Econ21410: Software

John Eric Humphries
Oliver Browne*

Updated: March 31, 2014

For this course, you are required to use R. You are also required to learn how to use LyX or another LaTeX editor. This note gives you some starting points, but user manuals and the internet are also great places to find explanations, tutorials and examples.

1 LaTeX and LyX

This document was created using the markup language LaTeX and Texmaker, a text editor specifically designed for use with LaTeX. If you do not already know how to make a document like this, we highly recommend that you invest some time in teaching yourself this skill. Being able to produce well-structured documents with mathematical equations in them will make it easier for you to save your work for later and to make corrections to it.

You have three options for using LaTeX. You can choose to use LyX, which is a WYSIWYG TeX editor. LyX doesn't require you to learn the TeX syntax, and therefore is the recommended choice for those who have not used LaTeX before. Installing LyX is easy, and it is free. Alternatively, you can choose one of the many TeX editors out there, most of which are also available free of charge. These editors allow you to work on a `.tex` file, which they then compile into a `.pdf` for you. To be able to do this, you will need to learn the syntax. Installing an editor requires you to first install a TeX distribution (recommended: MiKTeX for Windows, MacTeX for Mac OS, TeX Live for Linux/UNIX), and possibly additional packages. You will also need a PDF viewer to see your output. There are some start-up costs here, but it is worth the effort if you intend to take more advanced econ and math courses. A third option is to use an online editor. This doesn't require any installation, but it is not recommended because you can easily lose your work.

We will provide you with templates of `.lyx` and `.tex` files through Chalk. Lyx also comes with predefined templates. Play around with the templates a little bit until you understand how it works. You can also use the practice problems below.

Links for downloading the relevant software:

- Download LyX.
- A comparison of TeX editors (Wikipedia). Texmaker is recommended.
- The LaTeX wikibook: installation. This is a great resource in general, but this chapter will probably answer your questions about installation.
- Download Texmaker. Note that this requires a LaTeX distribution to be installed - see previous link if you have questions about this.

Some resources that you might find useful after you have installed an editor:

- The LaTeX wikibook.
- The not so short introduction to LaTeX by Tobias Oetiker.
- Detexify: recognizes handwritten symbols and returns suggestions for LaTeX code.
- GmailTeX.

*These prepatory materials are adapted from material prepared by Winnie Van Dijk for Econ 209

2 R and RStudio

R is a programming language and environment that is developed specifically for statistical use. It is designed for matrix handling, data storage, analysis and visualization. We recommended that you use RStudio, which is an open-source graphical interface for R. Both R and RStudio can be downloaded for free and are available for Windows, Mac, and Linux/UNIX.

Links for downloading the relevant software:

- Download R.
- Download RStudio.

Additional resources:

- An Introduction to R by W.N. Venables, D.M. Smith and the R Core Team.
- Google's R Style Guide: guidelines for syntax, naming of variables, and organization of your code.
- Stack Overflow. A great forum for programming questions.
- R reference card by Tom Short.

Example 1: Numerical integration - what you can do in R

Here's a simple example of how easy it is to use R for doing something that seems, at first sight, to be complicated: approximating an integral. We will go over some basic R commands and demonstrate how to use RStudio.

Suppose that we have a random variable $X \sim \mathcal{N}(0,1)$. We want to calculate, for $b > a$,

$$Pr\left(a < X < b\right) = \int_a^b \phi(x)dx,$$

where ϕ is the appropriate PDF. We know the expression for this PDF:

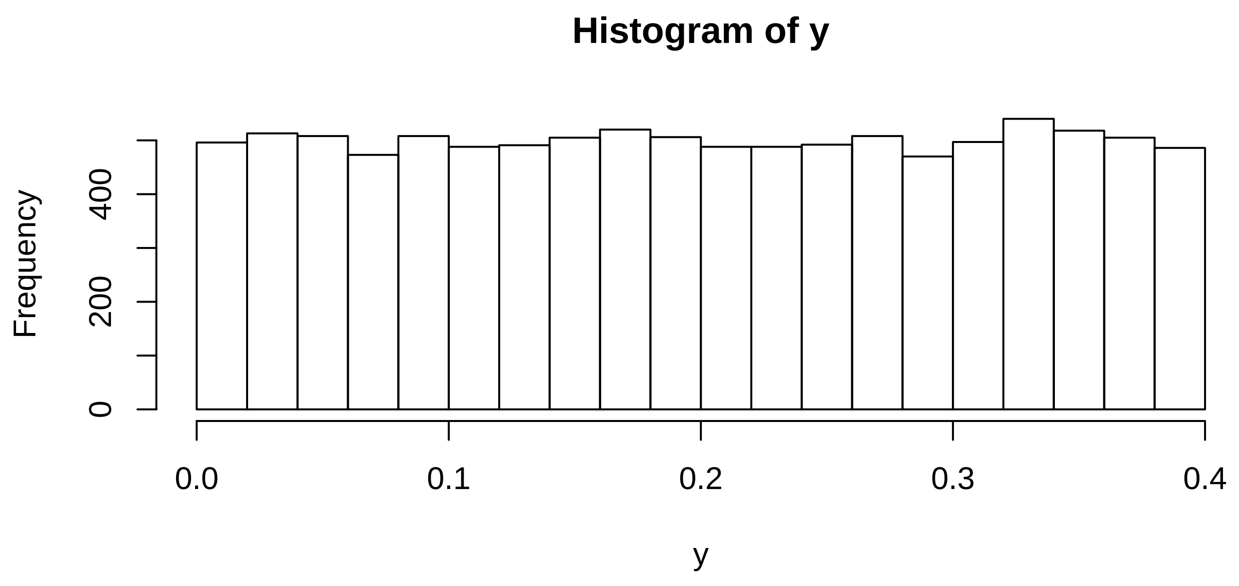
$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2}\right\}.$$

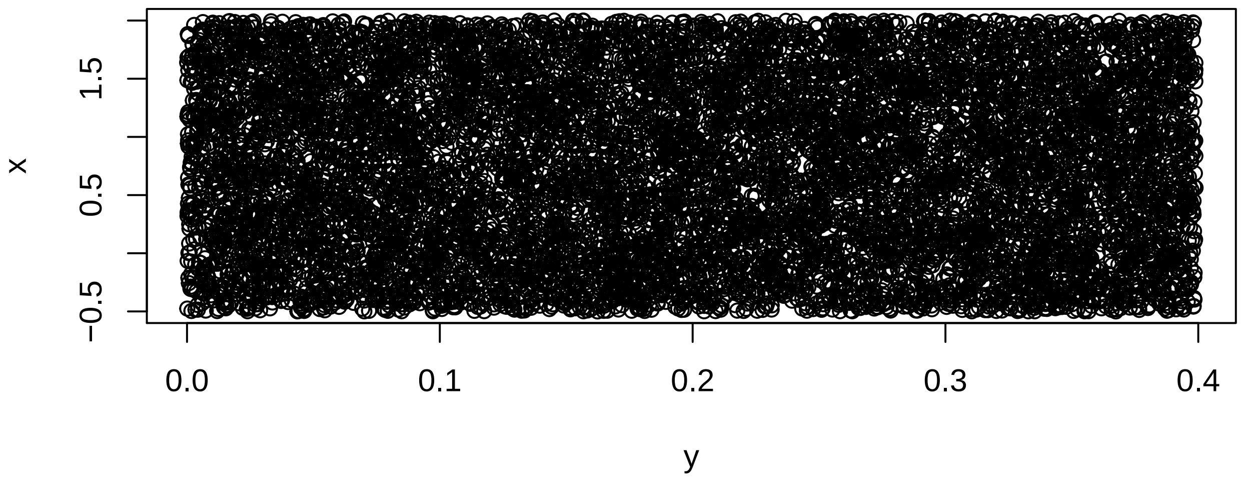
Let $a = -0.5$ and $b = 2$. We don't really want to calculate the integral. Instead, we can use simulation to approximate it. The idea is to "throw darts" at a rectangle of known area that encloses the area under the graph. In R, we can easily asses what fraction of the darts has landed in the area that we want to approximate (the fraction that has landed below the PDF), and take this fraction of the total known area as our approximation.

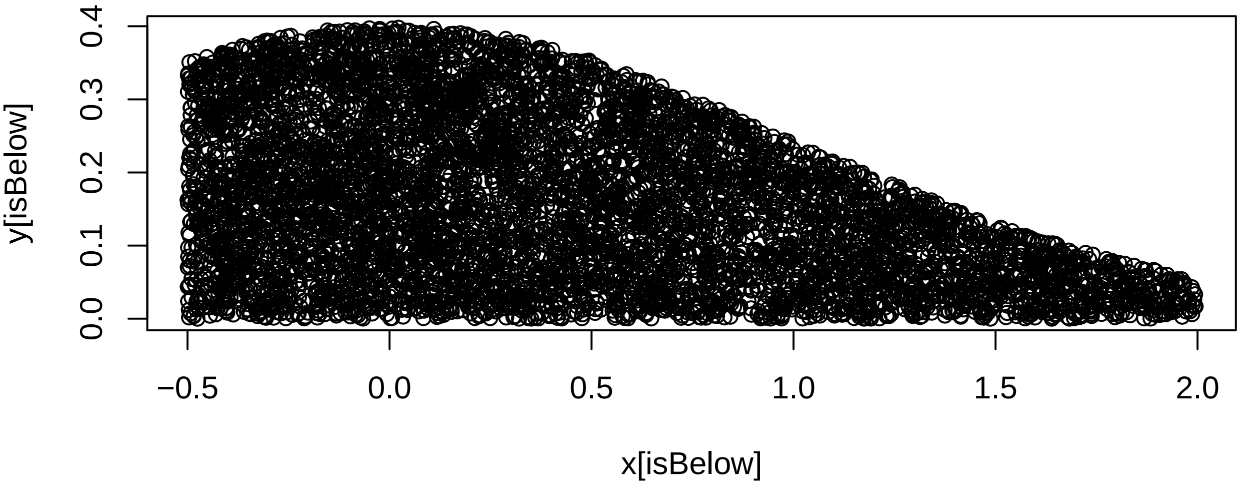
We simulate M random (x, y)-pairs on the rectangle $[a,b] \times \left[0, \frac{1}{\sqrt{2\pi}}\right]$.

```r
### EXAMPLE 1: NUMERICAL INTEGRATION ### WINNIE VAN DIJK DECEMBER 28, 2013

## Initialization setwd('C:/Users/WiNNiE/Dropbox/Teaching/Econ 209 TA/Review sessions') #
## Set the working directory

## Set parameters
set.seed(39)  # fix the randomization so that you get the same pseudo-random numbers next time you run t
M <- 10000  # set the number of simulated (x,y)-pairs
a <- -0.5  # set the left bound
b <- 2  # set the right bound
l <- 0  # set the lower bound
u <- 1/sqrt(2 * pi)  # calculate the upper bound
area <- (b - a) * (u - l)  # calculate the area of the rectangle

## Simulation
x <- runif(M, a, b)  # simulate random x-values from U[a,b]
y <- runif(M, l, u)  # simulate random y-values from U[l,u]
```

```
hist(y)  # create a histogram of the y-values
plot(y, x)

## Approximation
isBelow <- y < 1/sqrt(2 * pi) * exp(-x^2/2)  # create a vector with ones in the positions where the (x,y
fraction <- sum(isBelow)/(M)  # calculate the fraction of times the (x,y) pair is in the area that we wa
approximation <- fraction * area
approximation

## [1] 0.6645

## Scatter plot
plot(x[isBelow], y[isBelow])  # plot only those values that fell below the PDF

## Check the outcome
pnorm(b) - pnorm(a)  # use built-in functions to check the approximation

## [1] 0.6687
```







A few things to notice here:

- Setting the working directory
- Notation: see Google's R Style Guide
- A trick when working with matrices and vectors is to precede the name of a matrix by the letter 'm', and a vector by the letter 'v'. More in general, you could let the first letter of the variable name specify the object type to help you interpret more easily what your code is doing or why it is not working (but you don't have to do this).
- Comments and organization of code
- Variable assignment
- Operations on vectors: R adapts to the object it is asked to work on, so check whether it is doing what you want it to do (particularly for vectors and matrices).
- Setting the randomization seed
- Pseudo-random number generators

- Indexing of vectors
- Histogram, scatter plot
- Built-in probability functions `pnorm`, `dnorm`, `rnorm`, `qnorm`, `punif`, `dunif`, `runif`, `qunif`, `pexp`, etc. Use the help function to get details on their syntax.

In the TA session, we will go over the following in a demonstration of RStudio:

- Graphics window
- Object explorer
- Workspace (can be saved)
- Command history
- R Help
- Packages
- Command line
- Setting the working directory
- Writing `.R` files and sourcing code.

Example 2: Numerical integration - writing a function

In a continuation of the previous example, this example illustrates how to write a function in R.

Why would we want to write a function? Suppose that we want to reuse the code from the previous example to repeatedly approximate

$$Pr\left(a < X < b\right) = \int_a^b \phi(x)dx,$$

for general, user-specified, $b > a$ and M. Then it is convenient to write a function that takes as input a, b, and M.

```r
NumInt <- function(M, a, b) {

    # Computes an approximation to Pr(a<X<b), where X is standard Normally distributed.  INPUT:
    # M, the number of simulated (x,y)-pairs, integer a, left bound, double b, right bound,
    # double OUTPUT: approximation, double

    ## Error handling
    if (a > b) {
        print("Error: need a < b")
        return()
    }

    ## Set parameters
    l <- 0  # set the lower bound
    u <- 1/sqrt(2 * pi)  # calculate the upper bound
    area <- (b - a) * (u - l)  # calculate the area of the rectangle

    ## Simulation
    x <- runif(M, a, b)  # simulate random x-values from U[a,b]
    y <- runif(M, l, u)  # simulate random y-values from U[l,u]

    ## Approximation
    isBelow <- y < 1/sqrt(2 * pi) * exp(-x^2/2)  # create a vector with ones in the positions where the
    fraction <- sum(isBelow)/(M)  # calculate the fraction of times the (x,y) pair is in the area that w
    approximation <- fraction * area
    return(approximation)
}
```

After sourcing this code (look for the function name in the workspace to see if this was successful), we can call the function from the command line, or from another R file:

```r
NumInt(M = 10000, a = -0.5, b = 2)

## [1] 0.6677
```

Things to notice here:

- Comments: make it easy for another person to read your code.
- It is good practice to include error handling in your function. For example, the user might specify $a < b$, for which this code will return an error message.
- R suppresses intermediate output in a function. Using `print()` for printing intermediate output in a function can help you when testing/debugging your code.
- Syntax for an if statement.

Example 3: Loops

In this example you will see the syntax of a for loop and of a while loop. This is very intuitive, so we can get it over with quickly by examining two simple examples.

Here, we simulate some data using a for loop.

```r
N <- 20  # number of simulated values
x <- rnorm(N, 3, 1)  # simulated x values
e <- rnorm(N, 0, 1)  # simulated e values
a <- 2  # intercept
b <- 0.5  # slope

y <- rep(NA, N)  # it is necessary to define a vector of the correct length before assigning values to e
for (i in 1:length(x)) {
    y[i] <- a + x[i] * b + e[i]  # simulated y values
}
```

Of course, it would have been better (faster) to write

```r
y <- a + x * b + e
```

Things to notice here:

- Again, operations on vectors.
- The function `length()`. For matrices, use `dim()` to find out its dimensions.
- The syntax of the for loop.
- Creating a vector before assigning values to it.

Here is a while loop doing the exact same thing as the for loop:

```r
i <- 1  # initialize the counter
while (i < length(x)) {
    y[i] <- a + x[i] * b + e[i]
    i <- i + 1  # update the counter
}
```

Notice:

- A while loop can have more flexible stopping criteria.
- 'Self referencing': `i <- i+1`.

Example 4: Regression in R

R has many useful built-in functions for common tasks. In this example we go over reading data sets into R, dataframes, fitting a linear model, and some common statistics functions in R (e.g., `mean`, `sd`, and `summary`).

Building on the previous example, we have vectors `x` and `y` with simulated data. To save the data in a convenient format, use a dataframe object:

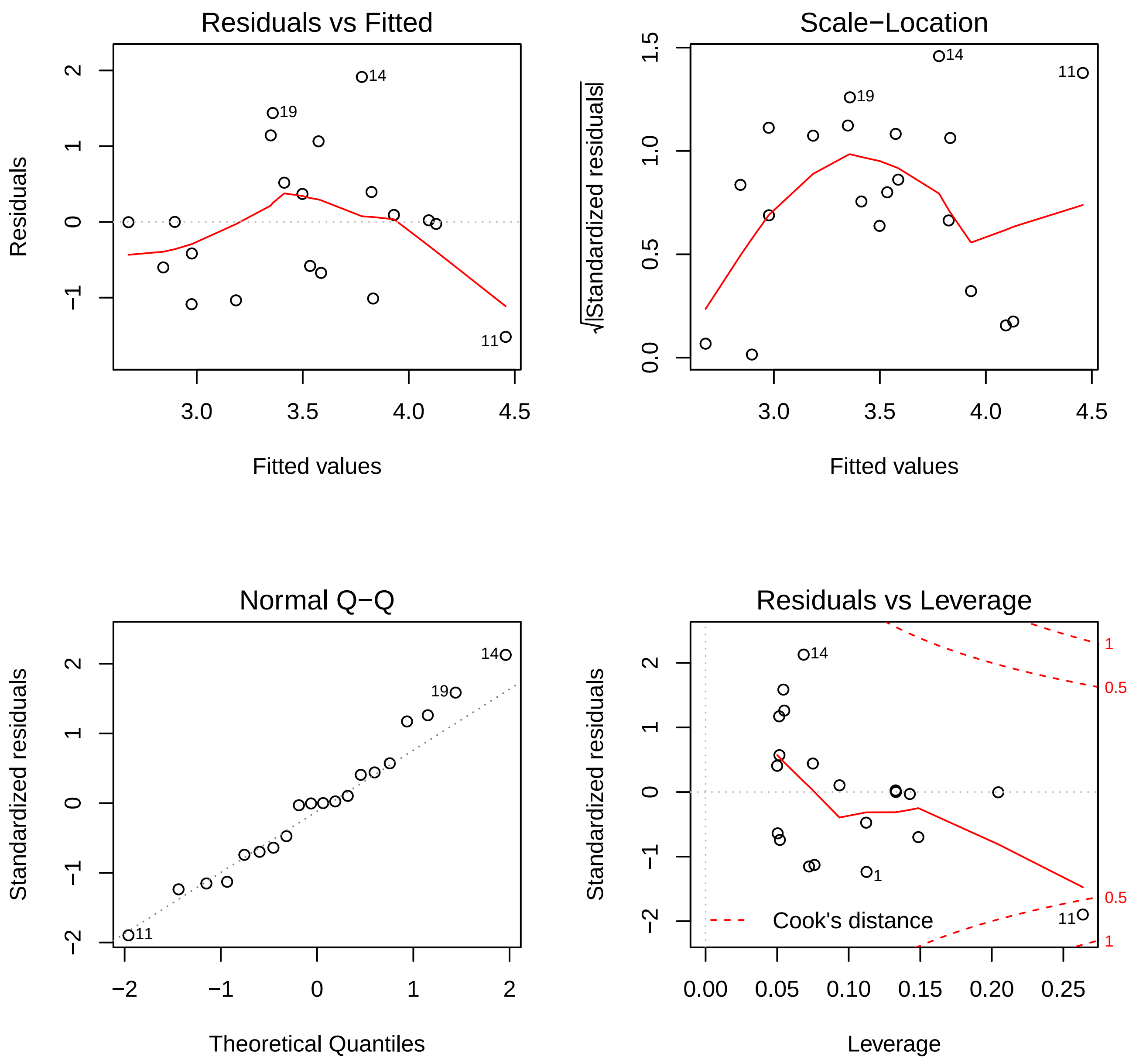
```
regdata <- data.frame(x, y)  # put data in a data frame
save(regdata, file = "regdata.dta")  # save in wd
load("regdata.dta")  # load (not necessary because was still in the workspace)
regdata$x  # print one variable

##  [1] 2.103 2.770 1.649 1.982 3.035 3.811 3.560 2.673 1.899 3.410 4.365 2.901 2.105 3.329
## [15] 3.399 3.017 2.956 2.423 2.687 3.864
```

A file will be saved in your working directory, you can simply load it next time you open RStudio.

Some useful commands for doing regression in R:

```
fit <- lm(y ~ x, data = regdata)  # linear regression with a constant
summary(fit)  # show results

##
## Call:
## lm(formula = y ~ x, data = regdata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5178 -0.6190 -0.0018  0.4258  1.9141
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.598      0.876    1.83    0.085 .
## x              0.655      0.294    2.23    0.039 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.932 on 18 degrees of freedom
## Multiple R-squared:  0.217,Adjusted R-squared:  0.173
## F-statistic: 4.98 on 1 and 18 DF,  p-value: 0.0386

fit$coefficients  # get the estimates

## (Intercept)           x
##      1.5980      0.6551

coefficients(fit)

## (Intercept)           x
##      1.5980      0.6551

fitted(fit)  # predicted values

##     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15
## 2.976 3.413 2.678 2.896 3.587 4.094 3.930 3.349 2.842 3.832 4.457 3.499 2.977 3.779 3.825
##    16    17    18    19    20
## 3.575 3.535 3.185 3.359 4.129

residuals(fit)  # residuals
```

```
##          1          2          3          4          5          6          7          8
## -1.0864957  0.5180722 -0.0038007  0.0001959 -0.6731714  0.0211339  0.0919359  1.1425335
##          9         10         11         12         13         14         15         16
## -0.6009989 -1.0117223 -1.5177606  0.3692733 -0.4167243  1.9141300  0.3950752  1.0639607
##         17         18         19         20
## -0.5813012 -1.0356556  1.4376919 -0.0263718

layout(matrix(c(1, 2, 3, 4), 2, 2))  # prepare plotting (4 graphs)
plot(fit)
```



More basic R commands

- `seq`, e.g., `seq(1,9,by=3)`, `seq(8,20,length=6)`
- `rep`, e.g., `rep(1:3,6)`, `rep(1:3,c(6,6,6))`, `rep(1:3,rep(6,3))`
- `c()`

- `matrix()`
- `sink`
- `list`
- `apply`

3 Practice problems

These exercises are selected to help you practice using the software packages described in this note. They ask you to find out how to do a few basic things that will come in handy at some point in this course. If you are a first-time user, it is highly recommended that you go through them, because it will save you a lot of time later on.

These exercises will not be graded and you will not be required to hand them in.

3.1 LaTeX and LyX

Once your have a working installation of a TeX editor on your computer, the key to doing these exercises is to use the manual and online resources to figure out how to do these common tasks. How to do them will depend on your editor.

1. Using the template `.tex` or `.lyx` file, try to export to a `.pdf` file. If you are using LyX, try exporting to a `.tex` file.
2. Try commenting out some text in the template.
3. Make a numbered list, such as the one you are reading. Make an un-numbered list, e.g.,
   - Item 1
   - Item 2.

   Make a list within a list, e.g.,

   (a) Item 1
   (b) Item 2.
4. Play around with section headers, subsections, subsubsections, etc. Include an automatically generated table of contents.
5. Replicate:

   (a) e^{e^2}
   (b) $\log_{10}(2)$
   (c) $\Phi(.5)$
   (d) $A \cup B$
   (e) $\mathbb{E}[X]$

   You may want to use Detexify for this.
6. Replicate the following:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

7. Make a table. For example, try to replicate this one:

	column 1	column 2
row 1	a	b
row 2	c	d

8. Figure out how to include or suppress an equation label, e.g.,

$$y = x + 3 \tag{1}$$

versus

$$y = x + 3.$$

9. Find out how to reference an equation, e.g., "see equation (1)". The number in this reference will automatically update after you change the position of the equation.

10. Find out how to automatically include the contents of an external `.txt`, `.R`, or `.do` file with your LaTeX output every time you compile it. This is very useful when you want to include code in the appendix to your write-up, but you don't want to copy-paste it every time you make a change. Whether this is possible may depend on the editor you use.

11. Find a package that looks interesting to you and include it in your preamble (not for LyX users).

12. Include a `.png` or `.pdf` figure in your output.

13. Find out how to align equations so that they become more readable, e.g.,

$$\begin{aligned} y &= x + 3 \\ z &= e^y \end{aligned}$$

instead of

$$\begin{gathered} y = x + 3 \\ z = e^y. \end{gathered}$$

3.2 R and RStudio

Using the command line:

1. Calculate the following:
   (a) e^{e^2}
   (b) $\log_{10}(2)$
   (c) $\Phi(.5)$, where $\Phi(\cdot)$ is the standard Normal CDF.

2. Define the vectors

$$x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \qquad y = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}.$$

   Do this once using the `c()` command and once using `seq()`. Also try $1:3$. What is the difference between `x*y` and `x%*%y`?

3. Using x and y as in the previous exercise, try `cbind(x,y)` and `rbind(x,y)`.

4. Define $x = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$.
   (a) Sum all the elements
   (b) Get the minimum
   (c) Calculate the mean
   (d) Replace the first element by 0
   (e) Add 1 to all elements
   (f) Add 1 to all odd elements
   (g) What is `x[c(1,3)]`? `x[2:3]`? `x[-2]`?

5. Create the following matrices:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \qquad B = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 6 \end{pmatrix}.$$

   To create B, use `diag()`.

   (a) What is the difference between `A*B` and `A%*%B`?

   (b) What does `t(A)` do? What does `solve(A)` do? Why doesn't this command work? Does it work on B?

   (c) Replace the $(2, 2)$ element of A by 1

   (d) Replace the diagonal of A by ones

   (e) Store the first row of A in a vector

   (f) Store the first two-by-two block matrix to a new matrix C

   (g) In what sense is B a weighting matrix?

6. Solve the following system of equations

$$\begin{aligned} x - y + z &= 9 \\ x - 6y + 4z &= 12 \\ -2x - 3y + 5z &= 1 \end{aligned}$$

7. Produce a graph of e^x where $x \in [-1, 1]$.

8. Go to `http://vincentarelbundock.github.io/Rdatasets/datasets.html` and find an interesting time series that you can import into R. Plot it and compute some summary statistics for your time series.

9. What does `proc.time()` do?

In a `.R` file:

1. Generate 100 pseudo-random numbers from a standard Normal distribution, and store them in a vector. Create a histogram with a vertical bar at the location of the mean.

2. Write a function that returns an approximation to $\pi/4$, along the lines of Example 1. The function should take as input M, the number of simulated (x, y) pairs. The first step to doing this is to simulate from the $U[0, 1]$ distribution to obtain (x, y)-pairs on the square $[0, 1] \times [0, 1]$, then calculate the fraction of (x, y)-pairs that lie within the unit circle (i.e., such that $x^2 + y^2 < 1$).