Data Programming with R

Karl Ho

9/18/2022

Table of contents

Preface

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 is an interpreted languages. In other words, the programs do not need compilation but will run in an environment to get the outputs. In this course, that is RStudio.

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 version 4.2.1 or later (https://cran.r-project.org)
2. RStudio version 1.2.x (https://www.rstudio.com)
3. Text editor of own choice (e.g. Atom, Sublime Text, Ultraedit)

Recommended websites/accounts:

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

1 Summary

This book is designed for the short course titled **Data Programming with R**, offered in the University of Texas at Dallas. It provides training for aspiring data scientists to program surrounding data using R. The primary goal is to build a comprehensive program preparing students in four major elements in Data Science:

1. Data collection
2. Data visualization
3. Data management
4. Data modeling

The major platform is R with the IDE (Integrated Development Environment) is Rstudio. However, the principles and applications can be used for other languages and platforms such as Python and Julia.

Part I

Introduction

2 Introduction

This chapter introduces the general principles for data programming 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 different inputs when used in different contexts, including the programmer himself or herself working in future times.

2.1 Principles of Data 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 :

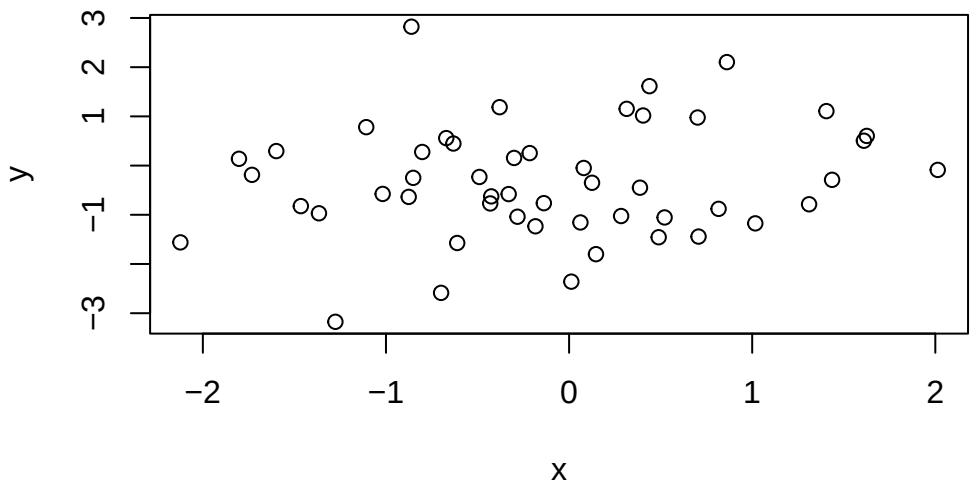
1. Data source
2. Documentation of data
3. Importing and exporting data
4. Management of data
5. Visualization of data
6. Data models

Sample R Programs:

R basics

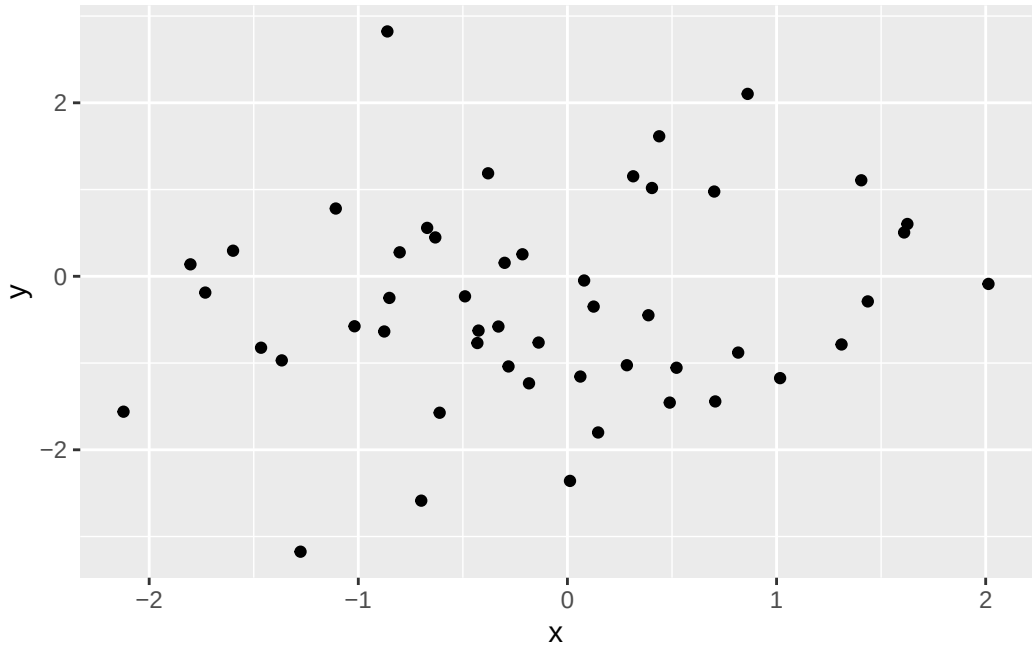
```
# Create variables composed of random numbers
x <-rnorm(50)
y = rnorm(x)

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



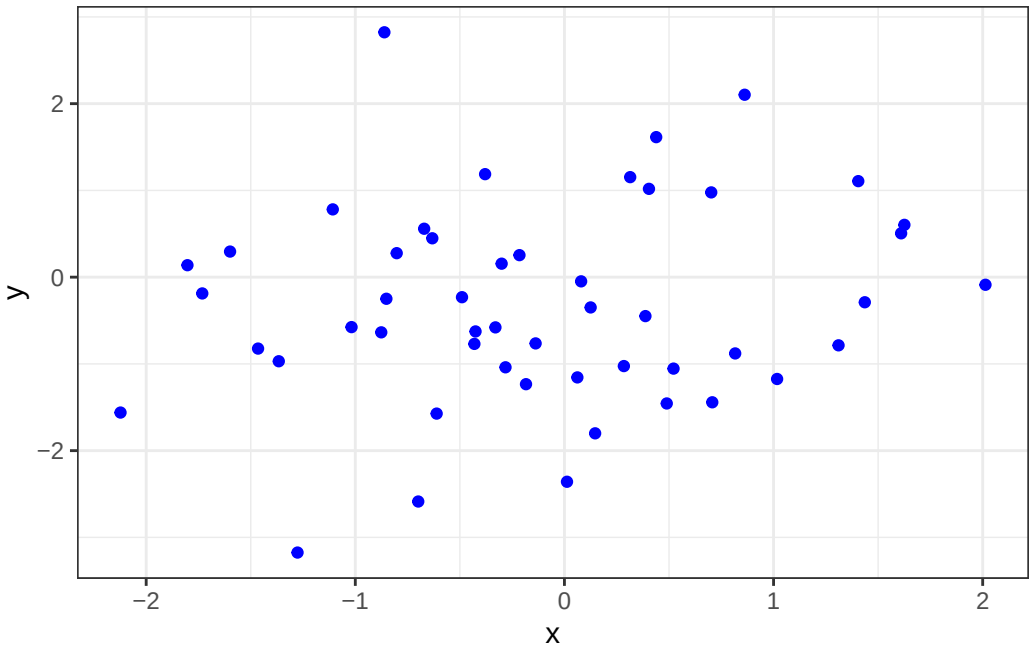
Using R packages

```r
# 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

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


2
0
−2
y
−2
−1
0
1
2
x

3 R Basic operations

This chapter introduces basic R operations including installing packages, operators, loading libraries and writing functions.

3.1 Create a project

Before diving into R programming, it is highly recommended creating a project. It will organize all related objects and files under one roof. Version control can be applied for automatically saving history (log) and data objects. Projects can facilitate additional specialized works such as:

- Writing a book using Quarto or bookdown
- Creating a website or a blog using Quarto
  - Highly recommended to use this feature to create personal website
- Software development
- Data visualization using Shiny
- Manage data in database servers or other platform

However, to start R programming with a project helps better organize the process of data analytics. A program is not just running a procedure but performing more data science tasks that last more than just one time execution. It scales!

For more detail, consult Using RStudio Projects

3.2 Operators

3.2.1 Assignment operators

<- and = are both the assignment operator where as = must be used as top level.

|> is the base R "pipe" operator, feeding value (arguement) into a function.

```
[1] 15
```

The other pipe operator %>% (package magrittr), which is more commonly used especially in tidyverse. Using %>%, the argument can be piped into the function without ().

```
[1] 15
```

3.2.2 Math operators

The common operators used in math are also applicable in the R environment.

- Arithmetic: +,-,*,/,^ (power)
- Logical: & (and), | (or), ! (Not)
- Relational: >,<, ==, >=, <=, !=

3.3 Package Management

3.3.1 Install packages

R comes with basic functions and demo datasets (e.g. mtcars, Titanic, iris). For additional functionalities or specialized functions, installing additional packages is needed. Use `install.packages("ISLR2")` for installation and it only needs be done the first time. However, for every new session, library() function is needed to call in (load) the package for remaining program.

3.3.2 Libraries

The `library()` function is used to load *libraries*, or groups of functions and data sets that are not included in the base `R` distribution. Basic functions that perform least squares linear regression and other simple analyses come standard with the base distribution, but more exotic functions require additional libraries. The $ISLR$ book uses the `MASS` package, which is a very large collection of data sets and functions. The `ISLR2` package also includes the data sets associated with this book for demonstration.

```
library(MASS)
library(ISLR2)
```

```
Attaching package: 'ISLR2'
```

```
The following object is masked from 'package:MASS':

    Boston
```

If you receive an error message when loading any of these libraries, it likely indicates that the corresponding library has not yet been installed on your system. Some libraries, such as `MASS`, come with `R` and do not need to be separately installed on your computer. However, other packages, such as `ISLR2`, must be downloaded the first time they are used. This can be done directly from within `R`. For example, on a Windows system, select the `Install package` option under the `Packages` tab. After you select any mirror site, a list of available packages will appear. Simply select the package you wish to install and `R` will automatically download the package. Alternatively, this can be done at the `R` command line via `install.packages("ISLR2")`. This installation only needs to be done the first time you use a package. However, the `library()` function must be called within each `R` session.

3.3.3 Speed and organization

The general principle of using packages is: only load what you need! There are over 20,000 packages but the memory is limited. This article gives some comparison on different methods of using and loading packages or libraries. Again, using Project to manage your resources and it is advised to restart the R session (Session –> Restart R) to start with a clean slate for each project.

3.4 Writing Functions

As we have seen, `R` comes with many useful functions, and still more functions are available by way of `R` libraries. However, we will often be interested in performing an operation for which no function is available. In this setting, we may want to write our own function. For instance, below we provide examples of simple functions. The first one reads in the `ISLR2` and `MASS` libraries, called `LoadLibraries()`. Before we have created the function, `R` returns an error if we try to call it.

```
LoadLibraries
```

```
Error in eval(expr, envir, enclos): object 'LoadLibraries' not found
```

```
LoadLibraries()
```

```
Error in LoadLibraries(): could not find function "LoadLibraries"
```

We now create the function. Note that the `+` symbols are printed by `R` and should not be typed in. The `{` symbol informs `R` that multiple commands are about to be input. Hitting *Enter* after typing `{` will cause `R` to print the `+` symbol. We can then input as many commands as we wish, hitting {*Enter*} after each one. Finally the `}` symbol informs `R` that no further commands will be entered.

```
LoadLibraries <- function() {
  library(ISLR2)
  library(MASS)
  print("The libraries have been loaded.")
}
```

Now if we type in `LoadLibraries`, `R` will tell us what is in the function.

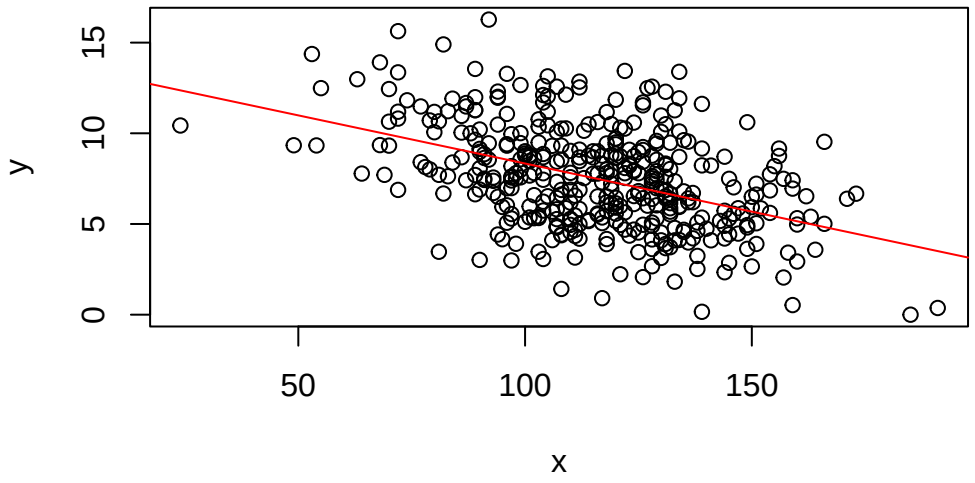
```
LoadLibraries()
```

```
[1] "The libraries have been loaded."
```

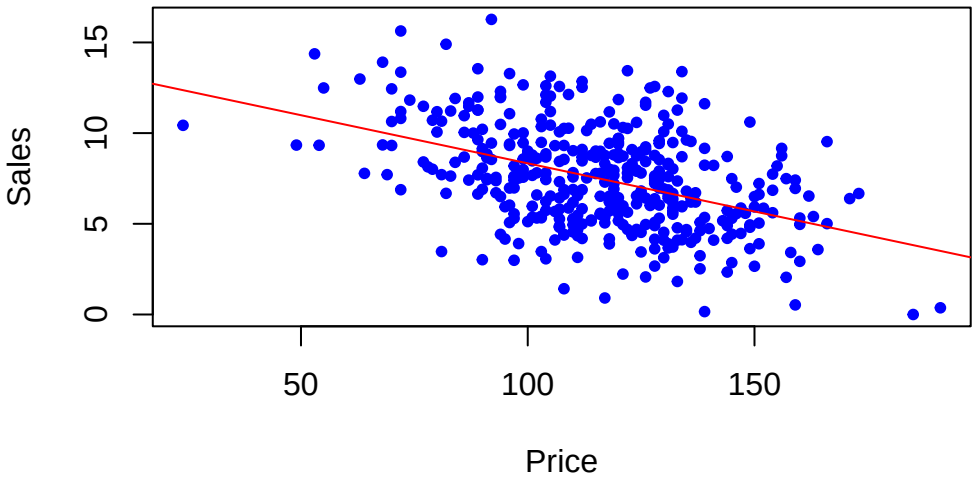
If we call the function, the libraries are loaded in and the print statement is output.

The following example demonstrates creating a function out of existing functions from different packages:

```
###Writing R functions
## Combine the lm, plot and abline functions to create a one step regression fit plot func
regplot=function(x,y){
  fit=lm(y~x)
  plot(x,y)
  abline(fit,col="red")
}
attach(Carseats)
regplot(Price,Sales)
```




```r
## Allow extra room for additional arguments/specifications
regplot=function(x,y,...){
  fit=lm(y~x)
  plot(x,y,...)
  abline(fit,col="red")
}  # "..." is called ellipsis, which is designed to take any number of named or unnamed ar
regplot(Price,Sales,xlab="Price",ylab="Sales",col="blue",pch=20)
```



The following example creates a function to deal with package management:

```r
# Create preload function
# Check if a package is installed.
```

```
# If yes, load the library
# If no, install package and load the library

preload<-function(x)
{
  x <- as.character(x)
  if (!require(x,character.only=TRUE))
  {
    install.packages(pkgs=x,  repos="http://cran.r-project.org")
    require(x,character.only=TRUE)
  }
}
```

Let's try preloading the package name **tidyverse** (be sure to wrap the name with double quotes " ":

3.5 Workshop 1

1. Create the following objects:
   - x <-rnorm(30)
   - y = rnorm(x)
2. Plot:
   - histogram of y (hint: use the hist() function)
   - Both x and y, using pch=20 (choose your own color using col=" ")
3. Check the environment
   1. Clean all objects using the following command:

      ```
      rm(list=ls())
      ```
4. Alternatively, you can Ctrl+Shift+F10 (Mac: Command+Shift+0) to restart R session

3.5.1 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

3.5.2 References:

Graham Williams 2011. *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge*

Part II

Communicative Programming using Quarto

4 Quarto

This chapter gives a brief introduction of communicative programming using Quarto. The idea is to incorporate data programming elements into a publishable document. Formats including PDF and HTML.

Part III

Data Collection

5 Data Collection with R

This chapter covers the basic methods of collecting data using R. Simple scrapping methods will be introduced like using *rvest* , *rtweet* and *academictwitteR*.

5.0.1 What is Big Data?

The [term] *Big Data* is about data that has huge volume, cannot be on one computer. Has a lot of variety in data types, locations, formats and form. It is also getting created very very fast (velocity) (Doug Laney 2001).

Big data is about the size of the data file. Big data is about the data generating process and data collection process.

Survey data, no matter how many cases, should be classified as small data instead of big data due to the collection process and design.

Burt Monroe, the founder of the Social Data Analytics program at the Penn State University, which is the first of its kind, gave new V's to Big Data.

- Volume
- Variety
- Velocity
- Vinculation
- Validity

He argues that Big Data is not just big, diverse and fast, it is inter-connected and must search for significance.

5.0.2 Why we need to collect data or original data?

One key element of data programming is the program is created for data, including collecting and updating data. Collection or production of data constitutes the major component in data science. No data scientist can count on others to provide data without knowing the source and generation method of the data. Knowing the data generation process is critical in preparing data for the rest of the data science processes including management, visualization and in particular modeling. Imagine if the date generation process is unknown, visualization and

model thus created could be faulty. Missing values, for instance, can mask the visualization and totally distort the modeling results.

5.0.3 Types of data by data generation

1. Made data vs. Found data
2. Structured vs. Semi/unstructured
3. Primary vs. secondary data
4. Derived data
5. metadata, paradata

5.0.3.1 Made data (by production)

1. Survey
2. Interviews
3. Experiments
4. Focus group

5.0.3.2 Found data (by collection)

1. Open data
2. API
3. Non-API
4. Open data

Open data refers to the type of data usually offered by government (e.g. Census), organization or research institutions (e.g. ICPSR. Some may require application for access and others may be open for free access (usually via websites or GitHub).

Since open data are provided by government agencies or research institutions, these data files are often:

- Structured
- Well documented
- Ready for data/research functions

2. API

API stands for Application Programming Interface. It is a web service that allows an interaction with, and retrieval of, structured data from a company, organization or government agency.

Example: Social media (e.g. Facebook, YouTube, Twitter), Government agency (e.g. Congress)

APIs can take many different forms and be of varying quality and usefulness. RESTful API (Representational State Transfer) is a means of transferring data using web protocols

Like open data, data available through API are generally:

- Structured
- Somewhat documented
- Not necessary fully open
- Subject to the discretion of data providers (e.g. Not all variables are available, rules may change without announcements, etc.)

Example:

- Crossref API
- Taiwan Legislative Yuan (Congress) API

3. Non-API

For the type of found data not available via API or open access, one can use non-API methods to collect this kind of data. These methods include scraping, which is to simulate web browsing but through automated scrolling and parsing to collect data. These data are usually non-structured and often times noisy. Researchers also have little control over data generation process and sampling design.

Non-API data are generally:

- Non-structured
- Noisy
- Undocumented with no or little information on sampling

5.0.4 Illustrations: Open data

5.0.4.1 Stock data

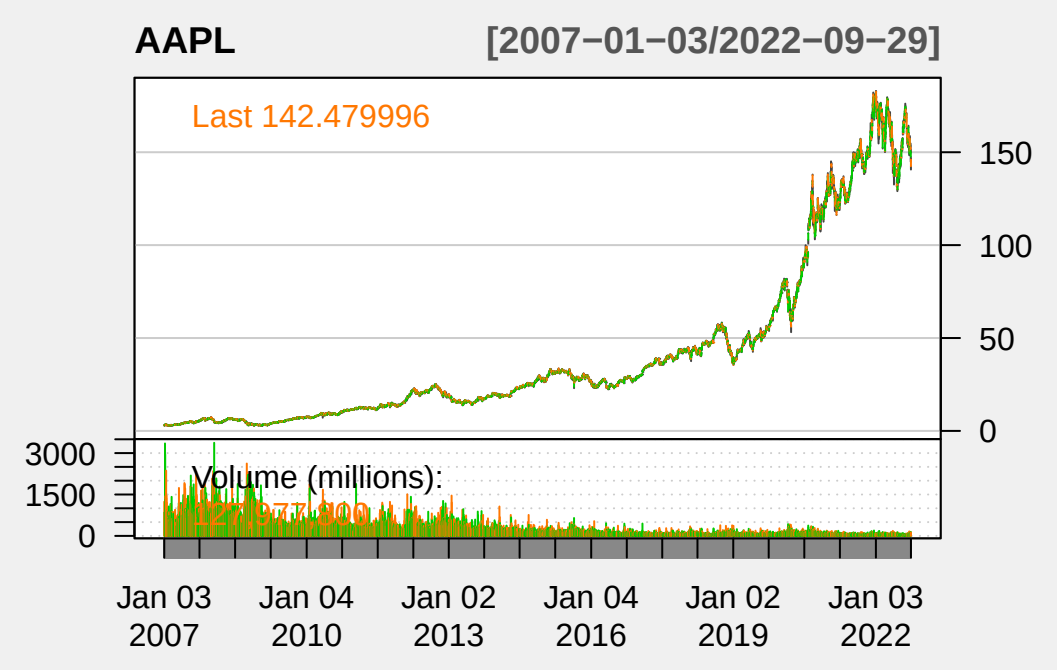
```
# Collecting Stock data and plotting stock data as time series
# install.packages(c("quantmod", "ggplot2", "magrittr","broom","googlesheet4"))
# lapply(c("quantmod", "ggplot2", "magrittr","broom","googlesheet4"), require, character.o
library(quantmod)
library(ggplot2)
```

```
library(magrittr)
library(broom)

# Setting time period
start = as.Date("2010-07-01")
end = as.Date("2022-09-30")
getSymbols("AAPL")
```

```
[1] "AAPL"
```

```
chartSeries(AAPL, theme="white")
```



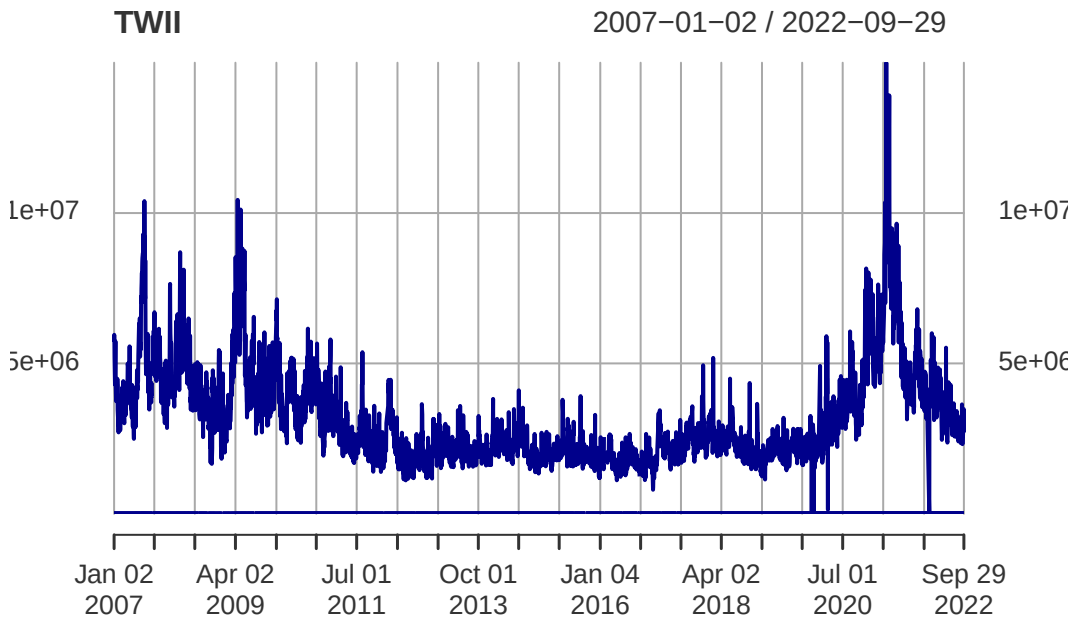

```
# Download Taiwan Weighted Index
getSymbols("^TWII", src="yahoo") # TWSE:IND
```

```
[1] "^TWII"
```

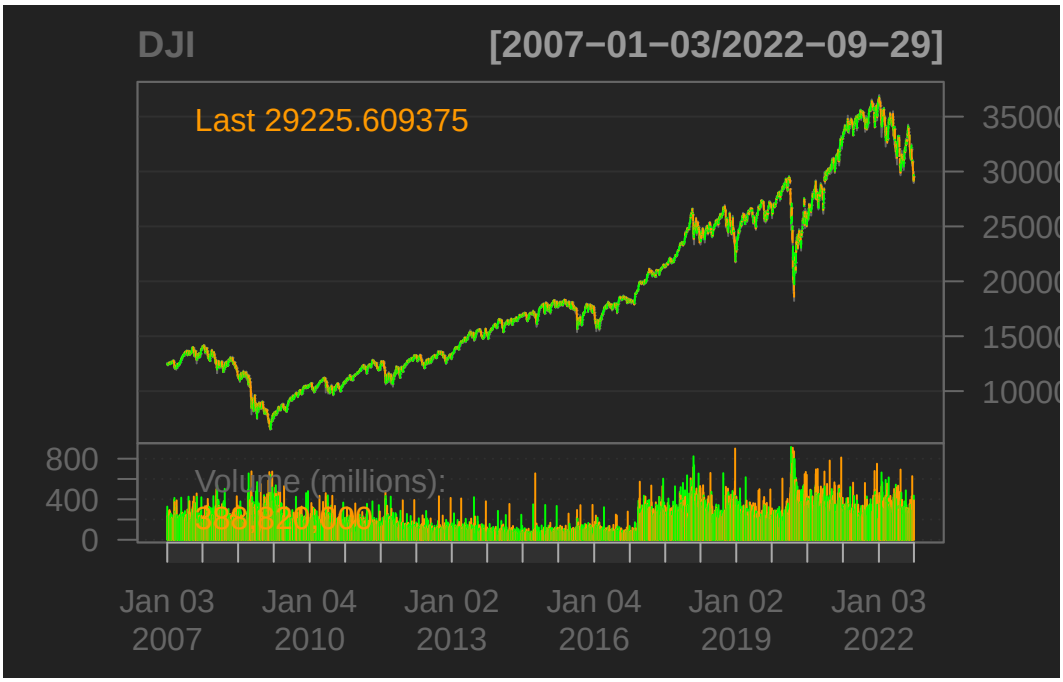
```
# Download Dow Jones Industrial Average
getSymbols("^DJI", src="yahoo") # Dow Jones Industrial Average
```

```
[1] "^DJI"
```

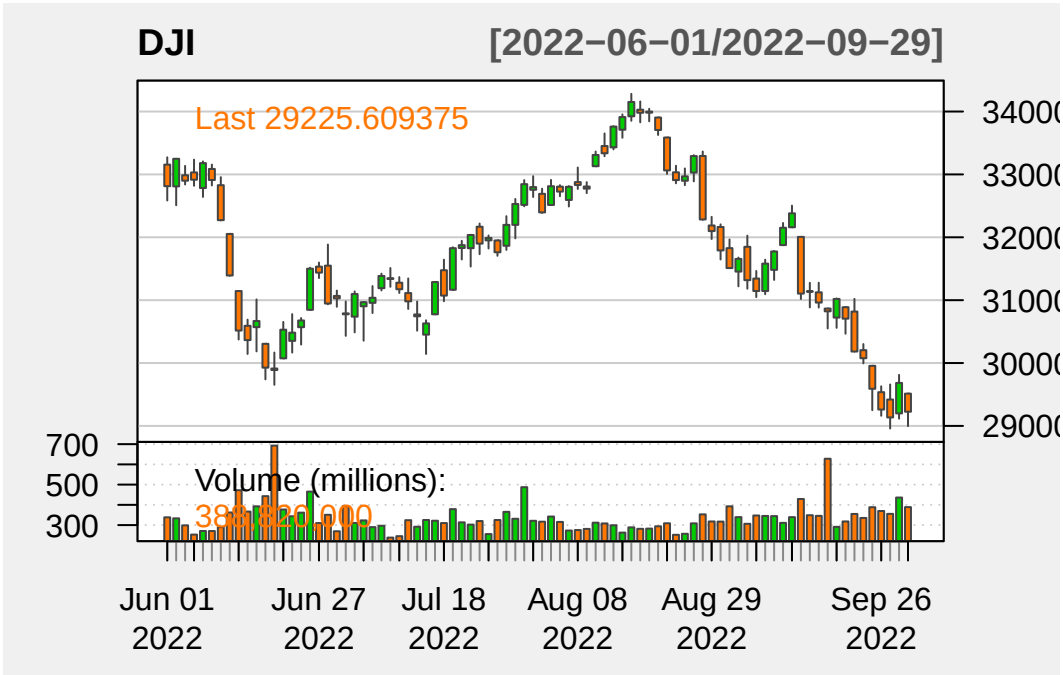
```
# Simple plot
plot(TWII, col="darkblue")
```



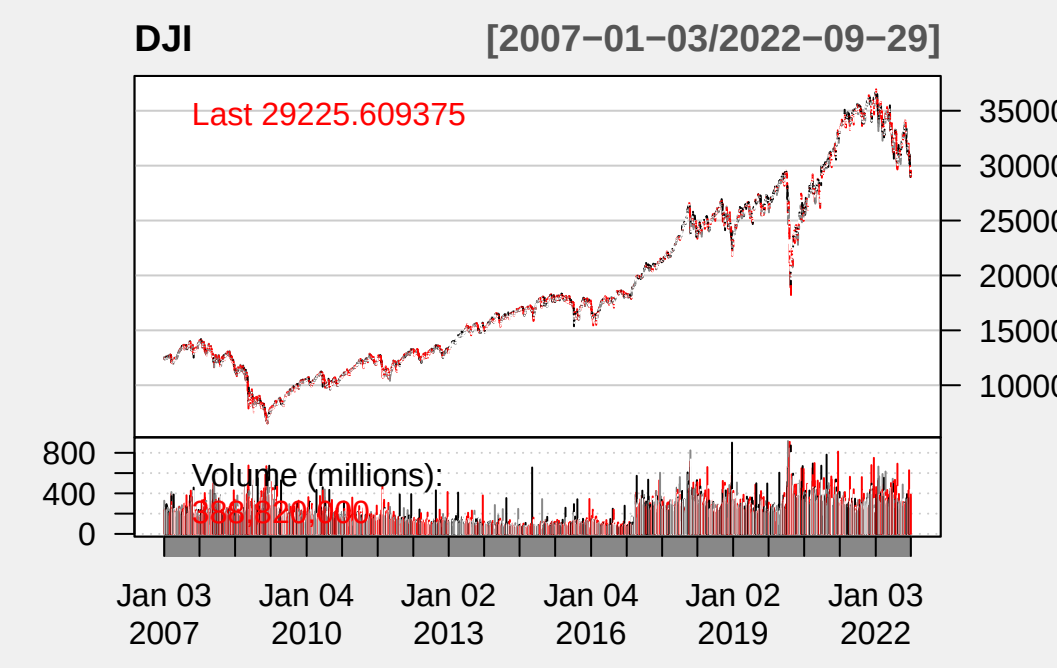

```
# Plot candle stick and other charts using quantmod
chartSeries(DJI)
```



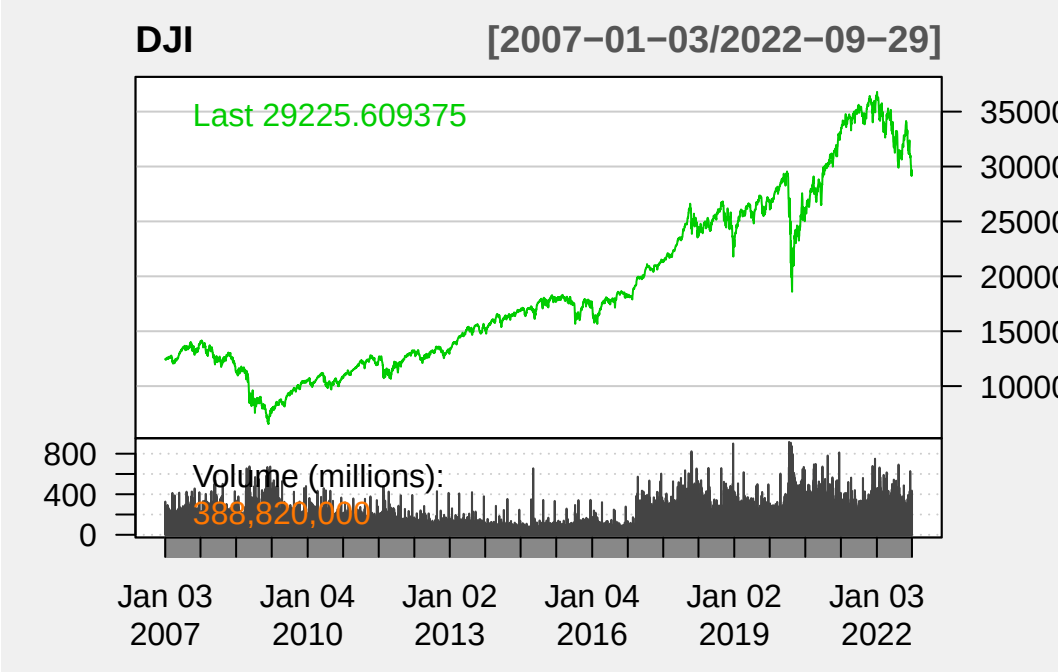

```
chartSeries(DJI, type = c("auto", "candlesticks", "matchsticks", "bars","line"), subset='l
```



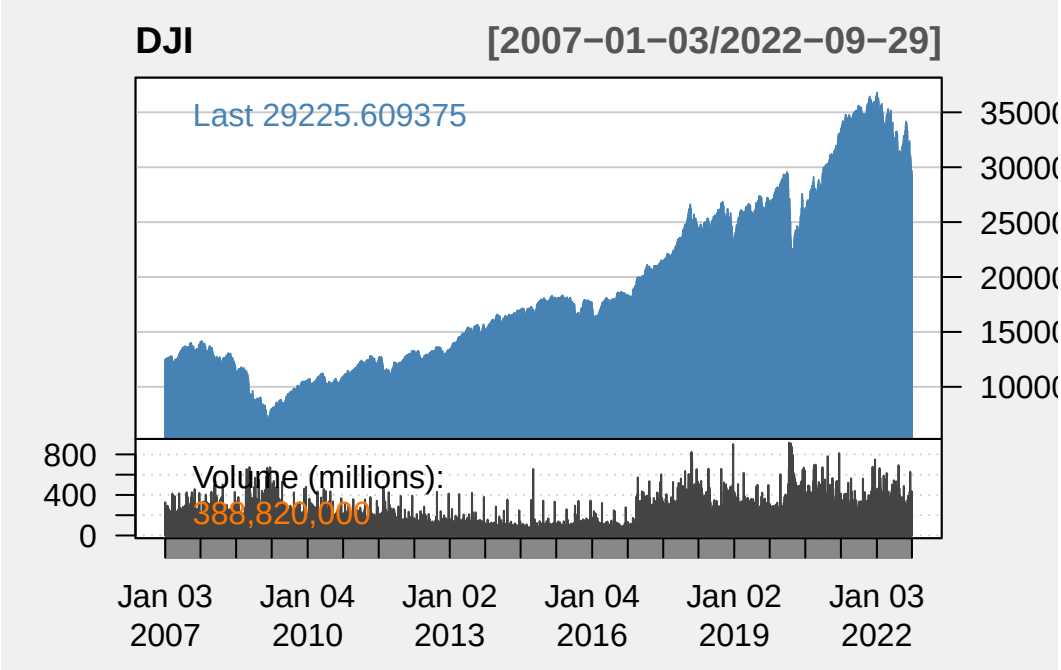
```
barChart(DJI,multi.col=TRUE,theme = 'white')
```



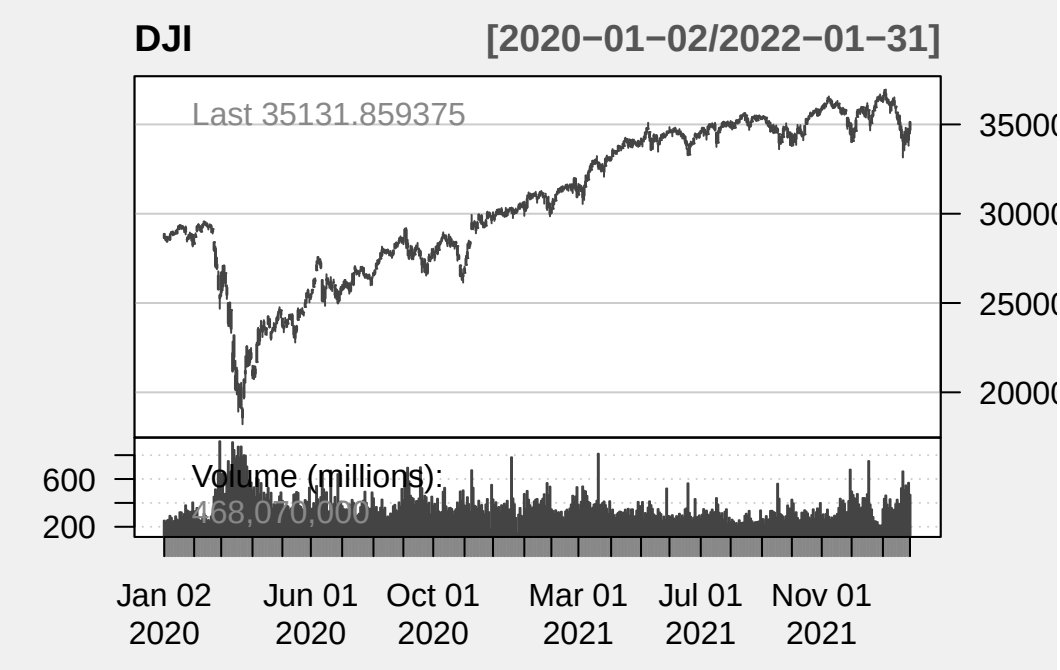

```
lineChart(DJI,line.type = 'l', theme = 'white') # line, choices include l, h, c, b
```




```
lineChart(DJI,line.type = 'h',theme = chartTheme('white', up.col='steelblue')) # histogram
```



```
candleChart(DJI,subset = '2020-01/2022-01', multi.col=TRUE,theme = chartTheme('white'))
```




```
## grey => Open[t] < Close[t] and Op[t] < Cl[t-1]
## white => Op[t] < Cl[t] and Op[t] > Cl[t-1]
## red => Op[t] > Cl[t] and Op[t] < Cl[t-1]
## black => Op[t] > Cl[t] and Op[t] > Cl[t-1]
```

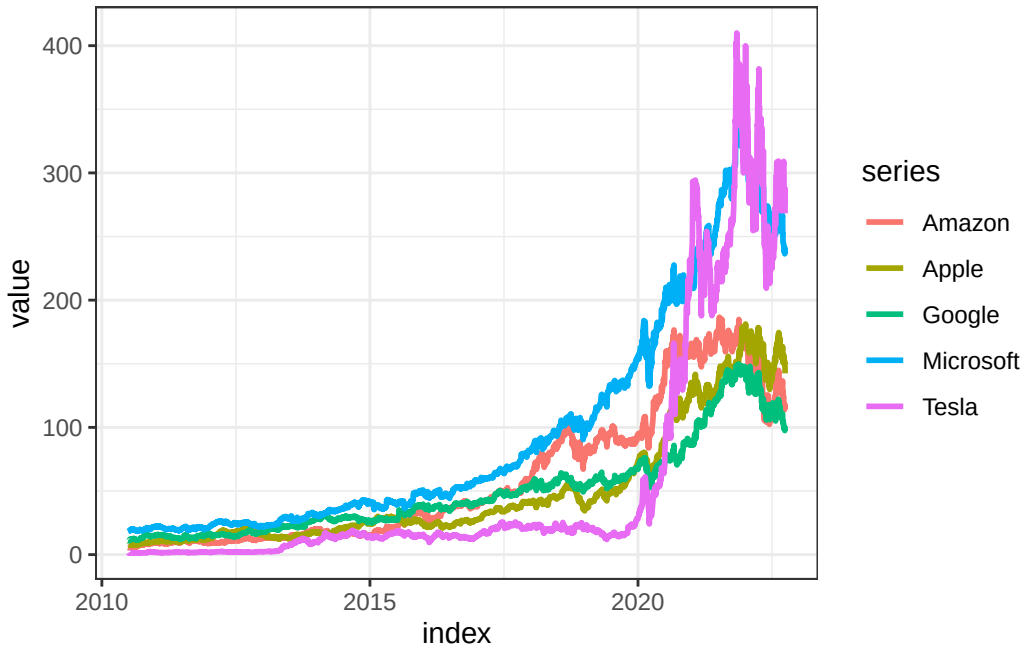
```
## Plotting multiple series using ggplot2
# Collect stock names from Yahoo Finance
getSymbols(c("AAPL", "MSFT", "AMZN", "TSLA", "GOOGL"), src = "yahoo", from = start, to = e
```

```
[1] "AAPL"  "MSFT"  "AMZN"  "TSLA"  "GOOGL"
```

```
# Prepare data as xts (time series object)
stocks = as.xts(data.frame(AAPL = AAPL[, "AAPL.Adjusted"],
                           MSFT = MSFT[, "MSFT.Adjusted"],
                           AMZN = AMZN[, "AMZN.Adjusted"],
                           GOOGL = GOOGL[, "GOOGL.Adjusted"],
                           TSLA = TSLA[, "TSLA.Adjusted"]))
```

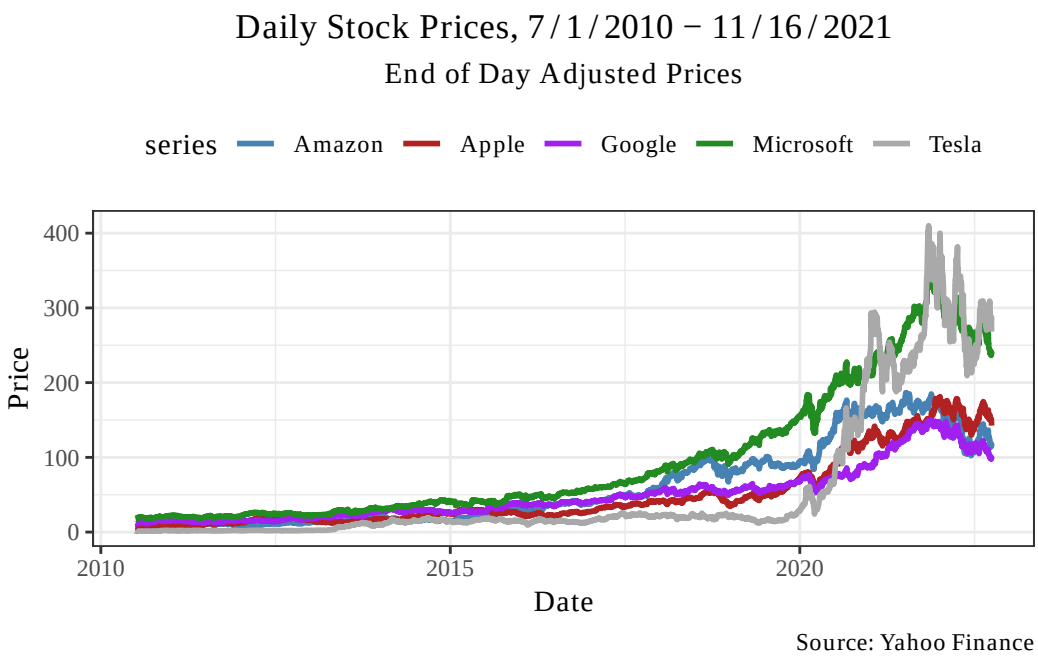
```r
# Index by date
names(stocks) = c("Apple", "Microsoft", "Amazon", "Google", "Tesla")
index(stocks) = as.Date(index(stocks))

# Plot
stocks_series = tidy(stocks) %>%
  ggplot(aes(x=index,y=value, color=series)) +
  geom_line(cex=1) +
  theme_bw()
stocks_series
```




```r
# Plot TWII
TWII_series = tidy(TWII$TWII.Adjusted) %>%
  ggplot(aes(x=index,y=value, color=series)) +
  geom_line(cex=1) +
  theme_bw()
TWII_series
```


```
stocks_series = tidy(stocks) %>%
  ggplot(aes(x=index,y=value, color=series)) +
  geom_line(cex=1) +
  theme_bw() +
  labs(title = "Daily Stock Prices, 7/1/2010 - 11/16/2021",
     subtitle = "End of Day Adjusted Prices",
     caption = "Source: Yahoo Finance") +
  xlab("Date") + ylab("Price") +
  scale_color_manual(values = c("steelblue", "firebrick","purple", "forestgreen","darkgray
  theme(text = element_text(family = "Palatino"), plot.title = element_text(hjust = 0.5),
  theme(legend.position="top")
stocks_series
```




```
# Fetching stock data using Googlesheet
library(googlesheets4)
library(httpuv)

# Read stock data using Googlesheet with public link
# Needs authentication of Google account via browser (when prompted)
# aapl = read_sheet("https://docs.google.com/spreadsheets/d/1vTdXZkqpIZwUsnxM8zXiXVyKI7BbZ
```

5.0.4.2 COVID

Many organizations put in tremendous efforts to collect, visualize and disseminate COVID data to facilitate research of the pandemic. The most well-known is the Johns-Hopkins Coronavirus Resource Center. Collecting data from all over the world data, these data are stored in the Center's GitHub ready for any researchers to get access. Other widely used data GitHub's includng New York Times and Our World in Data (OWID)

The following illustration uses the OWID data:

```
# Illustration: collecting open COVID data from OWID GitHub
## Packages used: vroom, tidyverse, finalfit
## Use install.packages() function to install these packages.
```

```r
library(vroom) # Fast reading in data
library(finalfit) #for checking missingness and output visualization
library(tidyverse)
library(RColorBrewer) #for choice of more colors
library(scales) # for controlling scales for x and y axes


# Reading all real time data directly from OWID GitHub
# vroom is the champion in reading github date, < 3 sec.
owidall = vroom("https://github.com/owid/covid-19-data/blob/master/public/data/owid-covid-
```

```r
as.factor(owidall[,1:3]) # Redefining first 3 columns as factor (categorical variables)
```

```
 iso_code continent  location
     <NA>      <NA>      <NA>
3 Levels: c("AFG", "OWID_AFR", "ALB", "DZA", "AND", "AGO", "AIA", "ATG", "ARG", "ARM", "ABW"
```

```r
# Subset by year

owid2022 = subset(owidall, format(as.Date(date),"%Y")==2022)
owid2021 = subset(owidall, format(as.Date(date),"%Y")==2021)
owid2020 = subset(vroom("https://github.com/owid/covid-19-data/blob/master/public/data/owi

# Clean up OWID*  cases
# Deselect cases/rows with OWID
owidall = owidall[!grepl("^OWID", owidall$iso_code), ]
owid2022 = owid2022[!grepl("^OWID", owid2022$iso_code), ]
owid2021 = owid2021[!grepl("^OWID", owid2021$iso_code), ]
owid2020 = owid2020[!grepl("^OWID", owid2020$iso_code), ]
owidall$location=as.factor(owidall$location)

# Subset by country: United States
owidus = subset(owidall, location=="United States")
owideu = subset(owidall, continent=="Europe")
owidasia = subset(owidall, continent=="Asia")
```

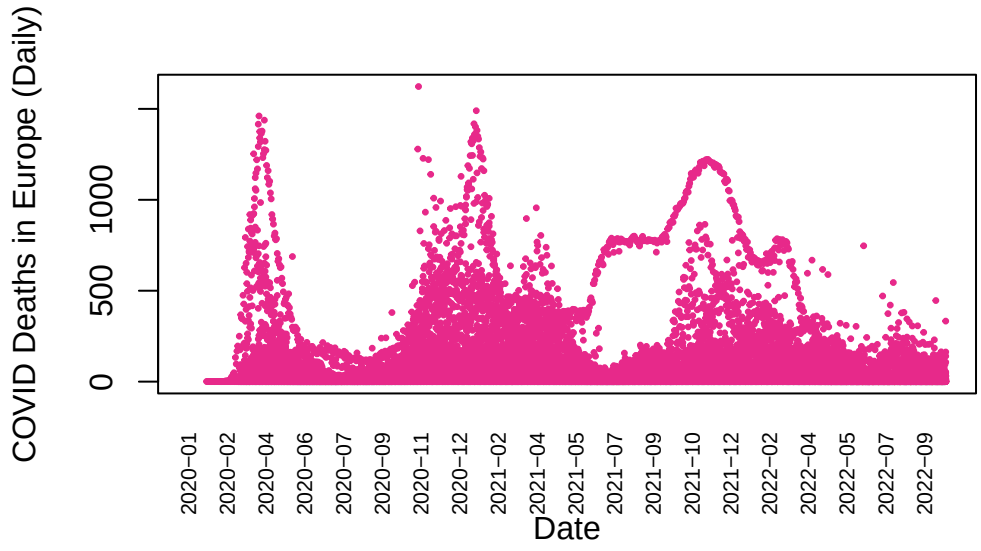
```r
# Get today's COVID data
owidtoday = subset(vroom("https://github.com/owid/covid-19-data/blob/master/public/data/ow

owid09302022 = subset(owidall,  date == "2022-09-30")
```

```r
owidUStoday = subset(owid2022, location == "United States" & date >="2022-09-30")
owidtw2022 = subset(owid2022, location == "Taiwan")
```

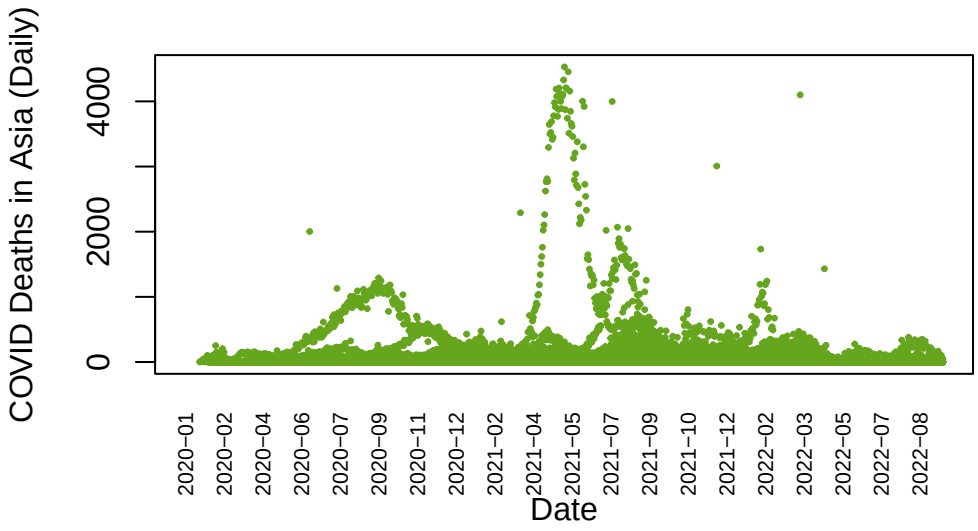
```r
options(scipen=999)
par(family = "Palatino")
```

```r
# Europe data
y = owideu$new_deaths
x = as.Date(owideu$date)
plot(x,y, pch=20, col="#E7298A", cex = .5, xaxt='n', xlab = "Date", ylab = "COVID Deaths i
axis(1, x, format(x, "%Y-%m"), cex.axis = .7, las = 3 , gap.axis =1.5, tick = FALSE)
```




```r
# identify(x,y,owideu$location, ps=8, atpen=TRUE)
# Use identify function to manually click on cases using mouse
```

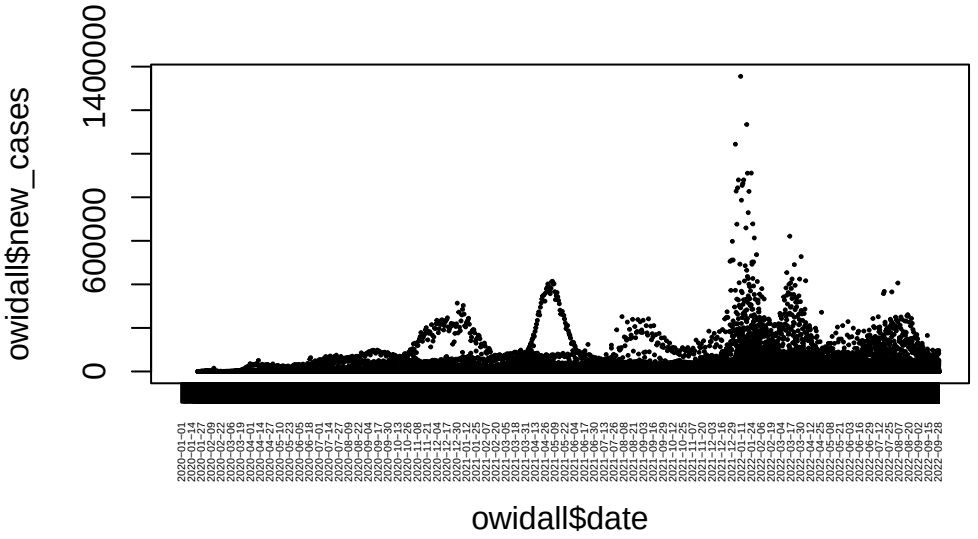
```r
# Asia data
y = owidasia$new_deaths
x = as.Date(owidasia$date)
plot(x,y, pch=20, col="#66A61E",  cex = .5, xaxt='n', xlab = "Date", ylab = "COVID Deaths
axis(1, x, format(x, "%Y-%m"), cex.axis = .7, las = 3 , gap.axis =1.5, tick = FALSE)
```



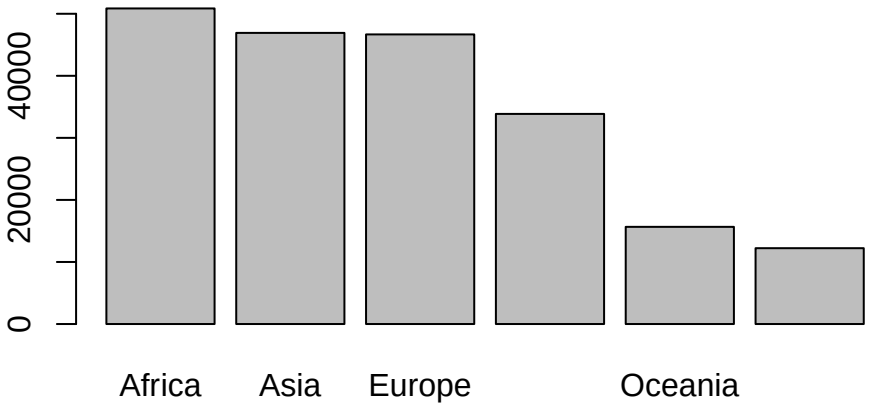

```
# identify(x,y,owidasia$location, ps=8, atpen=TRUE)

# Format date
owidall$date<-as.Date(owidall$date,format="%Y-%m-%d")
# World: new cases

plot(owidall$date,owidall$new_cases, pch = 20, cex = .3, xaxt='n')
points(owidtw2022$date,owidtw2022$total_tests, pch = 20, cex = .5, xaxt='n', col = "firebr
axis(1, owidall$date, format(owidall$date, "%Y-%m-%d"), cex.axis = .3, las = 3 )
```



```r
library(descr)
freq(owidall$continent)
```




```
owidall$continent
               Frequency Percent
Africa             50858  24.665
Asia               46919  22.754
Europe             46675  22.636
North America      33864  16.423
Oceania            15661   7.595
South America      12221   5.927
Total             206198 100.000
```

```r
options(scipen=999) # No sci notation
# format(new_cases, scientific = F)
```

Part IV

Data Management

6 Data Management with R

This chapter focuses on the "Data" part in Data programming. In other words, we will cover methods of managing data not just locally but on cloud. It will cover the use of *sparklyr* to interact with Spark. Some basic concepts of relational database, database management systems and interfacing with database servers will be introduced.

Part V

Data Visualization

7 Data Visualization with R

7.1 What is Data Visualization?

Data visualization is to deliver a message from your data. It is like telling a story using the chart or data applications. Sometimes the data is huge or the story to too long to tell. Visualization provides an ability to comprehend huge amounts of data. The important information from more than a million measurements is immediately available.

Visualization often enables problems with the data to become immediately apparent. A visualization commonly reveals things not only about the data itself but also about the way it is collected. With an appropriate visualization, errors and artifacts in the data often jump out at you. For this reason, visualizations can be invaluable in quality control.

Visualization facilitates understanding of both large-scale and small-scale features of the data. It can be especially valuable in allowing the perception of patterns linking local features.

Visualization facilitates hypothesis formation, inviting further inquiries into building a theory (Colin Ware 2012). It is exploratory data anlaysis (EDA) but can also provide the tools for hypothesis confirmation.

7.2 Learn to read data

Edward Tufte is one of the earliest data scientists emphasizing visual thinking. He postulates that one should first learn to read data, before moving on to visualize. He suggests training the visual thinking, then preparing the educated eyes. His newest book is titled SEEING WITH FRESH EYES: MEANING, SPACE, DATA, TRUTH, vividly testifying his philosophy of connecting the human perception with the data message.

For Tufte, number one thing to learn about data visualization is to discard the default.

"If you're not doing something different, you're not doing anything at all." - Edward Tufte

Figure 7.1: **Edward Tufte**

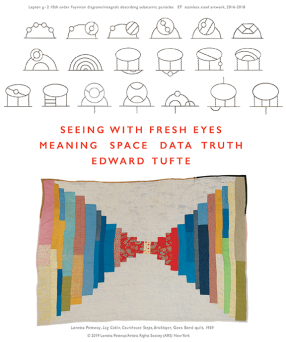


Figure 7.2: **Seeing with Fresh Eyes: Meaning, Space, Data, Truth**

7.3 References:

Graham Williams 2011. *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge*

Part VI

Data Modeling

8 Data Modeling with R

This chapter introduces methods of modeling data using R. Given the breadth of this topic, we will cover some basic machine learning models including:

- unsupervised learning
- supervised learning:
  - Classification
  - Regression
  - Tree-based models
- automated machine learning
  - H2O

9 What's next

This book will continue to incorporate new materials including new data science topics and developments.