R Tutorial

Rachel Levin, Gantavya Pahwa

September 1, 2021

Installing Base R and RStudio

See instructions on Canvas on how to download R and RStudio.

Navigating the RStudio interface

```
4 + 5     # hit Ctrl + Enter (Cmd + Enter) to run a line of code to the console
```

```
## [1] 9
```

Creating an R script

The first step is to set your working directory. You can do this by typing `setwd(filepath)` at the console, or clicking `Session` -> `Set Working Directory` -> `Choose Directory`, or navigating to your directory in the `Files` pane and then clicking `More` -> `Set as Working Directory`.

To check your current directory, either run the command `getwd()` to the console or navigate to the Files window in the bottom right pane.

Installing and loading R packages

Packages are a fundamental component of programming in R. R is an open source language, which enables programmers to create collections of functions, data sets, and other helpful tools that all R users can install and implement within their code. In this class, all of the data mining methods we will use will come from packages.

```
#install.packages("ggplot2")   # only run ONCE ever to install the package
library(ggplot2)               # run at the top of each new notebook to load the package
```

Basic R Programming

Data types

```
"a"     # character
8       # numeric
TRUE    # logical
```

```
TRUE + FALSE
```

```
## [1] 1
```

```
8 - TRUE
```

```
## [1] 7
```

Variable assignment

```
x <- c(5,6,7)                    # a vector is a sequence of data elements of the same type
y <- seq(from=1, to=3, by=1)     # alternatively, you can write this as seq(1,3)

z <- c(x,y)                      # combine two vectors into a third vector
z
```

```
## [1] 5 6 7 1 2 3
```

```
x + 3*y
```

```
## [1]  8 12 16
```

```
x.short <- c(5,6)
x.short
```

```
## [1] 5 6
```

```
x.short + y                    # adding two vectors of unequal lengths will result in an error
```

```
## Warning in x.short + y: longer object length is not a multiple of shorter object
## length
```

```
## [1] 6 8 8
```

Subsetting

Often times we will want to select one or multiple elements from a data storage object. There are several ways to do so.

```
# Numerically
z[1]
```

```
## [1] 5
```

```
z[c(1,3)]
```

```
## [1] 5 7
```

```
z[-1]
```

```
## [1] 6 7 1 2 3
```

```r
# With logicals
indices <- c(TRUE,TRUE,FALSE)
x[indices]
```

```
## [1] 5 6
```

Operations on variables and vectors

```r
s <- c(1,4,9,16,25,36,49)
sqrt(s)
```

```
## [1] 1 2 3 4 5 6 7
```

```r
log(s)
```

```
## [1] 0.000000 1.386294 2.197225 2.772589 3.218876 3.583519 3.891820
```

```r
# It is often helpful to compare the lengths of vectors
length(s)
```

```
## [1] 7
```

```r
# You can also subset vectors based on conditional criteria
s[s<10]
```

```
## [1] 1 4 9
```

R has built-in functions such as `sqrt()` and `log()` work elementwise, meaning they perform the operation on each element of the input vector.

R also has a variety of built-in functions for basic calculations and summary statistics.

Examples: `sum()`, `max()`, `min()`, `mean()`, `sd()`, `any()`, `all()`

Random number generation

```
rnorm(10)     # default parameters: mean=0, sd=1
```

```
##  [1] -0.4224126 -0.7722823  1.2019881  1.3434270  0.9656939  0.5674342
##  [7]  0.2484041  2.0225114  0.9719954  0.7693574
```

```
rexp(10)      # default parameters: lambda=1
```

```
##  [1] 1.14929449 0.04153129 0.16054885 0.09921146 0.06955087 0.30304114
##  [7] 0.97456446 1.76191551 0.44129349 0.50568495
```

```
runif(10)     # default parameters: min=0, max=1
```

```
##  [1] 0.81989368 0.89262669 0.89279340 0.51387899 0.56739820 0.89327035
##  [7] 0.39483431 0.86369885 0.05064259 0.95346874
```

Loops and iteration

```
r = c()
for (i in s){
  print(sqrt(i))
  r = c(r,sqrt(i))
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
```

```
r
```

```
## [1] 1 2 3 4 5 6 7
```

Matrices

A matrix is a collection of data elements of the same type organized in a 2-dimensional structure. Create a matrix using the function `matrix()`, with parameters data (input data), nrow, ncol, and byrow.

```
sample_data <- seq(1,12)                          # define data
mat <- matrix(data=sample_data, nrow=3, ncol=4)   # default parameter byrow=FALSE
dim(mat)                                          # matrix dimensions
```

```
## [1] 3 4
```

```
mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
matrix(sample_data, nrow=3, ncol=4, byrow=TRUE)   # fill matrix row-wise
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
```

```
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

Subsetting matrices

```
mat[1,]       # subset row 1
```

```
## [1]  1  4  7 10
```

```
mat[,3]       # subset column 3
```

```
## [1] 7 8 9
```

Matrix-vector multiplication

```
vec <- rep(x=1, times=4)       # create a vector with the number 1 repeated 4 times

mat %*% vec                    # multiplication of the matrix and vector
```

```
##      [,1]
## [1,]   22
## [2,]   26
## [3,]   30
```

Per matrix multiplication rules, make sure that the number of columns of the first matrix is equal to the number of rows of the second matrix (or vector).

Getting help

```
?matrix
help(matrix)
```

Other resources for getting help: Google, Piazza

R Markdown Overview

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. In STAT 471 we will output our code to PDF format for assignment submissions.

R Markdown serves two main purposes:

1. Lab notebook: keep track of work in progress, as an alternative to typing it in the console.
2. Report generation: a nice way of integrating text, code, and results in a single report.

In this class, we will be using R Markdown for the homework, midterm, and final project.

Embedding R Code

Include R code in your R Markdown file using either code chunks or inline code.

Code Chunks and Inline Code

Embedding enables you to output both the code and its output. The output can be a printed result, a figure, or a table. You can also write code inline with your text, which is more reproducible than copying and pasting the result. If you change something about the calculation, then the output of the inline code will change as well. Wrap inline code in single tick marks (‘).

This is how to produce an R code chunk and inline code:

````
```{r}
a <- seq(1,5)
a^2
```

The average of the vector a created above is `r mean(a)`.
````

Display Options

The R package `knitr` provides options to adjust how you want to display your output.

echo: whether to display code along with its results

warning: whether to display warnings

cache: whether to cache results for future renders. Note: caching does not check if the preceding code chunks have changed. To do smart caching (so that the chunks are re-run only if chunks on which they depend are changed), use the "dependson" chunk option. More information available at R4DS 27.4.4.

fig.width: width in inches for plots created in the code chunk

fig.height: height in inches for plots created in the code chunk

Document Organization

All submitted R Markdown documents should contain a YAML header at the top of the file, including title, author, and date. This is what a YAML header looks like:

```
---
title: "Introduction to R Markdown"
author: "[Insert Name Here]"
date: "2021-01-20"
---
```

In addition, good R coding practice includes frequent and clear use of headings throughout the document. The code

```
# Header 1

## Header 2

### Header 3

#### Header 4
```

produces

Header 1

Header 2

Header 3

Header 4

Text Formatting

You can also adjust text font. The code

```
plain text

*italics*

**bold**
```

produces

plain text

italics

bold

Inserting LaTeX Equations

LaTeX is a document preparation system designed for neatly writing mathematical and statistical equations. You can either write an equation in line with the text (wrapped in single dollar signs), or alternatively on its own separate line (wrapped in double dollar signs).

Inline: `$y_i = \beta_0 + \beta_1x_i + \varepsilon_i$ for $i = 1,...,n$` produces $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ for $i = 1, ..., n$

Separate Line:

The code `$$ \sigma^2_x = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2$$`

produces

$$\sigma_x^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2$$

Previewing and Exporting to PDF

Preview in Viewer Pane

Select "Preview in Viewer Pane" option for convenient previewing of the document inside RStudio. Then, click "Knit to HTML" to run the whole document and see what it will look like.

Export to PDF

Assignment submissions must be in PDF format. When you are ready to submit, click **Knit to PDF** to get a PDF output.

When you click the Knit to PDF button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

To generate PDF output, we will need to install LaTeX. Run the following two commands in the console once:

```
install.packages("tinytex")
tinytex::install_tinytex()
```

Additional Help

Base R Cheatsheet: https://rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf

R Markdown Cheatsheet: https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf