The objective of this section is to review the syllabus and to introduce the R environment. If there is remaining time, I'll work through some basic code puzzles that will require you to work in R, but will more likely leave them for you to play with on your own. The first two or three sections will feel slow for those of you substantial experience in R[1], but I promise we'll speed up soon.

Installing R

Download R: The download of R will vary by operating system, but it will begin here in any event:

cran.r-project.org

The online documentation and installer routines are comprehensive. If you are new to R, then it might make sense to use the Mac or Windows distribution, along with the built-in editor to write and evaluate code. `Rstudio` is a popular IDE that provides a somewhat more user-friendly interface than the base R installation. For the tech-oriented, the Linux distribution is very flexible; I use Emacs with the ESS package for editing. If you are interested in using the Linux distribution and are having trouble with the setup, please see me.

Learning R

I have included links to a few of the many resources on the web that provide gentle introductions to the R language. Those of you who have no experience with R or with programming in general will find it well worth your time to spend a few hours browsing those in your free time. In section I will focus on presenting examples of code piece-by-piece in order to illustrate certain concepts. My intention is to expose you to a small part of the R language in section so that you'll feel more comfortable exploring the rest.

Once you feel sufficiently competent in the language you will find that the optimal strategy for learning how to put together a particular piece of code is usually to search the web for "R [whatever you want to do]". RSeek is also an immensely useful resource. If you want to learn about a specific function, simply type `?func` into your R console, where `func` is the name of the function you want to look up.

A gentle introduction to matrix algebra

The lingua franca of this course is matrix algebra, so we will start by introducing some of the more common commands for working in matrix-world[2].

There are a variety of data objects in R, including numbers, vectors, matrices, strings, and dataframes. We will mainly be working with vectors and matrices, which are quick to create and manipulate in R. The `matrix` function will create a matrix, according to the supplied arguments.

[1] If you're bored, skip ahead to the puzzles

[2] Unfortunately not quite as cool as The Matrix, but probably cooler than The Matrix: Reloaded and undoubtedly cooler than The Matrix: Revisited.

```
matrix(1:6, ncol=3)
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

The `ncol` option specifies the number of columns for the output matrix; and the default behavior of `matrix` is to cycle through by column. To cycle through by rows, you'll have to set the optional argument `byrow=TRUE`.

```
matrix(1:6, ncol=3, byrow=TRUE)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Suppose we wanted to check to see if the first matrix was equal to the transpose of the second. This is clearly the case — we can see that it is. But in code, it would be cumbersome to check this condition using the previous two commands. Instead, we can assign the matrices to variables for use in subsequent manipulations. The `<-` operator assigns the arbitrary object to the supplied variable:

```
A <- matrix(1:6, ncol=2)
B <- matrix(1:6, ncol=3, byrow=TRUE)
```

The `=` operator also assigns values, with a slightly different behavior; and it is common practice to use the `=` assignment for function arguments.[3] The `==` comparison operator will yield `TRUE` or `FALSE`:

```
A == t(B)
```

```
     [,1] [,2]
[1,] TRUE TRUE
[2,] TRUE TRUE
[3,] TRUE TRUE
```

Note that `t()` will return the tranpose of the supplied matrix. Each element is checked individually, and each is identical in matrix $\mathbf{A}$ and $\mathbf{B}'$. To check the truthiness of the statement that all elements are identical, we need only to employ the `all` function:

```
all(A == t(B))
```

```
[1] TRUE
```

We can get a list of all the object currently available in memory with the `ls()` function, which is useful as the assignments begin to accumulate:

```
ls()
```

```
[1] "A" "B"
```

[3] See the Google style sheet for a description of other standard practices in `R`.

Note that without assignment, the transpose of $\mathbf{B}$, or `t(B)`, is created on the fly and not stored in memory.

When paired with the `rm()` function, we can use `ls()` to delete all of the objects in memory. This is similar to the command `clear` in Stata.

```
rm(list = ls())
```

What's going on here? `list` is actually the name of an argument built in to the `rm()` command. The default behavior of `rm` is to accept character strings; we could have alternatively specified `rm("A","B")` and the outcome would have been the same. But by passing it a list of all of the objects in memory, we are telling `rm()` to clear everything, not just the variables we name. Next week we will continue with more matrix operations.

Linear algebra puzzles

These notes will provide a code illustration of the Linear Algebra review in Chapter 1 of the lecture notes. Don't worry if you can't solve these puzzles, many of them require commands that we have not covered in section. Come back to them later, once we have gone over `R` code in more detail. There are many correct ways to solve these puzzles. If time remains, I will go over a couple of these next week.

1. Let $\mathbf{I}_5$ be a 5×5 identity matrix. Demonstrate that $\mathbf{I}_5$ is symmetric and idempotent using simple functions in `R`.

2. Generate a 2×2 idempotent matrix $\mathbf{X}$, where $\mathbf{X}$ is not the identity matrix. Demonstrate that $\mathbf{X} = \mathbf{X}\mathbf{X}$.

3. Generate two random variables, $\mathbf{x}$ and $\mathbf{e}$, of dimension $n = 100$ such that $\mathbf{x}, \mathbf{e} \sim N(0, 1)$. Generate a random variable $\mathbf{y}$ according to the data generating process $y_i = x_i + e_i$. Show that if you regress $\mathbf{y}$ on $\mathbf{x}$ using the canned linear regression routine `lm()`, then you will get an estimate of the intercept β_0 and the coefficient on $\mathbf{x}$, β_1, such that $\beta_0 = 0$ and $\beta_1 = 1$.

4. Show that if $\lambda_1, \lambda_2, \ldots, \lambda_5$ are the eigenvectors of a 5×5 matrix $\mathbf{A}$, then $\text{tr}(\mathbf{A}) = \sum_{i=1}^{5} \lambda_i$.