Programming Approaches for Bioinformatics Exam

Maiolino Aurelio - 923271

27-06-2025

Index

Introduction: Rileggi attentamente, potrebbero esserci dei CAZZI

The repository for this exam can be found on my GitHub

controlla i link, che forse cambi i nomi delle cartelle

rimuovi gli eval=FALSE

Part 1: Docker

Docker is this

The Dockerfile is that and can be found here

Building the Docker: the Dockerfile

Build a container from a base Ubuntu image, v20.04 was chosen due to its proven stability.

```
FROM ubuntu:20.04

ENV DEBIAN_FRONTEND=noninteractive

LABEL maintainer="aurelio.maiolino@edu.unito.it"
```

The base Ubuntu image is just that, an Ubuntu installation with not a lot more inside it. It is therefore necessary to install various packages.

Some notable are

- curl:
- b

ne mancano ancora

```
RUN apt-get update && apt-get install -y --no-install-recommends \
    software-properties-common \
    dirmngr \
    gpg \
    curl \
    build-essential \
    libcurl4-openssl-dev \
    libssl-dev \
    libxml2-dev \
    libfontconfig1-dev \
    libfreetype6-dev \
    libpng-dev \
    libtiff5-dev \
    libjpeg-dev \
    libharfbuzz-dev \
    libfribidi-dev \
    make \
    cmake \
    gfortran \
    libxt-dev \
    liblapack-dev \
    libblas-dev \
    sudo \
    wget \
    zlib1g-dev \
    libbz2-dev \
    liblzma-dev \
```

```
    libncurses5-dev \
    pandoc \
    git
```

After all of this a cleaning step can be useful:

```
RUN rm -rf /var/lib/apt/lists/*
```

Add the Cran Repository and install R

```
# Add the CRAN GPG key and repository for R
RUN curl -fsSL https://cloud.r-project.org/bin/linux/ubuntu/marutter_pubkey.asc | gpg
↪  --dearmor -o /usr/share/keyrings/cran.gpg \
    && echo "deb [signed-by=/usr/share/keyrings/cran.gpg]
    ↪  https://cloud.r-project.org/bin/linux/ubuntu $(lsb_release -cs)-cran40/" \
    | tee /etc/apt/sources.list.d/cran-r.list

# Update again and install R
RUN apt update && apt install -y --no-install-recommends r-base
```

Install Python and Jupyter Lab

```
# Install JupyterLab
RUN apt update && apt install -y python3 python3-pip python3-venv

# METTI il cazzo di &&
RUN python3 -m venv /opt/venv

# Activate virtual environment and install JupyterLab
RUN /opt/venv/bin/pip install --upgrade pip && /opt/venv/bin/pip install jupyterlab

# Set the virtual environment as the default Python path
ENV PATH="/opt/venv/bin:$PATH"

# Make R visible to jupyter
RUN R -e "install.packages('IRkernel')" \
    R -e "IRkernel::installspec(user = FALSE)"
```

Install R packages

```
# Install R packages
RUN R -e "install.packages(c('BiocManager', 'dplyr', 'ggplot2', 'data.table', 'future',
↪  'cowplot', 'remotes', 'R.utils', 'dplyr', rtracklayer', 'tinytex'))"
RUN R -e "BiocManager::install('tidyverse')"

# Install Seurat
RUN R -e "BiocManager::install('Seurat')"

# Install Signac
RUN R -e "remotes::install_github('stuart-lab/signac', ref = 'develop')"
```

My function have been put in a custom R package that needs to be downloaded from GitHub

```
Install custom R packages
RUN R -e "remotes::install_github('Maiolino-Au/')"
```

Create the Scripts folder and download the scripts from GitHub (for visualization purposes a space has been added)

```
#RUN mkdir -p /Scripts && \
  cd /Scripts && \
  curl -O
↪  https://raw.githubusercontent.com/Maiolino-Au/ProgrAppBioinfo_Exam/main/Scripts/
↪  First.R
```

end

```
ENV SHELL=/bin/bash
CMD jupyter lab --ip=0.0.0.0 --port=8888 --no-browser --allow-root
↪  --ServerApp.allow_origin='*' --ServerApp.token='' #last one disables the token
```

Runinng the Docker container

Command to run the container

for windows

```
@echo off
REM Ottieni il percorso corrente
set "CURRENT_DIR=%cd%"

REM Avvia Docker e monta la cartella come /sharedFolder
docker run -it --rm -p 8888:8888 -v "%CURRENT_DIR%:/sharedFolder" maiolino_exam2025
```

for linux

```
docker run -it -rm -p 8888:8888 -v "$(pwd)":/sharedFolder maiolino_exam2025
```

Part 2: Data processing and analysis

Step 0: Setup

Load the necessary R Packages

```
library(Seurat)
library(Signac)
library(Matrix)
library(readr)
library(ggplot2)
library(data.table)
library(GenomicRanges)
library(dplyr)
library(rtracklayer)
# library()
```

Create folders

For a better organization let's create, unless they are already present, some folders for the data and the results.

```
dir_data <- "/sharedFolder/Data"
if (!dir.exists(dir_data)) {
    dir.create(dir_data)
}

dir_data_raw <- paste0(dir_data, "/Raw_Data")
if (!dir.exists(dir_data_raw)) {
    dir.create(dir_data_raw)
}

dir_results <- "/sharedFolder/Results"
if (!dir.exists(dir_results)) {
    dir.create(dir_results)
}
```

Step 1: Downlad the data, load the sparse natrix and convert it to a full matrix

Download the data

```
a <- 1
```

unzip & organize

```
a <- 1
```

Load the data

```
data_path <- "data/matrix/"

matrix <- readMM(
  file = paste0(data_path, "matrix.mtx.gz")
  )

barcodes <- read_tsv(
  file = paste0(data_path, "barcodes.tsv.gz"),
  col_names = F,
  show_col_types = F
  )

features <- read_tsv(
  file = paste0(data_path, "features.tsv.gz"),
  col_names = c("id", "name", "type", "chr", "start", "end")
  )
```

Generate the full matrix as a data.table

first we give names to the col and row and transform the object "matrix" into a matrix

```
colnames(matrix) <- barcodes$X1
rownames(matrix) <- features$id
matrix <- as.matrix(matrix)
```

then we transform the object into a data.table, adding a column with the id for each row

```
dt <- as.data.table(matrix)
dt[, id := rownames(matrix)]
setcolorder(dt, c("id", setdiff(names(dt), "id")))
```

Step 2: Split gene expression and ATAC-seq

The datased we are using contains both expression data and ATAC-seq, we need to separate them. These two different type of data will have a different nomenclature in their id:

- Genes will have an Ensemble id, with begins with "ENSG"
- Peack will begin with the chromosome to which they have been mapped and their id will start with "chr"

With grepl we can generate a logic which will allow us to select only for one or the other

```
dt_genes <- dt[grepl("ENSG", dt$id)]
dt_atac <- dt[grepl("chr", dt$id)]
```

Step 3: Summarize the data

```
# Expression data
genes_summary <- dt_genes[, rowSum(.SD), .SDcols = -"id"]
names(genes_summary) <- dt_genes$id

# ATAC-seq
atac_summary <- dt_atac[, rowSum(.SD), .SDcols = -"id"]
names(atac_summary) <- dt_atac$id
```

Step 4: Create GenomicRanges objects

```
a <- 1
```

Step 5: Gene annotation for ATAC-seq peaks

```
a <- 1
```

Step 6: Finalize expression data

```
a <- 1
```

Step 7: Data Normalization and Integration

```
a <- 1
```

Step 8: Data Visualization

```
a <- 1
```

Part 3: Results

plots

Part 4: Generation of the R Markdown file

```
rmarkdown::render("Maiolino_Au.Rmd")
```