Session 1: Data Assembly

Samuel P Callisto

June 7, 2018

Contents

Using R Markdown

Useful resource: R Markdown Cheatsheet https://www.rstudio.com/resources/cheatsheets/

Output in LaTeX

Surrounding \theta_{ij} with "$" will generate θ_{ij}

Useful keyboard shortcuts

(Mac Users use command instead of Ctrl)

Ctrl-Alt-I: Insert new chunk

Ctrl-Enter: Run current line of code (multiple if highlighted)

Ctrl-Shift-C: Comment/uncomment selected lines

Importing data

```
## load packages
library(dataTools)
library(tidyverse)

## Excel files
excel <- read.csv("datasets/TPM_sim_dataset_20180607.csv", as.is = T, stringsAsFactors = T, header = T)
```

Can also use file.choose() or absolute filepath (instead of relative to working directory) to select files you want to import.

Arguments to modify output

include, warning, message, echo, etc.

I always import ALL datasets and packages in my first chunk of code. This way you don't need to hunt down the line where data was imported, just re-run the first chunk.

Data types in R

There are multiple ways to represent data in R. Most of the time we work with numbers, and R uses different names for different types of numbers:

- integer or int: any whole number.
- numeric or num: numbers with a decimal. Also called double.

When we are working with text, there are multiple ways to represent them in R as well:

- character or chr: basic representation of a string
- Factor: R's representation of an enumerated data type (set of named levels)

Most of the time we want to work with text as a character because they are easier to manipulate. Numbers can also be treated as a character, but this is typically undesirable because we cannot perform any arithmetic manipulations on numbers when they are in this format.

```
var <- as.integer(3)
str(var)
```

```
##  int 3
```

```
var2 <- var*2
str(var2)
```

```
##  num 6
```

```
var.c <- as.character(var)
str(var.c)
```

```
##  chr "3"
```

A third data type is the logical (or boolean) type (TRUE or T, FALSE or F). These are typically generated using logical tests.

Vectors contain the same data type. A vector of vectors create a matrix or a data.frame. Most of the time you will use data.frames to store your data since each vector can contain a different data type.

```
vec.1 <- 1:5
str(vec.1)
```

```
##  int [1:5] 1 2 3 4 5
```

```
vec.2 <- c("a", "b", "c", "d", "e")
str(vec.2)
```

```
##  chr [1:5] "a" "b" "c" "d" "e"
```

```
vec.3 <- c(TRUE,FALSE,F,F,T)
str(vec.3)
```

```
##  logi [1:5] TRUE FALSE FALSE FALSE TRUE
```

```
df <- data.frame(vec.1, vec.2, vec.3)
names(df) <- c("numbers", "letters", "vowels")

str(df)
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ numbers: int  1 2 3 4 5
##  $ letters: Factor w/ 5 levels "a","b","c","d",..: 1 2 3 4 5
##  $ vowels : logi  TRUE FALSE FALSE FALSE TRUE
```

```r
## convert from Factor to character data type
df$letters <- as.character(df$letters)
str(df)
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ numbers: int  1 2 3 4 5
##  $ letters: chr  "a" "b" "c" "d" ...
##  $ vowels : logi  TRUE FALSE FALSE FALSE TRUE
```

Some comments about reproducibility & readability

The importance of readability

A bad example of readability

```
write.csv(headr("datasets/TPM_sim_dataset_20180607.csv") %>%
  select(subjectid) %>%
  left_join(headr("datasets/Baseline_sim_dataset_AllSubjects_20180607.csv"),
            by= "subjectid"), "datasets/output.csv", row.names = F)
```

Can anyone guess what this line of code is accomplishing?

A good example of readability

```
## import data
baselineData <- read.csv("datasets/Baseline_sim_dataset_AllSubjects_20180607.csv")
topiramateData <- read.csv("datasets/TPM_sim_dataset_20180607.csv")

## create completers subset from topiramateData
completers <- select(topiramateData, subjectid)

## filter to keep only completers
filteredBaseline <- left_join(completers, baselineData, by= "subjectid")
```

```
## Warning: Column `subjectid` joining factors with different levels, coercing
## to character vector
```

```
## save filteredBaseline
# write.csv(filteredBaseline, "datasets/Baseline_sim_dataset_CompletersOnly_20180607.csv", row.names = F
```

Tip: Use "camel case" to separate words within a variable name by capitalizing the first letter of a new word, e.g. baselineData. This applies not only to coding within R, but also maintaining a proper audit trail for all your files!

Naming convention: for version control, best method of including dates is to use yyyymmdd, as this will sort chronologically when sorted alphabetically

Always write your code as though it were being read by a human, not a computer!

In addition to liberal commenting throughout your code, do not underestimate the importance of creating descriptive variable names. If a future graduate student takes over your project after you've graduated, will they be able to decipher what your code does without calling you to ask?

Version control

What's that mean?

Everytime you modify a file, you have created a new "version" of it. Version control allows you to keep a record of all the changes that you have made to a file. This way if you break a script that you have written, you can easily go back to the old version.

GitHub glossary

Commit: File changes saved to your local machine

Push: File changes saved to your online database

Pull: Bring online changes to your local workspace

Fork: Create a personal copy of someone else's work

Branch: Workspace to try implementing new feature without worrying about messing up existing function

Repositories

Can be either public or private. On github.com with a free account you can ONLY create public repositories. These can be seen by anyone. Enterprise or paid version of GitHub allows you to create private repositories which can only be seen by you and collaborators of your choice.

UMN GitHub

Students have free access to an Enterprise version of GitHub through UMN. Allows users to have private repositories. Users outside UMN cannot see anything in this account, but you can transfer repositories from the UMN GitHub to a public GitHub account.