Introduction to $\mathcal{R}$

*Babak RezaeeDaryakenari (*srezaeed@ASU.edu*)*
http://www.babakrezaee.com

Jan 6, 2017

Contents

1 A Brief Review of Mathematics and Statistics

1.1 Dimensionality

- $\mathbf{R}^1$ is the set of real numbers extending from $-\infty$ to $+\infty$ i.e. the real number line.
- $\mathbf{R}^n$ is an n-dimensional space (often referred to as Euclidean space), where each of the n axes extends from $-\infty$ to $+\infty$.
- Examples:
  - $\mathbf{R}^1$ is a one dimensional line.
  - $\mathbf{R}^2$ is a two dimensional plane.
  - $\mathbf{R}^3$ is a three dimensional space.

  Points in $\mathbf{R}^n$ are ordered n-tuples, where each element of the n-tuple represents the coordinate along that dimension.
- $\mathbf{R}^1$: (3)
- $\mathbf{R}^2$: (-5,5)
- $\mathbf{R}^3$: (4,2,5)

1.2 Interval Notation for $\mathbf{R}^1$

- **Open interval:** $(a, b) \equiv \{x \in \mathbf{R}^1 : a < x < b\}$. x is a one-dimensional element in which x is greater than a and less than b.
- **Closed interval:** $[a, b] \equiv \{x \in \mathbf{R}^1 : a \leq x \leq b\}$. x is a one-dimensional element in which x is greater than or equal a and less than or equal b.
- **Half open, half closed interval:** $(a, b] \equiv \{x \in \mathbf{R}^1 : a < x \leq b\}$. x is a one-dimensional element in which x is greater than a and less than or equal b.

1.3 Function

- One variable function: is a rule which assigns a number in $\mathbf{R}^1$ to each number in $\mathbf{R}^1$.For example, there is the function which assigns to any number the number which is one unit larger. We write this function as $f(x) = x + 1$. To the number 2, it assigns the number 3, and to the number $-\frac{3}{2}$, it assigns the number $-\frac{1}{2}$.

  Exercise: What is the function which assigns to any number its double?
- The input variable x is called the **independent variable**, or **exogenous variable**.
- The output variable y is called the **dependent variable**, or the **endogenous variable**.

Polynomials Analytically speaking, the simplest functions are the monomials, those functions which can be written as $f(x) = ax^k$ for some number a and some positive integer k. for example,

$$f_1(x) = 3x^4; f_2(x) = -x^7; \; and \; f_3(x) = -10x^2 \tag{1}$$

The positive integer exponent k is called degree of the monomial; the number a is called a **coefficient**. A function which is formed by adding together monomials is called a **polynomial**. For example, if we add the three monomials in (2), we obtain the polynomial

$$h(x) = -x^7 + 3x^4 - 10x^2 \tag{2}$$

- For any polynomial, the highest degree of any monomial that appears in it is called the **degree** of the polynomial.
- There are more complex types of functions:
  - **Rational functions**, which are ratios of polynomials, like

$$y = \frac{x^2+1}{x-1}, \; y = \frac{x^5+7x}{5}, \; y = \frac{1}{x} \tag{3}$$

  - **Rational functions**, in which variable x appears as an exponent, like $y = 10^x$, $y = e^x$.
- **Increasing and decreasing functions:**
  - A function is **increasing** if its graph moves upward from left to right. More precisely, a function is increasing if

$$x_1 > x_2 \; implies \; that \; f(x_1) > f(x_2) \tag{4}$$

  - A function is **decreasing** if its graph moves downward from left to right. More precisely, a function is increasing if

$$x_1 > x_2 \; implies \; that \; f(x_1) < f(x_2) \tag{5}$$

Exercise: In Figure 1, determine the type (increasing or decreasing) of each function.

The places where a function changes from increasing to decreasing and vice versa are also important. If a function f changes from decreasing to increasing at x_0, the graph f turns upward around the point $(x_0, f(x_0))$. This implies that the graph f lies above the point $(x_0, f(x_0))$ around that point. Such a point $(x_0, f(x_0))$ is called a **local minimum** of function f. If the graph of a function f never lies below $(x_0, f(x_0))$ then $(x_0, f(x_0))$ is called a **global minimum** of f.

Similarly, if function g changes from increasing to decreasing at z_0, the graph g cups downward at $(z_0, g(z_0))$ is called a local maximum

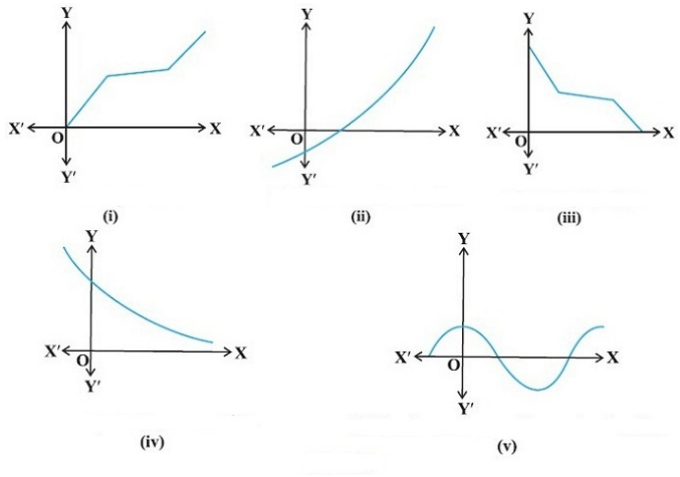


Figure 1: Decreasing and increasing functions

of g. If the graog of function g never goes above $(z_0, g(z_0))$ then is a **global maximum** of g.

Exercise: In Figure 1, determine local and global minimum and maximum of each function.

1.4 Matrix Algebra

A **matrix** is simply a rectangular array of numbers. So, any table of data is a matrix. The size of matrix is indicated by the number of its rows and the number of its columns. A matrix with k rows and n columns is called a $k \times n$ ("k by n") matrix.

The number in row i and column j is called the (i,j)th entry, and is often written a_{ij}.

Addition:

$$\begin{pmatrix} 3 & 4 & 1 \\ 6 & 7 & 0 \\ -1 & 3 & 8 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 7 \\ 6 & 5 & 1 \\ -1 & 7 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 8 \\ 12 & 12 & 1 \\ -2 & 10 & 8 \end{pmatrix}$$

Scalar multiplication:

$$2 \begin{pmatrix} 3 & 4 & 1 \\ 6 & 7 & 0 \\ -1 & 3 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 8 & 2 \\ 12 & 14 & 0 \\ -2 & 6 & 16 \end{pmatrix}$$

Matrix multiplication:

$$\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} aA + bC & aB + bD \\ cA + dC & cB + dD \\ eA + fC & eB + fD \end{pmatrix}$$

Exercise: Find the result of following below matrix multiplication.

$$\begin{pmatrix} 2 & 5 \\ 7 & 4 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ 3 & 2 \end{pmatrix} = \begin{pmatrix} X & Y \\ Y & X \\ X & X \end{pmatrix} \begin{pmatrix} a & b \\ X & Y \end{pmatrix} =$$

The $n \times n$ matrix

$$I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & 1 \end{pmatrix}$$

is called the $n \times n$ **identity matrix** because it is a multiplicative identity for matrices just as the number 1 is for real number.

Transpose: Finally, there is one other operation on matrices which we shall frequently use. The **transpose** of $k \times n$ matrix A is the $n \times k$ matrix obtained by interchanging the rows and columns of A.

This matrix is often written as A^T. The first row of A becomes the first column of A^T. The second row of A becomes the second column of A^T, and so on. Thus, the (i,j)th entry of A becomes (j,i)th entry of A^T. For example,

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$ You may need the following rules in POS 603 class:
$(A+B)^T = A^T + B^T$
$(A-B)^T = A^T - B^T$
$(A^T)^T = A$
$(rA)^T = rA^T$
$(AB)^T = B^T A^T$

1.5 Expectation

We often want to summarize some characteristics of the distribution of a random variable. The most important summary is the expectation (or expected value, or mean), in which the possible values of a random variable are weighted by their probabilities.

- **Expectation of a Discrete Random Variable:** The expected value of a discrete random variable Y is

$$E(Y) = \sum_y yP(Y = y) = \sum_y yp(y)$$

In words, it is the weighted average of all possible values of Y, weighted by the probability that y occurs. It is not necessarily the number we would expect Y to take on, but the average value of Y after a large number of repetitions of an experiment.

Exercise: For a fair die, $E(Y) =$

- **Expectation of a Continuous Random Variable:** The expected value of a continuous random variable is similar in concept to that of the discrete random variable, except that instead of summing using probabilities as weights, we integrate using the density to weight. Hence, the expected value of the continuous variable Y is defined by

  $E(Y) = \int_y yf(y)dy$

  Exercise: Find $E(Y)$ for $f(y) = \frac{1}{1.5},\ 0 < y < 1.5$.

  $E(Y) =$

- **Properties of Expected Value**:
  - $E(c) = c$
  - $E[E[Y]] = E[Y]$
  - $E[cg(Y)] = cE[g(Y)]$
  - $Linearity : E[g(Y_1) + \ldots + g(Y_n)] = E[g(Y_1)] + \ldots + E[g(Y_n)]$, regardless of independence
  - $E[XY] = E[X]E[Y]$, if X and Y are independent
- **Conditional Expectation:** With joint distributions, we are often interested in the expected value of a variable Y if we could hold the other variable X fixed. This is the conditional expectation of Y given $X = x$:

  1. Y discrete: $E(Y|X = x) = \sum_y yp_{Y|X}(y|x)$

  2. Y continuous: $E(Y|X = x) = \int_y yf_{Y|X}(y|x)dy$

  The conditional expectation is often used for prediction when one knows the value of X but not Y; the realized value of X contains information about the unknown Y so long as $E(Y|X = x) \neq E(Y)\ \forall x$.

1.6 Variance

We can also look at other summaries of the distribution, which build on the idea of taking expectations. Variance tells us about the "spread" of the distribution; it is the expected value of the squared deviations from the mean of the distribution. The standard deviation is simply the square root of the variance.

1. Variance: $\sigma^2 = Var(Y) = E[(Y - E(Y))^2] = E(Y^2) - [E(Y)]^2$

2. Standard Deviation: $\sigma = \sqrt{Var(Y)}$

1.7 Covariance and Correlation

The covariance measures the degree to which two random variables vary together; if the covariance is positive, X tends to be larger than it mean when Y is larger than its mean. The covariance of a variable with itself is the variance of that variable.

$$Cov(X, Y) = E[(X - E(X))(Y - E(Y))] = E(XY) - E(X)E(Y)$$

The correlation coefficient is the covariance divided by the standard deviations of X and Y. It is a unitless measure and always takes on values in the interval [-1,1].

$$\rho = \frac{Cov(X,Y)}{\sqrt{(Var(X)Var(Y))}} = \frac{Cov(X,Y)}{SD(X)SD(Y)}$$

- Properties of Variance and Covariance
  - $Var(c) = 0$
  - $Var[cY] = c^2 Var[Y]$
  - $Cov[Y, Y] = Var[Y]$
  - $Cov[X, Y] = Cov[Y, X]$
  - $Cov[aX, bY] = abCov[X, Y]$
  - $Cov[X + a, Y] = Cov[X, Y]$
  - $Cov[X + Z, Y + W] = Cov[X, Y] + Cov[X, W] + Cov[Z, Y] + Cov[Z, W]$
  - $Var[X + Y] = Var[X] + Var[Y] + 2Cov[X, Y]$

2 $\mathcal{R}$: The R Project for Statistical Computing

2.1 What is R? What is R-Studio? Why are they useful?

$\mathcal{R}$ is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. $\mathcal{R}$ can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

$\mathcal{R}$ provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and $\mathcal{R}$ provides an Open Source route to participation in that activity.

One of $\mathcal{R}$'s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formula where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

$\mathcal{R} - Studio$ is a free and open source Integrated Development Environment (IDE) for $\mathcal{R}$, a programming language for statistical computing and graphics.

$\mathcal{R} - Studio$ is available in two editions: $\mathcal{R} - Studio$ Desktop, where the program is run locally as a regular desktop application; and $\mathcal{R} - Studio$ Server, which allows accessing $\mathcal{R} - Studio$ using a web browser while it is running on a remote Linux server. Prepackaged distributions of $\mathcal{R} - Studio$ Desktop are available for Microsoft Windows, Mac OS X, and Linux.

$\mathcal{R} - Studio$ is written in the C++ programming language and uses the Qt framework for its graphical user interface. Work on $\mathcal{R} - Studio$ started at around December 2010 , and the first public BETA version (v0.92) was officially announced in February 2011.

2.2 Downloading $\mathcal{R}$ and $\mathcal{R} - Studio$

For downloading $\mathcal{R}$ go to: `http://cran.stat.ucla.edu/`, and there are install download links for Linux, Mac(OS), and Windows.

For downloading $\mathcal{R} - Studio$ go to: `http://www.rstudio.com/products/RStudio/`, and there are both Desktop (recommended) and Server version.

Now, install $\mathcal{R}$ and $\mathcal{R} - Studio$ on your laptops! Please go to above links and follow the instructions for downloading and installing $\mathcal{R}$ and $\mathcal{R} - Studio$.

2.3 The $\mathcal{R}$ and $\mathcal{R}-Studio$ user interface

Now open $\mathcal{R}$ software! After $\mathcal{R}$ is started, there is a console awaiting for input. You can enter commands one at a time at the command prompt (>) or run a set of commands from a source file.

Type following simple calculations and commands, and see what are the results:

```
2+2
```

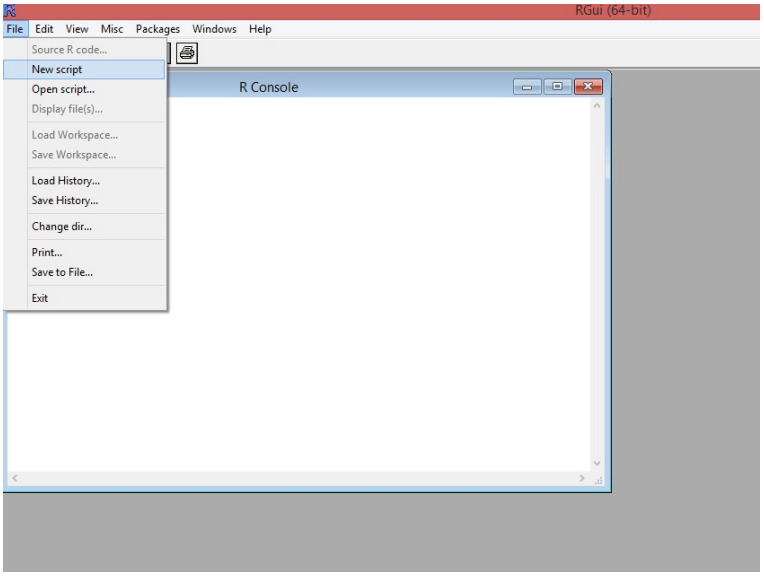
```
print(2+2)
```

```
print('2+2')
```

```
print('Hello, world!')
```

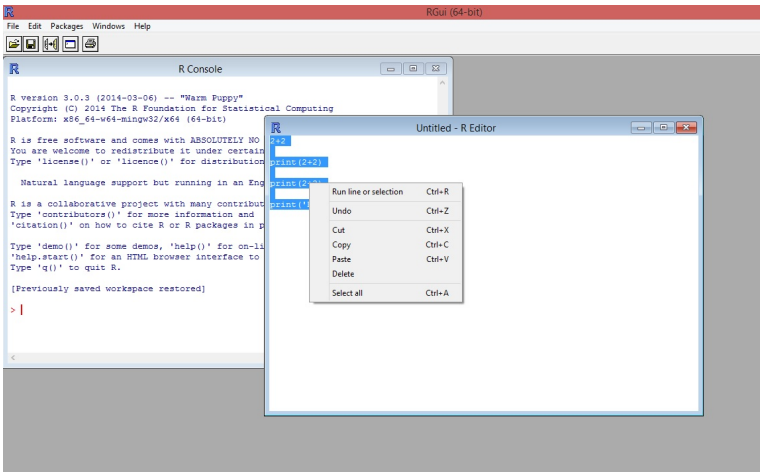
```
print(Hello, world!)
#Did you get any error message? Why that happened?
```

Now, on File tab, choose new script. And, redo what you did on Console window. In script window, for running the commands you can either use $Ctrl + R$ ($Cmmd + R$ in Mac) or make a right click and choose Run line or selection.



Could you tell what is the difference in running a command in $\mathcal{R}$ console and $\mathcal{R}$-Editor ($\mathcal{R}$ Script)?

Now open $\mathcal{R}-Studio$ software and repeat what you did in $\mathcal{R}-Studio$. (You can choose to use either $\mathcal{R}-Studio$ or $\mathcal{R}$ for remaining part of this workshop based on your preferences. But, for now, we want to get familiar with the interface of both software! For benefiting the advantages of embedding $\mathcal{R}$ in LaTeX, it is preferred to use $\mathcal{R}-Studio$.)



Results of calculations can be stored in objects using the assignment operators:

- An arrow (<-) formed by a smaller than character and a hyphen without a space!
- The equal character (=)

These objects can then be used in other calculations. To print the object just enter the name of the object. There are some restrictions when giving an object a name:

- Object names cannot contain 'strange' symbols like !, +, -, # .
- A dot (.) and an underscore (_) are allowed, also a name starting with a dot.
- Object names can contain a number but cannot start with a number.
- R is case sensitive, X and x are two different objects, as well as temp and temP.

```
x = 1
print(x)

x<-1
print(x)

x<-2
print(x)

X=2
x+X
xX=x+X
print(xX)

y='a'
print(y)
```

```
w="a"
print(w)

z<-'Which one do you prefer? R or R-Studio'
print(z)
```

How can you drop/delete/remove an object? Try `rm()` command!

```
rm(x)
print(x)

rm(z)
print(z)
```

Now, create two small vectors with data. The following apply the function `c()` to combine three numeric values into a vector.

```
V1=c(1,2,3)
V2=c(1,2,'a')
print(V1)
print(V2)
```

Then, make one vector which include 1,2,3,4, and 5. Store this vector, and name it x1.

```
c(1,2,3,4,5)
x1<-c(1,2,3,4,5)

#Now try x2<-c(1:5) and name it x2.

x2<-c(6:10)
print(c1)
print(x2)
```

All text after the pound sign "#" within the same line is considered a comment.

2.4 Help!

$\mathcal{R}$ provides extensive documentations. For example, entering ?c or help(c) at the prompt gives documentation of the function c in R. Please give it a try.

```
?c
help(c)
```

2.5 Vector

Vectors can be combined via the function c. For examples, the following two vectors n and s are combined into a new vector containing

elements from both vectors.

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc", "dd", "ee")
c(n, s)
```

Arithmetic operations of vectors are performed member-by-member, i.e., member-wise.

For example, suppose we have two vectors a and b.

```
a = c(1, 3, 5, 7)
b = c(1, 2, 4, 8)
```

Then, if we multiply a by 5, we would get a vector with each of its members multiplied by 5.

```
5 * a
```

And if we add a and b together, the sum would be a vector whose members are the sum of the corresponding members from a and b.

```
a + b
```

Similarly for subtraction, multiplication, and division, we get new vectors via member wise operations.

```
a - b
```

```
a * b
```

```
a / b
```

Recycling Rule: If two vectors are of unequal length, the shorter one will be recycled in order to match the longer vector. For example, the following vectors u and v have different lengths, and their sum is computed by recycling values of the shorter vector u.

```
u = c(10, 20, 30)
v = c(1, 2, 3, 4, 5, 6, 7, 8, 9)
u + v
```

2.6 Matrix

There are various ways to construct a matrix. When we construct a matrix directly with data elements, the matrix content is filled along the column orientation by default. For example, in the following code snippet, the content of B is filled along the columns consecutively.

```
B = matrix(
   c(2, 4, 3, 1, 5, 7),
   nrow=3,
   ncol=2)
```

```
B              # B has 3 rows and 2 columns
```

Transpose: We construct the transpose of a matrix by interchanging its columns and rows with the function `t()`.

```
t(B)           # transpose of B
B<-t(B)
```

Combining Matrices: The columns of two matrices having the same number of rows can be combined into a larger matrix. For example, suppose we have another matrix C also with 3 rows.

```
C = matrix(
   c(7, 4, 2),
   nrow=3,
   ncol=1)

C              # C has 3 rows
```

Then we can combine the columns of B and C with `cbind()`.

```
cbind(B, C)
```

Similarly, we can combine the rows of two matrices if they have the same number of columns with the `rbind()` function.

```
D = matrix(
   c(6, 2),
   nrow=1,
   ncol=2)

D              # D has 2 columns

rbind(B, D)
```

2.7 Importing Data

Importing data into R is fairly simple. For Stata and Systat, use the *Foreign* package. For SPSS and SAS I would recommend the *Hmisc* package for ease and functionality. See the Quick-R section on these packages, for information on obtaining and installing the these packages. Before working with some examples of importing data, we need to learn about $\mathcal{R}$-packages and Working Directories.

2.7.1 $\mathcal{R}$-packages

$\mathcal{R}$-packages are reproducible and reusable $\mathcal{R}$-Codes written, tested, and confirmed by $\mathcal{R}$-community.

To use an $\mathcal{R}$-package, you need to first download and install it. Assume, we want to install package *Foreign*, type following syntax and run it to see the results:

```
install.packages("foreign")
```

You also can install multiple package with one syntax. The following syntax installs both *foreign* and *Hmisc* packages.

install.packages(c("foreign", "Hmisc"))

Take-Home Exercise: You can install $\mathcal{R}$ through the menu as well, how?

For using $\mathcal{R}$-packages, you need to install them only once, but for using them, you need to call them every time you open your $\mathcal{R}$/$\mathcal{R}$-studio. To call/load the installed packages in your $\mathcal{R}$-library, you can use following syntax:

```
library(foreign)
library(Hmisc)
```

Take-Home Exercise: How can we load multiple packages at once? *Hint*: One way is using a *for*-loop.

2.7.2 Working Directories

There are two ways to set your working directory:

1. Through the menu
   - In Windows: go to the File menu, select Change Working Directory, and select the appropriate folder/directory
   - In Macs: go to the Misc menu, select Change Working Directory, and select the appropriate folder/directory

2. By using the function

   ```
   setwd("...")
   ```

   in which, the "..." is the specific pathway, e.g.,

   in Windows:

   ```
   setwd("C:/Users/User Name/Documents/FOLDER")
   ```

   in Macs:

   ```
   setwd("/Users/User Name/Documents/FOLDER")
   ```

   In this example, the working directory has been set to a folder, *FOLDER*, within the Documents directory. You have to type in, or copy and paste the appropriate pathway on your computer, of course.

You can check that your working directory has been correctly set by using `getwd()`

How to use "~/"? As long as you have got your desired target folder consistently under these directories, you can always set your new working directories like this:

```
setwd("~/FOLDER")
```

or

```
setwd("~/Documents/FOLDER")
```

where, again, the "~/" refers to the home directory.

2.7.3 From A Comma Delimited Text File

We use Gandrud and Grafstrom (2014) data set in this workshop.[1] Set your working directory accordingly, and load the CSV file you downloaded as follow:

```
mydata<-read.csv("GB_FRED_cpi_2007.csv")
```

For SPSS and STATA, we have:

```
data <- read.spss("~/myfile.sav", to.data.frame=TRUE)

data <- read.dta("~/myfile.dta")\\
```

Take-Home Exercise: Assume your file is saved as an *MS Excel* file, how would you open this file? *Hint:* Google it and you will find millions of web pages on this question. Here is an example `https://www.datacamp.com/community/tutorials/r-tutorial-read-excel-into-r#gs.qJX3nOA`.

3 Refining datasets

Use the assignment operator `<-` to create new variables. A wide array of operators and functions are available here.

Load your *GB_FRED_cpi_2007.csv* data, and open it using `View()` command:

```
View(mydata)
```

Then, find *DebtGDP* and *ExpenditureGDP*. Now, we want to make a new variable and add it to our data. This new variable, lets call it ExpenditureDebtRatio, should be computed through dividing ExpenditureGDP by DebtGDP.

```
mydata$ExpenditureDebtRatio<-mydata$ExpenditureGDP/mydata$DebtGDP
```

[1] Go to the journal of Political Science and Research and Method harvard dataverse web page, and find "Inflated Expectations: How government partisanship shapes bureaucrats' inflation expectations" by Gandrud and Grafstrom (2014). On Data and Analysis section of study, and download *GB_FRED_cpi_2007.csv* file. Then import this *csv* file into your $\mathcal{R}$ workspace.

View your data again, is the new variable added to your data?

3.0.1 Re-coding Variables

Now, we want to code economic conditions as a categorical variable. Assume, if *ExpenditureDebtRatio* is larger than 0.5, the economy is bad, otherwise it is good. We can do it as follow[2]:

```
mydata$ExpenditureDebtRatio<-mydata$DebtGDP/mydata$ExpenditureGDP

mydata$condition <-
ifelse(mydata$ExpenditureDebtRatio > 0.5,c("Bad"), c("Good"))
```

3.0.2 Renaming Variables

You can rename variables interactively or through programming.

```
# rename interactively
fix(mydata) # results are saved on close
```

Take-Home Exercise: How can we rename a variable through commands? (Hint: Ask Google!)

3.0.3 Data-frame

A data frame is used for storing data tables. It is a list of vectors of equal length. For example, the following variable *df* is a data frame containing three vectors *n, s, b*.

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc")
b = c(TRUE, FALSE, TRUE)
df = data.frame(n, s, b)       # df is a data frame
```

We want to use a built-in data-frame in $\mathcal{R}$ for our tutorials. For example, here is a built-in data-frame called *mtcars*.

```
mtcars
mtcars[1, 2]
nrow(mtcars)    # number of data rows
ncol(mtcars)    # number of columns
```

4 Creating graphs with $\mathcal{R}$

There are different packages for making graphs in $\mathcal{R}$. *ggplot2* and *lattice* are among the well-known packages.

Let's make some graphs with *ggplot2*. First, install and load *ggplot2*.

[2] It is just an example, we never evaluate economic conditions this way!

4.1 Scatter Plots

We want to create a scatter graph for variables of *deflator* vs. *cpi_change*. Try following commands, and how changes in the `qplot` command affect the created graph.

```
qplot(deflator, cpi_change, data=mydata)

qplot(deflator, cpi_change, data=mydata,
        main="Scatter plot of deflator vs. cpi_change ")

qplot(deflator,cpi_change,data=mydata, main= "Scatter plot")

qplot(deflator, cpi_change, data=mydata,
        main="Scatter Plot of Deflator vs. CPI Change ",
        xlab="GDP Deflator", ylab="Change in Customer Price Index")

qplot(deflator, cpi_change, data=mydata,
        main="Scatter plot",
     xlab="GDP Deflator", ylab="Change in Customer Price Index")
```

4.2 Bar Plots

Now, assume we want to compare the ratio of government expenditure to government debt across different US presidents.

```
ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
  geom_bar(stat="identity")

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
  geom_bar(stat="identity")+
  geom_bar(aes(fill=president), stat="identity")

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
  geom_bar(aes(fill=president), stat="identity")

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio))+
  geom_bar(colour="black", stat="identity")
```

4.3 Line Plots

How can we make line graph to compare the ratio of government expenditure to government debt across different US presidents?

```
ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
  geom_line()

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
  geom_line()

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
```

```
  geom_line(aes(group=1))

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio))+
  geom_line(aes(group=1))

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
  geom_line(aes(group=1))+geom_point()

ggplot(data=mydata, aes(x=president, y=ExpenditureDebtRatio)) +
  geom_line(aes(group=1))+geom_point (color="blue")
```

5 LaTeX

For high-quality typesetting, we can use LaTeX. It is used mostly in medium-to-large technical or scientific documents. There are several packages in $\mathcal{R}$ and STATA that generate LaTeX codes for publishing nice and professional statistical results. LaTeX is also compatible with $\mathcal{R}$, and they can be knitted together for a better management and documentation of your scientific research (Quantitative and Qualitative).

There are different ways of implementing LaTeXcodes. If you learn coding in LaTeX, you easily can move across these different platforms to see which one works better for you. Since I am a big fan of working and benefiting cloud-based platforms, I explain coding in LaTeXusing *Overleaf* project. *Overleaf* is an online and free real-time collaborative writing and publishing tool which uses LaTeXcommands. There are thousands of templates saved on this online platform which can facilitate your work.

Let's start with a simple document. Go to `www.overleaf.com`, and sign up an account[3]. Go to *My Projects*, and then click on *New Project*, and open a blank project.

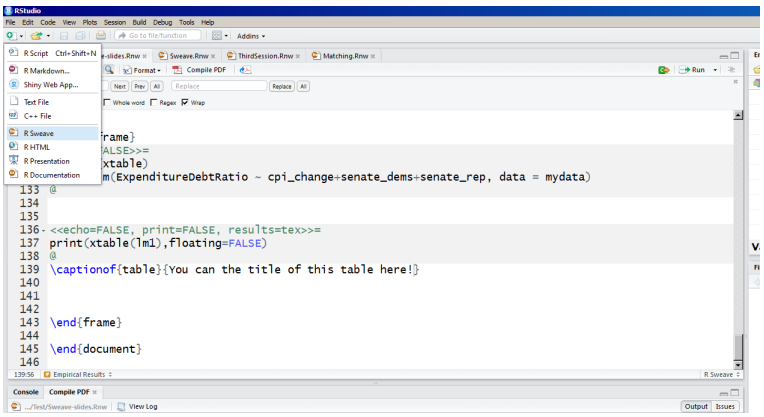
Now, go to `https://www.overleaf.com/read/xcqpdsfgtjvg`, and find the template that I prepared for this class. We will discuss basic LaTeXcommands using this template.

6 Sweave

Sweave is a tool in $\mathcal{R}$ enabling integration of R code into LaTeXdocuments. This tool facilitates managing your empirical results and decreases typos and mistakes which can happen during transferring figures and tables created by $\mathcal{R}$ to your manuscript.

[3] Overleaf has been used by well-known academic publishers such as Cambridge and Oxford for sharing their manuscript templates. You usually receive tips and suggestions from this website, and I barely can remember they sent marketing or spam emails to me. Thus, I suggest using this platform. If you are not comfortable signing up an account with this website, please let me know so I suggest an alternative platform.

Go to $\mathcal{R}$-Studio and as below picture shows open a new $\mathcal{R}$ Sweave document.



First, copy and paste the LaTeXdocument we worked on, and click 'Compile PDF' to see what happens.

Now, go to `https://www.dropbox.com/s/ftq3tqjm5yxc8xp/Sweave-SPGS2017Works` `Rnw?dl=0`, and find the Sweave template that I prepared for this class. The file is named 'Sweave-SPGS2017Workshop'. We will discuss basic Sweave commands using this template.

You also can create a slide file to present your work using Sweave. A template can be downloaded from `https://www.dropbox.com/s/` `hh4gvccn7pvj453/Sweave-SPGS2017Workshop-Slides.Rnw?dl=0`. This template is named 'Sweave-SPGS2017Workshop-Slides'.