Homework 0: Installing R

Last edited August 31, 2021

- Due September 10th, 5:00 pm EST
- Submit `hw0.Rmd` and `hw0.pdf` to canvas.
- Please post questions and discussion you have related to this homework on the Homework 0 discussion thread on Ed (link).

Overview

This assignment will show you how to write a report using `R` and Rmarkdown.

HW 0 might be challenging, especially if this is your first time using `R`. First, you'll need to install several pieces of software. That can be certainly be frustrating. If you get stuck, the TAs will help you troubleshoot. Please post the problem you are running into on Ed. You will also need to read and modify some `R` code. Doing that for the first time is challenging. HW 0 should help you to get started.

If you are already comfortable using `R` and Rmarkdown then this assignment will be simple. Download `hw0.Rmd`, modify one line of code, then render a `.pdf`. This "write then render" workflow is central to making reports in Rmarkdown.

In this course, you aren't required use `R` exclusively. Nevertheless, we will sometimes use `R`. For example, in precept we will discuss how ideas from lecture can be put into practice using `R`. Having some understanding of `R` will help you to follow these discussions.

Complete the class survey on Canvas

Please complete the course survey that is available on Canvas under the Lecture 1 module (link). The purpose of this survey is to assess your level of expertise in programming which will be useful for future assignments and help sessions. The survey will be graded only for completion.

Installing R and Rstudio

Here is a walk-through of installing R and Rstudio (link). Rstudio is currently the most popular integrated development environment (IDE) for working with R. We recommend that you use it in order to get started quickly. You'll need to install R before installing Rstudio.

If you get stuck installing R Getting stuck on this can be frustrating, but the TAs will help you troubleshoot. Please post the problem you are running into on the Ed discussion thread.

Alternatively, you can complete this assignment without installing R on your PC. Rstudio cloud can run R through a web browser (link). You can set up an account for free.

Navigating the Rstudio IDE

By default, the Rstudio IDE is split into several panes. You'll likely spend most of your time working with the *source editor*, which is the upper left-hand pane. From here you can write and evaluate R. To evaluate a line in the source editor type CTRL-Enter. This will send the line to the *console* and evaluate it. The *console* is below the *source editor*.

Depending on what you're doing in the *source editor* you may want to refer to some of the other panes.

- **getting help:** you can show documentation in the lower right corner. For example, evaluating `?mean` will show the documentation for the `mean` function.
- **refining a plot:** You can display a plot in the lower right-hand pane by clicking on the plots tab. The arrows at the top of this pane allows you to scroll between recent plots. For many R programmers, iterating back and forth between writing R and plotting is an important part of the workflow.
- **looking up variables:** If you want to check what a variable is named or get info on how it is structured, you can use the *environment* tab. `Environments` are used by R to associate names to values, hence the two columns of the *environment* tab.

Reproducible reports using Rmarkdown

In this class you'll be writing reports in Rmarkdown. Some helpful instructions include:

- a tutorial by RStudio (link).
- a more concise guide (link).

You can also use the `.Rmd` that generated this homework as a template. It is available on canvas.

Rmarkdown includes a light-weight set of markup elements that allow you to do some basic formatting of your document. For example, you can easily specify

- **bold text**,
- and lists.

The key feature of Rmarkdown is the ability to incorporate code and computation into your writing. To describe your methods you could directly show the code you used to generate the results. You could express those same ideas in an equally valid way using written/spoken language. However, Rmarkdown can seamlessly connect results to their underlying methods. We'll see some examples of this later in the assignment. Bottom line, Rmarkdown is designed to facilitate effective communication both between programmers and to a broader audience.

Installing Rmarkdown

To compile an Rmarkdown document you need to install the `rmarkdown` package in `R`. This package has its own set of dependencies including `latex` and `pandoc`. If you are using Rstudio, it will automatically install `pandoc`. Otherwise, you'll need to install these dependencies yourself. *R markdown: the definitive guide* includes a chapter with instructions for installing `rmarkdown` (link).

Rendering an Rmarkdown document

Once you have the `rmarkdown` package installed you have a few options for rendering an Rmarkdown document.

- If you use the Rstudio IDE then you can simply click a button to render your document. This button looks like a ball of yarn and is above the source editor. The shortcut to render is Ctrl+Shift+K.

- Alternatively, you can do this from the command line. First move to the directory that contains your Rmarkdown source code. Next render the document using something like

```
Rscript -e "rmarkdown::render('./hw0.Rmd')"
```

This assignment was written using a `.Rmd` file. The later parts of the assignment will ask you to modify it. Before making changes you should download the `.Rmd` from canvas and render it to produce a `.pdf` file.

Yeast cell cycle

We'll begin by downloading a dataset that examines changes in yeast gene expression over the course of the cell cycle. This is a famous dataset that has been analyzed numerous times. In this dataset, at each of 14 time points as the yeast were dividing, the authors measured the levels at which 6178 yeast genes were expressed. We will learn more about gene expression and these technologies later in the class. More abstractly, we can think of the measured data as a 6178 by 14 matrix of real numbers.

First we download the dataset (from Bioconductor, using the code below), and format it for plotting. You don't need to modify the following code chunk. Don't worry if it doesn't all make sense to you, we will go over any parts you need to know in office hours and/or help sessions.

```
## set up for downloading data from bioconductor
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
if (!requireNamespace("yeastCC", quietly = TRUE))
    BiocManager::install("yeastCC")
library(yeastCC)

## consider only the elutriation experiment
timecourse_keep <- "elu"
is_elu <- pData(yeastCC)[, "Timecourse"] == timecourse_keep
exp <- yeastCC[, is_elu]
covariates <- pData(exp)

## format data for plotting
phase <- covariates$Phase  # growth phase of the yeast cells
phase <- factor(phase, levels = as.character(unique(phase)))
exp <- exprs(exp)  # log2 fold change in gene expression vs asynchronous culture
                                        # ^^^ rows index genes
                                        # ^^^ columns index time
time <- gsub("elu", "", covariates$Timepoint) # units: minutes after elutriation
```

From a computer science perspective, `exp` is a matrix with elements of type `numeric`. It also includes some some metadata giving names to rows and columns. This metadata consists of a list containing two character vectors. `time` is a vector of type `character`. To get a quick summary of a variable you can use the "structure" command, for example `str(yeastCC)`.

At a high level, what we've done so far is to take an `ExpressionSet` object, `yeastCC`, and use it to get a matrix and a vector. The `ExpressionSet` object is important because it helps bioconductor packages to interface to one another. Although, for our specific goal of making a plot we only need a few pieces of the original object.

From a biology perspective, `exp` represents gene expression measurements with rows corresponding to transcripts and columns corresponding to samples. Before being observed, these yeast were cell-cycle synchronized by a process called elutriation. Elutriation separates cells according to size (Rosebrock, 2017). At time zero, the smallest cells are present. These cells correspond to the G1 phase of the cell cycle. At each timepoint, RNA abundance was measured relative to unsynchronized cells in exponential growth.

These measurements were made to get a more comprehensive understanding of the cell-devision cycle. This experiment increased the number of mRNAs known to be cell cycle dependent by eight fold. The 800 genes identified in this experiment are about a tenth of all yeast genes.

Homework: modifying a plot

You'll be making a modification to the plot below. Currently the x-axis represents column indexes from the data matrix. To make the plot more scientifically informative, modify the x-axis to represent the the timepoint of the measurement.

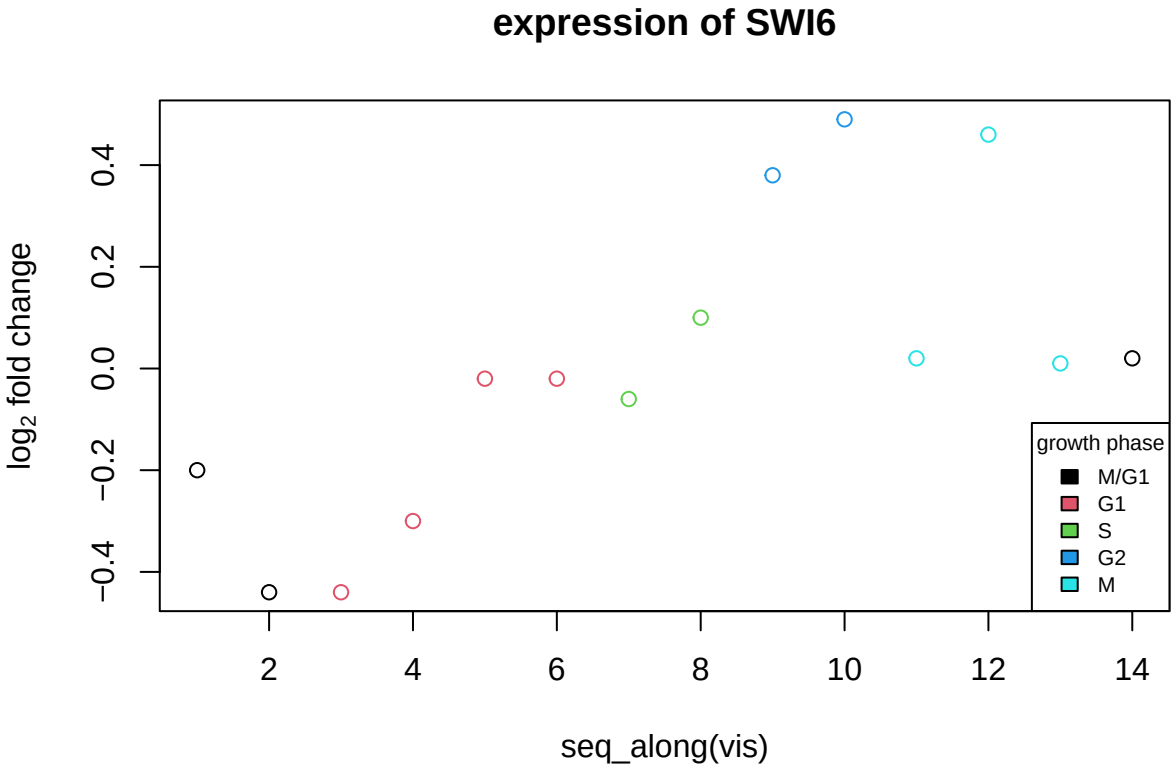
The information you need is in the variable `time` below. For example, consider the i^{th} timepoint. `vis[i]` represents the expression observed at time `time[i]`. To read documentation for the `plot` function you can run `?plot`.

```
time  # note the variable TIME is already available
```

```
##  [1] "0"   "30"  "60"  "90"  "120" "150" "180" "210" "240" "270" "300" "330"
## [13] "360" "390"
```

```
## examine expression of a gene regulating the cell cycle
vis_systematic <- "YLR182W"  # SWI6 regulates G1 to S transition
vis_standard <- "SWI6"
vis <- exp[vis_systematic, ]

plot(x = seq_along(vis),  # TODO modify this line
     y = vis, col = phase,
     main = paste("expression of", vis_standard),
     ylab = expression(paste("log"[2], " fold change")))
legend("bottomright", legend = levels(phase),
       fill = sort(unique(phase)),
       cex = 0.7,
       title = "growth phase")
```



The arguments of the plot function are:

- x: a vector of x-axis coordinates. Here `seq_len` is being used to construct a vector 1, 2, . . . , `length(vis)`.
- y: a vector of y-axis coordinates. Here this is the observed log fold-change in RNA abundance.
- main: this text will form the title of the plot. The function `paste` is being used here to make the plot simpler to modify. If someone updates `vis_standard` then the title of the plot will reflect this change. It would be better to map the systematic name to the standard name, rather than depending on the user to keep them consistent. For the sake of simplicity, we'll take the approach above.
- ylab: this is the y-axis label. `expression` is being used to format the subscript nicely.

Submitting the assignment You can complete this assignment by modifying only the line marked `TODO` above. Once you modify the plot above, render your `.Rmd` file to produce a `.pdf`. There are instructions for doing this in the *Rendering an Rmarkdown document* section above. Submit both your `.Rmd` and `.pdf` file to canvas.

References

Spellman, P.T., et al. 1998. Comprehensive identification of cell cycle–regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Molecular biology of the cell, 9(12), pp.3273-3297.

Rosebrock, A.P., 2017. Synchronization of budding yeast by centrifugal elutriation. Cold Spring Harbor Protocols, 2017(1), pp.pdb-prot088732.

```
sessionInfo(package = NULL)
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
```

```
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] yeastCC_1.32.0      Biobase_2.52.0      BiocGenerics_0.38.0
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.27       magrittr_2.0.1      evaluate_0.14
##  [4] highr_0.9           rlang_0.4.11        stringi_1.7.3
##  [7] rmarkdown_2.10      tools_4.1.1         stringr_1.4.0
## [10] xfun_0.25           yaml_2.2.1          compiler_4.1.1
## [13] BiocManager_1.30.16 htmltools_0.5.1.1   knitr_1.33
```