Data Programming

Karl Ho

2019-05-29

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 data programmng or coding involving data. Data programming is a practice that works and evolves with data. Unlike the point-and-click approach, programming allows the user to manage most closely the data and process data in more effective manner. Programs are designed to be replicable, by user and collaborators. A data program can be developed and updated iteratively and incrementally. In other words, it is building on the culminated works without repeating the steps. It takes debugging, which is the process of identifying problems (bugs) but, in fact, updating the program in different situations or with differnt inputs when used in different contexts, including the programmer himself or herself working in future times.

2.1 Principles of Programming

Social scientists Gentzkow and Shapiro (2014) list out some principles for data 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/Modularity

- Organize by functions and data chunks

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

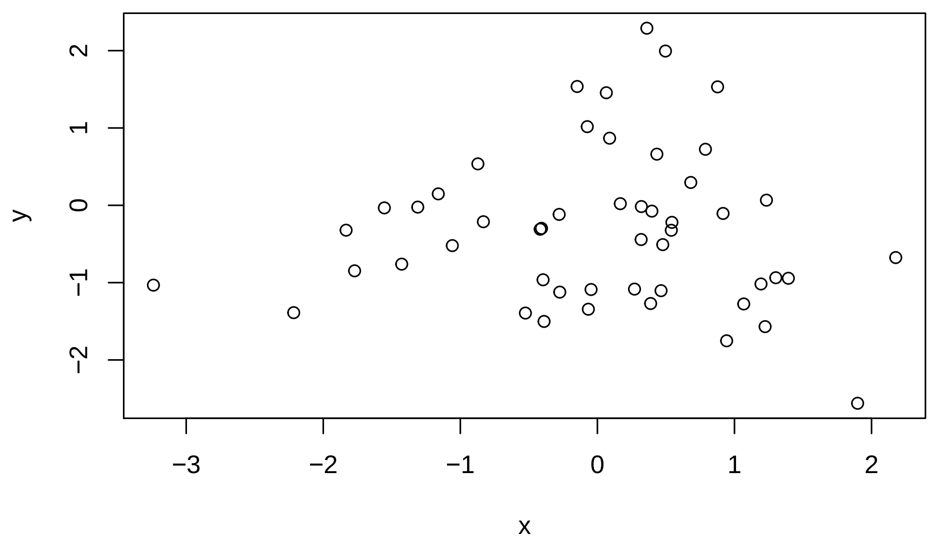
A data program can provide or perform :

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

Sample R Programs:

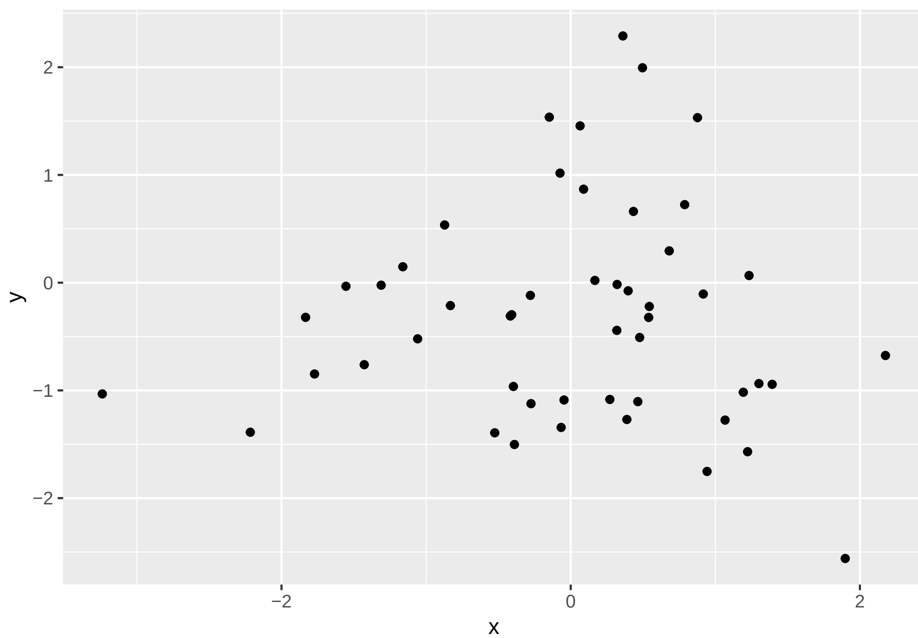
R basics

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

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



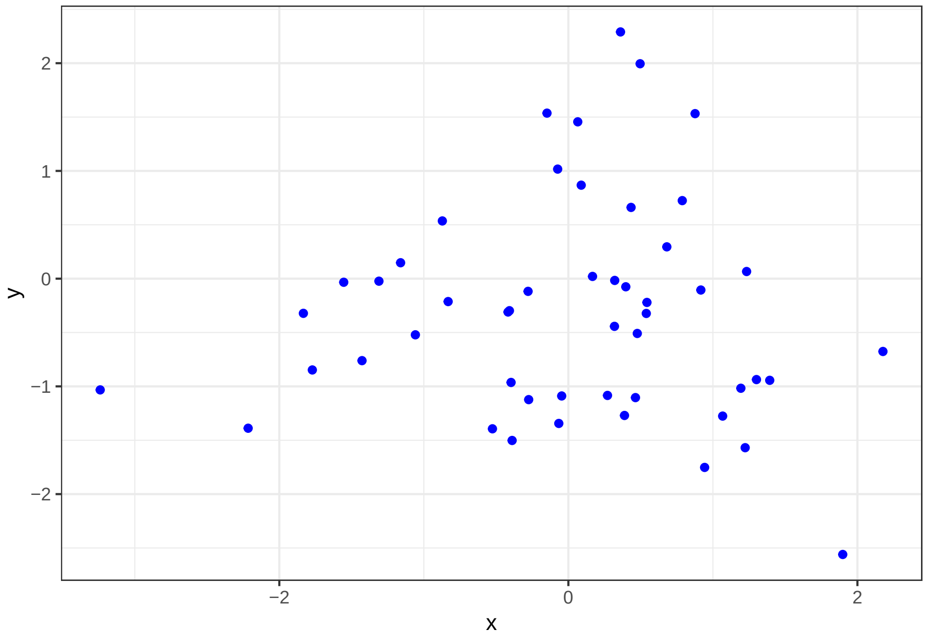
Using R packages

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



More R Data Visualization

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



Sample Python Programs (## represents output)

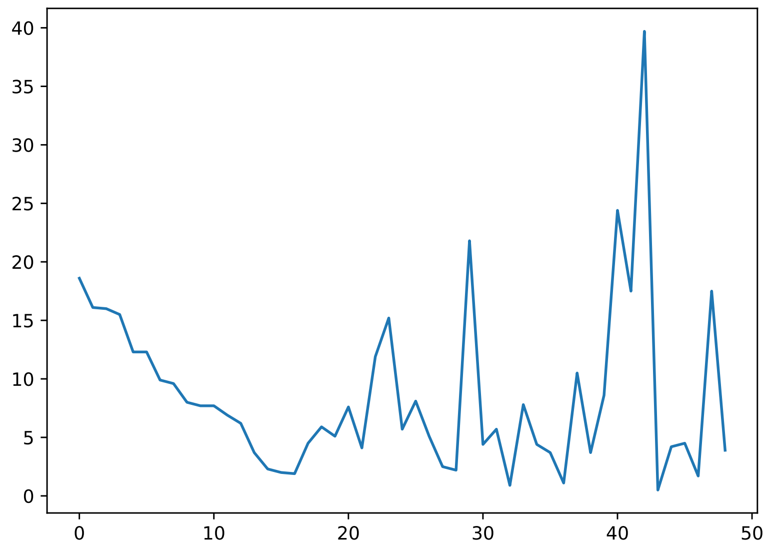
Python using Pandas

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

# 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%
```

Python using Matplotlib

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



In the subsequent chapters, sample programs will be provided to illustrate how to code or program in different environments to perform data science 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 over 14,000 available packages (as of May, 2019).
- Graphics!!!
- Scaleble and Portable
- Interface with other platform/langaugues (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)

3.4 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


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

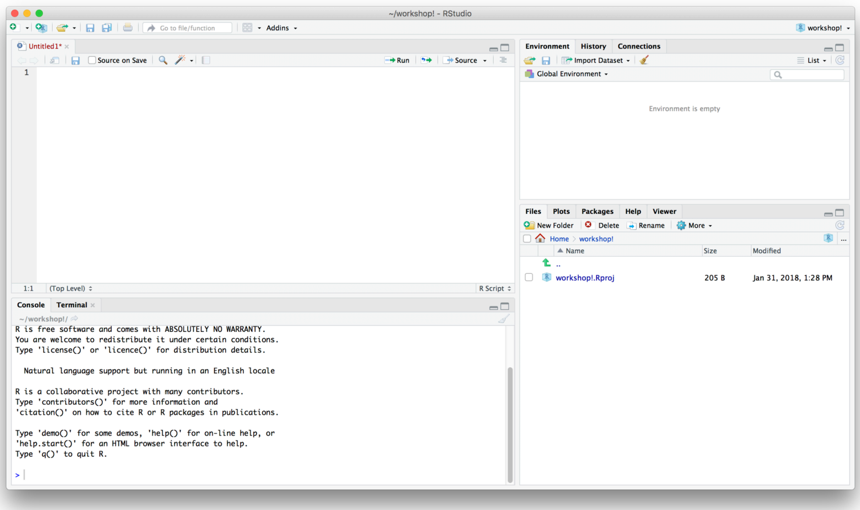


Figure 3.4: RStudio screenshot

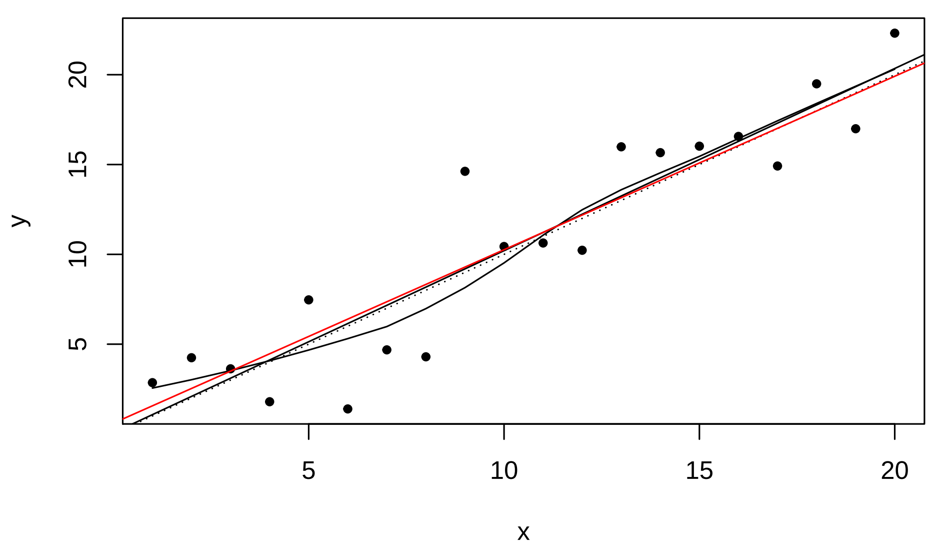
```
# Generate x, y and w to demontrate linear models and plots.
# Make x = (1,2,...,20).

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.

summary(fm)
```

```
##
## Call:
## lm(formula = y ~ x, data = dummy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7524 -2.3274  0.3992  1.8265  5.4309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06297    1.21496   0.052    0.959
## x            1.01420    0.10142  10.000 8.93e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.615 on 18 degrees of freedom
## Multiple R-squared:  0.8475, Adjusted R-squared:  0.839
## F-statistic: 99.99 on 1 and 18 DF,  p-value: 8.935e-09
```

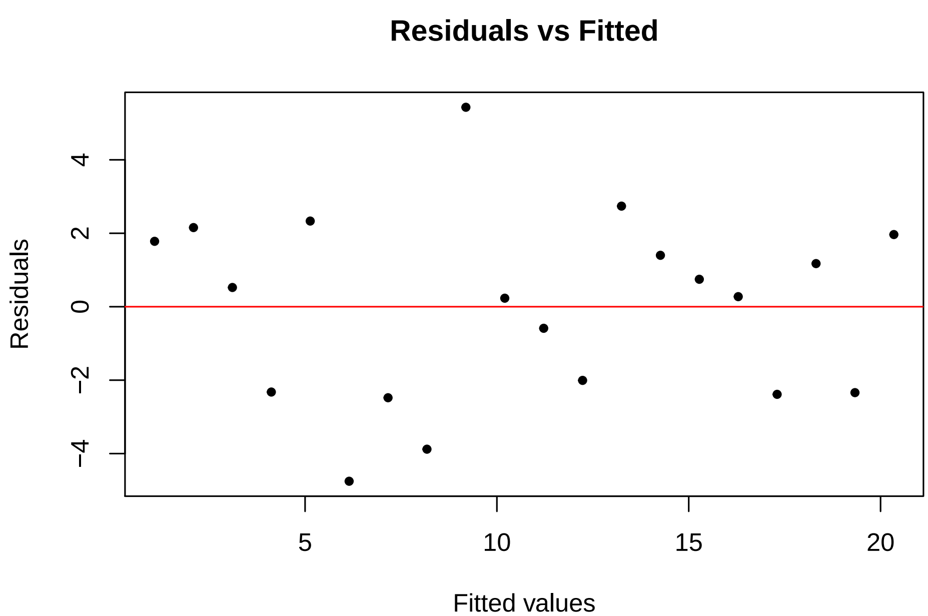
```r
# Use w for a weighted regression.

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
## -2.2475 -0.6936  0.1204  0.7708  2.1317
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6026     0.9960   0.605    0.553
## x             0.9655     0.1016   9.507 1.93e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.101 on 18 degrees of freedom
## Multiple R-squared:  0.8339, Adjusted R-squared:  0.8247
## F-statistic: 90.38 on 1 and 18 DF,  p-value: 1.934e-08
```

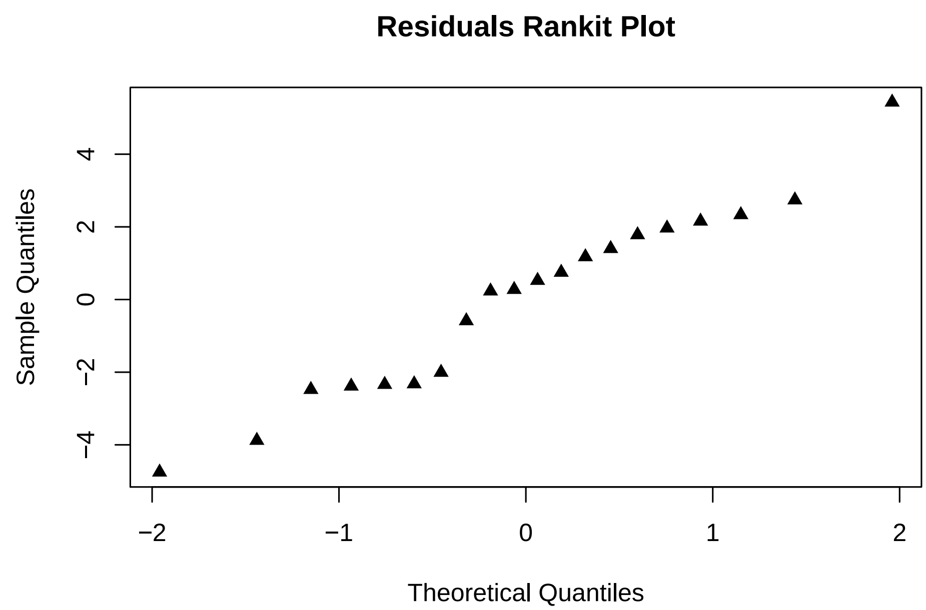
```
# Make the columns in the data frame visible as variables.
attach(dummy)
# Make a nonparametric local regression function.
lrf <- lowess(x, y)
# Standard point plot, with plotting character (pch) as bullet.
plot(x, y,pch=20)
# Add in the local regression.
lines(x, lrf$y)
# The true regression line: (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 Illustration

In this section, we demonstrate exploring data about Taiwan elections in 2016. The Taiwan Election and Democratization Study (TEDS) is one of the longest and most comprehensive elections studies starting in 2001. TEDS collects data through different modes of surveys including face-to-face interviews, telephone interviews and internet surveys. More detail of TEDS can be found at the National Chengchi University Election Study Center website at https://esc.nccu.edu.tw/main.php.

Taiwan Election and Democratization Study 2016 data

```
# Import the TEDS 2016 data in Stata format using the haven package
## install.packages("haven")

library(haven)
TEDS_2016 <- read_stata("https://github.com/datageneration/home/blob/master/DataProgra

# Prepare the analyze the Party ID variable
# Assign label to the values (1=KMT, 2=DPP, 3=NP, 4=PFP, 5=TSU, 6=NPP, 7="NA")

TEDS_2016$PartyID <- factor(TEDS_2016$PartyID, labels=c("KMT","DPP","NP","PFP", "TSU",
```

Take a look at the variable:

```
# Check the variable
attach(TEDS_2016)
```

```
## The following objects are masked from TEDS_2016 (pos = 13):
##
##     age, Age, Arear, Blue, Career, Career8, District, DPP,
##     Econ_worse, econworse5, edu, Edu, Ethnic, female,
##     Govt_dont_care, Govt_for_public, green, Green, highincome,
##     income, income_nm, Independence, Inequality, inequality5, KMT,
##     lowincome, Mainland_father, Minnan_father, nI2, No_Party,
##     noparty, north, npp, Party, PartyID, pfp, pubwelf5, Sex,
##     South, sq, Taiwanese, Tondu, Tondu3, Unification, voteblue,
##     voteblue_nm, votedpp_1, votekmt, votekmt_1, votekmt_nm,
##     votetsai, votetsai_all, votetsai_nm, whitecollar
```

```
head(PartyID)
```

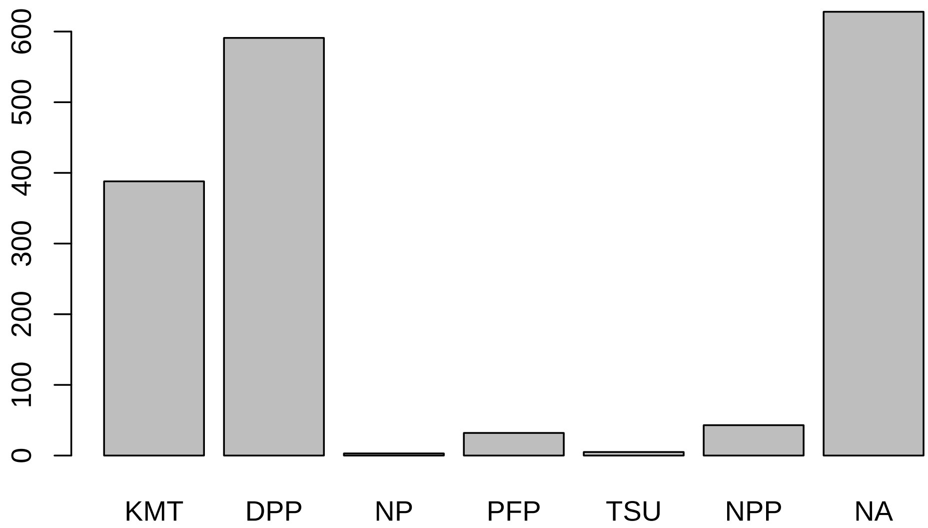
```
## [1] NA   NA   KMT NA   NA   DPP
## Levels: KMT DPP NP PFP TSU NPP NA
```

```
tail(PartyID)
```

```
## [1] NA   NA   DPP NA   NA   NA
## Levels: KMT DPP NP PFP TSU NPP NA
```

Frequency table:

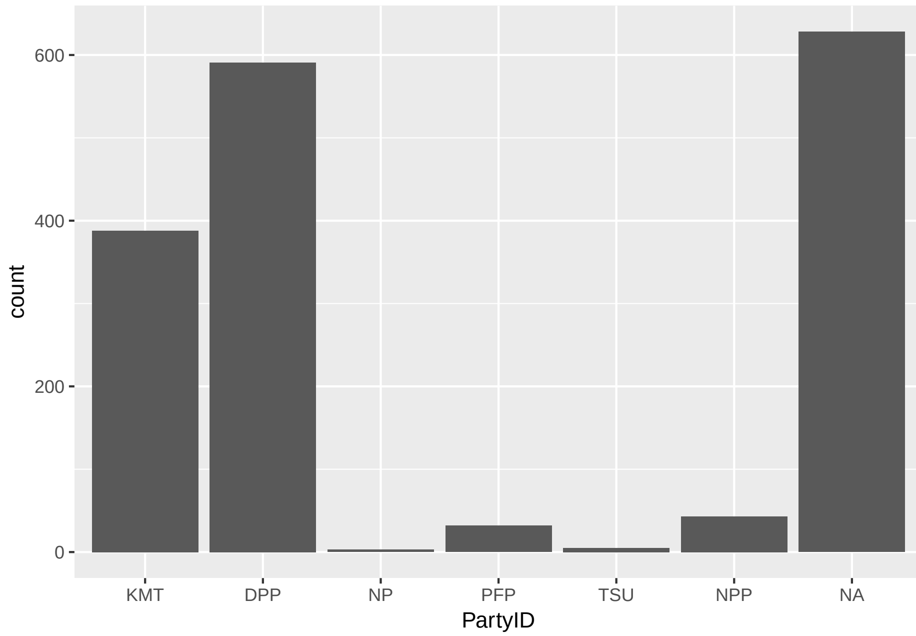
```
# Run a frequency table of the Party ID variable using the descr package
## install.packages("descr")
library(descr)
freq(TEDS_2016$PartyID)
```




```
## TEDS_2016$PartyID
##       Frequency  Percent
## KMT         388  22.9586
## DPP         591  34.9704
## NP            3   0.1775
## PFP          32   1.8935
## TSU           5   0.2959
## NPP          43   2.5444
## NA          628  37.1598
## Total      1690 100.0000
```

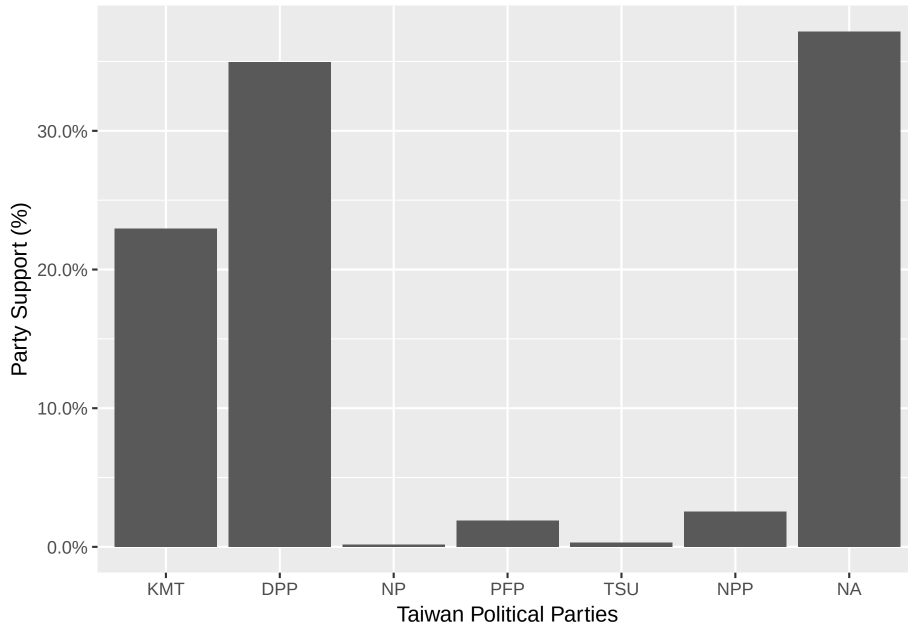
Get a better chart of the Party ID variable:

```
# Plot the Party ID variable
ggplot(TEDS_2016, aes(PartyID)) +
  geom_bar()
```



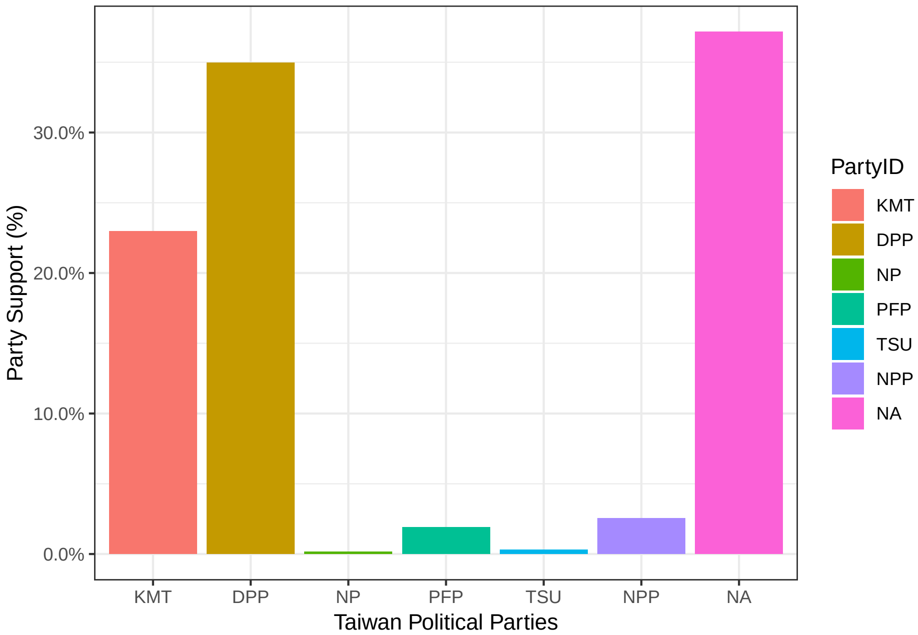
We can attend to more detail of the chart, such as adding labels to x and y axes, and calculating the percentage instead of counts.

```
ggplot(TEDS_2016, aes(PartyID)) +
  geom_bar(aes(y = (..count..)/sum(..count..))) +
  scale_y_continuous(labels=scales::percent) +
  ylab("Party Support (%)") +
  xlab("Taiwan Political Parties")
```



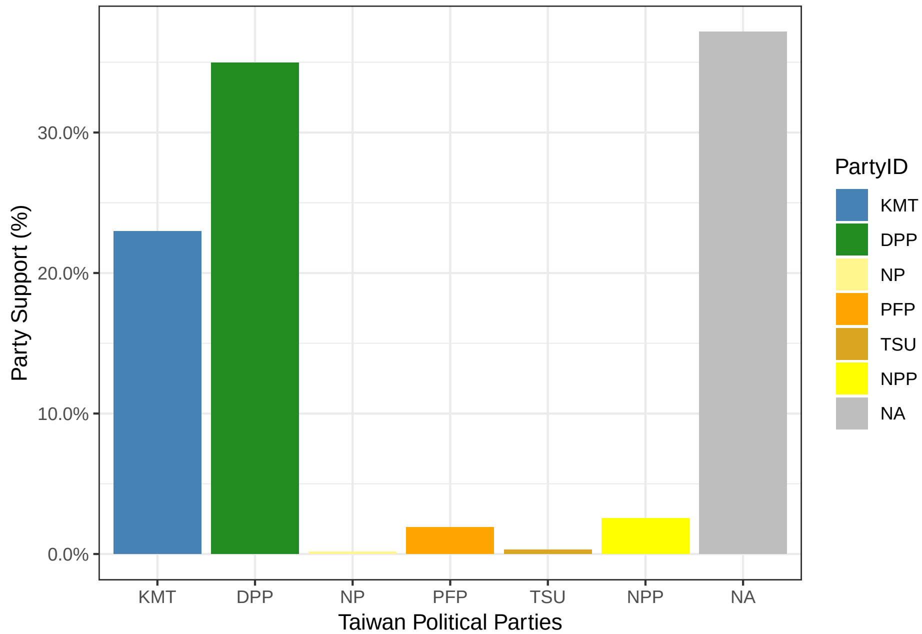
Adding colors, with another theme:

```
ggplot(TEDS_2016, aes(PartyID)) +
  geom_bar(aes(y = (..count..)/sum(..count..),fill=PartyID)) +
  scale_y_continuous(labels=scales::percent) +
  ylab("Party Support (%)") +
  xlab("Taiwan Political Parties") +
  theme_bw()
```



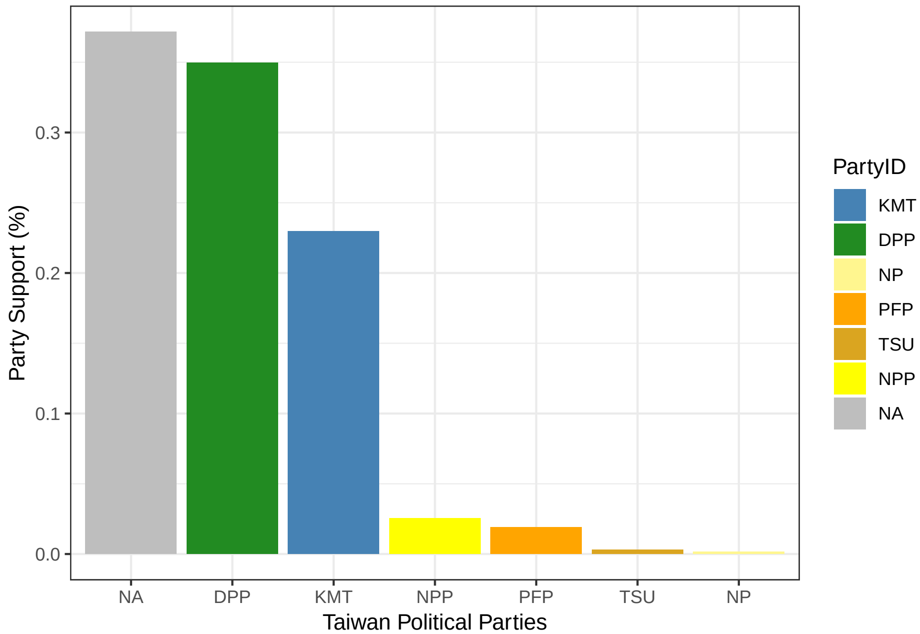
Hold on, colors are not right!

```
ggplot(TEDS_2016, aes(PartyID)) +
  geom_bar(aes(y = (..count..)/sum(..count..),fill=PartyID)) +
  scale_y_continuous(labels=scales::percent) +
  ylab("Party Support (%)") +
  xlab("Taiwan Political Parties") +
  theme_bw() +
  scale_fill_manual(values=c("steel blue","forestgreen","khaki1","orange","goldenrod","yellow","g
```



To make the chart more meaningful, we can use a package called tidyverse to manage the data.

```
##install.packages("tidyverse")
library(tidyverse)
TEDS_2016 %>%
  count(PartyID) %>%
  mutate(perc = n / nrow(TEDS_2016)) -> T2
ggplot(T2, aes(x = reorder(PartyID, -perc),y = perc,fill=PartyID)) +
  geom_bar(stat = "identity") +
  ylab("Party Support (%)") +
  xlab("Taiwan Political Parties") +
  theme_bw() +
  scale_fill_manual(values=c("steel blue","forestgreen","khaki1","orange","goldenrod","
```



3.6 Exercise

Analyze the Tondu (□□□variable using the following procedures:

1. Assign label to each category
2. Run a frequency table using descr
3. Plot the variable using ggplot2

Hint:

- Prepare the analyze the Tondu variable using these labesl: ("Unification now","Status quo, unif. in future","Status quo, decide later","Status quo forever", "Status quo, indep. in future", "Independence now","No response")
- Sample codes:

```
TEDS_2016$Tondu<-factor(TEDS_2016$Tondu,labels=c("Unification now","Status quo, unif. in >future'
```

3.7 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

4.1 What is Python?

- Interpreted, high level computer language
- Invented by Dutch programmer Guido van Rossum
- Named after the TV Show *Monty Python's Flying Circus*
- Open sourced programming language

4.1.1 Why Python?

- Simplicity
- Large ecosystem of domain-specific tools to facilitate scientific - computing and data science
- User-built packages
- Data management

Figure 4.1: Python Inventor Guido van Rossum

  - Web data
  - Data munging

4.2 Python basic packges:

1. NumPy - manipulation of homogeneous array-based data
2. Pandas - manipulation of heterogeneous and labeled data
3. SciPy - for common scientific computing tasks
4. Matplotlib - data visualizations
5. IPython - interactive execution and sharing of code using Jupyter notebook
6. Scikit-Learn - machine learning

4.3 Python IDE

Choice of Integrated Desktop Environment matters! There are plenty of IDE available for python programming and developments. To name a few:

- IDLE
- Pycharm
- Jupyter Notebook
- Spyder
- Rodeo
- R Studio

4.4 Basic operations and object assignment

```
# 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
```

4.4.1 Import libraries

```
#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
```

4.4.2 Import data

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

# 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%
```

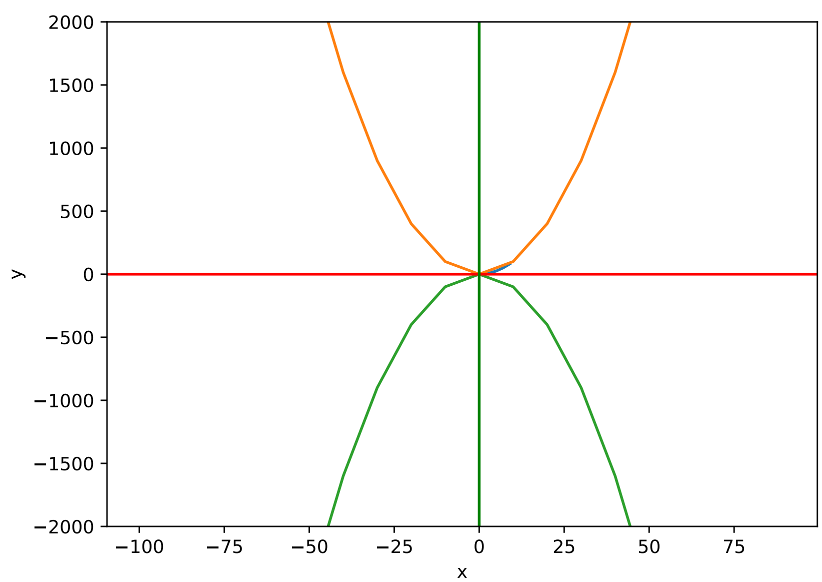
4.4.3 Simple plot

```
# Creating variables
xs = [1,3,5,7,9]
ys = [x**2 for x in xs]
```

```
# Simple plot
plt.plot(xs, ys)

xs = range(-100,100,10)
x2 = [x**2 for x in xs]
negx2 = [-x**2 for x in xs]

# Combined plot

plt.plot(xs, x2)
plt.plot(xs, negx2)
plt.xlabel("x")
plt.ylabel("y")
plt.ylim(-2000, 2000)
```

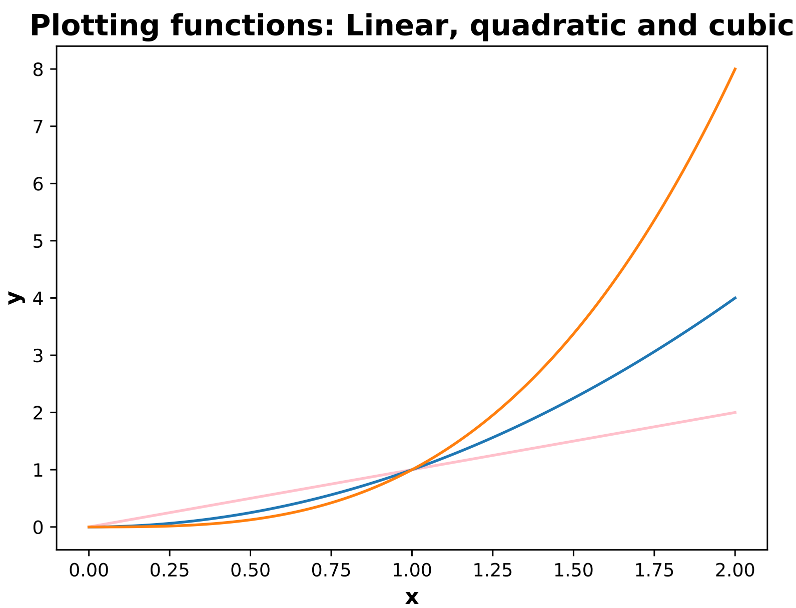
```
## (-2000, 2000)
```

```
plt.axhline(0,color="red") # horiz line
plt.axvline(0,color="green") # vert line
plt.show()
```



4.4.4 Visualizing data

```
import matplotlib.pyplot as plt

x = np.linspace(0, 2, 100)
plt.plot(x, x, label='linear',color="pink")
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x',fontsize=12,fontweight='bold')
plt.ylabel('y',fontsize=12,fontweight='bold')

plt.title("Plotting functions: Linear, quadratic and cubic", fontsize=16,fontweight='bold')
```



4.5 Exercise

This exercise is designed to run in class. Students are advised to install Anaconda 3 to own computer.

1. Launch Jupyter Notebook from Anaconda
2. On Applications pulldown menu, choose anaconda3
3. Run sample programs from class GitHub (https://github.com/datageneration/dataprogramming/)

4.6 Recommended Python Resources:

- A Whirlwind Tool of Python: Getting started
- Datacamp: Online training courses
- Matplotlib.org: Data visualization

Chapter 5

JavaScript

5.1 What is JavaScript?

- JavaScript is not related to Java
- Created by Brendan Eich in 1995
- Originally developed as a prototype language for web browser (Client-side).
- Now used in server-side (Node.js) as well.
- Not related to Java, just named similarly for marketing purpose.
- C style syntax but got inspiration from Functional programming
- for, while, continue, break, if/else, switch are similar to C
- operators (+,-,*,/,%) are also similar (except ==,!=,||)
- include function operations such as map, reduce, forEach.

5.1.1 JavaScript Data Types

Data Types

- Numbers: 42, 3.14159

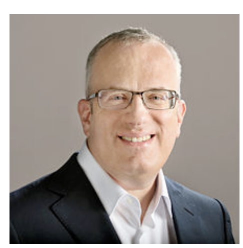


Figure 5.1: JavaScript Inventor Brendan Eich

- Logical: true, false
- Strings: "Hello", 'Taiwan'
- null
- undefined* - undefined is not null!

5.1.2 JSON

- JavaScript Object Notation
- JavaScript as an XML alternative for storing data
- e.g.

[{"Station":"Alishan","Temperature":14.5,"Precipitation":812.4,"Humidity":95,"Pressure":762.5,"dayrain":30},….]

5.2 What is D3?

- D3 stands for Data-Driven Documents. -d3.js (D3) is "a JavaScript library for manipulating documents based on data".
- D3 can be used in conjunction with HTML and CSS (amongst others) to visualize data on a webpage.
- It's an open framework.
- It embeds or includes data in scripts to create images in webpages.

> "With D3, designers selectively bind input data to arbitrary document elements, applying dynamic transforms to both generate and modify content."
>
> —Bostock, Ogievetsky and Heer, 2011

5.2.1 D3 and web documents

- D3 is web-based, working with following components:
  - HTML (Hypertext Markup Language)
  - CSS (Cascade Style Sheet)
  - JavaScript(js)
  - SVG (Scalable Vector Graphics), interpreted graphic output

All of the above can be coded using a text editor. Output needs a browser with JavaScript console

5.2.2 Sample D3 graphics

Interactive Ladder Graph

Click here to access the online version

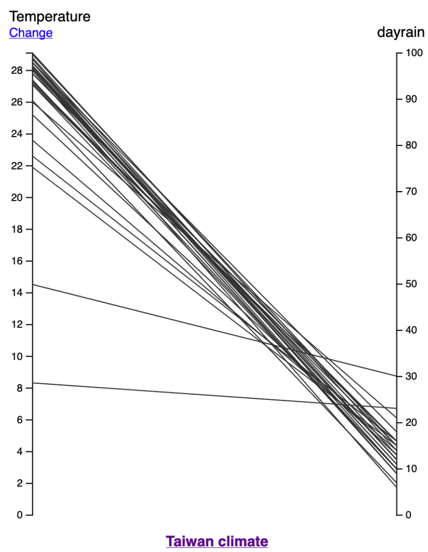


Figure 5.2: D3: Ladder graph

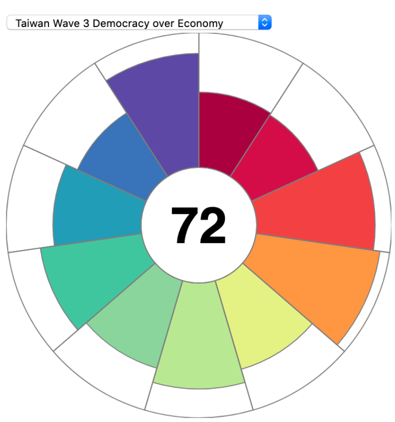


Figure 5.3: D3: Aster graph

Interactive Aster Graph

Click here to access the online version

Interactive Network Graph

Chapter 6

Summary

This manuscript provides brief notes and sample programs for the “Data Programming” course covering the basic programming for Data Science. The languages included in this volume are primarily R, Python and JavaScript. There will be more developments to add materials and sample programs to build this manuscript into a full-blown codebook for data science.

More materials can be accessed at the GitHub:

https://www.github.com/datageneration/dataprogramming/