SingleCell Workshop

Yuanhua lab

2021-07-02

Contents

Chapter 1

Introduction

Cell interaction and Cell differentiation trajectory

- Cell interaction with ligand and receptors
- Pseudo-time and trajectory analysis
- RNA velocity

Cellular genetics

- mtDNA variants for lineage inference from single-cell omics
- Copy number variation estimation from scRNA-seq
- Nuclear SNV analysis in single-cell omics

Chapter 2

Prerequisites

2.1 (Optional) Install Windows Subsystem for Linux

Note, this is for Windows users only. While some required softwares only support Linux or macOS, you could install WSL to use Linux inside Windows.

If you already have Linux or macOS, you can skip this section and jump directly to the section of `conda installation` for Linux or for macOS.

The whole process of installing WSL requires at least 2G disk space. Note that this process was tested on Windows 10 (Version 2004, build 19041.1052).

2.1.1 What is the Windows Subsystem for Linux (WSL)?

According to the Microsoft Docs,

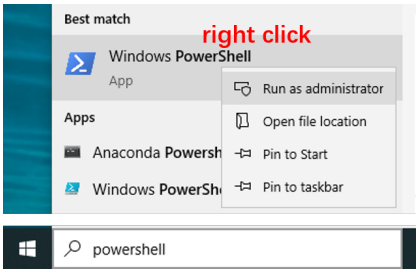
> *"The Windows Subsystem for Linux lets developers run a GNU/Linux environment – including most command-line tools, utilities, and applications – directly on Windows, unmodified, without the overhead of a traditional virtual machine or dualboot setup."*

2.1.2 Manual Installation Steps

Step 1. Enable required feature in Windows PowerShell

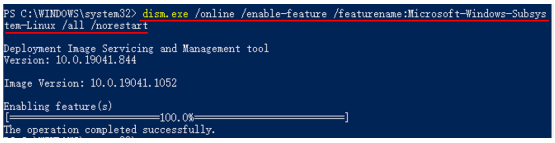
It is necessary to enable the required feature for WSL before installing it.

- Type `powershell` in the search box of the Windows taskbar.
- Right click `Windows PowerShell` and select `Run as administrator`.



- Type the command below in PowerShell.

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /r
```

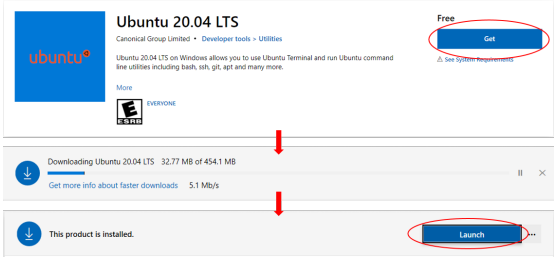


Step 2. Download and install WSL

Microsoft now supports several Linux distributions as WSL, such as Ubuntu, openSUSE, Fedora, etc (a full list here), among which we choose Ubuntu as an example.

Ubuntu WSL could be freely downloaded and installed through Microsoft Store.

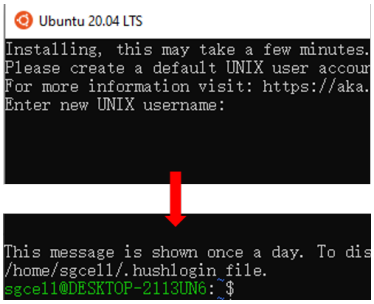
- Go to the webpage for Ubuntu in Microsoft Store.
- Click on the `Get` button.
- Wait for completion of downloading and installation.
- Click on the `Launch` button.



Step 3. Create a new account for Ubuntu

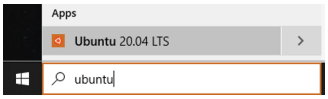
After successfully installing Ubuntu, a new user account should be created.

- Type user name and password following the prompts on the screen. Note, it is normal that the password is invisible when you are typing.



Now Congratulations! You have successfully installed and set up Ubuntu in your Windows System!

Next time you can re-open Ubuntu through the search box of the Windows taskbar.



More information about the usage of WSL can be found at Microsoft Docs.

2.2 Install Conda Environment

The whole process of installing conda environment requires at least 1G disk space.

2.2.1 What is conda?

Conda is part of the Anaconda platform. According to the Anaconda docs,

> *"Anaconda Individual Edition is a free, easy-to-install package manager, environment manager, and Python distribution with a collection of 1,500+ open source packages with free community support. Anaconda is platform-agnostic, so you can use it whether you are on Windows, macOS, or Linux."*

For quick installation and configuration, we choose to install Miniconda instead of the whole Anaconda.

> *"Miniconda is a free minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, the packages they depend on, and a small number of other useful packages, including pip, zlib and a few others."*

2.2.2 Installation on Windows

Although it is easy to install Miniconda itself on Windows, there are several conda softwares (e.g., bcftools, cellsnp-lite) required for this workshop that are well-supported only on Unix-Like systems. Linux or macOS is essential to use those softwares.

Good news is that Microsoft has provided the Windows Subsystem for Linux (WSL), which enables you to use several Linux distributions on Windows 10. The installation of WSL is easy and details can be found here.

After installing WSL, you could follow Section Installation on Linux to install Miniconda on WSL.

2.2.3 Installation on Linux

This section is adapted based on Miniconda installation guides for Linux. In this section, we select the Miniconda with Python 3.9 as an example.

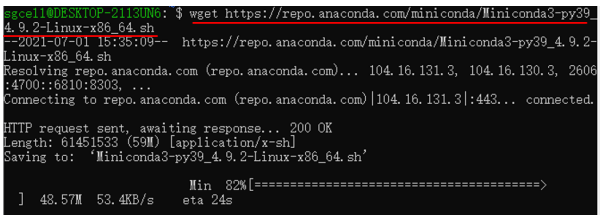
Step 1. Open Shell

- Open Linux Shell / WSL Shell (see Section Install WSL for details of WSL).

Step 2. Download Miniconda installer

The Miniconda installer is a `.sh` file containing metadata of installation. Run `wget <installer_link>` to download the installer. e.g.,

- Type in Shell `wget https://repo.anaconda.com/miniconda/Miniconda3-py39_4.9.2-Linux-x8`

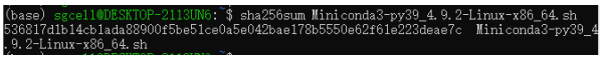


You could choose other Miniconda installers based on your platform. A full list of installers can be found here.

Step 3. Verify your installer hashes (Optional)

An unmatched hash indicates there were some error during the downloading process. We could simply verify the installer hashes by typing `sha256sum <installer_file>` in the Shell. e.g.,

- Type `sha256sum Miniconda3-py39_4.9.2-Linux-x86_64.sh` and check if the output hash is `536817d1b14cb1ada88900f5be51ce0a5e042bae178b5550e62f61e223deae7c`.



A full list of hashes can be found here.

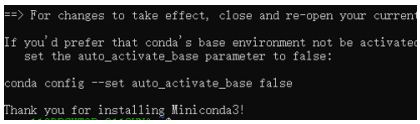
Step 4. Initialize Miniconda

Run `bash <installer_file>` in Shell. e.g.,

- Type `bash Miniconda3-py39_4.9.2-Linux-x86_64.sh`.

- Follow the prompts on the installer screens.

When you are asked `Do you wish the installer to initialize Miniconda3 by running conda init`, we recommend "yes". When the initialization is done, it looks like,



Step 5. Close and re-open Shell

- Close and then re-open your Shell, to make the changes take effect.

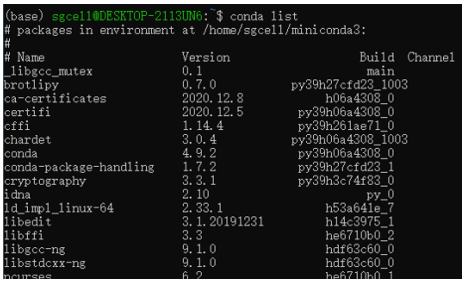
The screen should have a new `(base)` prefix, which looks like,

It means the Shell is now in the `base` environment of conda.

Step 6. Test your installation

- Type `conda list` in Shell.

A list of installed packages appears if it has been installed correctly.



2.2.4 Installation on macOS

You could refer to the Section Installation on Linux as the processes of installing Miniconda on macOS and Linux are quite similar. We briefly list the steps here for your quick reference.

Step 1. Open macOS Terminal

Step 2. Download Miniconda installer

- Type in Terminal `wget https://repo.anaconda.com/miniconda/Miniconda3-py39_4.9.2-MacO`

Step 3. Verify your installer hashes (Optional)

- Type `shasum -a 256 Miniconda3-py39_4.9.2-MacOSX-x86_64.sh` in Terminal and check if the output hash is `b3bf77cbb81ee235ec6858146a2a84d20f8ecdeb61467803`

Step 4. Initialize Miniconda

- Type in Terminal `bash Miniconda3-py39_4.9.2-MacOSX-x86_64.sh`.
- Follow the prompts on the installer screens.

When you are asked `Do you wish the installer to initialize Miniconda3 by running conda init`, we recommend "yes".

Step 5. Close and re-open Terminal

Step 6. Test your installation

- Type `conda list` in Terminal.

A list of installed packages appears if it has been installed correctly.

2.3 other prepare

2.3.1 Conda Configuration

To simplify the configuration of conda environment, we have pre-compiled a conda environment, named `sgcell`, containing all required softwares for this workshop. The corresponding metafile `sgcell.yml` of this environment was exported, based on which we can easily re-build the pre-compiled environment.

Assuming you have installed conda and activated any environment in the Shell / Terminal (if not, please install conda based on Section Install Conda Environment first). Then,

- Open Linux Shell / macOS Terminal / WSL Shell
- Add conda channels,

```
conda config --add channels bioconda
conda config --add channels conda-forge
```

- Re-build the pre-compiled environment,

```
conda env create -f sgcell.yml
```

- Activate the new environment `sgcell`,

```
conda activate sgcell
```

- Verify if the new environment was installed correctly,

```
conda env list
```

The `sgcell` sould appear in the listed environments.

2.3.2 Example of R Markdown

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter @ref(RNA velocity).

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

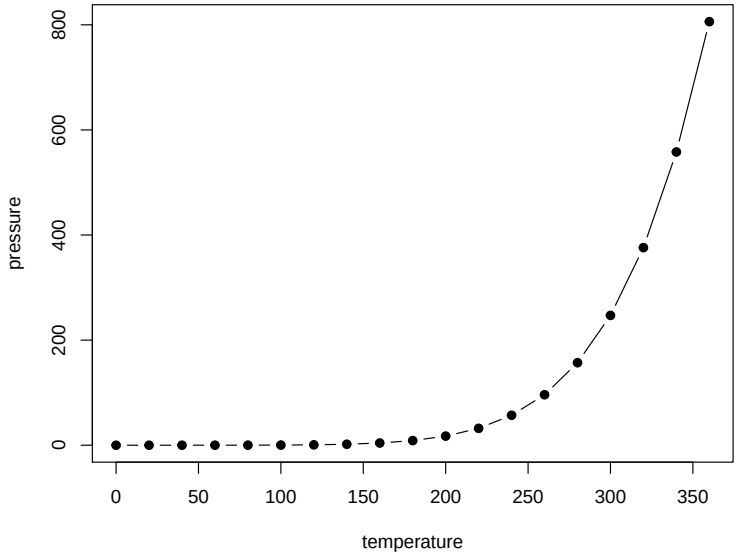


Figure 2.1: Here is a nice figure!

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2021) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports, e.g., a math equation $a^2 + b^2 = c^2$.

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

Chapter 3

Pseudo-time and trajectory analysis

Here is a review of existing methods.

Chapter 4

RNA velocity

We describe our methods in this chapter.

Chapter 5

Copy number variation estimation from scRNA-seq

Rongting Huang

2021-06-29

5.1 Method: inferCNV

InferCNV: Inferring copy number alterations from tumor single cell RNA-Seq data

5.1.1 install inferCNV

Software Requirements

- JAGS
- R (>3.6)

In order to run infercnv, JAGS (Just Another Gibbs Sampler) must be installed.

Download JAGS from https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/ and install JAGS in your environment (windows/MAC).

If you use inferCNV on server, install JAGS via `conda install` in your conda environment is recommended.

```
conda install -c conda-forge jags
```

More details refer to inferCNV wiki page

Five options for installing inferCNV

Option A: Install infercnv from BioConductor (preferred)

```
if (!requireNamespace("BiocManager", quietly = TRUE))
     install.packages("BiocManager")
BiocManager::install("infercnv")
```

For more other options, refer to Five options for installing inferCNV

Data requirements

- a raw counts matrix of single-cell RNA-Seq expression
- an annotations file which indicates which cells are tumor vs. normal.
- a gene/chromosome positions file

File-Definitions

5.1.2 getting started

If you have installed infercnv from BioConductor, you can run the example data with:

```
library(infercnv)

infercnv_obj = CreateInfercnvObject(raw_counts_matrix=system.file("extdata", "oligodenc
                                    annotations_file=system.file("extdata", "oligodendr
                                    delim="\t",
                                    gene_order_file=system.file("extdata", "gencode_dow
                                    ref_group_names=c("Microglia/Macrophage","Oligodenc

infercnv_obj = infercnv::run(infercnv_obj,
                             cutoff=1, # cutoff=1 works well for Smart-seq2, and cutoff
                             out_dir=tempfile(),
                             cluster_by_groups=TRUE,
                             denoise=TRUE,
                             HMM=TRUE)
```

If you can run the getting started part with demo data provided by inferCNV, then it is installed successfully.

Demo Example Figure

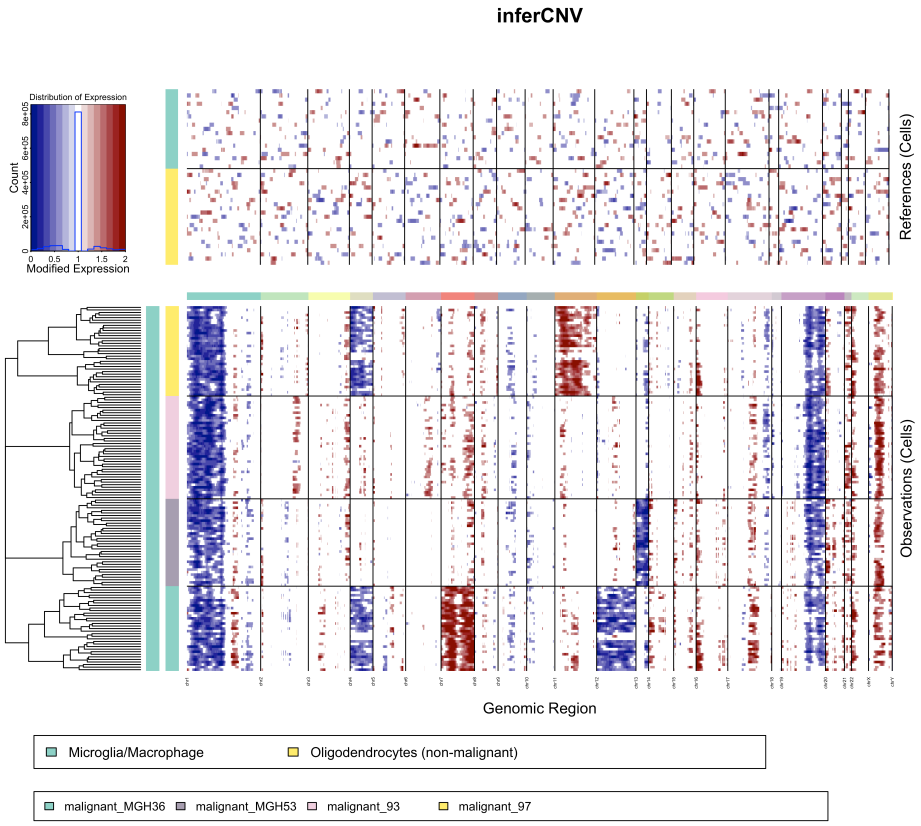


Figure 5.1: Demo Example Figure

5.2 Application on TNBC1

5.2.1 data description

TNBC1 is a triple negative breast cancer tumor sample of high tumor purity (72.6%) with 796 single tumor cells and 301 normal cells. The dataset is available on NCBI GEO under the accession number GSM4476486.

Table 5.1: Details of TNBC1 dataset (from published articles, copyKAT).

TNBC1	Number of clones	Number of tumor clones	Tumor clone-specific copy gain
Triple negative breast cancer	3	2	• C1: 4p, 7q, 9, 17q • C2: 3p, 6q, 7p, 11q, X

- Expression

Table 5.2: Subclusters of TNBC1 dataset (from gene expression analysis-Seurat).

Clone A	Clone B	Normal
488	307	302

Notes: 488,307, 247, 55

- B Allele Frenquency (BAF)

- BAF V.S. Expression

5.2.2 run inferCNV

data_download

demo1_log_file

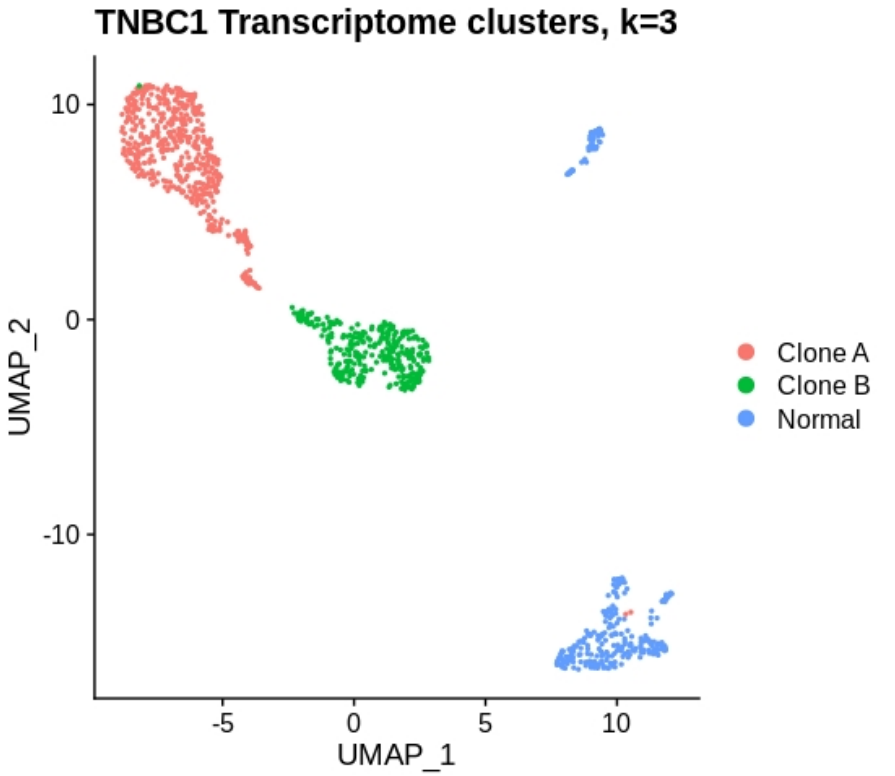


Figure 5.2: umap

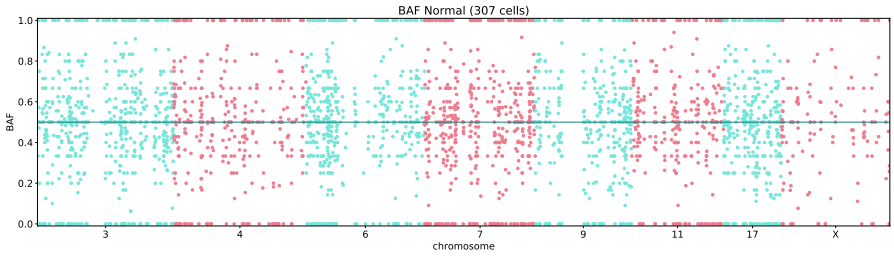


Figure 5.3: baf1

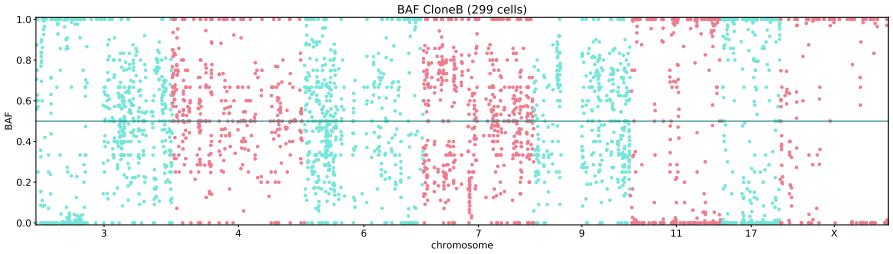


Figure 5.4: baf2

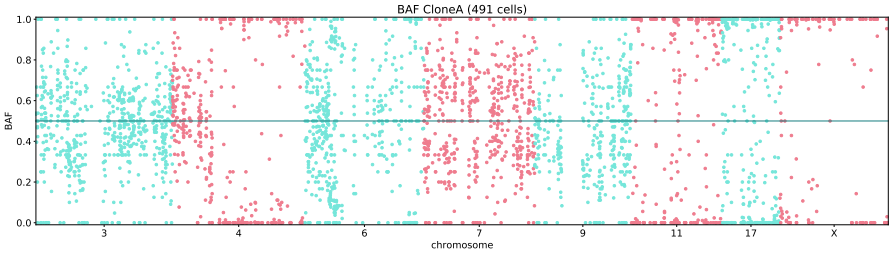


Figure 5.5: baf3

demo2_log_file

output_files

```
# library(infercnv)
# library(utils)
# library (BiocGenerics)

## DEMO1
#
# tnbc <- read.delim("C://Users/Rongting/Documents/GitHub_repos/combinedTNBC1.txt")
# anno <- tnbc[2,]
# anno <- t(anno)
# anno <- as.data.frame(anno)
#
# gex <- tnbc[-c(1:2),]
# gex <- type.convert(gex)
#
# gene_file <- "C://Users/Rongting/Documents/GitHub_repos/gene_note_noheader_unique.tx

#
# infercnv_obj = CreateInfercnvObject(raw_counts_matrix=gex,
```

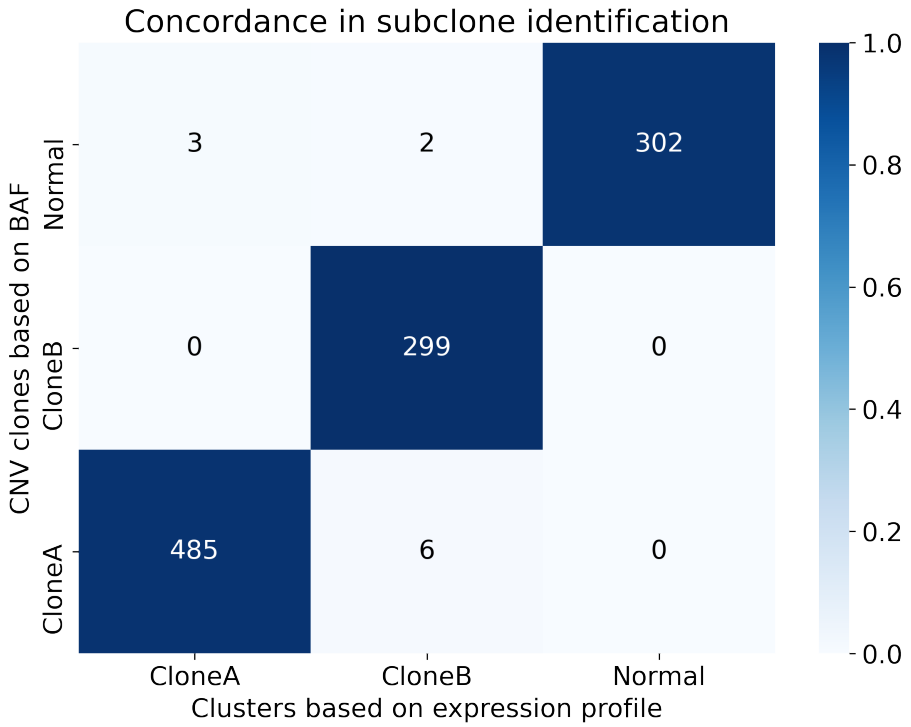


Figure 5.6: confusion heatmap

```
#                                       annotations_file=anno,
#                                       delim='\t',
#                                       gene_order_file=gene_file,
#                                       ref_group_names= "N")

# output = "C://Users/Rongting/Documents/GitHub_repos/tnbc1_demo"

# infercnv_obj = infercnv::run(infercnv_obj,
#                              cutoff=0.1,
#                              out_dir= output ,
#                              cluster_by_groups=T,
#                              denoise=T,
#                              HMM=T)

## DEMO2

# gene_file <- "C://Users/Rongting/Documents/GitHub_repos/gene_note_noheader_unique.tx
#
# anno_file <- 'C://Users/Rongting/Documents/GitHub_repos/tnbc-3cluster-id.txt'
#
# infercnv_obj2 = CreateInfercnvObject(raw_counts_matrix=gex,
#                                       annotations_file=anno_file,
#                                       delim='\t',
#                                       gene_order_file=gene_file,
#                                       ref_group_names= "Normal")
#
# output = "C://Users/Rongting/Documents/GitHub_repos/tnbc1_demo2"
#
# infercnv_obj2 = infercnv::run(infercnv_obj2,
#                              cutoff=0.1,
#                              out_dir= output,
#                              cluster_by_groups=T,
#                              denoise=T,
#                              HMM=T)
```

```
##################
##Notes
##################

## load the package
library(Seurat)
library(infercnv)

## prepare the data (cellranger output)
```

```
### load count matrix (example)
matrix_path <- "../cellranger/xxxx/count_xxxxx/outs/filtered_gene_bc_matrices/GRCh38/"


### read count matrix
gex_mtx <- Seurat::Read10X(data.dir = matrix_path)

### run inferCNV with loop
celltype = c('CloneA', 'CloneB', 'Normal')

for (i in celltype){
  infercnv_obj1 = CreateInfercnvObject(raw_counts_matrix=gex_mtx,
                                       annotations_file=anno_file,
                                       delim='\t',
                                       gene_order_file=gene_file,
                                       ref_group_names=c(i))

  output <- paste0('/groups/cgsd/rthuang/processed_data/inferCNV/xxxx/','xxxx_', i)

  infercnv_obj1 = infercnv::run(infercnv_obj1,
                                cutoff=0.1,
                                out_dir= output ,
                                cluster_by_groups=T,
                                denoise=T,
                                HMM=T)
}
```

5.2.3 inferCNV result

- demo1

- demo2

5.3 ref

https://www.r-bloggers.com/2012/04/getting-started-with-jags-rjags-and-bayesian-modelling/

```
bookdown::render_book("index.Rmd", "bookdown::gitbook")
```

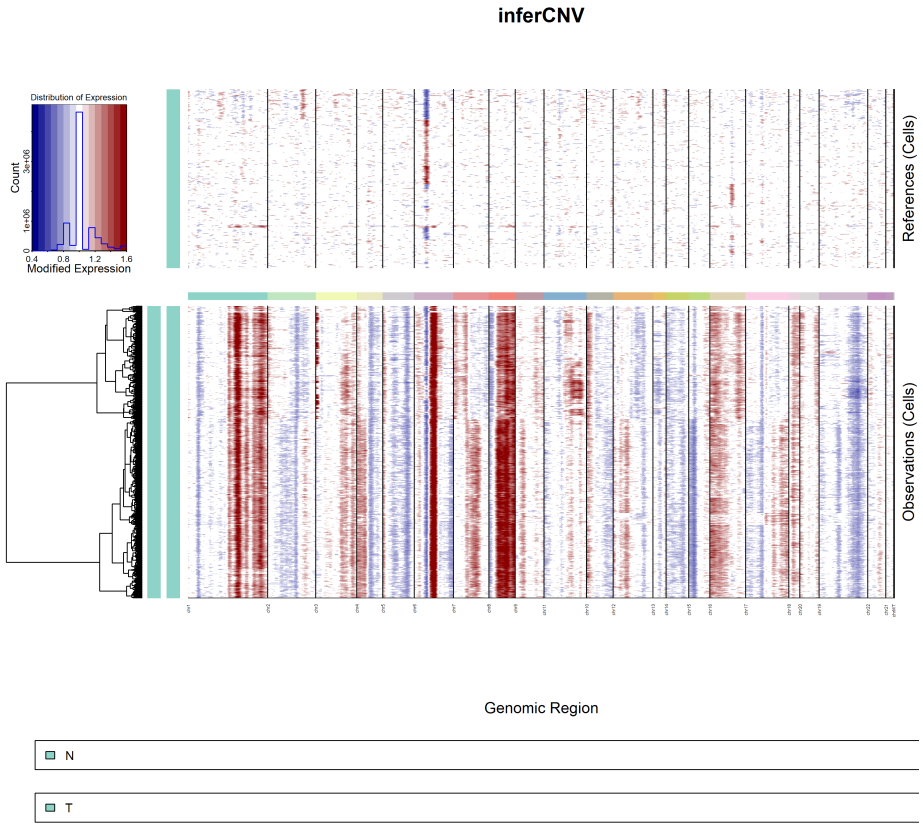


Figure 5.7: infercnv1

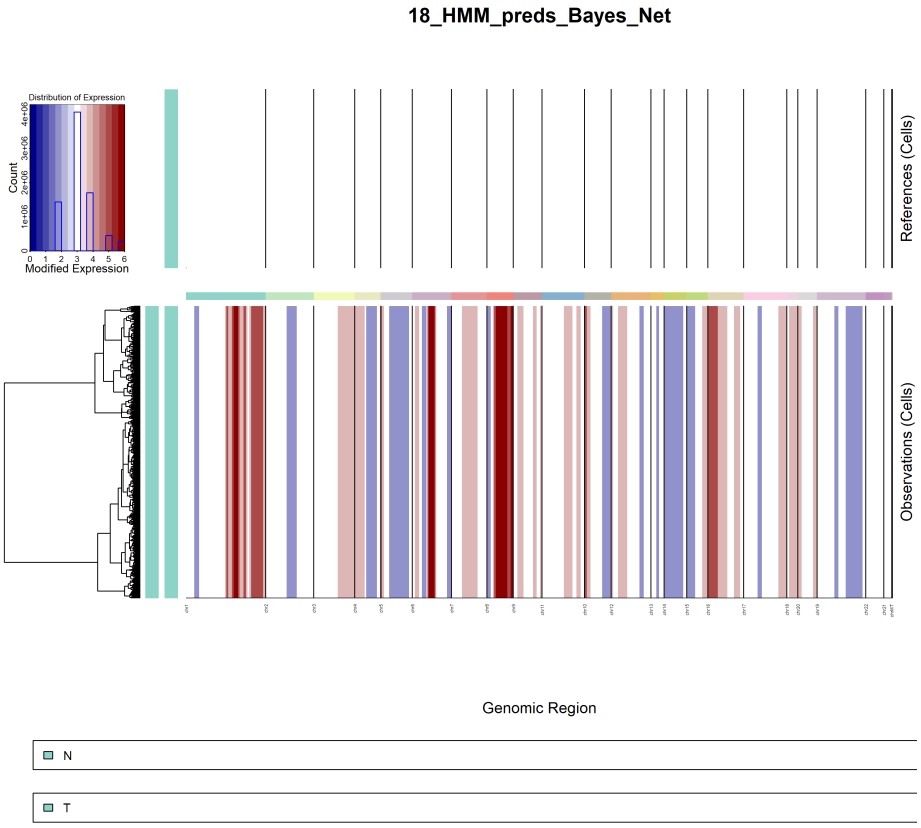


Figure 5.8: infercnv2

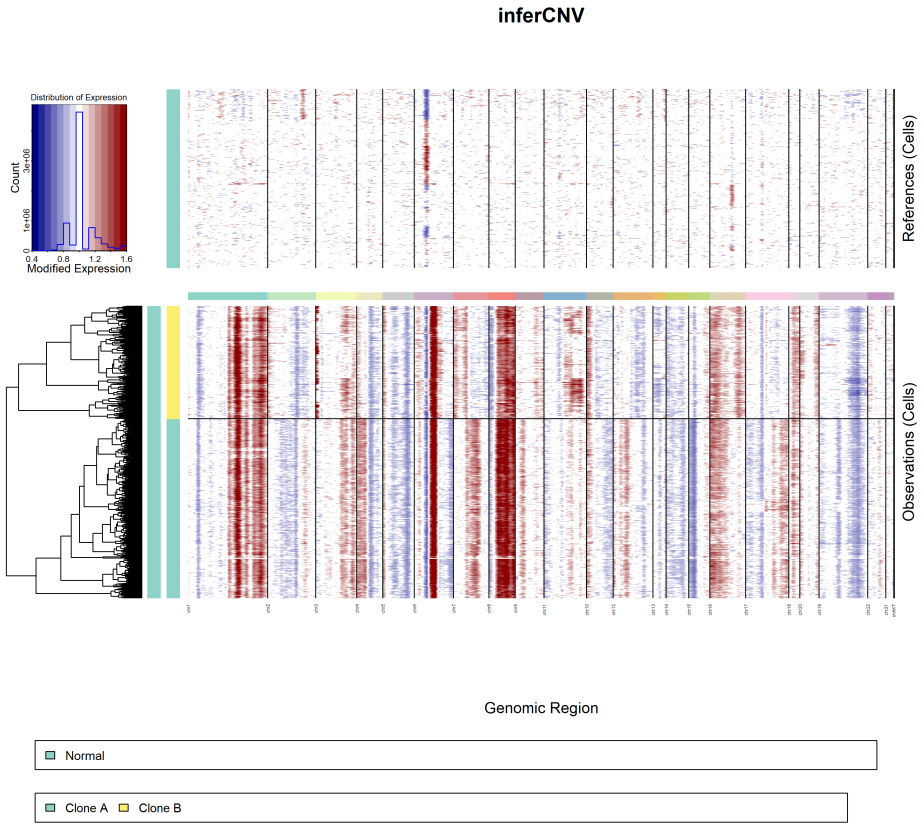


Figure 5.9: infercnv3

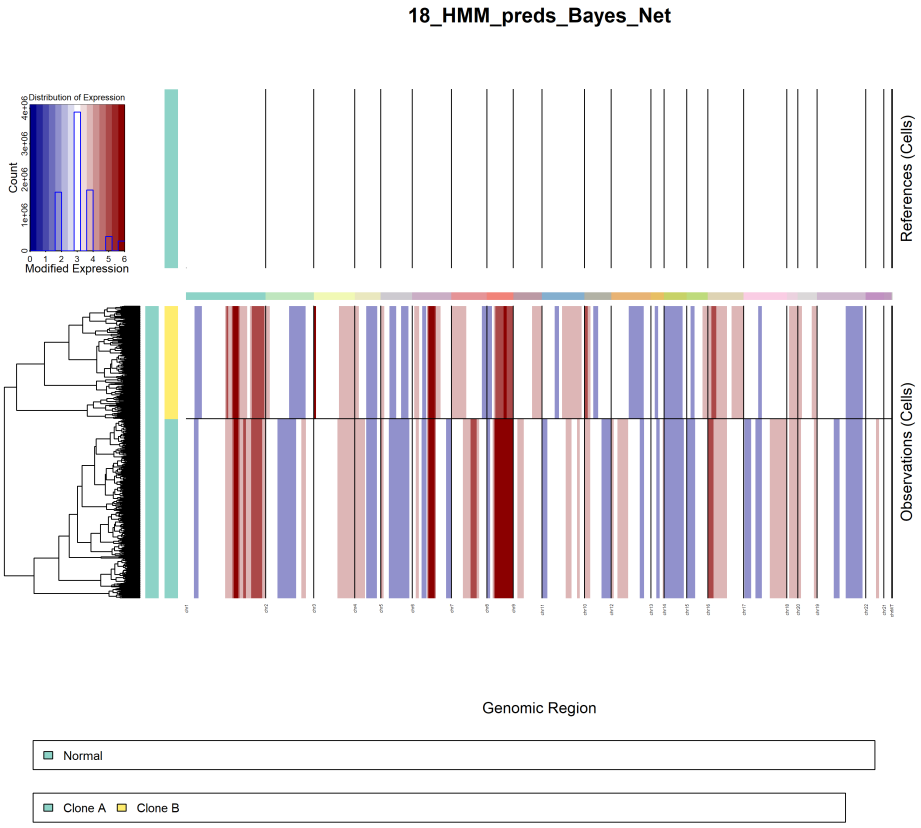


Figure 5.10: infercnv4

Chapter 6

mtDNA variants for lineage inference from single-cell omics

6.1 Pileup with cellsnp-lite

Cellsnp-lite is designed to perform efficient pileup and genotyping for both bulk and single cell sequencing data. It could be easily installed via conda with `conda install -c bioconda -c conda-forge cellsnp-lite`.

In this example, we use cellsnp-lite to pileup chrM on a SMART-seq2 dataset, whose output could be used as inputs of MQuad model for mitochondrial clone analysis.

Note that we have uploaded the pileup results of cellsnp-lite to sgcellworkshop repo on sourceforge. You can directly download the `joxm.hg19.cellsnp.mode2b.tar.gz` file and then run `tar zxvf joxm.hg19.cellsnp.mode2b.tar.gz` to unpack it.

If you want to repeat the pileup processing, please follow the steps below.

- Step 1. Download the bam files

The 77 SMART-seq2 bam files (~3.5G) were packed into one file `joxm.bam.all.77.tar.gz`. This file, together with another two files `joxm.hg19.bam.lst` and `joxm.sample.lst`, should be downloaded from sgcellworkshop repo on sourceforge. Please put the three files in the same directory.

- Step 2. Unpack the bam files

Unpack the bam files with `tar zxvf joxm.bam.all.77.tar.gz`. The bam files, together with the `.bai` files, should be in the `sort/` directory.

- Step 3. Run cellsnp-lite

```
cellsnp-lite      \
  -S ./joxm.hg19.bam.lst    \
  -i ./joxm.sample.lst      \
  -O ./cellsnp              \
  --chrom MT        \
  --cellTAG None    \
  --UMItag None     \
  --minCOUNT 20     \
  --minMAF 0        \
  --genotype        \
  --gzip
```

6.2 Clonal analysis with MQuad

We have finished a nice book.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2021). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.22.