class_11.R

moffer

2024-01-03

```
# Getting started in R

# Download R: https://www.r-project.org/
# Download RStudio: https://www.rstudio.com/products/rstudio/download
# To compile reports, you'll need a LaTeX installation as well.

## This file is an R script. ----

# When working in R, it's preferable to write your code in a script, so that
# you have a record of all the commands you ran in order, and can reproduce
# your work.

# Comments have the hashtag before them; this text is not evaluated by R.

# Code is evaluated and output is reported in the console.
3
```

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

```
# You can treat R as a calculator.
1 + 1
```

```
## [1] 2
```

```
1 - .4556
```

```
## [1] 0.5444
```

```
1e3 / 3i
```

```
## [1] 0-333.3333i
```

```
# R has some built in functions.
log(4)
```

```
## [1] 1.386294
```

```
sin(-pi)
```

```
## [1] -1.224647e-16
```

```
atan(3 * pi)
```

```
## [1] 1.465089
```

```
sqrt(49)
```

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

```r
# You can read a function's documentation by putting the question mark before it
?log


# You can also assign quantities to named objects; you'll see these objects in
# the environment.
a <- 2 # using the assignment operator, `<-`
a
```

```
## [1] 2
```

```r
3 * a
```

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

```r
1 / a
```

```
## [1] 0.5
```

```r
## Reading data into R (and working directories) ----

# When you open R directly it opens to your home directory.
getwd() # (this will look different when we compile vs. using R interactively)
```

```
## [1] "/Users/moffer/Documents/Git/SOSC13200-W24/in-class"
```

```r
# (What is a working directory? What is directory structure?)

# You can reset the working directory to the location of the script you are
# working in, by going to:
# Session > Set Working Direction > To Source File Location

# Check the working directory again to see if it's different.
getwd()
```

```
## [1] "/Users/moffer/Documents/Git/SOSC13200-W24/in-class"
```

```r
# Best practice is to open R from a script every time. When you open R from a
# script, R opens to the directory that the script is in. That way, you're
# always working in the right place.
# When you compile an R script, it works the same way.


# Read some data in.
dat <- read.csv("../data/card-krueger.csv", as.is = TRUE)

# Look for the data in the Environment pane.

# We keep files for a given project nearby each other, so we can load data in
# using *relative* file paths.
# (What is the difference between *relative* and *absolute* file paths?)

# Check out the file pane. You can navigate around it.

# Check out the data.
head(dat)
```

```
##   id nj d d_nj bk kfc roys wendys co_owned centralj southj pa1 pa2   fte   ft
## 1  1  0 0    0  1   0    0      0         0         0      0   1   0 40.50 30.0
## 2  2  0 0    0  0   1    0      0         0         0      0   1   0 13.75  6.5
## 3  3  0 0    0  0   1    0      0         1         0      0   1   0  8.50  3.0
## 4  4  0 0    0  0   0    0      1         1         0      0   1   0 34.00 20.0
## 5  5  0 0    0  0   0    0      1         1         0      0   1   0 24.00  6.0
## 6  6  0 0    0  0   0    0      1         1         0      0   1   0 20.50  0.0
##     pt mgrs wage meal hrsopen bonus ncalls status type inctime firstinc nregs
## 1 15.0    3   NA 2.58    16.5     1      0      1    1      19       NA     3
## 2  6.5    4   NA 4.26    13.0     0      0      1    1      26       NA     4
## 3  7.0    2   NA 4.02    10.0     0      0      1    1      13     0.37     3
## 4 20.0    4  5.0 3.48    12.0     1      0      1    1      26     0.10     2
## 5 26.0    5  5.5 3.29    12.0     1      0      1    1      52     0.15     2
## 6 31.0    5  5.0 2.59    12.0     0      2      1    1      26     0.07     2
```

```
## Compiling reports ----
# We will compile reports for homework assignments using a method called
# "spinning." This shows your R code evaluated from start to finish, and shows
# that your results are reproducible.
```

Spinning

When spinning, we can use a different kind of comment, roxygen comments, to do more advanced formatting. I'm switching to roxygen comments now. The formatting only shows up when we compile the document.

We can make text **bold** or *italicized*.

You can also create headers and sub-headers for sections.

And, we can write code inline and have it evaluated using backticks. For example: 4.

```
# This doesn't work in standard comments: `r a + a`
```

You can also write equations.

For example, we can write equations inline if we put it between two dollar signs, `$`. For example, $Y = \alpha + \beta_0 + \epsilon$.

You can also write multi-line equations:

$$\begin{aligned} \int_0^1 x^2 \, dx &= \frac{1}{3}x^3 \Big|_0^1 \\ &= \frac{1}{3}1^3 - \frac{1}{3}0^3 \\ &= \frac{1}{3} \end{aligned}$$

$$\begin{aligned} X &= \frac{15+30}{2} - \frac{15+20+20+10+15}{5} \\ &= 6.5 \end{aligned}$$

A convenient and helpful "cheatsheet" for LaTeX math is here: http://reu.dimacs.rutgers.edu/Symbols.pdf

A tool on the web that might be helpful for writing expressions in a relatively intuitive way is here: http://www.codecogs.com/latex/eqneditor.php?lang=en-en, where you click on symbols and things, and the latex code comes out in the yellow box.

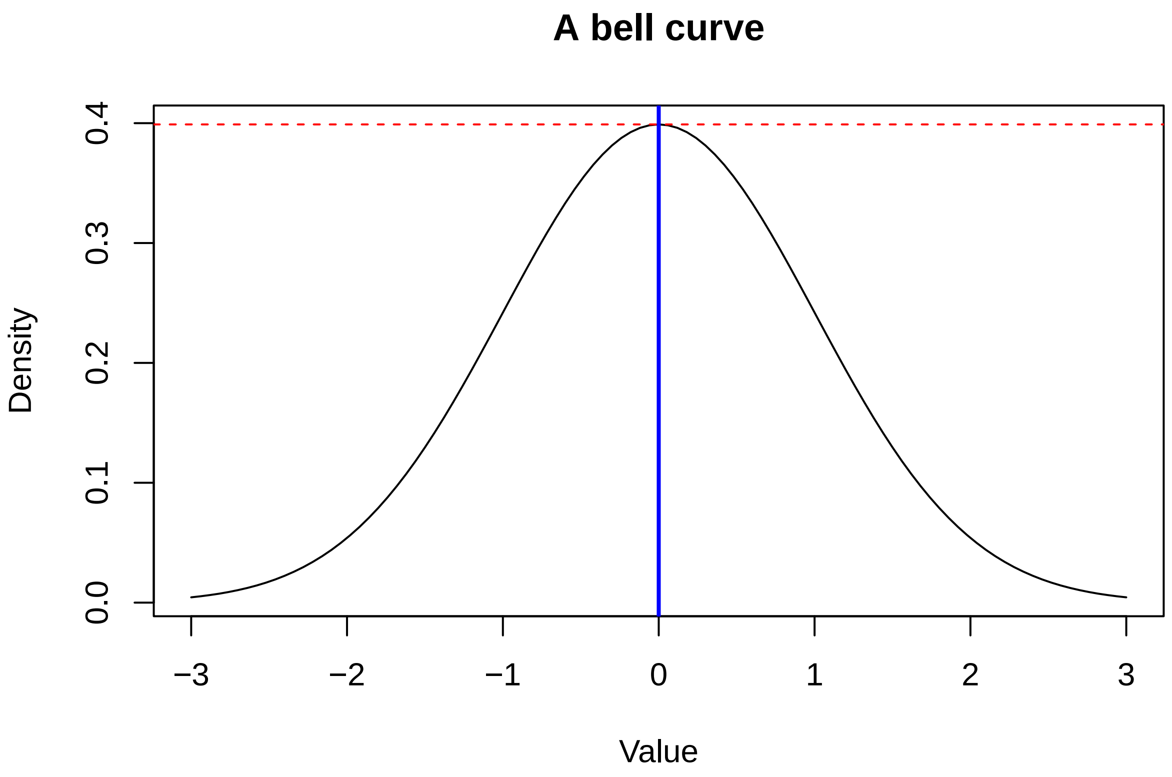
You can also include images, using the `knitr::include_graphics` function.

```
knitr::include_graphics("../assets/tree.jpeg")
```

We set the scale of the picture using the out.width argument (here, it's 50%) of the page.

And we can include plots generated in R.

```r
plot(dnorm, -3, 3, main = "A bell curve", ylab = "Density", xlab = "Value",
     type="l")
abline(v = 0, lwd = 2, col = "blue") # a vertical line
abline(h = dnorm(0), lty=2, col = "red") # a horizontal dotted line
```



A note on compiling: Reports are compiled using a knitr function called "spin." You can look up `knitr::spin` if you want to learn more options for your text and code presentation. Knitr uses Markdown formatting, which you can look up for things like styling text and creating lists.

Let's compile the document now.