Data Programming

Karl Ho

2019-05-25

Contents

Chapter 1

Prerequisites

This course requires no prior experience in programming. Yet, if you have some programming experience (e.g. SPSS, Stata, HTML), it will be helpful. R, Python and JavaScript are all interpreted languages. In other words, the programs do not need compilation but will run in an environment to get the outputs.

All packages and accounts are free and supported by open sources. It is recommended students bring their own computers (not mobile device) running MacOS, Linux or Windows operating systems.

Recommended software and IDE's:

1. R (https://cran.r-project.org)
2. RStudio (https://www.rstudio.com)
3. Anaconda 3 (https://www.anaconda.com)*
4. Text editor of own choice (e.g. Atom, Sublime Text, Ultraedit)

Recommended websites/accounts:

1. GitHub (https://github.com)
2. RStudio Cloud

(*) – Python 3.x only.

Chapter 2

Introduction

This chapter introduces the general principles for coding or programming involving data.

Gentzkow and Shapiro (2014) list out some principles for data programming.

2.1 Principle of Programming

1. Automation
   - For replicability (future-proof, for the future you)
2. Version Control
   - Allow evolution and updated edition
   - Use Git and GitHub
3. Directories
   - Organize by functions
4. Keys
   - Index variable (relational)
5. Abstraction
   - KISS (Keep in short and simple)
6. Documentation
   - Comments for communicating to later users
7. Management
   - Collaboration ready

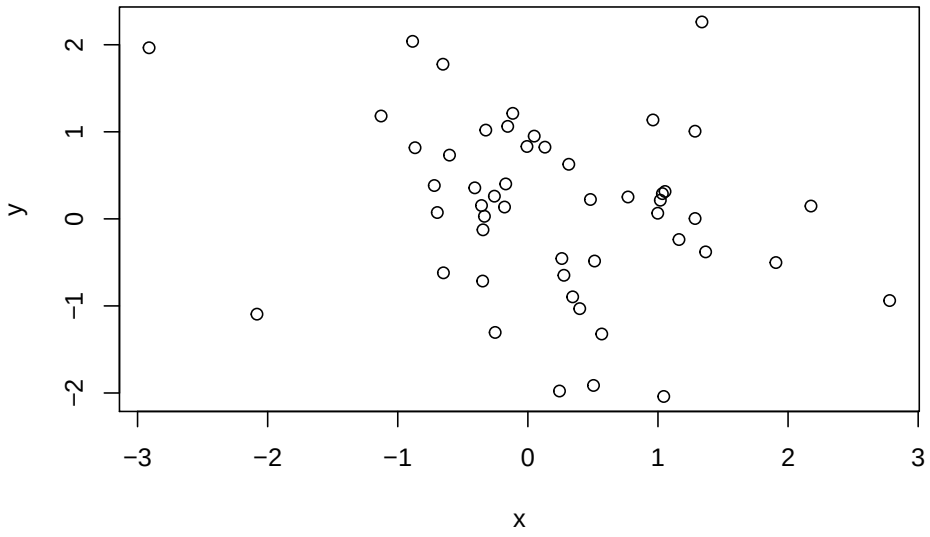
2.2 Functionalities of Data Programs

A data program can perform :

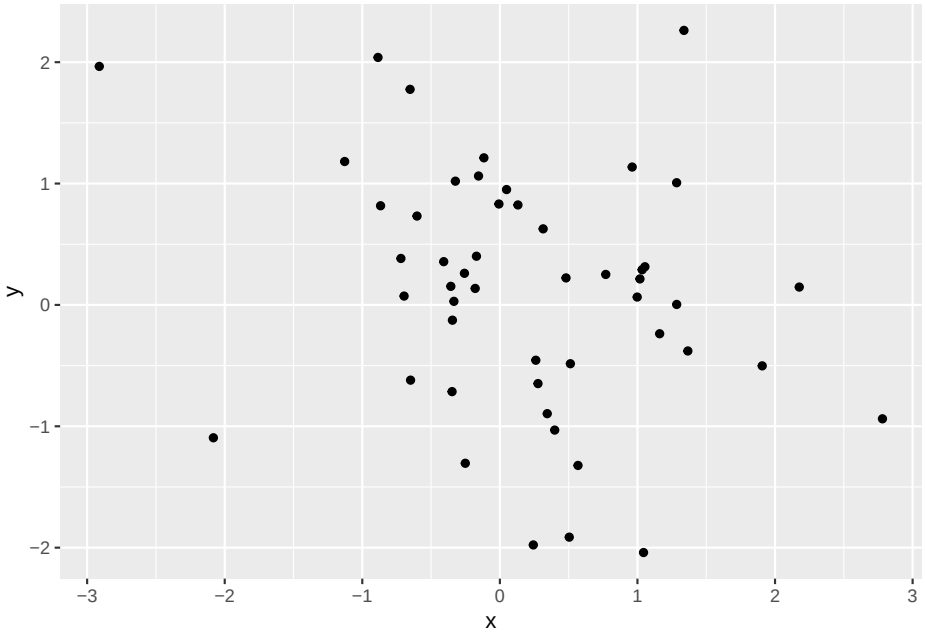
1. Documentation of data
2. Importing and exporting data
3. Management of data
4. Visualization of data
5. Data models

Sample R Programs:

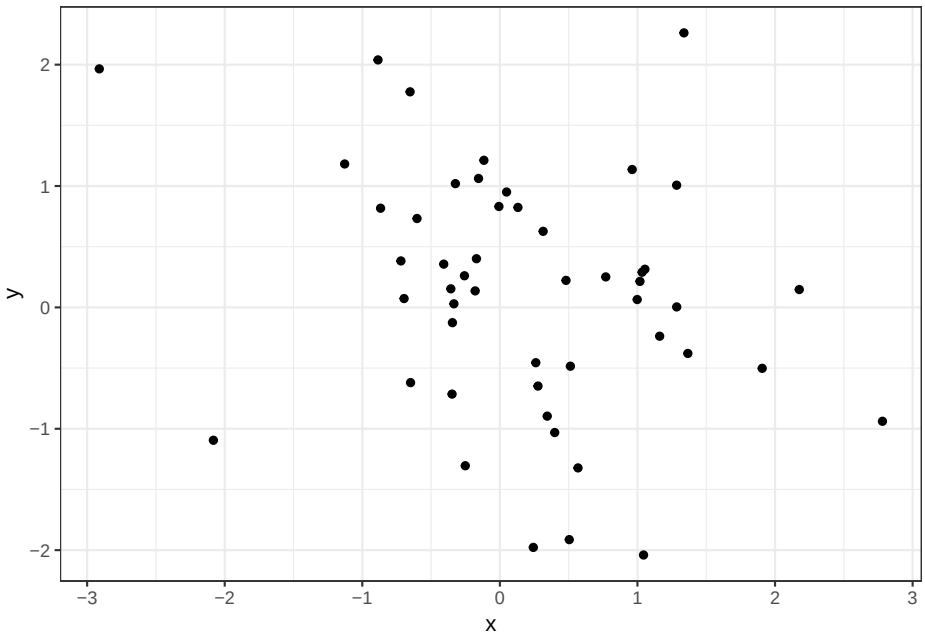
```
# Create variables composed of random numbers
x <-rnorm(50)
y = rnorm(x)

# Plot the points in the plane
plot(x, y)
```




```
# Plot better, using the ggplot2 package
## Prerequisite: install and load the ggplot2 package
## install.packages("ggplot2")
library(ggplot2)
qplot(x,y)
```




```
# Plot better better with ggplot2
ggplot(,aes(x,y)) + theme_bw() + geom_point()
```



Sample Python Programs (## represents output)

```
# Python example program 0
# Some basics

# Print a one-line message
print ("Hello NCHU friends!!")

# Create some variables
```

```
## Hello NCHU friends!!
```

```
x=5
y=3

# Perform some mathematical operations
x*y
```

```
## 15
```

```
x**y
```

```
## 125
```

```
x%y
```

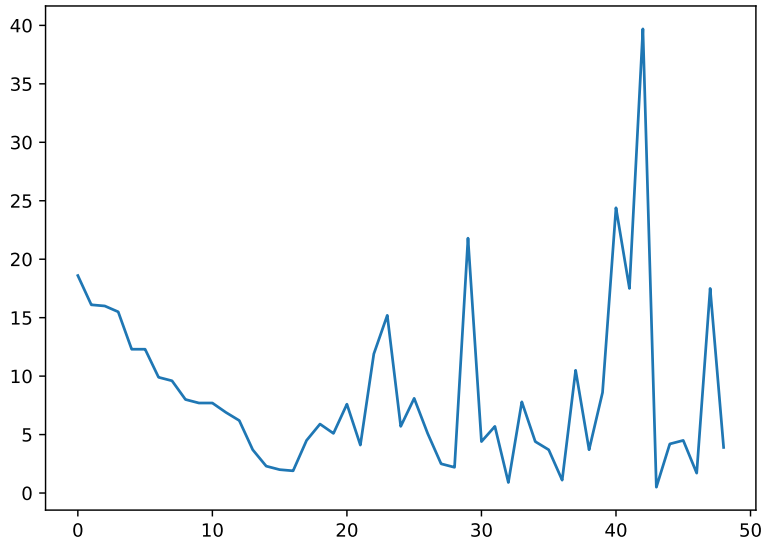
```
## 2
```

```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Import a text file in csv format
import pandas as pd
CO2 = pd.read_csv("https://raw.githubusercontent.com/kho777/data-visualization/master/
# Take a glimpse of the data file
CO2.head()
```

```
##                country              CO2 _kt  CO2pc  CO2percent
## 0                  Australia    446,348   18.6       1.23%
## 1            United States  5,172,336   16.1      14.26%
## 2             Saudi Arabia    505,565   16.0       1.39%
## 3                   Canada    555,401   15.5       1.53%
## 4                   Russia  1,760,895   12.3       4.86%
```

```
# Using matplotlib to do a simple plot
import matplotlib.pyplot as plt
CO2pc=CO2["CO2pc"]
plt.plot(CO2pc)
```



In the following chapters, sample programs will be provided to illustrate these functionalities.

Chapter 3

R Programming

3.1 What is R?

The R statistical programming language is a free, open source package based on the S language developed by John Chambers.

3.1.1 Some history of R and S

S was further developed into R by Robert Gentlemen (Canada) and Ross Ihaka (New Zealand)

Source: Nick Thieme. 2018. R Generation: 25 years of R

Figure 3.1: R Inventors

Figure 3.2: Prominent R Developers

3.1.2 It is:

- Large, probably one of the largest based on the user-written add-ons/procedures
- Object-oriented
- Interactive
- Multiplatform: Windows, Mac, Linux

According to John Chambers (2009), six facets of R:

1. an interface to computational procedures of many kinds;
2. interactive, hands-on in real time;
3. functional in its model of programming;
4. object-oriented, “everything is an object”;
5. modular, built from standardized pieces; and,
6. collaborative, a world-wide, open-source effort.

Source: Nick Thieme. 2018. R Generation: 25 years of R

3.2 Why R?

- A programming platform environment
- Allow development of software/packages by users
- Currently, the CRAN package repository features 12,108 available packages (as of 1/31/2018).
- Graphics!!!
- Scaleble and Portable
- Interface with other platform/langauges (e.g. C++, Python, JavaScript, Stan, SQL)
- Comparing R with other software?

Source: Oscar Torres-Reyna. 2010. Getting Started in R~Stata Notes on Exploring Data

Features	Stata	SPSS	SAS	R
Learning curve	Steep/gradual	Gradual/flat	Pretty steep	Pretty steep
User interface	Programming/point-and-click	Mostly point-and-click	Programming	Programming
Data manipulation	Very strong	Moderate	Very strong	Very strong
Data analysis	Powerful	Powerful	Powerful/versatile	Powerful/versatile
Graphics	Very good	Very good	Good	Excellent
Cost	Affordable (perpetual licenses, renew only when upgrade)	Expensive (but not need to renew until upgrade, long term licenses)	Expensive (yearly renewal)	Open source

Figure 3.3: R Compared with other statistical programs/platforms

3.3 RStudio

RStudio is a user interface for the statistical programming software R.

- Object-based environment
- Window system
- Point and click operations
- Coding recommended

- Expansions and development
- a multi-functional Integrated Development Environment (IDE)

3.3.1 Getting started

3.4 Basic operations and object assignment

Arithmetic Operations:

+, -, *, /, ^ are the standard arithmetic operators.

Assignment

To assign a value to a variable use “<-” or “=”:

```
## Introduction to R sample program
## file: introR02.R
## Adapted from Venables, W.N., Smith, D.M. and Team, R.C., 2018. An Introduction to R, Version 3
```

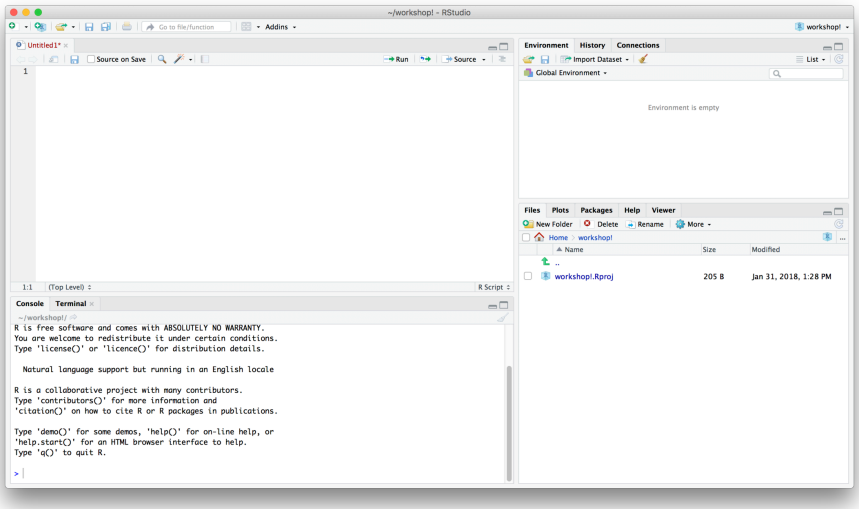


Figure 3.4: RStudio screenshot

```
# Clear any existing objects
rm(list = ls())

# Generate x, y and w to demontrate linear models and plots.
# Make x = (1,2,...,20).

x <- 1:20

# Create A 'weight' vector of standard deviations.

w <- 1 + sqrt(x)/2

# Create a data frame of two columns, x and y.

dummy <- data.frame(x=x, y= x + rnorm(x)*w)

# Fit a simple linear regression
# With y to the left of the tilde then x, meaning y being dependent on x.
# Unlike other statistical packages, R does not display all output.  It is recommended
# to create an object to store the estimates.

fm <- lm(y ~ x, data=dummy)

# Display the summary of the output of model fm.
```

```r
summary(fm)
```

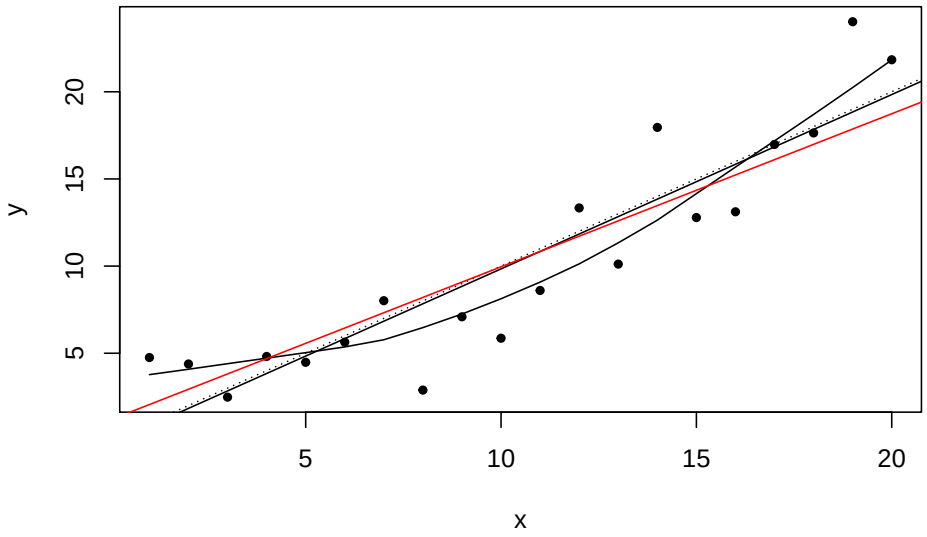
```
##
## Call:
## lm(formula = y ~ x, data = dummy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9525 -2.1010 -0.2029  1.6209  5.1769
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.1631     1.3046  -0.125    0.902
## x             1.0002     0.1089   9.184 3.26e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.808 on 18 degrees of freedom
## Multiple R-squared:  0.8241, Adjusted R-squared:  0.8144
## F-statistic: 84.35 on 1 and 18 DF,  p-value: 3.255e-08
```

```r
# Use w for a weighted regression.

fm1 <- lm(y ~ x, data=dummy, weight=1/w^2)

# Display the summary of the output of model fm1.

summary(fm1)
```

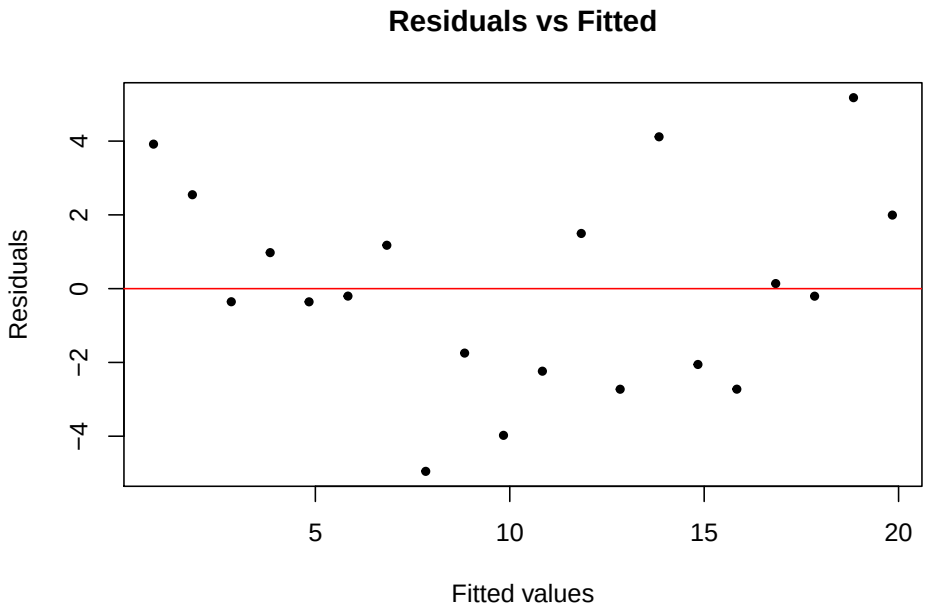
```
##
## Call:
## lm(formula = y ~ x, data = dummy, weights = 1/w^2)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2022 -0.7335 -0.1511  0.6572  1.9363
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1781     1.0186   1.157    0.263
## x             0.8781     0.1039   8.454 1.11e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.126 on 18 degrees of freedom
## Multiple R-squared:  0.7988, Adjusted R-squared:  0.7876
```

```
## F-statistic: 71.47 on 1 and 18 DF,  p-value: 1.107e-07
```

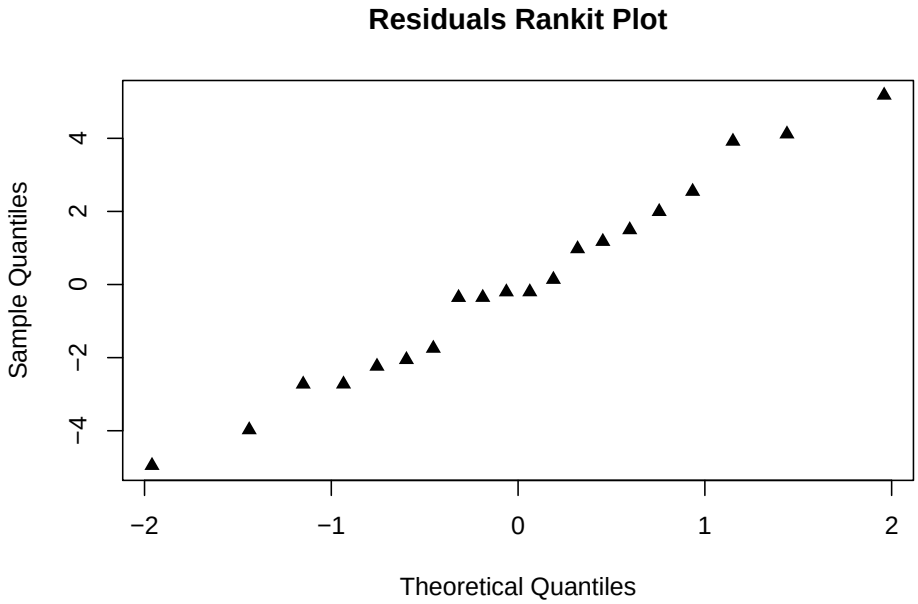
```
# Make the columns in the data frame visible as variables.

attach(dummy)

# Make a nonparametric local regression function.

lrf <- lowess(x, y)

# Standard point plot, with plotting character (pch) as bullet.

plot(x, y,pch=20)

# Add in the local regression.

lines(x, lrf$y)

# The true regression line: (intercept 0, slope 1, with dotted line type )

abline(0, 1, lty=3)

# Unweighted regression line.

abline(coef(fm))

# Weighted regression line.

abline(coef(fm1), col = "red")
```



```
# A standard regression diagnostic plot to check for heteroscedasticity. Can you see it?

plot(fitted(fm), pch=20, resid(fm), xlab="Fitted values", ylab="Residuals", main="Residuals vs Fi

# How about now?

abline(0,0, col="red")
```




```
# A normal scores plot to check for skewness, kurtosis and outliers.

qqnorm(resid(fm), main="Residuals Rankit Plot", pch=17)
```




```
# Cleaning up

rm(list = ls())
```

3.5 Recommended R Resources:

- The R Journal
- Introduction to R by W. N. Venables, D. M. Smith and the R Core Team
- Introduction to R Seminar at UCLA
- Getting Started in Data Analysis using Stata and R by Data and Statistical Services, Princeton University

Chapter 4

Python Programming

We describe Python in this chapter.

Chapter 5

JavaScript

Some JavaScript examples are demonstrated in this chapter.

5.1 Example one

5.2 Example two

Chapter 6

Final Words

We have finished a nice book.