprocesing considerations

- Host: Human
  - Reference used: hg19 (GRCh37) genome assembly

```
wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_39/GRCh37_mappi
wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_39/GRCh37_mappi
```

5.3.1 Cleaning process

Ran in SLURM. The `minlength` set for this experiment was 40.

```
sbatch clean_wbbduk.sh SRR_listAcc_PRJNA636173.txt
```

```
#!/bin/bash
#clean_wbbduk.sh
#SBATCH --job-name=clean
#SBATCH --partition=short
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=32
#SBATCH --mem=30gb
#SBATCH --time=1-00:00:00
#SBATCH --output=clean_%j.log

<< "clean_wbbduk.sh"
For each sample in fq_dir:
- Clean the fq.gz with bbduck.sh

MJ
2022/02/21
clean_wbbduk.sh

# paths
samples=$1
```

```
adapters="/storage/evsysvir/TimeSeries/references/adapters.fa"
fq_dir="/storage/evsysvir/TimeSeries/PRJNA636173/raw_data"
clean_dir="/storage/evsysvir/TimeSeries/PRJNA636173/clean_data"
qc2_dir="/storage/evsysvir/TimeSeries/PRJNA636173/QC2"

minlength=40

while read sample
do
        echo "****************** PROCESSING $sample ******************"
        bbduk.sh \
        in1=${fq_dir}/${sample}_1.fastq \
        in2=${fq_dir}/${sample}_2.fastq \
        out1=${clean_dir}/${sample}_clean_1.fq.gz \
        out2=${clean_dir}/${sample}_clean_2.fq.gz \
        ref=${adapters} \
        ktrim=r k=21 mink=11 \
        qtrim=r trimq=10 maq=5 minlength=$minlength

done < $samples
```

5.3.2 Alignment vs complete genome with STAR

We are going to align versus the **complete genome**, not the transcriptome (option used in the original paper). The reason is that we want to do allways the same process and for *A. thaliana* there are not a good transcriptome available (for *C. elegans* I don't know yet).

- Building index (careful with this!):

Interactively in garnatxa (The process is high memory consuming and in that way can't finish the process)

```
path\to\STAR --runThreadN 6 \
--runMode genomeGenerate \
--genomeDir genome_hg19_index \
--genomeFastaFiles /storage/evsysvir/TimeSeries/references/genome_hg19_index/GRCh37.primary_assem
--sjdbGTFfile /storage/evsysvir/TimeSeries/references/genome_hg19_index/gencode.v39lift37.annotat
--sjdbOverhang 50 # readlength-1
```

Necessary to run in SLURM:

```
#!/bin/bash
#generateIndex2STAR.sh
#SBATCH --job-name=generateIndex2STAR
#SBATCH --partition=long
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=32
#SBATCH --mem=200gb
#SBATCH --time=1-00:00:00
#SBATCH --output=generateIndex2STAR_%j.log

<< "generateIndex2STAR"
In some cases (large genomes), we need a lot of memory to generate the index
needed to align with STAR.

Use:
arg1 = path to genome.fasta
arg2 = path to genome.gtf
arg3 = max(readlength) - 1
arg4 = path to the output directory

generateIndex2STAR

genomeFasta=$1
genomeGTF=$2
length=$3
outDir=$4

STAR="/home/maolu/programs/STAR-2.7.9a/bin/Linux_x86_64/STAR"

$STAR --runThreadN 32 \
--runMode genomeGenerate \
--genomeDir $outDir \
--genomeFastaFiles $genomeFasta \
--sjdbGTFfile $genomeGTF \
--sjdbOverhang $length
```

- Alignment + markDuplicates + remove duplicates (keeping the dedup file too):

In *slurm* as a batch

```
#!/bin/bash
#map&dedup.sh
#SBATCH --job-name=map+dedup
```

```
#SBATCH --partition=long
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=32
#SBATCH --mem=100gb
#SBATCH --time=10-00:00:00
#SBATCH --output=map+dedup_%j.log

<< "mapdedup"
For each sample in SRR_acc_list:
- Align paired end reads to the complete genome with STAR
- Mark duplicates with gatk4 (need to activate env)
- Generate a copy of bam files without duplicates

MJ
2022/02/21
mapdedup

samples=$1
dir_genome="/storage/evsysvir/TimeSeries/references/genome_hg19_index"
clean_dir="/storage/evsysvir/TimeSeries/PRJNA636173/clean_data"
dir_alignments="/storage/evsysvir/TimeSeries/PRJNA636173/alignments"
dir_dedup="${dir_alignments}/dedup"
dir_nodup="${dir_alignments}/nodup"
dir_dedup_metrics="${dir_alignments}/metrics"

gatk="/home/maolu/programs/gatk-4.2.2.0/gatk"
STAR="/home/maolu/programs/STAR-2.7.9a/bin/Linux_x86_64/STAR"

while read sample
do
    # Align vs complete genome hg19
    $STAR --runThreadN 12 \
    --genomeDir $dir_genome \
    --readFilesCommand gunzip -c\
    --readFilesIn ${clean_dir}/${sample}_clean_1.fq.gz ${clean_dir}/${sample}_clean_2.fq.gz \
    --outFileNamePrefix ${dir_alignments}/${sample}_ \
    --outSAMtype BAM SortedByCoordinate \
    --outSAMunmapped Within \
    --outSAMattributes NH HI NM MD AS

    # Mark duplicates
    $gatk MarkDuplicates \
    -I ${dir_alignments}/${sample}_Aligned.sortedByCoord.out.bam \
    -O ${dir_dedup}/${sample}_dedup.bam \
    -M ${dir_dedup_metrics}/${sample}_metrics.txt
```

```
    # Remove duplicates
    samtools view -hbF0x400 $dir_dedup/${sample}_dedup.bam > $dir_nodup/${sample}_nodu

done < $samples
```

5.4 Samples

Samples analyzed:

SRR11884692 SRR11884693 SRR11884694 SRR11884695 SRR11884696
SRR11884697 SRR11884698 SRR11884699 SRR11884700 SRR11884701
SRR11884702 SRR11884703 SRR11884704 SRR11884705 SRR11884706
SRR11884707 SRR11884708 SRR11884709 SRR11884710 SRR11884711
SRR11884712 SRR11884713 SRR11884714 SRR11884715 SRR11884716
SRR11884717 SRR11884718 SRR11884719 SRR11884720 SRR11884721
SRR11884723 SRR11884724 SRR11884725 SRR11884726 SRR11884727

5.5 DEG analysis

5.5.1 Exploration of samples

Bibliography

Arinjay Banerjee, Nader El-Sayes, Patrick Budylowski, Rajesh Abraham Jacob, Daniel Richard, Hassaan Maan, Jennifer A. Aguiar, Wael L. Demian, Kaushal Baid, Michael R. D'Agostino, Jann Catherine Ang, Tetyana Murdza, Benjamin J.M. Tremblay, Sam Afkhami, Mehran Karimzadeh, Aaron T. Irving, Lily Yip, Mario Ostrowski, Jeremy A. Hirota, Robert Kozak, Terence D. Capellini, Matthew S. Miller, Bo Wang, Samira Mubareka, Allison J. McGeer, Andrew G. McArthur, Andrew C. Doxey, and Karen Mossman. Experimental and natural evidence of SARS-CoV-2-infection-induced activation of type I interferon responses. *iScience*, 24(5):102477, 2021. ISSN 25890042. doi: 10.1016/j.isci.2021.102477. URL https://doi.org/10.1016/j.isci.2021.102477.