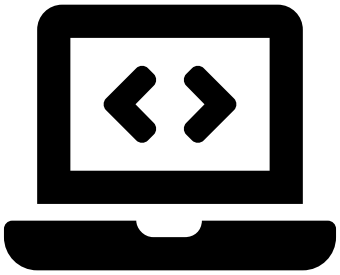
Financial modeling in R.



Dr. Martín Lozano.

<martin.lozano@udem.edu>

https://sites.google.com/site/mlozanoqf/

https://github.com/mlozanoqf/

julio 05, 2021. 22:43:34

Abstract

This is an introductory applied tutorial suitable for undergraduate students interested in the area of quantitative finance modeling. The main objective is to show how to analyze financial data and extract basic insights from financial models using the powerful R programming language. We extend examples and codes freely available in Internet, and some of my own. As in the philosophy of Donald Knuth [Knuth, 1984], the objective of this document is to explain to human beings what we want a computer to do as literate programming. This is a work in progress and it is under revision.

Index.

1 Introduction.

The main objective of this document is to show how to extract, visualize and analyze financial data in the context of asset pricing models, asset allocation models, a few financial econometrics techniques, and review some the most cutting edge technologies applications such as blockchain, using the powerful R programming language.

Here, we are interested in explaining concepts and real-life applications by developing examples that start from the data extraction, the problem statement, the proposed solution and the evaluation of the solution. We start by explaining what finance is about and why we require to incorporate a programming language as a way to conduct financial analysis. Any quantitative analysis requires gathering and manipulating data at some initial step, so we continue our journey by showing how to extract a wide variety of financial and economic information making special emphasis on stock market information and in data visualization. Then, we conduct some basic financial analysis based on the firm's financial statements and time series of firm's stock prices. Then, we move from analyzing prices to analyzing asset returns and we introduce the concept of financial risk. With these foundations, then we move into some asset pricing applications to understand what drives asset returns, we discuss the role of risk factors and how asset pricing can help us to make better financing and investment decisions. Given the relevance of future asset prices changes we also incorporate some foundations of asset prices forecast. Then, we introduce the portfolio analysis by showing how an investor could take informed decisions to optimize the performance of his or her investment portfolio, how to reduce risk by implementing diversification tools and financial algorithms, as well as evaluate investment strategies with the help of R. Finally, we review the concept of blockchain and illustrate how it works. For starters, this technology uses data elements encrypted in blocks of computer code. The blocks are chained together across a shared ledger through cryptology. If someone tries to hack the ledger, it is immediately known by the involved parties and the chain falls apart. Blockchain has the potential to reshape processes that are defined inside finance, primarily because of its cost and control benefits.

1.1 Finance.

According to [OECD, 2020], developed and emerging countries and economies have become increasingly concerned about the level of financial literacy (ability to understand and properly apply financial management skills) of their citizens, including young people. This initially stemmed from concern about the potential impact of shrinking public and private welfare

systems, shifting demographics, including the ageing of the population in many countries, and the increased sophistication and expansion of financial services. We all face financial decisions and we demand and offer financial services in this evolving context. As a result, financial literacy is now globally recognized as an essential life skill.

We all have different interests and incentives to learn finance. For me, it is a way to better understand the world we are living in. In finance we mainly study financing and investment decisions under uncertainty. Financing decisions are about looking for funds whereas investment decisions are about assigning funds to run a project. Thus, financing and investment are basically two sides of the same coin called project, business idea, firm, financial asset, countries, etc. A business project is a series of inputs, outputs, investment plans and tasks that need to be completed in order to reach a specific expected outcome in the future. Projects are uncertain by nature because many things can go wrong in the future: a firm might go bankrupt, a business idea can be stolen, sales projections might not be as good as expected, plans could change because of coronavirus pandemic, and so on. This is why we say that financing and investment decisions are taken under uncertainty. Finance decisions are taken today, but their results are seen or realized in the uncertain future. Risky projects and uncertain projects are not necessarily bad projects as uncertainty and risk boost innovation, demands a high-quality quantitative analysis, represent opportunities for entrepreneurs, and returns for investors.

An individual takes financing decisions in the job market, trying to get the maximum salary in the most convenient job to get the funds to pay for food, housing and leisure. This individual takes an investment decision when she decides to use her savings to start a small firm. If this firm performs well in the first year, then the firm will apply for a bank loan to finance a business expansion. Once the firm gets the new funds, the firm will have to invest this money wisely in productive assets, technology and hire experts in the relevant business field. As the firm generates profits, the owner who initially decided to risk her money will get returns. These returns may attract other investors willing to help the manager to export to other countries in exchange for some participation in the firm's profits. When we replicate this in the economy many times, it is easy to understand how good finance and investment decisions contribute to the economic growth.

Finance can be applied by individuals and firms as illustrated above. Governments get funds from taxpayers to invest in public assets such as education, health, public infrastructure, etc. Governments also can contribute to create certainty about the stability of economic indicators and maintain a rule of law to stimulate private investment. If public spending remains higher than income, the country might fall into a public deficit and this could lead

to financial instability which dampers not only public finances but personal and private investment decisions as well.

Most finance decisions are taken based on a risk-return analysis. In particular, if after doing the corresponding maths and consulting with the pillow you conclude that the expected return is higher than the associated risk, then you will most likely go for it. On the other hand, if the risk looks quite high compared with the expected return, you will surely rethink or abandon the project. We, as individuals, perceive the risk and the return differently. Consider the following real example. For each of the past 17 years, the All-England Lawn Tennis Club has paid for an insurance policy to guard against losses if Wimbledon should have to be cancelled in the event of a worldwide pandemic. This was considered for some as an excessive cost until recently. Wimbledon received about £114 million because 2020 tournament was cancelled due to the coronavirus. A similar thing happens when you buy a used car. Surely the buyer considers the car cheap enough, and the seller considers the car price expensive enough to close the deal. So, it is good to have different perspectives about prices, risk and returns as this allows commercial and financial transactions to exist. We also perceive risk and return differently as we use different methods, data and procedures to estimate risk and return.

Although we all perceive risk and return differently by nature, we may also perceive the risk and return wrong by lack of knowledge. This is not uncommon as we may apply political decisions to finance problems, or make financial decisions without the relevant knowledge in the field, or ignoring the power of data analysis. Underestimating risks and overestimating returns may be as harmful as overestimating risk and underestimating returns. The first could lead to an excess of risk and the latter could lead to forego a good business. We are not suggesting everybody should become a finance expert, but finance professionals are expected to contribute to make better and correct financial decisions most of the times.

Finance is not a pure exact area of knowledge, it borrows some principles of physics and mathematics to develop financial models.[1] For instance, we are sure that at standard atmospheric pressure, water boils at approximately 100 degrees Celsius. But we do not know

[1]The Black-Scholes formula for pricing European call and put options is one of the most famous equations in financial mathematics. See Scholes [1973] and Merton [1973]. This equation is so important that Robert Merton and Myron Scholes received the 1997 Nobel Price for Economics in honour of their work. Unfortunately, Fischer Black, who clearly contributed extensive work to the formula, passed away in 1995. Interestingly, the Black-Scholes formula is basically a partial differential equation (PDE) well known in physics as the "heat equation" which describes the distribution of heat in a given region over time. Moreover, there are close parallels between random movements of particles in a fluid (called physical Brownian motion) and price fluctuations in financial markets (known as financial Brownian motion). Thus, finance seems to follow not only human behaviour but also some physics principles.

for sure whether my business profits will grow at 10% next year. Returns are uncertain, this is why we call them expected returns. In fact, after doing some financial analysis, I can estimate that my profits will grow in the range of 5% to 15%, this range shows how uncertain my profits are. This is why returns and risk are two main pillars in finance. The ability of understanding the economic conditions, the market, and the firm will determine the success of financing and investment decisions. This means that finance requires some knowledge of economics, statistics, math, accounting, probability, marketing, psychology, and data science to transform data into intelligent decisions.

As an economist, I consider finance an area within economics. The Journal of Economic Literature (JEL) classification system is used to classify articles, dissertations, books, book reviews, and working papers in EconLit, and in many other applications. The JEL classify finance as "financial economics" and includes

- G00. Financial Crises.
- G1. General Financial Markets, Portfolio Choice, Investment Decisions, Asset Pricing, Trading Volume, Bond Interest Rates, Contingent Pricing, Futures Pricing, Information and Market Efficiency, Event Studies, Insider Trading, International Financial Markets, Financial Forecasting and Simulation, Government Policy and Regulation.
- G2. Financial Institutions and Services, Banks, Depository Institutions, Micro Finance Institutions, Mortgages, Insurance, Insurance Companies, Actuarial Studies, Non-bank Financial Institutions, Financial Instruments, Institutional Investors, Investment Banking, Venture Capital, Brokerage, Ratings and Ratings Agencies, Government Policy and Regulation.
- G3. Corporate Finance and Governance, Capital Budgeting, Fixed Investment and Inventory Studies, Capacity, Financing Policy, Financial Risk and Risk Management, Capital and Ownership Structure, Value of Firms, Goodwill, Bankruptcy, Liquidation, Mergers, Acquisitions, Restructuring, Corporate Governance, Payout Policy, Government Policy and Regulation.
- G4. Behavioral Finance, Role and Effects of Psychological, Emotional, Social, and Cognitive Factors on Decision Making in Financial Markets.
- G5. Household Finance, Household Saving, Borrowing, Debt, and Wealth, Insurance, Financial Literacy.

In this tutorial we focus on only two areas of finance: asset pricing and portfolio analysis. Asset pricing allows us to understand the risk-return relationship of financial instruments and portfolio analysis allows us to take optimal investment decisions in the financial markets.

1.2 Computers.

Every area of knowledge requires computers to conduct interesting analysis and applications. Traditionally, people use good commercials (and unfortunately expensive) software such as Microsoft Excel, SPSS, STATA, E-Views, and many others. These commercial software are good. However, you have to be aware that these programs are fully controlled by private firms who genuinely seek to create value for their shareholders, so there is no guarantee that their associated file formats could be readable in the future, or even exist in the future, which negatively impacts reproducibility. I never advise not to learn commercial software like the ones listed above; but I always encourage learning and use R (or Python) for serious data analysis. These computer languages are user-oriented and are created and constantly improved by a growing scientific community with an immense online presence to assist users.

Commercial software products as the ones listed above are definitely important in the job market, but you also have to realize that the main interaction with these programs is by using the mouse to click on pre-defined, limited and inflexible menus. This kind of user-interaction is most of the times ephemeral and unrecorded, so that many of the choices made during a full quantitative procedure are frequently undocumented and this turns out to be highly problematic because there is no trace about how an analysis was conducted, and also because it becomes hard to propose an extension to the analysis in phases or replication in different contexts. Coding allows us to conduct and develop reproducible research. Learning how to code is equivalent as writing a cooking recipe and every time you click run you get the dish done. Although, chefs have to pay for ovens, kitchen items and even ingredients, while in finance most of our inputs are free data and the technology is also free as R is an open source software.

Other commercial products in which you do not have to code like Microsoft Excel, SPSS, STATA, E-Views and many others, have high licensing fees and also rely on mysterious black boxes to produce a battery of results. These black boxes are problematic because the data comes in and the result comes out as magic, showing no details about the procedure followed to produce the final results, and the user could sadly get the wrong illusion that he or she understands data analysis. This might be convenient in some specific and limited cases but in others you miss the fun that represents having access to all the details of the computation and limit the extent to which you can customize or extend to innovative and create new improved applications. The general alternative to using a point-and-click program is to familiarize with languages like R which allows writing scripts to program algorithms for economic and financial analysis and visualizations.

1.3 R and RStudio.

R is a language and environment for statistical computing and graphics. R is a powerful integrated suite of software facilities for data manipulation, calculation and graphical display. R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS. Given its popularity and flexibility, R is currently implemented in virtually all areas of knowledge including finance by students, practitioners, researchers, universities, institutions, firms, think tanks, and policy makers around the world.

Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. This is why R is a popular choice for finance and economic modeling. R can be extended (easily) via packages as we will show in this tutorial. There are about eight default packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics, data science and finance applications.

Let's see how many packages are there as today using R code.

```
# The function is available.packages and we store the result
# in R_packages variable.
R_packages <- available.packages(filters = "duplicates",
                                 repos="https://cran.rstudio.com")
# Now, we combine paste and print functions to produce a sentence.
# Note that nrow simply counts the number of rows in R_packages.
print(paste("There are", nrow(R_packages), "R packages available in CRAN as of",
            Sys.Date()))
```

```
## [1] "There are 17816 R packages available in CRAN as of 2021-07-05"
```

Every R package has its own PDF online documentation and there are many online examples developed by users as well. My recommendation here in case you have a question about this is Google it. Many times, we do not know how to deal with an error message, and we can find our way out by Google it. I have been using R, Matlab, Python, Octave and other languages for the last 15 years and I can confirm every time I do not know how to code something in R, I can easily find a solution online either in discussion forums, documentation, or in YouTube videos.

R offers numerous advantages for data analysis compared with other alternatives like Mi-

crosoft Excel. R is free; it is easy to do reproducible research (self-documenting, repeatable); it is scalable (applicable to small or large problems); there is a big and growing R online community by discipline and by region (including R-Ladies groups); Stack Overflow; plenty of learning resources (quantity and quality); many R books and resources (see the reference list at the end of this tutorial). Finally, R is 'becoming' the new norm in data science and specifically in finance analysis. Even if you are interested in other languages like Python, at the end learning one language can help you to understand others. Microsoft Excel is a great tool and we are expected to learn and use it very well, but it is not the best alternative for data analysis.

While some people find the use of a commandline environment for coding daunting, it is becoming a necessary skill for managers, management analysts, and data scientists as the volume and variety of data has grown. Thus, scripting or programming has become a third language for modern professionals, in addition to their native language, and discipline specific terminology.

Below is the way we show R code and R output throughout this tutorial.

```
# We write comments in italics. The R code looks like this:
print("hello world")
```

```
## [1] "hello world"
```

The output of the R code above is a string text: "hello world". The R function `print` prints in the screen the quoted string in parenthesis. We will use a number of R functions to illustrate finance applications and examples. Nobody expects you to memorize each function and its syntax as you can always access R documentation to help you out with R functions syntax and even examples. This is done by typing `?print` in the console. You will see a right panel with the corresponding help. This is part of the help documentation when you type `?print`:

Print Values. Description. `print` prints its argument and returns it invisibly (via invisible(x)). It is a generic function which means that new printing methods can be easily added for new classes. Usage. `print(x, ...)`

Now, a simple numerical example to show R code and R output.

```
# Define the value of a.
a = 2 + 2
# Print the value of a.
print(a)
```

```
## [1] 4
```

```
# Or simply
a
```

```
## [1] 4
```

So, `a` is equal to 4. Of course, R is more than a Fisher-Price calculator. But it is always useful to see simple examples first.

In the following sections, we show and explain examples that the participants can replicate for their own purposes and interests. In particular, we expect the participants to "copy and paste" the structure of the code to replicate some analysis on their own. For example, imagine you are interested in producing a variable `b` that contains the result of $3+3$. Given the previous example, you know you can do this by typing $b = 3 + 3$. A variable `c` which is the product of `a` and `b` is then $c = a \times b$. You will soon realize that there are many ways in which we can define `c`. An alternative is $c = (2+2) * b$, or $c = a * (3+3)$, $c = (2+2) * (3+3)$, and even $c = 4 * 6$. This means that there are many equivalent ways to do one task.

2 Financial data.

Data is everywhere and finance has always been about data.[2] Industries perceive data as an essential commodity and fuel to take decisions under uncertainty. As a matter of fact, data science and finance go hand in hand. Even before the term data science was coined, finance was using it. Data Science is widely used in areas like risk analytics, credit risk, fraud detection, risk management, pricing, and algorithmic trading. Financial institutions were among the earliest users and pioneers of data analytics. We need data to perform financial analysis, estimate financial models, forecast financial variables, take financing and investment decisions, and more. The alternative of data is to listen to your guts, your intuition, or experience. This alternative is not bad by itself, but it would not be our main approach here.

Financial data is usually free and available in electronic sources for downloading. There are exemptions as private firms are not obligated to make their financial information public. Also, there are some kind of financial information that public firms are not obligated to make public given the international regulations. Sometimes, financial data is not fully available but we have summary statistics that can be quite useful to simulate the data by ourselves. Financial markets are a great source of financial information like exchange rates, commodities, financial instruments, etc. Public institutions and international organizations are useful sources of financial information as well. In any case, we have more financial data than time and resources to analyze it, and the gap is getting bigger and bigger.

2.1 Financial statements.

Financial data comes in a wide variety of formats, and is reported and stored in numerous electronic sources. Financial statements represent a great source to analyze and understand the financial health of a firm, individuals and even governments. Financial statements are standard and well-organized reports that summarize important financial accounting information of a business. The traditional approach to analyze financial statements is looking at those in the firm's annual reports, or downloading them in PDF or Excel format in the firm's investor website site.

In R, we can download public firm's financial statements from the US Securities and Ex-

[2] According to market intelligence company IDC, the 'Global Datasphere' in 2018 reached 18 zettabytes. The vast majority of the world's data has been created in the last few years and this astonishing growth of data shows no sign of slowing down. In fact, IDC predicts the world's data will grow to 175 zettabytes in 2025. One zettabyte is 100000000000000000000 bytes. In scientific notation this is 1e+21 or 1 with 21 zeros at the right, we call it one sextillion.

change Commission using the *finreportr* package (there are others like *finstr*). This is useful to evaluate a firm performance and conduct financial analysis as we do not need to visit the firm site. In fact, we could replicate or extend the full analysis by running the code instead of downloading newer financial statements.

In case you are not familiar with R packages, you should know you need to download the packages you need before using them. This is an easy process in RStudio. First, select the "Tools" menu, and then "Install Packages..." option. Now, type the name of the package you need, in this case type *finreportr* for accessing financial statements (later we will need *tidyquant* as well and others), leave the default options as they are, and click "Install". You will see a typical bar progress of this download process and R will let you know when this is done. You only need to download packages once, but you have to load the package (using the function *library*) every time you will use the package in your script. There are plenty of YouTube tutorials showing this installation process step by step, for example https://youtu.be/JBcVi-fAT_k

If you are interested in working with financial statements, you should look for the *finreportr* package documentation online or users' examples available in Internet. Perhaps the main drawback of this package is that it may be slow, you normally need to wait for R to download the financial reports. Let's start with some examples here. First, we load the previously downloaded packages.

```
# Load the finreportr package to download financial reports.
library(finreportr)
# Load the tidyquant package to manipulate the data and more.
library(tidyquant)
library(dplyr)
library(tidyr)
```

Let's analyze Facebook.

```
FB.Info <- CompanyInfo("FB")
print(FB.Info)
```

```
##        company        CIK  SIC state state.inc FY.end   street.address
## 1 Facebook Inc 0001326801 7370    CA        DE   1231 1601 WILLOW ROAD
##            city.state
## 1 MENLO PARK CA 94025
```

This is the physical address of the firm. Now some information about Facebook's annual

reports.

```
FB.reports <- AnnualReports("FB")
FB.reports$accession.no
```

```
##  [1] "0001326801-21-000014" "0001326801-20-000013" "0001326801-19-000009"
##  [4] "0001326801-18-000009" "0001326801-17-000007" "0001326801-16-000063"
##  [7] "0001326801-16-000043" "0001326801-15-000010" "0001326801-15-000006"
## [10] "0001326801-14-000007" "0001326801-13-000003"
```

These are keys to name specific financial reports. You can Google one of those keys above and see the kind of information available at SEC website. Now, let's download a balance sheet.

```
# Store the balance sheet of Facebook (FB) in BS_FB.
BS_FB <- GetBalanceSheet("FB", 2018)
# Show the value in the screen (in paper).
BS_FB
```

```
##                                                  Metric Units      Amount
## 1              Cash and Cash Equivalents, at Carrying Value   usd  4315000000
## 2              Cash and Cash Equivalents, at Carrying Value   usd  4907000000
## 3              Cash and Cash Equivalents, at Carrying Value   usd  8903000000
## 4              Cash and Cash Equivalents, at Carrying Value   usd  8079000000
## 5                           Marketable Securities, Current   usd 20546000000
## 6                           Marketable Securities, Current   usd 33632000000
## 7                     Accounts Receivable, Net, Current   usd  3993000000
## 8                     Accounts Receivable, Net, Current   usd  5832000000
## 9       Prepaid Expense and Other Assets, Current   usd   959000000
## 10      Prepaid Expense and Other Assets, Current   usd  1020000000
## 11                                  Assets, Current   usd 34401000000
## 12                                  Assets, Current   usd 48563000000
## 13            Property, Plant and Equipment, Net   usd  8591000000
## 14            Property, Plant and Equipment, Net   usd 13721000000
## 15   Intangible Assets, Net (Excluding Goodwill)   usd  2535000000
## 16   Intangible Assets, Net (Excluding Goodwill)   usd  1884000000
## 17                                       Goodwill   usd 18026000000
## 18                                       Goodwill   usd 18122000000
## 19                                       Goodwill   usd 18221000000
```

```
## 20                                    Other Assets, Noncurrent   usd  1312000000
## 21                                    Other Assets, Noncurrent   usd  2135000000
## 22                                                      Assets   usd 64961000000
## 23                                                      Assets   usd 84524000000
## 24                                  Accounts Payable, Current   usd   302000000
## 25                                  Accounts Payable, Current   usd   380000000
## 26                             Accounts Payable, Other, Current   usd   280000000
## 27                             Accounts Payable, Other, Current   usd   390000000
## 28                                 Accrued Liabilities, Current   usd  2203000000
## 29                                 Accrued Liabilities, Current   usd  2892000000
## 30                       Deferred Revenue and Credits, Current   usd    90000000
## 31                       Deferred Revenue and Credits, Current   usd    98000000
## 32                                       Liabilities, Current   usd  2875000000
## 33                                       Liabilities, Current   usd  3760000000
## 34                               Other Liabilities, Noncurrent   usd  2892000000
## 35                               Other Liabilities, Noncurrent   usd  6417000000
## 36                                                 Liabilities   usd  5767000000
## 37                                                 Liabilities   usd 10177000000
## 38                               Commitments and Contingencies   usd        <NA>
## 39                               Commitments and Contingencies   usd        <NA>
## 40                                 Common Stock, Value, Issued   usd           0
## 41                                 Common Stock, Value, Issued   usd           0
## 42                                  Additional Paid in Capital   usd 38227000000
## 43                                  Additional Paid in Capital   usd 40584000000
## 44 Accumulated Other Comprehensive Income (Loss), Net of Tax   usd  -703000000
## 45 Accumulated Other Comprehensive Income (Loss), Net of Tax   usd  -227000000
## 46                     Retained Earnings (Accumulated Deficit)   usd 21670000000
## 47                     Retained Earnings (Accumulated Deficit)   usd 33990000000
## 48                Stockholders' Equity Attributable to Parent   usd 36096000000
## 49                Stockholders' Equity Attributable to Parent   usd 44218000000
## 50                Stockholders' Equity Attributable to Parent   usd 59194000000
## 51                Stockholders' Equity Attributable to Parent   usd 74347000000
## 52                                     Liabilities and Equity   usd 64961000000
## 53                                     Liabilities and Equity   usd 84524000000
##    startDate    endDate
## 1       <NA> 2014-12-31
```

```
## 2       <NA> 2015-12-31
## 3       <NA> 2016-12-31
## 4       <NA> 2017-12-31
## 5       <NA> 2016-12-31
## 6       <NA> 2017-12-31
## 7       <NA> 2016-12-31
## 8       <NA> 2017-12-31
## 9       <NA> 2016-12-31
## 10      <NA> 2017-12-31
## 11      <NA> 2016-12-31
## 12      <NA> 2017-12-31
## 13      <NA> 2016-12-31
## 14      <NA> 2017-12-31
## 15      <NA> 2016-12-31
## 16      <NA> 2017-12-31
## 17      <NA> 2015-12-31
## 18      <NA> 2016-12-31
## 19      <NA> 2017-12-31
## 20      <NA> 2016-12-31
## 21      <NA> 2017-12-31
## 22      <NA> 2016-12-31
## 23      <NA> 2017-12-31
## 24      <NA> 2016-12-31
## 25      <NA> 2017-12-31
## 26      <NA> 2016-12-31
## 27      <NA> 2017-12-31
## 28      <NA> 2016-12-31
## 29      <NA> 2017-12-31
## 30      <NA> 2016-12-31
## 31      <NA> 2017-12-31
## 32      <NA> 2016-12-31
## 33      <NA> 2017-12-31
## 34      <NA> 2016-12-31
## 35      <NA> 2017-12-31
## 36      <NA> 2016-12-31
## 37      <NA> 2017-12-31
```

```
## 38      <NA> 2016-12-31
## 39      <NA> 2017-12-31
## 40      <NA> 2016-12-31
## 41      <NA> 2017-12-31
## 42      <NA> 2016-12-31
## 43      <NA> 2017-12-31
## 44      <NA> 2016-12-31
## 45      <NA> 2017-12-31
## 46      <NA> 2016-12-31
## 47      <NA> 2017-12-31
## 48      <NA> 2014-12-31
## 49      <NA> 2015-12-31
## 50      <NA> 2016-12-31
## 51      <NA> 2017-12-31
## 52      <NA> 2016-12-31
## 53      <NA> 2017-12-31
```

We are done. We have the balance sheet information of Facebook in `BS_FB` in basically two lines of code without even visiting the firm's website. However, it does not look the same as we are used to seeing in a typical format like PDF or Excel. To be honest, it is difficult to read to say the least. R takes the information from the annual report, so if we ask for 2018, we retrieve information for the previous years.

Let's review how we can manipulate `BS_FB` contents to get a more readable result. Balance sheets are annual, so we do not need the *startDate* column, and we know it is expressed in US dollars, so we do not need the *Units* column as well. Imagine we are interested in a clear version of the 2017 balance sheet.

```
# First make a copy.
BS_2017 <- BS_FB |>
  # convert Amount from character to numeric.
  mutate(Amount = as.numeric(Amount)) |>
  # Let's extract only 2017.
  filter(endDate == "2017-12-31") |>
  # Let's show only two columns.
  select(Metric, Amount)
# Show the value in the screen (in paper).
BS_2017
```

```
##                                                    Metric      Amount
## 1                Cash and Cash Equivalents, at Carrying Value  8079000000
## 2                                  Marketable Securities, Current 33632000000
## 3                             Accounts Receivable, Net, Current  5832000000
## 4                   Prepaid Expense and Other Assets, Current  1020000000
## 5                                            Assets, Current 48563000000
## 6                           Property, Plant and Equipment, Net 13721000000
## 7                 Intangible Assets, Net (Excluding Goodwill)  1884000000
## 8                                                 Goodwill 18221000000
## 9                                   Other Assets, Noncurrent  2135000000
## 10                                                  Assets 84524000000
## 11                                Accounts Payable, Current   380000000
## 12                              Accounts Payable, Other, Current   390000000
## 13                                 Accrued Liabilities, Current  2892000000
## 14                       Deferred Revenue and Credits, Current    98000000
## 15                                       Liabilities, Current  3760000000
## 16                               Other Liabilities, Noncurrent  6417000000
## 17                                             Liabilities 10177000000
## 18                               Commitments and Contingencies          NA
## 19                                 Common Stock, Value, Issued           0
## 20                                  Additional Paid in Capital 40584000000
## 21 Accumulated Other Comprehensive Income (Loss), Net of Tax  -227000000
## 22                      Retained Earnings (Accumulated Deficit) 33990000000
## 23                Stockholders' Equity Attributable to Parent 74347000000
## 24                                     Liabilities and Equity 84524000000
```

This looks more like a familiar balance sheet.

This is the first time we introduce the use of pipes |> in the code. Pipes are a powerful tool for clearly expressing a sequence of multiple operations mainly in the context of *tidyverse* packages. In the code above we first create a variable for the 2017 balance sheet `BS_2017` as the same as the previously created variable `BS_FB`, then we transform one column to numeric values so we can make further operations, then we filter the whole information to keep only 2017 values, and finally we select only two columns (the name and value of each account). There might be even more efficient ways to achieve the same result. Note that every sequence is linked with a pipe |>. Pipes process a data-object using a sequence of operations by passing the result of one step as input for the next step.

A balance sheet is a financial statement that reports a company's assets, liabilities and shareholders' equity. The balance sheet is one of the three (income statement and statement of cash flows being the other two) core financial statements used to evaluate a business performance. The balance sheet allows us to verify the so-called accounting equation: $assets = liabilities + equity$. This is equivalent as saying that the firm has what the firm owes.

In plain English the balance sheet shows the value of what the firm has to produce (assets), and where did the money come from (banks in the form of liabilities and/or owners in the form of equity). The balance sheet is used to measure the firm size, the evolution of the firm's debt, and the participation of the owners to finance the firm. It is also useful to measure firm's liquidity and credit risk, among many other things.

Let's extract the 2016 and 2017 balance sheets and show the most relevant rows to verify how the accounting equation holds.

```
# Define the accounts we need in our balance sheet summary.
accounts <- c("Assets" , "Liabilities",
              "Stockholders' Equity Attributable to Parent",
              "Liabilities and Equity")
# At the beginning, the summary is a copy of the full statement.
BS_2016_2017_summary <- BS_FB |>
  # Convert Amount to numeric values (so we can do math operations).
  mutate(Amount = as.numeric(Amount)) |>
  # This is convenient but not strictly necessary.
  mutate(Metric = factor(Metric, levels = unique(Metric))) |>
  # Here we spread rows into columns to have information per year.
  spread(endDate, Amount) |>
  # Select only three columns.
  select(Metric, "2016-12-31", "2017-12-31") |>
  # Select only the relevant accounts we are interested in.
  filter(Metric %in% accounts) |>
  # Net income account is lengthy, let's rename it.
 mutate(Metric = recode_factor(Metric,
        `Stockholders' Equity Attributable to Parent` = "Equity"))
# Now, show the results.
BS_2016_2017_summary
```

```
##                   Metric 2016-12-31 2017-12-31
## 1                 Assets 6.4961e+10 8.4524e+10
## 2            Liabilities 5.7670e+09 1.0177e+10
## 3                 Equity 5.9194e+10 7.4347e+10
## 4 Liabilities and Equity 6.4961e+10 8.4524e+10
```

As expected, the accounting equation holds in both 2016 and 2017. We were not expecting a different result. The purpose of this example is to show how we can download and manipulate financial statements of firms. As shown above, the only thing we need is the ticker symbol of the firm and voilà. A ticker symbol or stock symbol is an abbreviation used to uniquely identify publicly traded firm shares. A stock symbol may consist of letters, numbers or a combination of both.

Looking at the USD values is difficult to have a gasp about which balance sheet account increased more or increased less in this period of time. This is, looking at absolute values (USD) makes it difficult to clearly see which account increased more than others, so we normally transform the data to relative terms (ratios, percentage changes, or proportions given a total).

Let's propose a different way to see the balance sheet statement.

```
# First we make a copy.
BS_2016_2017_change <- BS_2016_2017_summary |>
  # Calculate the percentage change from 2016 to 2017.
  mutate(Year.Change = (`2017-12-31`-`2016-12-31`)/`2016-12-31`)
# Show the results.
BS_2016_2017_change
```

```
##                   Metric 2016-12-31 2017-12-31 Year.Change
## 1                 Assets 6.4961e+10 8.4524e+10   0.3011499
## 2            Liabilities 5.7670e+09 1.0177e+10   0.7646957
## 3                 Equity 5.9194e+10 7.4347e+10   0.2559888
## 4 Liabilities and Equity 6.4961e+10 8.4524e+10   0.3011499
```

Liabilities increased more than twice compared with assets from 2016 to 2017. Should we be worried about Facebook financial health? One simple way to dig deeper into this issue is to show an alternative balance sheet showing the accounts as relative proportions with respect to the assets.

```
# Create a copy of the summary.
BS_2016_2017_prop <- BS_2016_2017_summary |>
  # Proportions with respect to the assets (2016).
  mutate(prop2016 = `2016-12-31`/`2016-12-31`[1]) |>
  # Proportions with respect to the assets (2017).
  mutate(prop2017 = `2017-12-31`/`2017-12-31`[1])
# Show the results.
BS_2016_2017_prop
```

```
##                   Metric 2016-12-31 2017-12-31   prop2016  prop2017
## 1                 Assets 6.4961e+10 8.4524e+10 1.00000000 1.0000000
## 2            Liabilities 5.7670e+09 1.0177e+10 0.08877634 0.1204037
## 3                 Equity 5.9194e+10 7.4347e+10 0.91122366 0.8795963
## 4 Liabilities and Equity 6.4961e+10 8.4524e+10 1.00000000 1.0000000
```

We are now less worried about the financial health of Facebook. Even though the liabilities increased about 76% from 2016 to 2017, these values are still low with respect to the firm's assets. In 2016 their liabilities were about 9% of their assets and 12% in 2017.

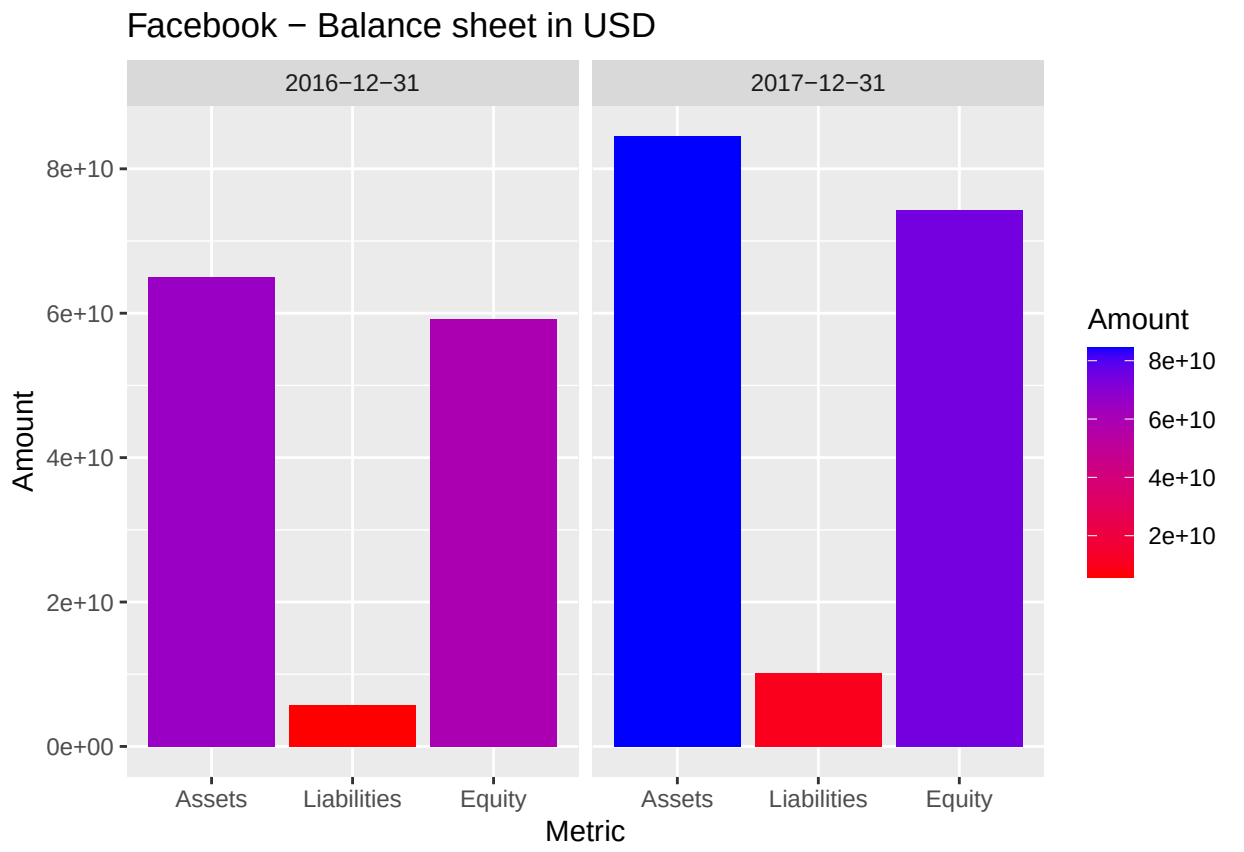
Let's plot.

```
accounts_BS <- c("Assets" , "Liabilities",
              "Stockholders' Equity Attributable to Parent")

BS_2016_2017_plot <- BS_FB |>
  mutate(Amount = as.numeric(Amount)) |>
  mutate(Date = as.Date(endDate)) |>
  mutate(Metric = factor(Metric, levels = unique(Metric))) |>
  select(Metric, Amount, Date) |>
    filter(Date == "2016-12-31" | Date == "2017-12-31") |>
  filter(Metric %in% accounts_BS) |>
 mutate(Metric = recode_factor(Metric,
`Stockholders' Equity Attributable to Parent` = "Equity")) |>
 mutate(Metric =
           factor(Metric, levels = c("Assets",
                                     "Liabilities", "Equity")))
BS_2016_2017_plot
```

```
##         Metric     Amount       Date
## 1       Assets 6.4961e+10 2016-12-31
## 2       Assets 8.4524e+10 2017-12-31
## 3 Liabilities 5.7670e+09 2016-12-31
## 4 Liabilities 1.0177e+10 2017-12-31
## 5       Equity 5.9194e+10 2016-12-31
## 6       Equity 7.4347e+10 2017-12-31
```

Having the data as in `BS_2016_2017_plot` makes it easier to use `ggplot`.

```
library(ggplot2)
g <- ggplot(BS_2016_2017_plot,
            aes(Metric, Amount, fill = Amount)) +
  ggtitle("Facebook - Balance sheet in USD") +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "red", high = "blue") +
  facet_wrap(~ Date)
g
```



Imagine all balance sheet accounts increased as in the case of Facebook, but the firm is losing money according to the income statement. In this case, we could say there is something wrong with Facebook. On the other hand, if these increases in the balance sheet accounts are supported by an increase in profits for the same period, then things are going well in the firm. Thus, we need Facebook income statement to verify the firm's profits and evaluate whether we should be worried about the increase in the liabilities.

Let's download the income statement and show only the revenues and profit for 2016 and 2017. Note that the process in R is basically the same as in the case of the balance sheet before. This is important as coding allows us to replicate or reproduce the analysis easily. As the process was recorded in the balance sheet, it is only a matter of adapting the same instructions to perform a similar manipulation for the income statement. You might see several lines of code but in reality, it is almost a copy-paste from the previous code. In fact, we are using the comments to emphasize that this code is not new, and is basically a copy of our previous procedure.

```
# Define accounts just as we did before for the balance sheet.
accounts_IS <- c("Revenues" ,
  "Net Income (Loss) Available to Common Stockholders, Basic")
# Instead of GetBalanceSheet, we use GetIncome.
IS_FB <- GetIncome("FB", 2018)
  # Make a copy (as before).
IS_2016_2017_summary <- IS_FB |>
  # Exactly the same as before.
  mutate(Amount = as.numeric(Amount)) |>
  # Exactly the same as before.
  mutate(Metric = factor(Metric, levels = unique(Metric))) |>
  # Select the required columns.
  select(Metric, Amount, endDate) |>
  # Exactly the same as before.
  spread(endDate, Amount) |>
  # Exactly the same as before.
  select(Metric, "2016-12-31", "2017-12-31") |>
  # Similar as before.
  filter(Metric %in% accounts_IS) |>
  # Similar as before.
  mutate(Metric = recode(Metric,
```

```
  `Net Income (Loss) Available to Common Stockholders, Basic`
  = "Net Income"))
# Similar as before.
IS_2016_2017_summary
```

```
##       Metric 2016-12-31 2017-12-31
## 1   Revenues 2.7638e+10 4.0653e+10
## 2 Net Income 1.0188e+10 1.5920e+10
```

In this code above we explicitly repeat how many times we reproduce the same or very similar code as we did in the case of the balance sheet. This is because it is important to highlight that coding allows us to reproduce our analysis easily. Imagine you are interested to compare balance sheet and income statements for 5 companies using a software that requires using the mouse to click on pre-defined menus. In that case you will have to repeat the procedure step by step and hope to remember so you do not skip anything. In R we code so our procedure is recorded in a series of instructions. Now consider that you need to add two accounts to your analysis. In that case you will only have to change this line:

```
accounts_IS <- c("Revenues" , "Net Income (Loss) Available to Common Stockholders,
Basic")
```

and add a third or fourth account name and that's it.

Now we have the relevant information we need. The profits increased more than 50% from 2016 to 2017, so apparently the firm knows how to make money. The new asset investments allow the firm to increase the profits. We can calculate a simple financial ratio to find out the relationship between assets (in the balance sheet, representing the firm investment) and profits (in the income statement, shows the ability of the firm to make money). A financial ratio or accounting ratio is a relative magnitude of two selected numerical values taken from an enterprise's financial statements. There are many standard ratios used to try to evaluate the overall financial condition of a corporation.

- $AP_{2016} = 6.4961/1.0188 = 6.376227$
- $AP_{2017} = 8.4524/1.5920 = 5.309296$

We normally are interested in making these calculations automatically, without copying the values and taking them directly from the correspondent variables:

```
BS_2016_2017_summary[1,2] / IS_2016_2017_summary[2,2]
```

```
## [1] 6.376227
```

```
BS_2016_2017_summary[1,3] / IS_2016_2017_summary[2,3]
```

```
## [1] 5.309296
```

AP_{2016} and AP_{2017} represent the ratio of assets and profits for 2016 and 2017. Then, the firm requires 6.38 USD asset investment to make 1 USD in profits in 2016 and only 5.31 USD in 2017. We can also view the same thing the other way around, this is profits over assets. If we do this, then the firm makes 0.1568326 USD for every dollar invested in the asset in 2016 and 0.1883489 USD in 2017. This is clearly good news as the firm's assets are more productive according to this simple analysis.

Let's calculate the revenues with respect to profits.

```
# Make a copy.
IS_2016_2017_prop <- IS_2016_2017_summary |>
  # Calculate proportion (2016).
  mutate(prop2016 = `2016-12-31`/`2016-12-31`[2]) |>
  # Calculate proportion (2017).
  mutate(prop2017 = `2017-12-31`/`2017-12-31`[2])
# Show results.
IS_2016_2017_prop
```

```
##       Metric 2016-12-31 2017-12-31 prop2016 prop2017
## 1   Revenues 2.7638e+10 4.0653e+10 2.712799  2.55358
## 2 Net Income 1.0188e+10 1.5920e+10 1.000000  1.00000
```

According to this ratio, in 2016 the firm transformed 2.7 USD of sales into 1 USD profit, and in 2017 the firm required only 2.5 USD. Accounting ratios like this one can be used for financial managers as corporate targets or firm's objectives. Imagine most of the rest of the firms in the same industry only need 2 USD to generate 1 USD profit. In that case, the financial manager might propose an objective for next year to reduce costs so they can transform 2 USD sales into 1 USD profits for the next year.

Let's plot the income statement basics. First, some necessary changes.

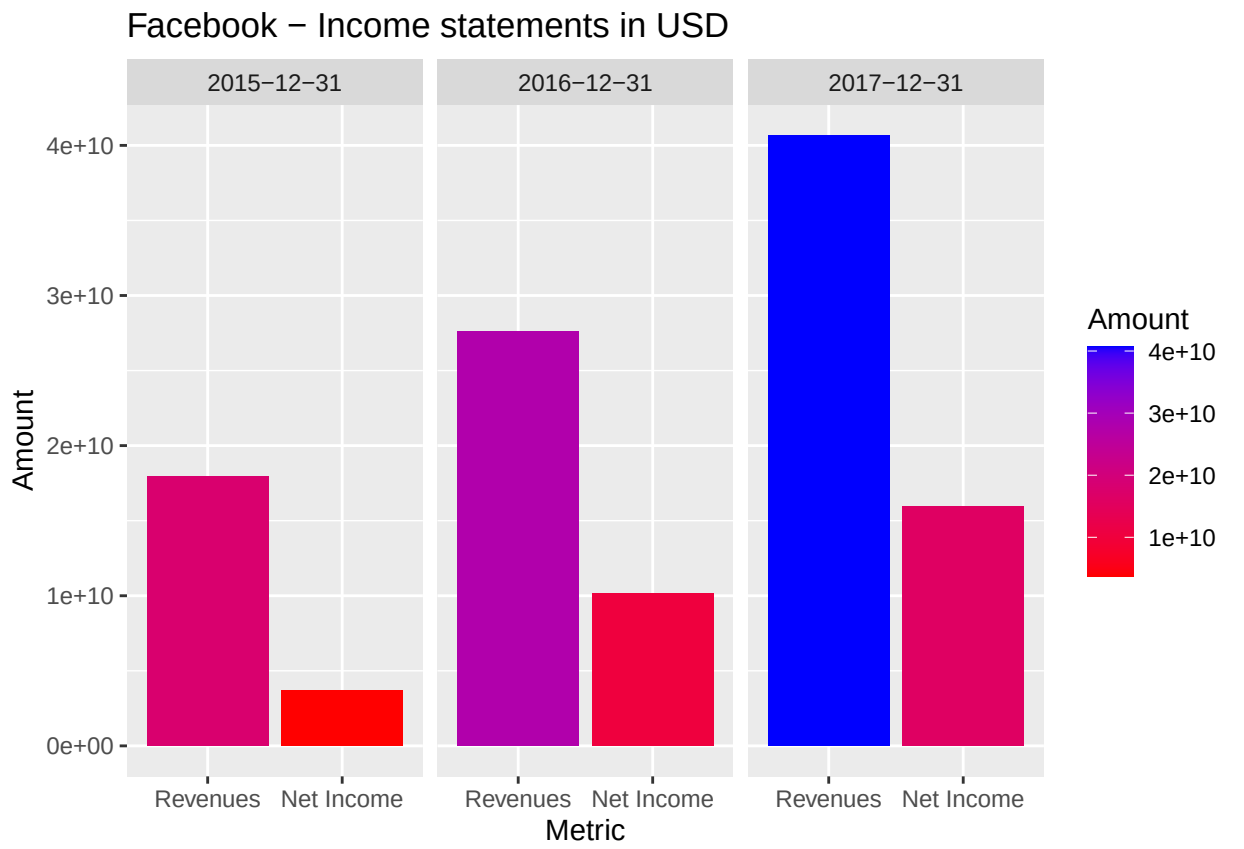
```
IS_2016_2017_plot <- IS_FB |>
  mutate(Amount = as.numeric(Amount)) |>
  mutate(Date = as.Date(endDate)) |>
  mutate(Metric = factor(Metric, levels = unique(Metric))) |>
  select(Metric, Amount, Date) |>
```

```
  filter(Metric %in% accounts_IS) |>
  mutate(Metric = recode(Metric,
  `Net Income (Loss) Available to Common Stockholders, Basic`
  = "Net Income"))
IS_2016_2017_plot
```

```
##       Metric     Amount       Date
## 1   Revenues 1.7928e+10 2015-12-31
## 2   Revenues 2.7638e+10 2016-12-31
## 3   Revenues 4.0653e+10 2017-12-31
## 4 Net Income 3.6690e+09 2015-12-31
## 5 Net Income 1.0188e+10 2016-12-31
## 6 Net Income 1.5920e+10 2017-12-31
```

Now, a nice plot.

```
library(ggplot2)
g <- ggplot(IS_2016_2017_plot,
            aes(Metric, Amount, fill = Amount)) +
  ggtitle("Facebook - Income statements in USD") +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "red", high = "blue")+
  facet_wrap(~ Date)
g
```



This is why we are interested firms to perform well in the economy. If firms are well managed, they are expected to take good strategic decisions to increase the value of the firm. In particular, the firm looks for funds from banks or owners, then invests in assets to produce or provide a service to the market. If things go well, sales increase and if revenues are greater than costs, the firm makes profits. When the firm increases their assets, they might also need more people to produce or manage those assets, then employment increases. More jobs mean more families with available income, and families will spend their income and stimulate the whole economy further. Even government may increase their tax revenues to (hopefully) spend them into high-quality public goods to increase society's welfare. Not only that, as a firm performs well, the whole value chain straightens as the firm requires inputs and services from other firms to operate. We are interested firms to perform well in the economy because that is good for the population in general and to other firms as well. Private investment stimulates the whole economy.

Strictly speaking, firms (regardless of their size) do not own what they have. Firm's assets were originally bought or financed using others' money (liabilities) and/or owners' money (equity). In simple terms, people are interested to finance the firm activities because the firm is supposed to know how to invest those funds in productive assets to make profits as

in the case of Facebook. When the financial statements show that firms take bad decisions, their assets do not produce profits and people will not be very interested in financing a non-profitable firm. In good times, people who finance productive firms receive interests, returns, or dividends as a compensation for putting their money at risk.

Firms have incentives to be and keep being financially healthy in the future to attract financing more easily. Public firms (as those participating in the stock market) are subject to strict regulations and public scrutiny so everybody can constantly evaluate the firm performance. Regulations do not prevent all frauds or bankruptcies, but they are intended to minimize irregular and illegal practices. We were able to freely download financial statements because regulators ask public firms to do so given some deadlines and specific guidelines. Yearly and quarterly financial statements are analyzed in detail to evaluate firms as projects, and the main interest is the relationship between risk and return, so people can make better investment decisions. The decision of financing a firm operation is risky, but the reward is supposed to be attractive enough to compensate that risk. In finance we are interested to measure how risky are the investment opportunities in the form of firms, projects, and financial assets in general. When we do a financial analysis, we also compare the performance of one company not only with respect to previous years, but also with respect to other firms, and also with respect to their own industry or sector. This is not difficult as the regulators ask public firms to publish their financial statements in a similar manner, so comparisons among different firms are valid.

Finance is about people taking decisions under uncertainty with the help of data analysis. Think in two kind of people in the world: those who have money but no ideas (investors), and those who have ideas but no money (firms or entrepreneurs). We refer to ideas as business ideas, innovations, business projects that require funds to get started. Those with ideas and money are rare in this world, and those without ideas and without money are in trouble. When the financial system works well, productive projects will succeed at attracting funds, firms will invest in productive assets, private investment grows, and we should expect economic growth, and hopefully economic development improves. When the financial system fails, good projects might fail to get funding and probably bad projects get financing. This could be frustrating.

People with money and no ideas are looking for opportunities for their money to grow. They basically face two alternatives: to save their money (put it in the bank and receive a small but certain return), or invest their money in risky projects offered by firms or entrepreneurs. The final decision is taken based on a risk-return analysis, with the help of data analysis and financial models. Imagine the bank offers a secure 4%, and the firm offers a risky 3.5%

return (assume it could be as low as -1% or as high as 4%). In that case, most of us might agree (if you are risk averse as me) that the funds are better in a bank. Now assume a different situation. The bank offers a secure 4%, and the firm offers a risky 6% return (it could be as low as 3.5% and as high as 8%). In that case, there might be some investors that after doing some analysis finally decides to put their money at risk. Things become interesting and complex because the same project can be attractive for some investors and bad for others as they might have different tools to analyze risky projects and they might also have different risk appetite levels.

People with ideas but no money are looking for funds to run their projects. If the project is impressively good, then the firm could ask for funds in exchange of a reduced return. The project is then not too risky and the return is also low. On the other hand, if the project looks OK but entails a high risk, the firm or entrepreneur should be willing to pay high returns to investors to compensate for the associated risk. This is why risk and return move together, the higher the risk the higher the return and vice-versa. If this rule fails, then we are in trouble.

Interestingly, an excess of private investment is not always good for the economy. Imagine the interest rate (return of savings, and the cost of asking for a loan) is low and the economic perspectives look good. In that case, we have good conditions for the private investment to grow as it is cheap to ask for financing. In the extreme, families now have more money and they start spending a lot. Aggregate demand for all goods and services increase and consequently the prices of all goods and services increase as well. The country is experiencing an increase in the inflation levels. Then, even though people have money in their pockets, they are not able to buy as prices increase faster than families' income. In order to reduce inflation levels, the monetary authority increases the interest rates, making savings more attractive and investment less attractive. If they succeed, people will spend less and save more. This decreases the aggregate demand for goods and services and prices could decrease again to desired levels. To add more drama, imagine that this increase in interest rates is so high that now economic growth is slowing down and the risk of falling into a recession increase. If that happens, then the monetary authority should decrease the interest rates again to incentive people to stop saving and start investing again. This is why you see that interest rates go up and down, although we have had low interest rates for a while.

Let's go back to finance and illustrate the process of a firm looking for funds, the difference between private and public firms, and see how a firm can grow or go bankrupt in a simple but illustrative example.

Firms can sell bonds or stocks as a way to get funds. Let's focus on the stocks. By selling stocks, the firm can increase their equity and consequently increase their assets. People who buy stocks expect the firm to make wise investment decisions to get a return in the future. Initially, in time zero $t = 0$, assume the firm is private and has the following balance sheet:

```
# The firm is private now.
private_firm_bs <- c(assets = 100, liabilities = 40, equity = 60)
# Print results.
private_firm_bs
```

```
##      assets liabilities      equity
##         100          40          60
```

Private firms cannot sell stocks in the stock market, only public firms can. Selling stocks in the stock market is attractive for some firms because it is a way for them to grow. Stock markets allow more participant firms as long as they fulfill some strict requirements. Given the balance sheet above, the owner of this private firm invested 60 USD. When the firm goes 100% public, then the owner will sell the 60 USD investment to others.

In $t = 1$, with the objective of getting fresh funds, the firm decides to go 100% public and sell 60 stocks at 1 USD each. Imagine that the firm succeeded at selling all 60 stocks at 1 USD. This is because investors who participate in the stock market consider that this is a fair value for the firm. These investors become new owners; this is why we say the firm is now public. So, by the end of $t = 1$ we have:

```
# The firm is public now.
public_firm_bs <- c(assets = 100, liabilities = 40,
                    equity_60x1 = 60)
# Print results.
public_firm_bs
```

```
##      assets liabilities equity_60x1
##         100          40          60
```

In $t = 2$ more investors realize that the firm is well managed, its market is growing and there is no close competition. Now, investors actually think that paying 1 USD per stock is actually a bargain, so they are willing to pay even more than 1 USD to participate in this project. So, the demand for those stocks increases, and as a result the price goes from 1 to 2 USD per share. By the end of $t = 2$ we have:

```r
# Stock price increased from 1 to 2.
public_firm_bs <- c(assets = 100+60, liabilities = 40,
                    equity_60x2 = 120)
# Print results.
public_firm_bs
```

```
##      assets liabilities equity_60x2
##         160          40         120
```

Now the firm has grown by going public and by showing good perspectives.

However, in $t = 3$ we sadly get to know that the firm's financial statements were overestimating profits, and a big fraud was detected today. Big scandal and all newspapers talk about it. Then, investors are now interested in selling all their stocks as quickly as they can. Investors are so desperate to sell their stocks that they drop the stock price to almost zero. By the end of $t = 3$ we have:

```r
# Stock price decrease to 0.
public_firm_bs <- c(assets = 40, liabilities = 40,
                    equity_60x0 = 0)
# Print results.
public_firm_bs
```

```
##      assets liabilities equity_60x0
##          40          40           0
```

There are still 60 stocks in the market with a unit price of zero, so the market value of equity is null. This hypothetical firm is virtually bankrupt as they need to sell all their assets to pay the debt. In simple terms, a firm is bankrupt when the liabilities are greater than the assets. The firm will hardly get fresh funds as investors realize this was a bad project.

Many things can happen when a firm faces this difficult situation. Sometimes it is worthwhile to try to improve the firm's financial condition, hiring professional managers, and hope to recover the investor's confidence. Before bankruptcy, firms could also negotiate with their debt holders. If negotiation succeeds, the firm could get a longer debt maturity, a reduced debt to help the firm remain alive and hope to recover its operations, or a more innovative solution. There are also other firms that look for firms in trouble to buy them for a wide variety of reasons. In the US, you may hear about "Chapter 11", which is a form of bankruptcy that involves a reorganization of a debtor's business affairs, debts, and assets,

and for that reason is known as "reorganization" bankruptcy. Corporations generally file Chapter 11 if they require time to restructure their debts. This version of bankruptcy gives the debtor a fresh start. However, the terms are subject to the debtor's fulfillment of its obligations under the plan of reorganization.

As we show before, the stock price evolution is generally a good indicator about how the investors interpret the future of the firm. When the stock price increases, then investors are demanding more stocks. A higher stock price is associated with a more valuable equity and this is good for the firm. When the stock price decreases, then there are more people trying to sell stocks compared to those trying to buy so the stock price falls. There might be some cases in which a drop in the stock price is associated with a firm good performance. Consider the stock price has increased in the last week, investors bought at 10 USD and today's price is 15 USD. Those investors might be interested to sell now to make a 5 USD profit. This increase in the supply of stocks could be significant enough to drop prices in the short run.

Remember this firm needed to go public before issuing and selling stocks in the stock market. Normally, these firms are big, although there are some stock markets for medium-sized firms. Small firms can hardly access these kinds of markets to get funding in a direct way. This might be problematic because small firms are the ones which require more alternatives to get financing so they can grow. The literature regarding small firms show that one of the most important problems they face is precisely the lack of financing or the limited alternatives to get funds. This remains an open challenge for the financial services and an opportunity for fintech companies. In the world there are significantly more small firms compared with big firms, so any improvement in the credit conditions of small firms could have a significant impact on the income of many people.

In the following section we move from a financial statement analysis to the analysis of stock prices. Note that both are linked by the equity and overall performance of public firms.

2.2 Stock prices and other financial data.

Financial markets not only facilitate financial transactions between sellers and buyers, they also represent a rich source of financial data. Here, we show how to use R to download stock price data from financial markets by using a few examples.

R packages like *quantmod* and *tidyquant* make the process of downloading financial data to perform financial analysis very straightforward. The *quantmod* package for R is designed to assist the quantitative trader in the development, testing, and deployment of statistically based trading models. The *tidyquant* package is part of the so-called *tidyverse*, an opinion-

ated collection of R packages introduced by Hadley Wickham and designed for data science. All *tidyverse* packages share an underlying design philosophy, grammar, and data structures.

In the past, when we were interested in performing a financial analysis in a rudimentary statistical software, we had to open the financial market site, download the data in Excel or text format, and then convert it to a compatible format given the statistical software requirements. In ancient times, people had to gather financial data from the newspaper. Fortunately for us, now we have R packages to make this process not only easy but also free, extremely efficient and immediate.

Let's download Apple stock prices in one step (we already loaded the *tidyquant* package as step zero).

```
# Get stock prices for Apple stock from Yahoo! finance site.
aapl_stock_prices <- tq_get("AAPL")
```

We are done. The special variable `aapl_stock_prices` contains Apple stock prices and other relevant information. Many things happened here in a very few steps. First, we need to have Internet access, R and R Studio working on our computer. Then, we load the *tidyquant* package. This package presumably integrates the best resources for collecting and analyzing financial data in R. The `tq_get` function belongs to the *tidyquant* package, and *tidyquant* belongs to the *tidyverse* packages. Specifically, `tq_get` connects to a default financial site, which in this case is the finance.yahoo.com and looks for the "AAPL" symbol which corresponds to Apple Inc. (an American multinational technology company). Finally, we store the data into `aapl_stock_prices`.

To see what it is inside the `aapl_stock_prices` object we can do the following.

```
# str function is a way to display the structure of an R object.
str(aapl_stock_prices)
```

```
## tibble [2,643 x 8] (S3: tbl_df/tbl/data.frame)
##  $ symbol  : chr [1:2643] "AAPL" "AAPL" "AAPL" "AAPL" ...
##  $ date    : Date[1:2643], format: "2011-01-03" "2011-01-04" ...
##  $ open    : num [1:2643] 11.6 11.9 11.8 12 11.9 ...
##  $ high    : num [1:2643] 11.8 11.9 11.9 12 12 ...
##  $ low     : num [1:2643] 11.6 11.7 11.8 11.9 11.9 ...
##  $ close   : num [1:2643] 11.8 11.8 11.9 11.9 12 ...
##  $ volume  : num [1:2643] 4.45e+08 3.09e+08 2.56e+08 3.00e+08 3.12e+08 ...
##  $ adjusted: num [1:2643] 10.1 10.2 10.3 10.2 10.3 ...
```

According to the `str` output, `aapl_stock_prices`, we have 2643 daily observations for 7 variables. As you can see, we have not only prices but also volume. Let's consider a different way to see and have a grasp of what it is inside `aapl_stock_prices`. In particular, we can see the first and the last part of this lengthy daily database.

The first observations.

```
# See the first observations of aapl_stock_prices.
head(aapl_stock_prices)
```

```
## # A tibble: 6 x 8
##   symbol date        open  high   low close    volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1 AAPL   2011-01-03  11.6  11.8  11.6  11.8 445138400     10.1
## 2 AAPL   2011-01-04  11.9  11.9  11.7  11.8 309080800     10.2
## 3 AAPL   2011-01-05  11.8  11.9  11.8  11.9 255519600     10.3
## 4 AAPL   2011-01-06  12.0  12.0  11.9  11.9 300428800     10.2
## 5 AAPL   2011-01-07  11.9  12.0  11.9  12.0 311931200     10.3
## 6 AAPL   2011-01-10  12.1  12.3  12.0  12.2 448560000     10.5
```

The last observations.

```
# See the last observations of aapl_stock_prices.
tail(aapl_stock_prices)
```

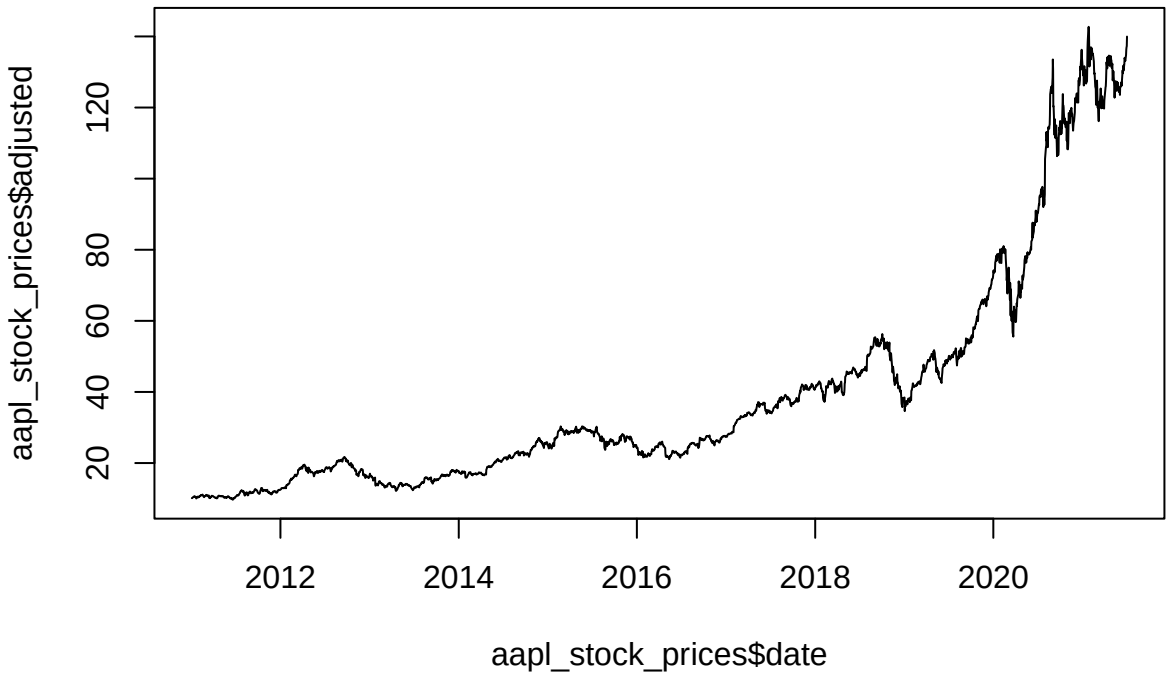
```
## # A tibble: 6 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 AAPL   2021-06-25  133.  134.  133.  133. 70783700     133.
## 2 AAPL   2021-06-28  133.  135.  133.  135. 62111300     135.
## 3 AAPL   2021-06-29  135.  136.  134.  136. 64556100     136.
## 4 AAPL   2021-06-30  136.  137.  136.  137. 63261400     137.
## 5 AAPL   2021-07-01  137.  137.  136.  137. 52485800     137.
## 6 AAPL   2021-07-02  138.  140   138.  140. 78852600     140.
```

By default, the `tq_get` function downloads the latest set of data available. You can verify that the last date of `aapl_stock_prices` above approximately corresponds to the production date of this tutorial. If there is a difference it is simply because the stock market is still close or we are running the code in a weekend. In this case:

- Last `aapl_stock_prices` observation: 2021-07-02.
- Today is: 2021-07-05.

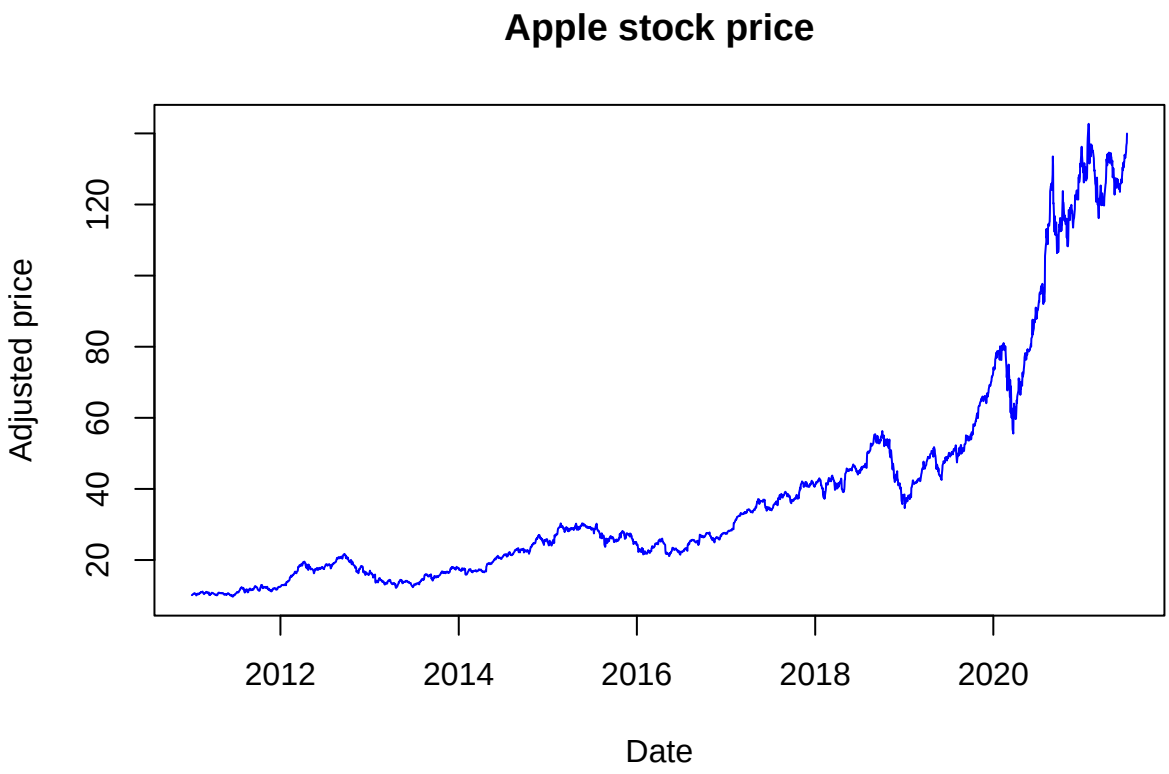
We also like to show the data in a plot.

```
# Date is in the x axis, price in the y axis. The type l stands for line.
plot(aapl_stock_prices$date, aapl_stock_prices$adjusted,
     type = "l")
```



We can add a few instructions to improve the format of our plot.

```
# x and y labels, the title, and the color of the line.
plot(aapl_stock_prices$date, aapl_stock_prices$adjusted,
     type = "l",
     xlab = "Date", ylab = "Adjusted price",
     main = "Apple stock price", col = "blue")
```



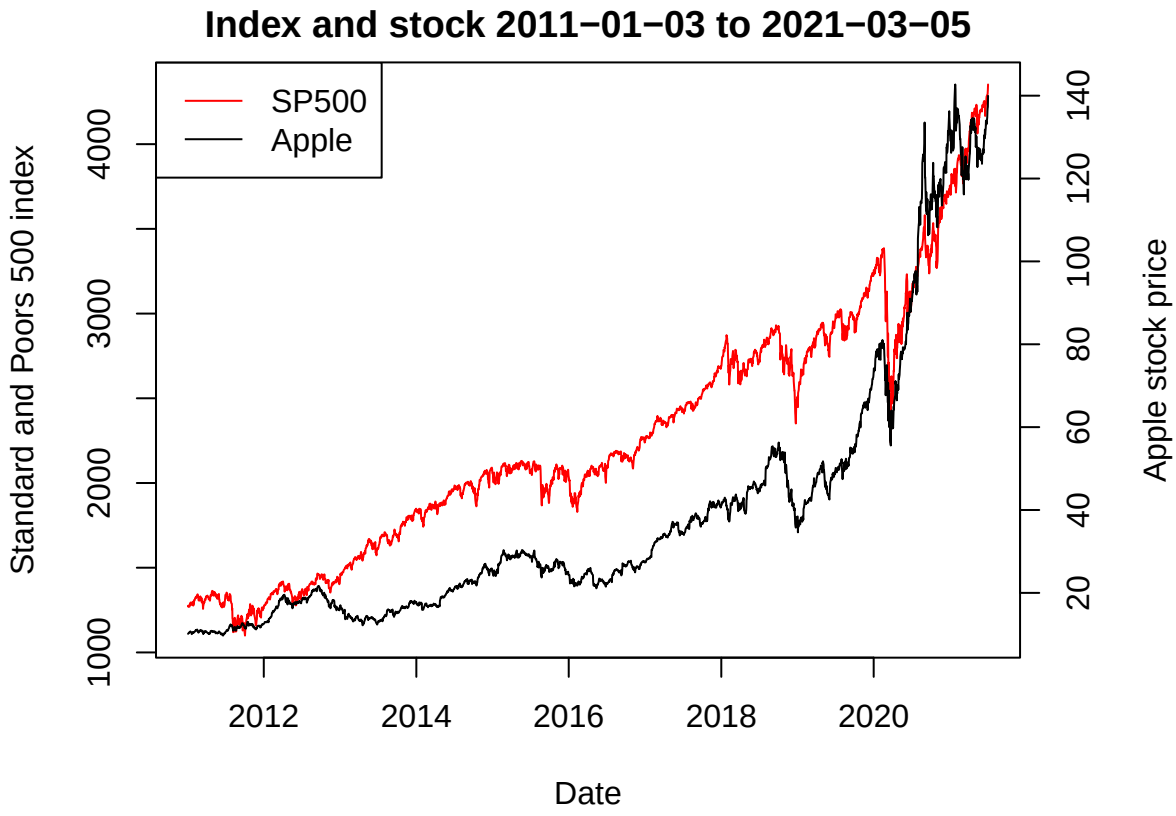
We can compare Apple versus a stock index like S&P500.

```
# Download the S&P500 index.
SP <- tq_get("^GSPC")
```

Plot both assets together.

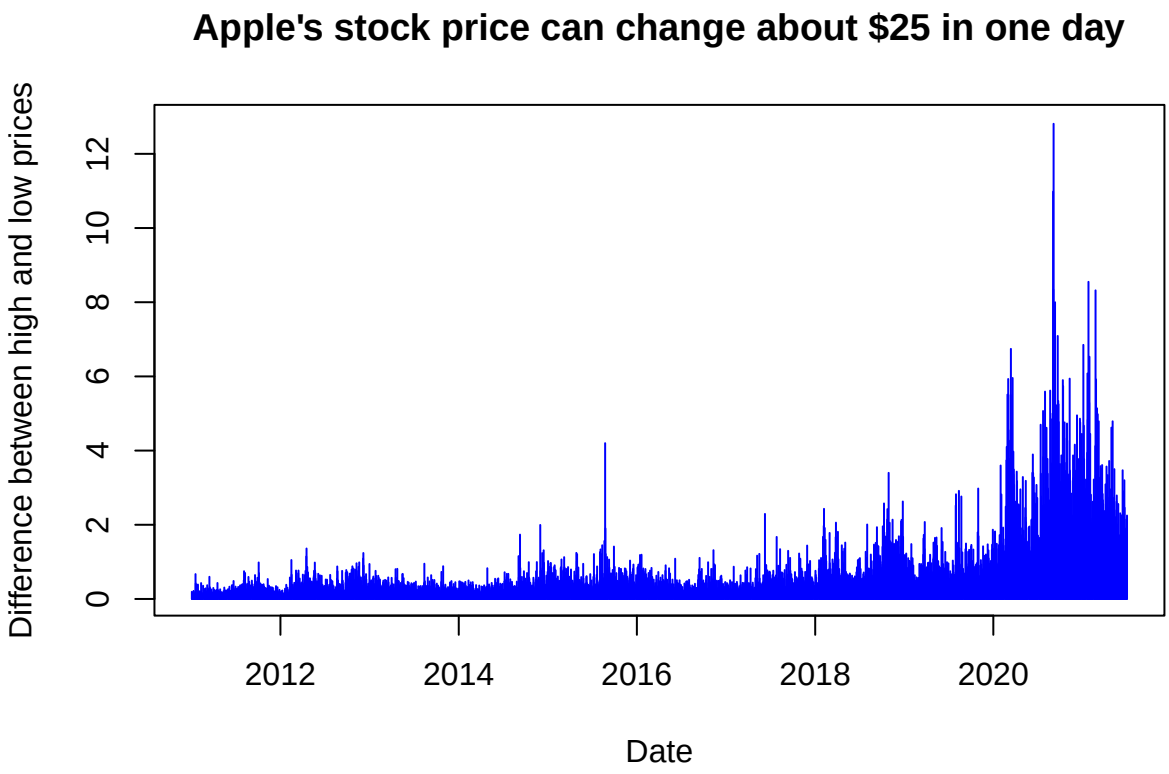
```
par(mar = c(5, 5, 2, 5))
plot(SP$date, SP$adjusted, type = "l", col = "red",
     ylab = "Standard and Poors 500 index",
     xlab = "Date",
     main = "Index and stock 2011-01-03 to 2021-03-05")
par(new = T)
plot(aapl_stock_prices$date, aapl_stock_prices$adjusted,
     type = "l", axes = F, xlab = NA, ylab = NA, cex = 1.2)
axis(side = 4)
mtext(side = 4, line = 3, "Apple stock price")
legend("topleft",
       legend=c("SP500", "Apple"),
```

```
       lty = 1, col = c("red", "black"))
```



Let's continue with the Apple stock analysis. Stock prices change throughout the trading day. This is, the open price changes every minute (sometimes in milliseconds), so the open price is normally different from the closing price. We can calculate the difference between the high and low price of each day and then show the result in a plot.

```
# Create a new variable.
aapl_diff = aapl_stock_prices$high - aapl_stock_prices$low
# Plot the new variable.
plot(aapl_stock_prices$date, aapl_diff, type = "h",
     xlab = "Date",
     ylab = "Difference between high and low prices",
     main = "Apple's stock price can change about $25 in one day",
     col = "blue")
```



The difference between high and low prices show that the stock price changes during a trading day. These changes can be considerably high. Moreover, this volatility has increased in recent times. Is it possible to know the exact date in which this difference is the highest? One alternative is to sort all observations, but there is a simpler approach.

```
# Which observation has the maximum value of aapl_diff?
highest_change = which.max(aapl_diff)
# Show results.
highest_change
```

```
## [1] 2436
```

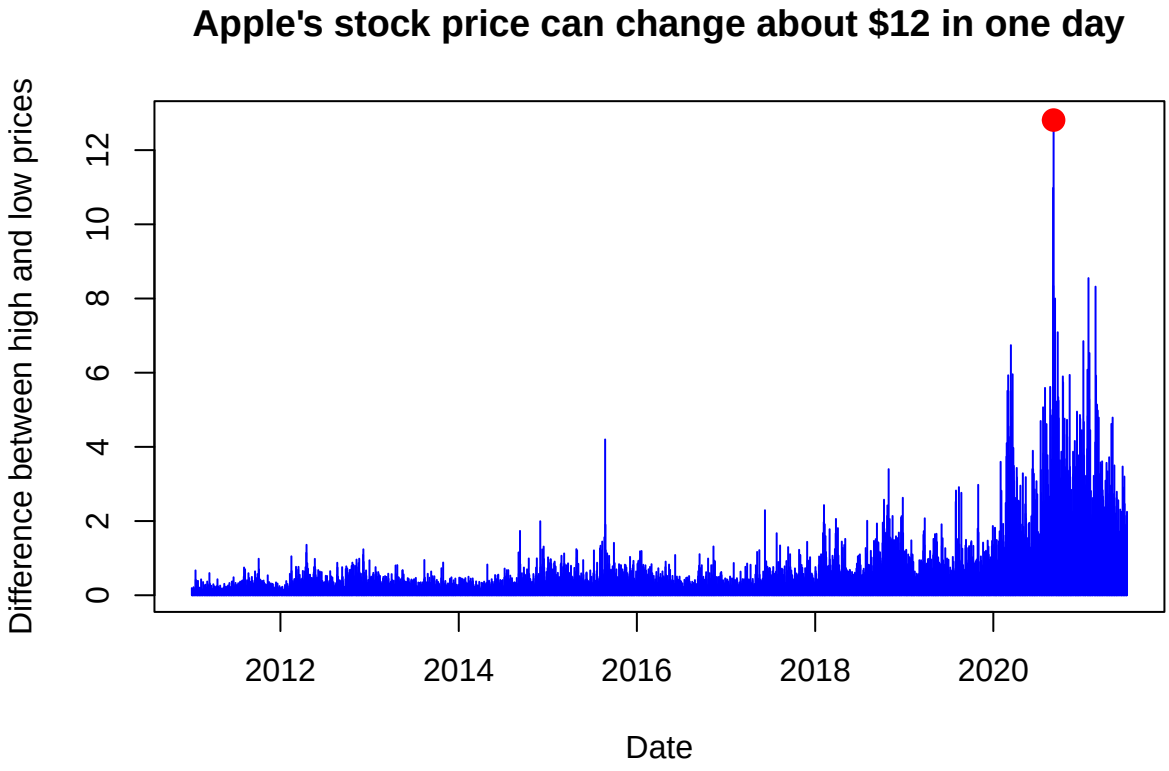
Consider the value of `highest_change` as an index. This is, the number of the row that contains the highest `aapl_diff` value. We can use this index to extract the date and the actual `aapl_diff` value.

```
# Now, extract the date and the actual value of aapl_diff.
# See how brackets [] are used to extract a single row value.
when = aapl_stock_prices$date[highest_change]
```

```
top = aapl_diff[highest_change]
```

The output above shows when (in the variable `when`) Apple stock had its highest price change (`top`) in one day.

```
# We can use when and top variables to improve our plot.
plot(aapl_stock_prices$date, aapl_diff, type = "h",
     xlab = "Date",
     ylab = "Difference between high and low prices",
     main = "Apple's stock price can change about $12 in one day",
     col = "blue")
# Here, we add the red point.
points(when, top, pch = 19, col = "red", cex = 1.5)
```



Up to know, we have been using default options as we simply ask for stock prices of Apple without any other kind of instruction. As a result, we get data starting from 2010: `tq_get("AAPL")`. However, we can change default options if needed to get data back to 1990. I will take some online available examples developed by the author of *tidyquant* Matt

Dancho.

```
# Download Apple stock prices.
aapl_prices  <- tq_get("AAPL", get = "stock.prices",
                       from = " 1990-01-01")
# Show results.
aapl_prices
```

```
## # A tibble: 7,938 x 8
##    symbol date        open  high   low close    volume adjusted
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
##  1 AAPL   1990-01-02 0.315 0.335 0.312 0.333 183198400    0.267
##  2 AAPL   1990-01-03 0.339 0.339 0.335 0.335 207995200    0.269
##  3 AAPL   1990-01-04 0.342 0.346 0.333 0.336 221513600    0.270
##  4 AAPL   1990-01-05 0.337 0.342 0.330 0.337 123312000    0.271
##  5 AAPL   1990-01-08 0.335 0.339 0.330 0.339 101572800    0.273
##  6 AAPL   1990-01-09 0.339 0.339 0.330 0.336  86139200    0.270
##  7 AAPL   1990-01-10 0.336 0.336 0.319 0.321 199718400    0.258
##  8 AAPL   1990-01-11 0.324 0.324 0.308 0.308 211052800    0.247
##  9 AAPL   1990-01-12 0.306 0.310 0.301 0.308 171897600    0.247
## 10 AAPL   1990-01-15 0.308 0.319 0.306 0.306 161739200    0.246
## # ... with 7,928 more rows
```

Sometimes we are interested in periodicity aggregation from daily to monthly. We cannot transform from monthly to daily, but we can always transform from daily to monthly, monthly to yearly and so on. FANG is a dataset containing the daily historical stock prices for the "FANG" tech stocks, "FB", "AMZN", "NFLX", and "GOOG", spanning from the beginning of 2013 through the end of 2016.

Let's aggregate data from daily to monthly frequency.

```
# Here, we are using a pre-loaded dataset.
data("FANG")
# Transform from daily to monthly stock prices.
FANG |>
    group_by(symbol) |>
    tq_transmute(select = adjusted, mutate_fun = to.monthly,
                 indexAt = "lastof")
```

```
## # A tibble: 192 x 3
## # Groups:   symbol [4]
##    symbol date       adjusted
##    <chr>  <date>        <dbl>
##  1 FB     2013-01-31     31.0
##  2 FB     2013-02-28     27.2
##  3 FB     2013-03-31     25.6
##  4 FB     2013-04-30     27.8
##  5 FB     2013-05-31     24.4
##  6 FB     2013-06-30     24.9
##  7 FB     2013-07-31     36.8
##  8 FB     2013-08-31     41.3
##  9 FB     2013-09-30     50.2
## 10 FB     2013-10-31     50.2
## # ... with 182 more rows
```

The *tidyquant* package can also access other kinds of data from diverse sources like the Federal Reserve Economic Data (FRED). Federal Reserve Economic Data is a database maintained by the Research division of the Federal Reserve Bank of St. Louis that has more than 765,000 economic time series from 96 sources. Consider the WTI Crude Oil Prices.

```
# See https://fred.stlouisfed.org/series/DCOILWTICO
# Download oil prices from FRED.
wti_price_usd <- tq_get("DCOILWTICO", get = "economic.data")
# Show results.
wti_price_usd
```

```
## # A tibble: 2,736 x 3
##    symbol     date       price
##    <chr>      <date>     <dbl>
##  1 DCOILWTICO 2011-01-03  91.6
##  2 DCOILWTICO 2011-01-04  89.4
##  3 DCOILWTICO 2011-01-05  90.3
##  4 DCOILWTICO 2011-01-06  88.4
##  5 DCOILWTICO 2011-01-07  88.1
##  6 DCOILWTICO 2011-01-10  89.2
##  7 DCOILWTICO 2011-01-11  91.1
##  8 DCOILWTICO 2011-01-12  91.8
```

```
##  9 DCOILWTICO 2011-01-13  91.4
## 10 DCOILWTICO 2011-01-14  91.5
## # ... with 2,726 more rows
```

Now that we have oil prices in basically one single step, we can plot them. It is interesting because recently the oil prices have turned negative in one day. Negative prices are rare but not impossible, especially in commodities.

```
# ggplot2 is a system for creating graphics see https://amzn.to/2ef1eWp.
library(ggplot2)
# Create a simple line plot to show the oil price evolution.
ggplot(wti_price_usd, aes(date, price)) +
  geom_line() + theme_bw() +
geom_hline(yintercept = 0, linetype = "dashed", color = "red")
```

It is easy to find out the exact day when the oil price was negative.

```
# Extract one row given a condition.
subset(wti_price_usd, price < 0)
```

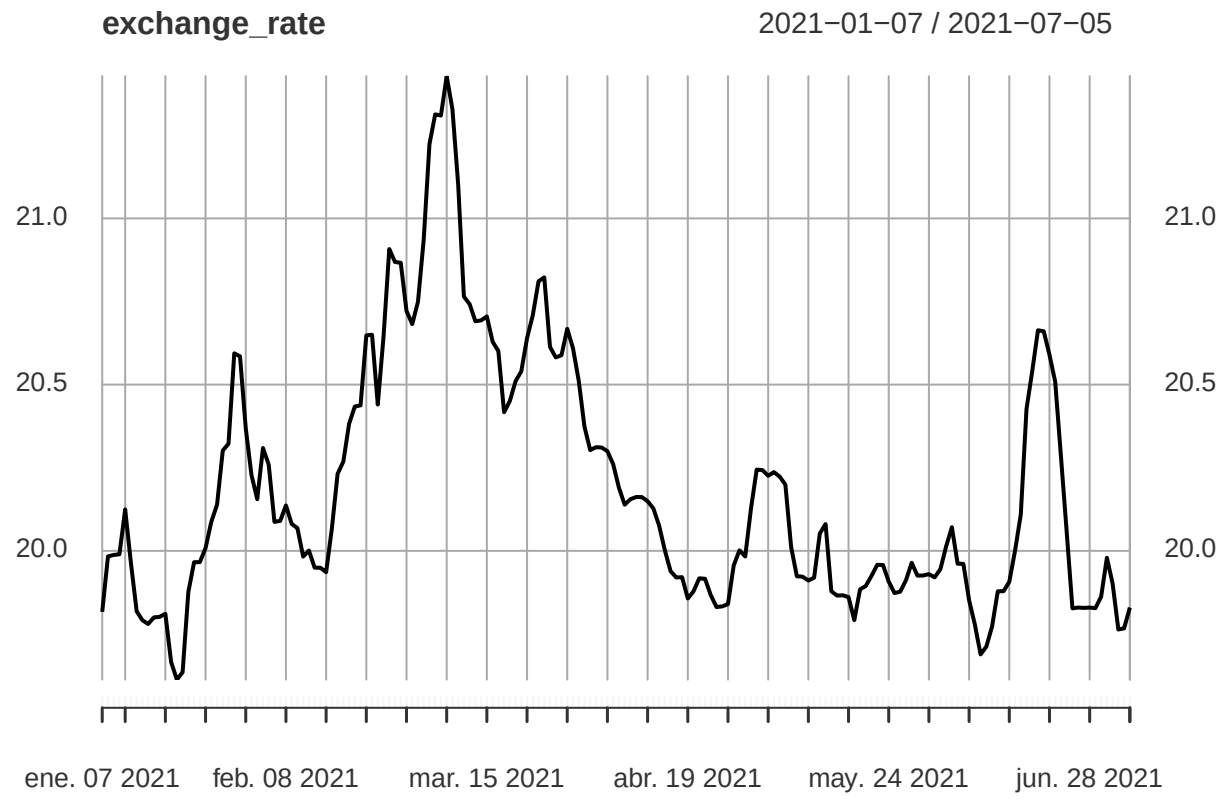
```
## # A tibble: 1 x 3
##   symbol     date       price
##   <chr>      <date>     <dbl>
## 1 DCOILWTICO 2020-04-20 -37.0
```

Without going into technical arguments, we can interpret negative prices as the following. Imagine (as it actually happened) that storing oil is expensive, not only that, imagine oil producers have no further physical space to store oil, so they are desperate to get rid of the oil production. On the other hand, oil buyers realize that the economic perspectives in the world look bad. If the economic activity suddenly stops, then we expect a lower demand for fuel including oil as firms are producing less. Then, producers want to sell and buyers are not interested to buy as they do not need as countries have more than enough inventories. This could go to the extreme in which producers are (sadly) willing to pay people in exchange of taking the oil out of their hands. This is why commodity prices can be negative.

There are other explanations, for example one related to the maturity of oil futures contracts. The price that went negative on Monday 2020-04-20 was for futures contracts to be delivered in May. Those contracts expired on Tuesday 2020-04-21. Upon expiration of the futures contract, the clearinghouse matches the holder of a long contract against the holder of a short position. The short position delivers the underlying asset to the long position. So, on Monday, traders — who were not equipped to take physical deliveries — were rushing to sell them to buyers who have booked storage.

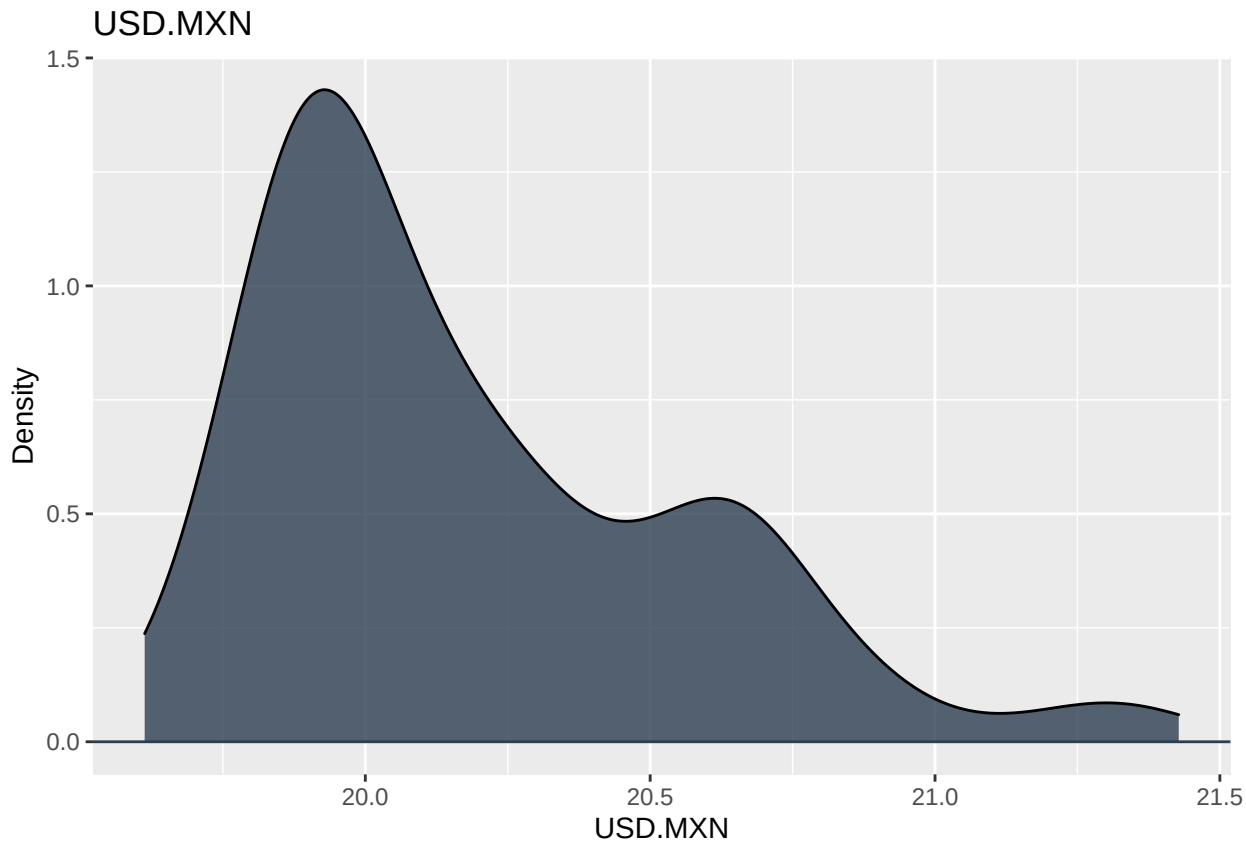
The *quantmod* package can retrieve exchange rates easily.

```
# Load the package.
library(quantmod)
# Download USD/MXN exchange rate from Oanda site.
exchange_rate <- getSymbols("USD/MXN", src = "oanda",
                            auto.assign = FALSE)
# Plot the results.
plot(exchange_rate)
```



Here, 1 USD equals 19.82952 MXN as 2021-07-05. We can show the time series information into a density plot.

```
# Density plots.
ggplot(exchange_rate, aes(x = USD.MXN, fill = "")) +
  geom_density(alpha = 0.8) +
  geom_hline(yintercept = 0, color = palette_light()[[1]]) +
  labs(title = "USD.MXN", x = "USD.MXN", y = "Density") +
  theme(legend.position = "none", legend.title = element_blank()) +
  scale_fill_tq()
```



We can conveniently download the data directly from the FRED API. Let's see data of "Beer, Wine, and Distilled Alcoholic Beverages Sales". For the full database details see:

https://fred.stlouisfed.org/series/S4248SM144NCEN

```
# Beer, Wine, Distilled Alcoholic Beverages, in Millions USD
beer_sales_tbl <- tq_get("S4248SM144NCEN", get = "economic.data",
                         from = "2010-01-01", to = "2021-01-01")
```

Let's have a look at the data set. By default, it says *price*, but these are basically sales figures. According to the main FRED reference, these are in millions of dollars, not seasonally adjusted.

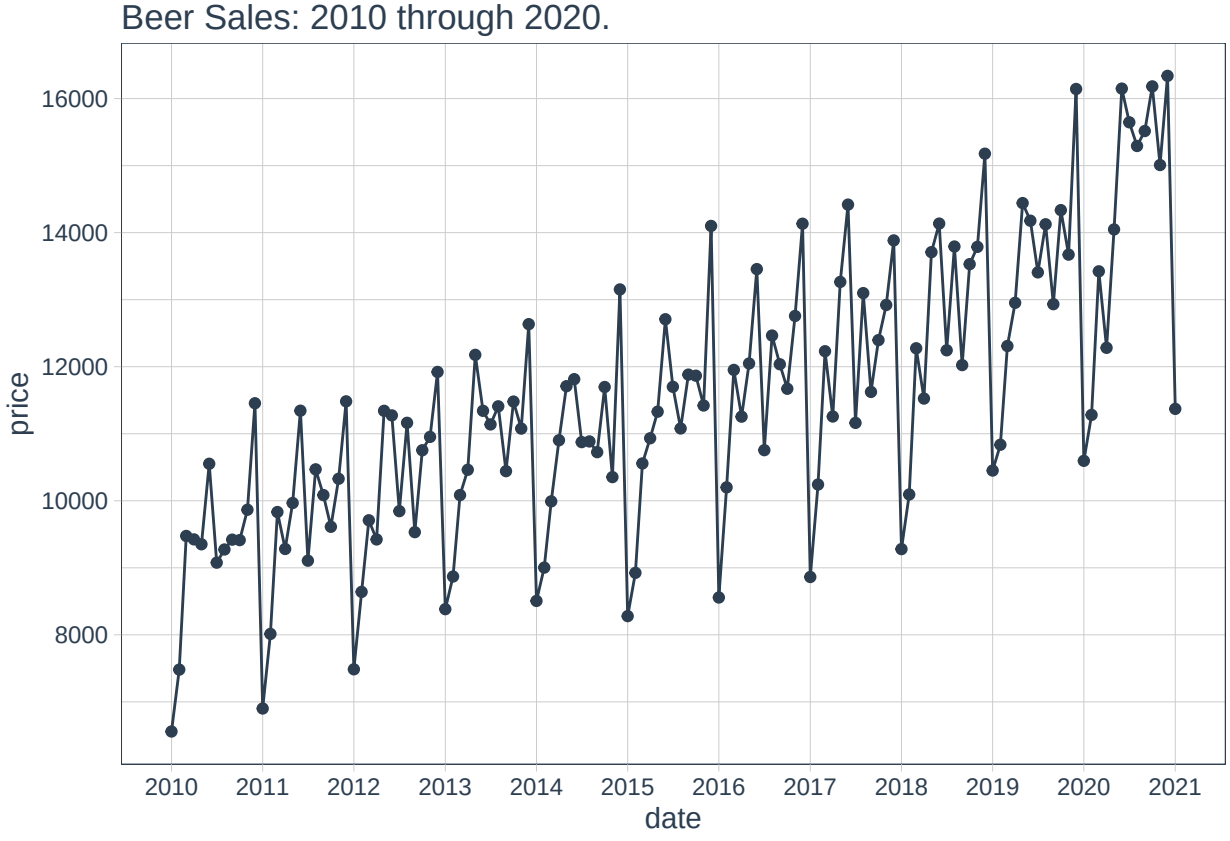
```
# See part of the data.
glimpse(beer_sales_tbl)
```

```
## Rows: 133
## Columns: 3
## $ symbol <chr> "S4248SM144NCEN", "S4248SM144NCEN", "S4248SM144NCEN", "S4248SM1~
## $ date   <date> 2010-01-01, 2010-02-01, 2010-03-01, 2010-04-01, 2010-05-01, 20~
```

```
## $ price  <int> 6558, 7481, 9475, 9424, 9351, 10552, 9077, 9273, 9420, 9413, 98~
```

Visualization is particularly important for time series analysis and forecasting. It is also important to see the time series because normally the models will perform better if we can identify time series basic characteristics such as trend and seasonality. This data set clearly has a trend and a seasonality.

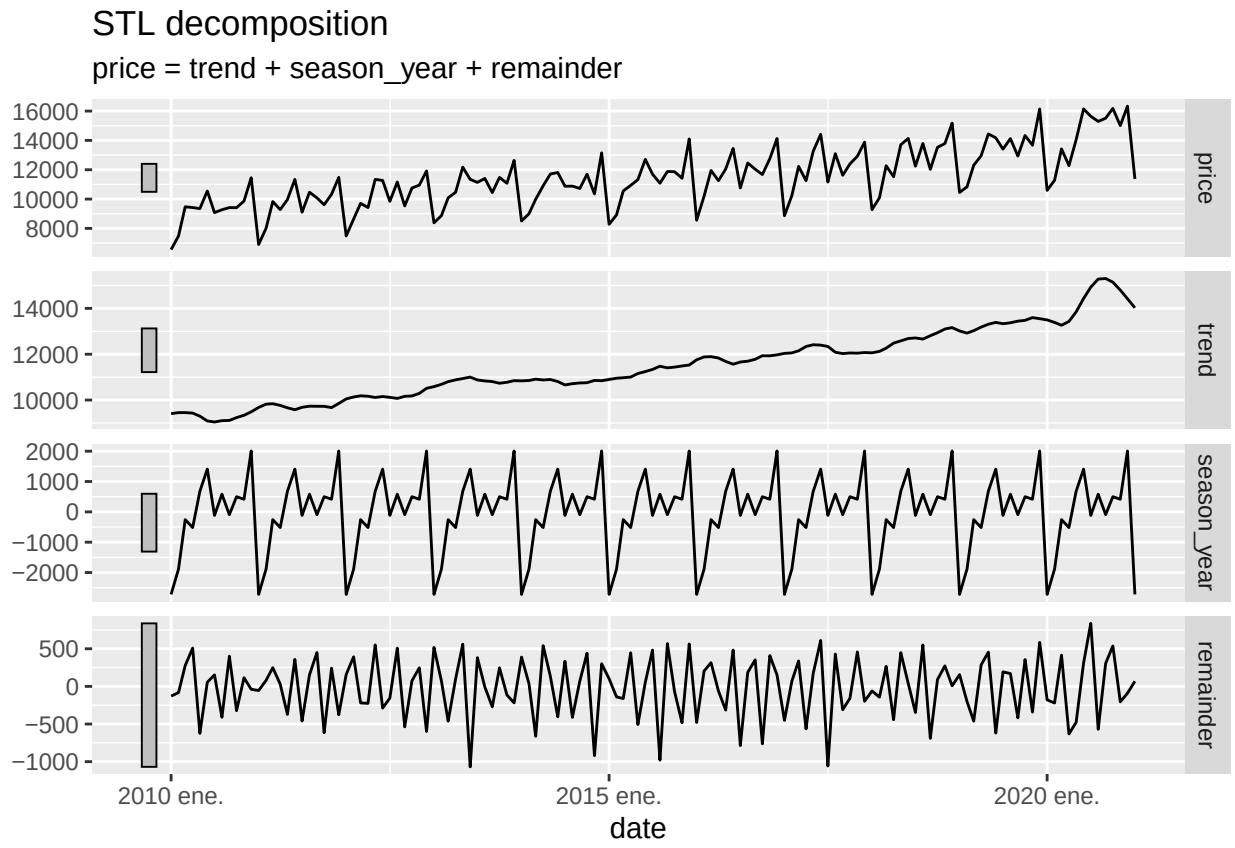
```
# Plot the data.
beer_sales_tbl |>
  ggplot(aes(date, price)) +
  geom_line(col = palette_light()[1]) +
  geom_point(col = palette_light()[1]) + theme_tq() +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Beer Sales: 2010 through 2020.")
```



A time series can be decomposed as shown below.

```
library(fpp3)


stl_dcmp <- beer_sales_tbl |>
```

```
  mutate(date = yearmonth(date)) |>
  as_tsibble(index = date)

stl_dcmp |>
  model(STL(price ~ trend(window = 7) +
              season(window = "periodic"), robust = TRUE)) |>
  components() |>
  autoplot()
```



We can also create some interesting visualizations about US employment and the US recessions (in the shaded area) over time. The code below is not so compact but it works.

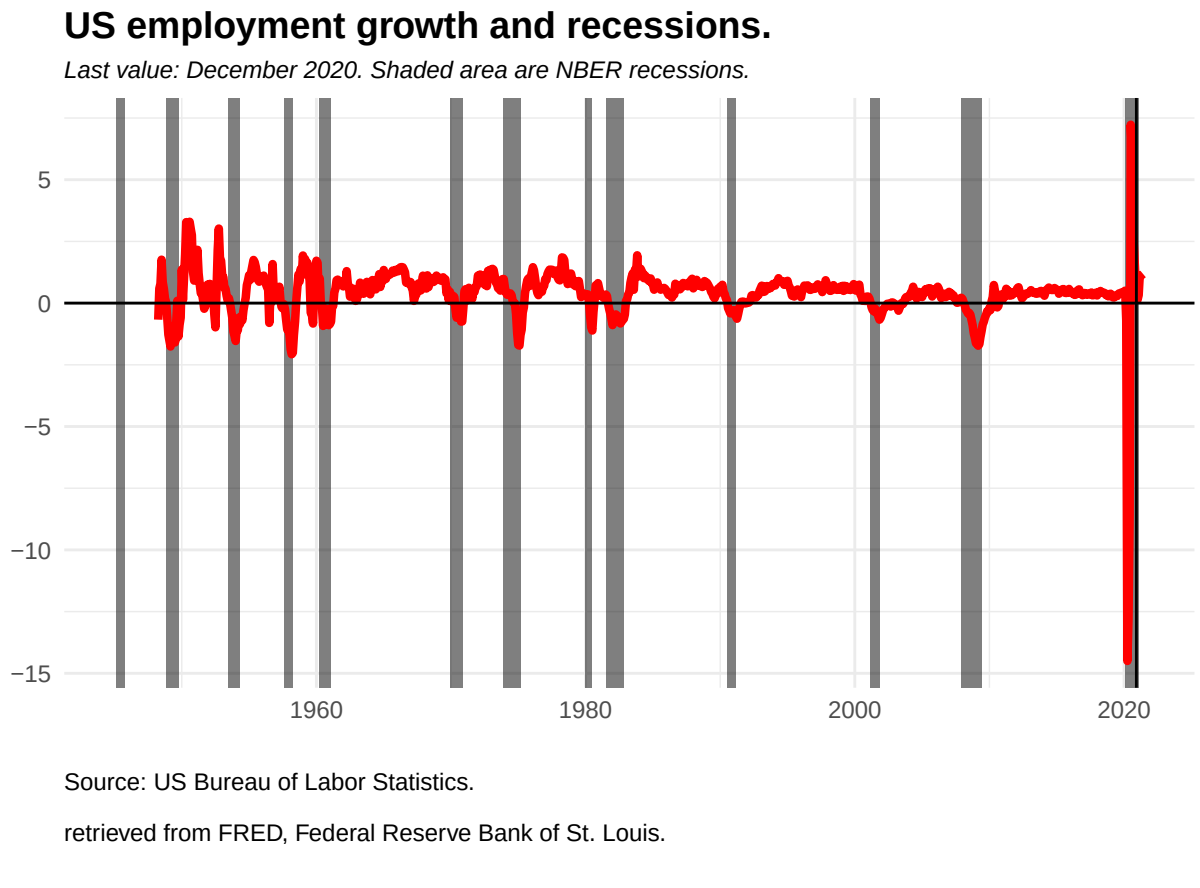
```
# NBER Recession indicator and US nonfarm payroll employment
tickers<- c("USREC", "PAYEMS")
df <- tq_get(tickers, get = "economic.data", from = "1948-01-01")
# recession df (for plotting)
recessions.df = read.table(textConnection(
  "Peak, Trough
```

```
  1945-02-01, 1945-10-01
  1948-11-01, 1949-10-01
  1953-07-01, 1954-05-01
  1957-08-01, 1958-04-01
  1960-04-01, 1961-02-01
  1969-12-01, 1970-11-01
  1973-11-01, 1975-03-01
  1980-01-01, 1980-07-01
  1981-07-01, 1982-11-01
  1990-07-01, 1991-03-01
  2001-03-01, 2001-11-01
  2007-12-01, 2009-06-01
  2020-02-01, 2021-01-30"), sep = ',',
  colClasses = c('Date', 'Date'), header = TRUE)
rec3 <- filter(df, df$symbol == "PAYEMS")
my_trans <- function(in.data,transform = "pctdiff3") {
  switch(transform, logdiff  = c(NA, diff(log(in.data))),
         pctdiff3 = 100 * Delt(in.data, k = 3),
        logdiff3 = c(rep(NA, 3), diff(log(in.data), 3)))
}
vlist <- c("PAYEMS")
df2 <- df |> group_by(symbol) |>
  mutate(x = ifelse(symbol %in% vlist, my_trans(price), price))
df2 |> select(-price)  |> spread(symbol, x) |>
  mutate(REC12 = lead(USREC, 12)) -> df3
df41 <- filter(df3, year(date) > 1945)
```

We have the data, now we can plot.

```
ggplot(data = df41, aes(x = date, y = PAYEMS)) +
  geom_rect(data = recessions.df, inherit.aes = FALSE,
            aes(xmin = Peak, xmax = Trough,
                ymin = -Inf, ymax = +Inf),
            fill = 'black', alpha = 0.5) + theme_minimal() +
  geom_line(color = "red", size = 1.5) +
  labs(x = "", y = "",
       title = "US employment growth and recessions.",
```

```
      subtitle = "Last value: December 2020. Shaded area are NBER recessions.",
      caption = "Source: US Bureau of Labor Statistics.
      \nretrieved from FRED, Federal Reserve Bank of St. Louis.") +
geom_hline(yintercept = 0) +
geom_vline(xintercept = as.numeric(as.Date("2020-12-01")),
           type = 3) +
theme(plot.caption = element_text(hjust = 0),
      plot.subtitle = element_text(face = "italic", size = 9),
      plot.title = element_text(face = "bold", size = 14))
```



Here we can see the magnitude of the current and upcoming financial and economic crisis. The fall in the US employment is impressive as you can see. Note how recessions (gray bars) are very closely related with falls in the employment over time.

Given the current (2020) health crisis many sectors and firms in the US economy stopped or reduced operations as a way to reduce the spread of the virus. As firms produce less (or nothing), they sell less (or nothing). Firms face two kinds of costs, variable and fixed. Variable costs depend on production, so if production decreases the variable costs decreases

as well. The problem here is fixed costs because they never disappear, they are fixed and they have to be paid no matter what. Some firms might have the possibility to pay fixed costs with reduced or null revenues for a while (days, weeks, probably a little bit more), but definitely not for long. I believe it is clear what happen next. Some firms did not make it, they could not survive and they simply went bankrupt and definitely closed operations. These firms had employees and now they are unemployed. Even those firms who are still operating, in some cases they had to reduce the payroll, and some employees are now unemployed. Less aggregate production and more families without an income reduce the aggregate supply and demand. This is why this current-next economic crisis has no precedent.

Having easy access to financial and economic data is important to facilitate the data analysis. However, it is even more important to be able to manipulate the financial and economic data correctly to communicate facts and discover new insights to make better decisions.

2.3 Technical analysis.

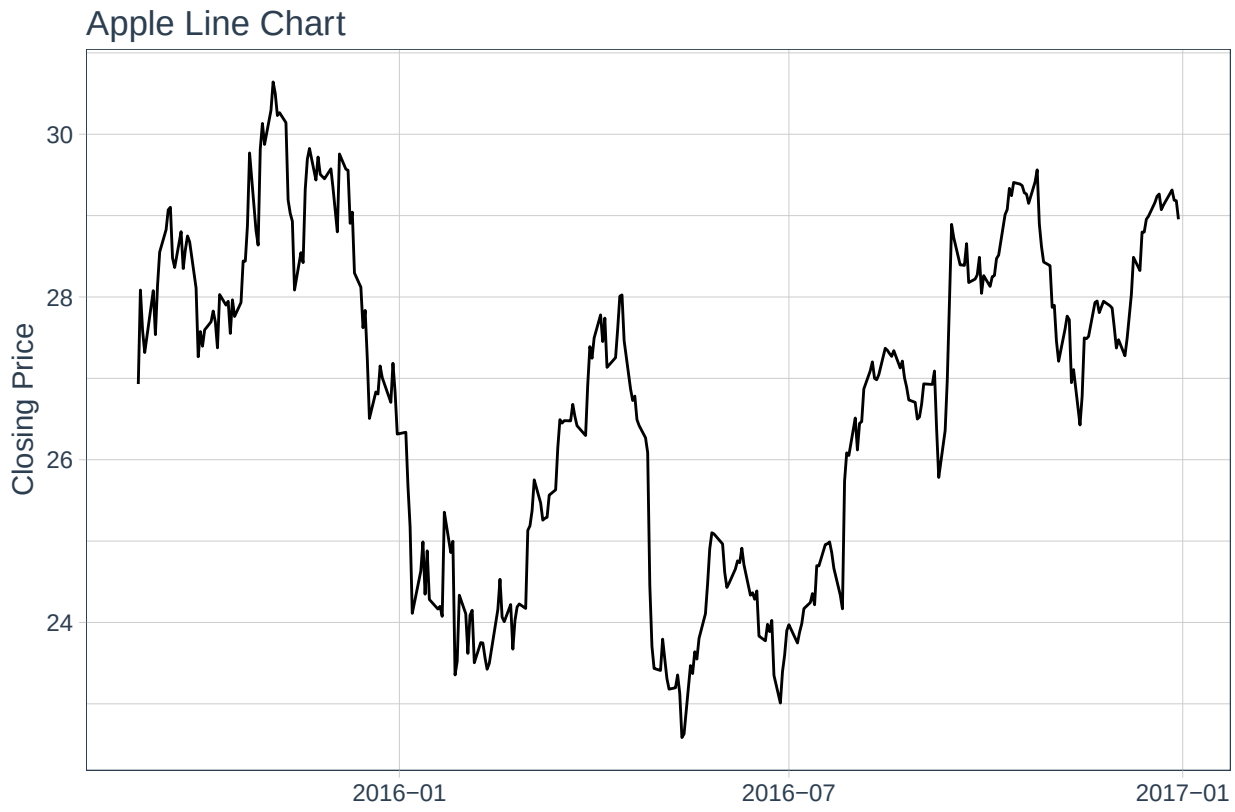
Technical analysis is the forecasting of future financial price movements based on an examination of past price movements and volume of trading. Technical analysis is applicable to stocks, indexes, commodities, futures or any tradable instrument where the price is influenced by the forces of supply and demand. This analysis is limited because it fails to incorporate other forces and factors that can influence the price of the security. Although this analysis is limited, it is still quite popular, and given that we now know how to get financial prices, it seems convenient to illustrate the basics of technical analysis.

Let's use FANG data again.

```
# Get AAPL and AMZN stock prices.
AAPL <- tq_get("AAPL", get = "stock.prices", from = "2015-09-01",
               to = "2016-12-31")
```

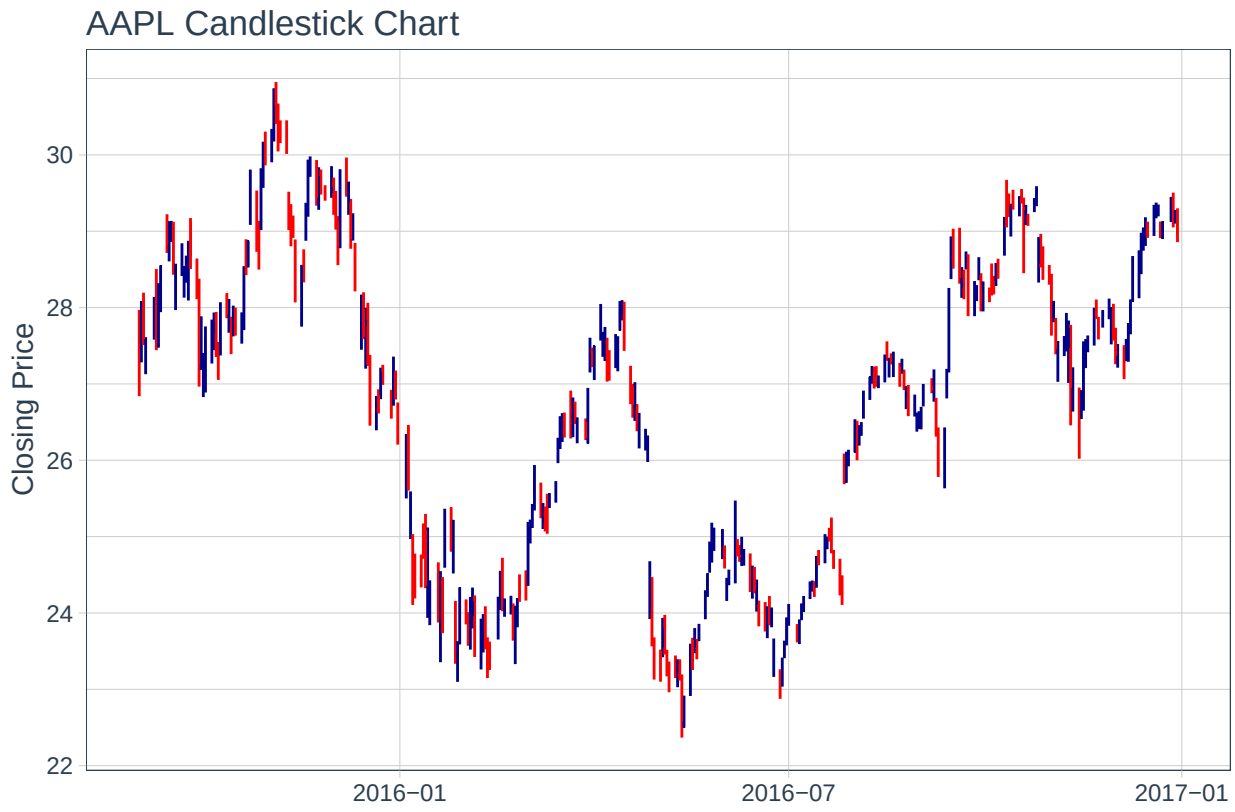
Let's begin with one plot.

```
# Plot the data.
AAPL |>
    ggplot(aes(x = date, y = close)) + geom_line() +
    labs(title = "Apple Line Chart",
         y = "Closing Price", x = "") +
    theme_tq()
```



The plot above is about closing prices. However, we also have open and close prices per day. We could take advantage of this by plotting vertical lines per day. The length of the daily vertical lines represents the difference between the open and close prices. To make it more informative, blue lines are those cases when the close price is greater than the open price, and red lines are those cases when the close price is lower than the open price.

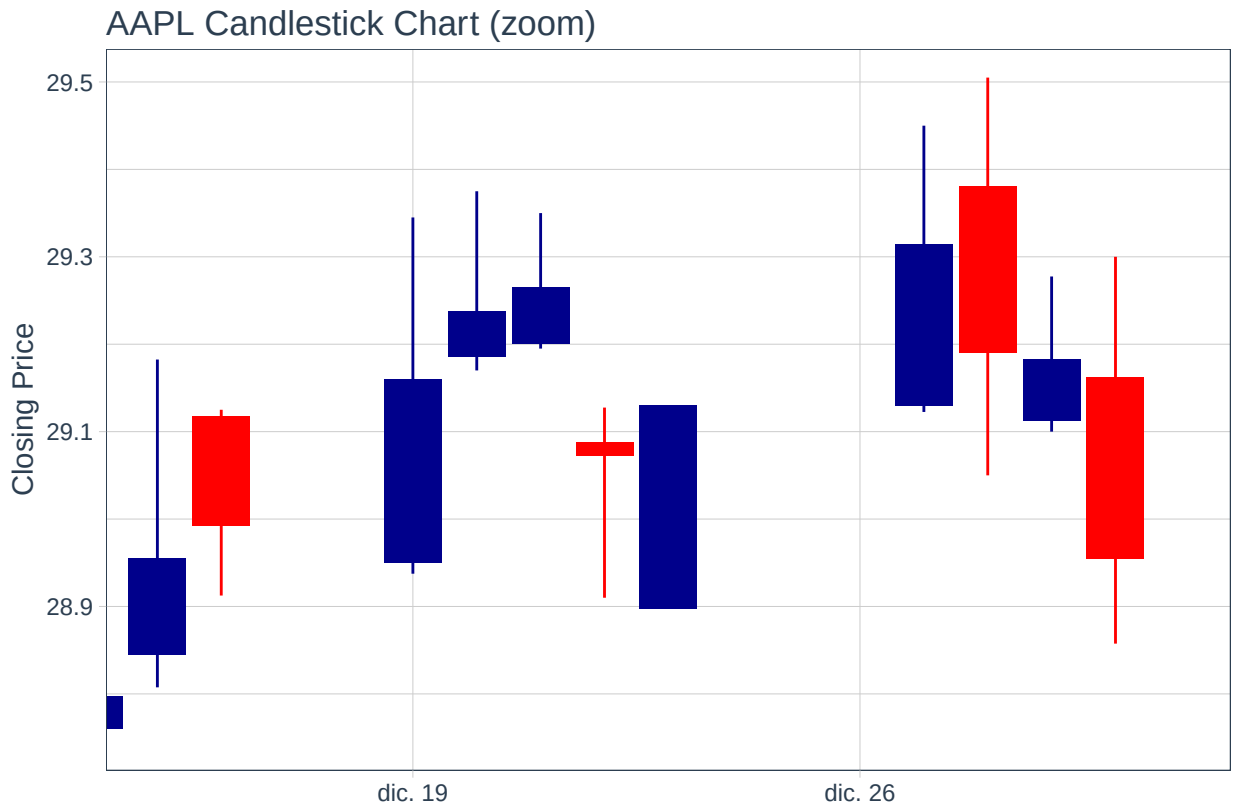
```
# Plot the data.
AAPL |>
    ggplot(aes(x = date, y = close)) +
    geom_candlestick(aes(open = open, high = high, low = low,
                         close = close)) +
    labs(title = "AAPL Candlestick Chart", y = "Closing Price",
         x = "") + theme_tq()
```



There are some blank spaces. This is simply because the close price of yesterday is not always exactly the same as the open price of today. Local stock markets close on weekends, and there are also holidays.

These kinds of charts are difficult to read when we have many observations. Let's do a zoom:

```
# A candlestick plot.
AAPL |>
    ggplot(aes(x = date, y = close)) +
    geom_candlestick(aes(open = open, high = high, low = low,
                         close = close)) +
  coord_x_date(xlim = c("2016-12-15", "2016-12-31"),
               ylim = c(28.75, 29.5)) +
    labs(title = "AAPL Candlestick Chart (zoom)",
         y = "Closing Price", x = "") + theme_tq()
```



A formal interpretation is difficult given the limitations of this analysis. However, the main idea is the following. Consider the stock is increasing quickly, then it is believed that before a fall in the price, the stock will increase at a lower speed. Big blue bars are eventually followed by smaller blue bars before exhibiting a price fall (a red bar).[3] The same logic can be applied the other way around. A dramatic fall in the price eventually reach a floor. This is, big red bars are eventually followed by smaller red bars before experiencing a price increase.

Could you guess what will happen with the price given the chart above? Seems difficult as the last four observations are consecutive blue-red-blue-red. Then, this indicator alone, without any complementary analysis, does not seem to deliver a clear signal.

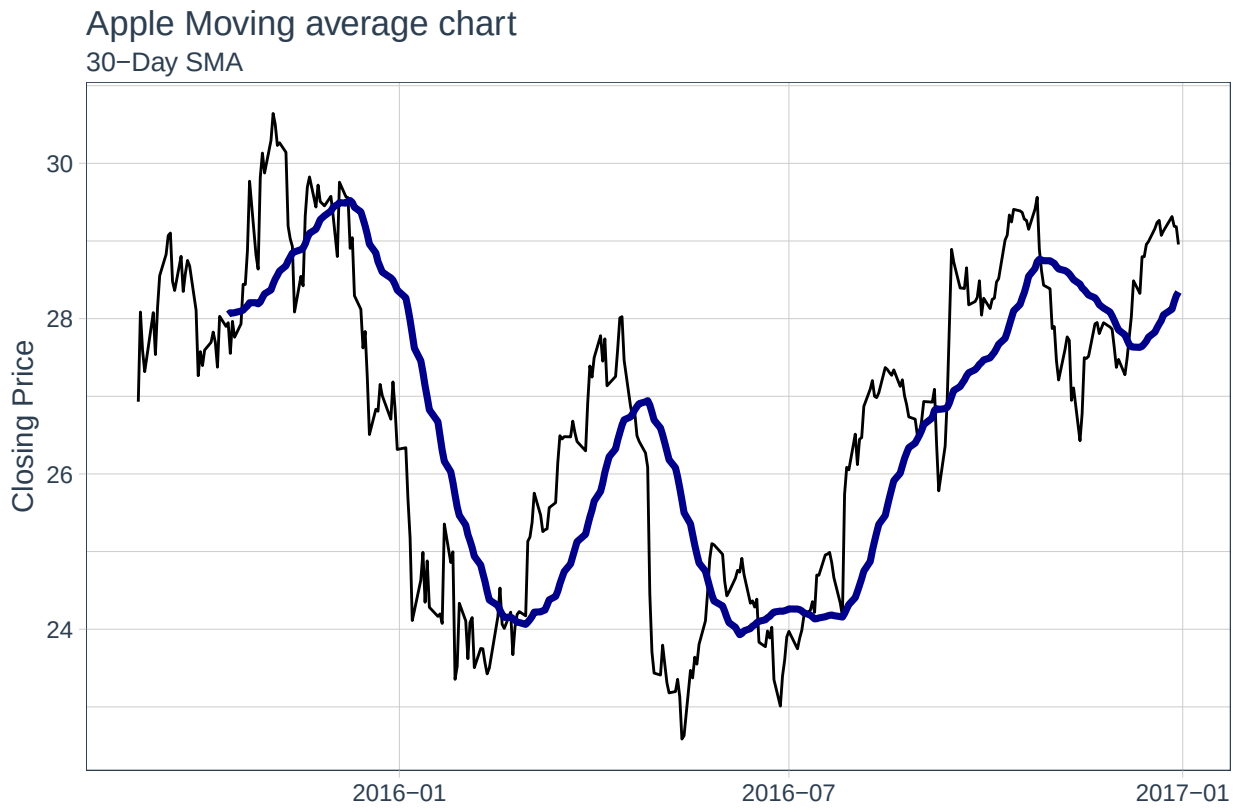
Moving averages are supposed to anticipate price movements. The idea is the following. We have two line plots, one is the original evolution of the price chart in black and the second is the moving average in blue. In this case we have a 30-day moving average. The blue line

[3]The logic behind this is not very different from what we have heard about "flatten the curve" in the context of the 2020 pandemic. As the curve increases, the high speed of increase is captured by big vertical blue bars. It is expected that the speed of increase slows down before the curve is flat, and this slowing down is captured by smaller vertical blue lines. So, before the curve is flat and starts to decrease, the blue vertical lines become smaller and smaller.

is simply the average of the last 30-day stock prices. In the code below, this value is easily modified by changing the value of n, in this case $n = 30$. The blue line (moving average) seems to anticipate what happens to the black line. The way we interpret the moving average is simple. If the black line crosses the blue from top to down in t, then we are expected to anticipate a drop in the stock price in $t + 1$. In case the black line remains below the blue after $t + 1$, then we expect the stock price to keep falling in the following periods.

If the black line crosses the blue from down to top, then we are supposed to anticipate an increase in the stock price. The sign is stronger when the black line crosses the blue line showing a steeper slope. What happens in practice is that the value of n should be calibrated until we fit the moving average with the behaviour of the price stock. This indicator alone is far from being perfect as there might be a lot of contrary signals which might make it difficult to follow the interpretation as we explained above.

```
# The simple moving average (SMA) plot.
AAPL |>
    ggplot(aes(x = date, y = close)) +
    geom_line(aes(open = open, high = high, low = low,
                        close = close))  +
    geom_ma(ma_fun = SMA, n = 30, linetype = 1, size = 1.25) +
    labs(title = "Apple Moving average chart",
         subtitle = "30-Day SMA",
         y = "Closing Price", x = "") + theme_tq()
```



There is a price fall a few weeks before January 2016 and a few weeks after January 2016. This price fall was correctly anticipated by the moving average. Then the stock price increased a bit more than 110 USD, this was also correctly anticipated by the moving average. The period around July 2016 is less clear. By the end of the time-series, the black line is above the blue, so trying to follow our previous explanation this means that we should not expect a significant drop in the stock price.

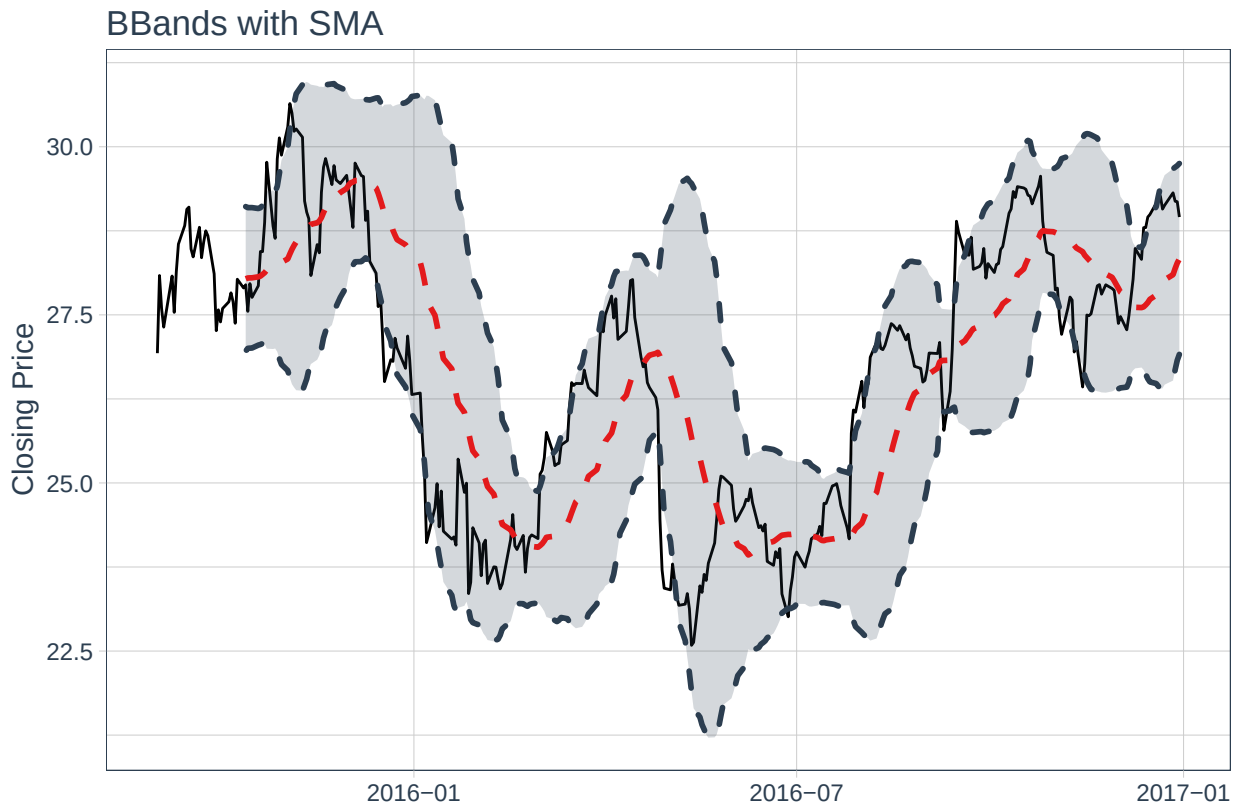
```
AAPL_recent <- tq_get("AAPL", get = "stock.prices",
                      from = "2016-12-30", to = "2017-01-12")
AAPL_recent
```

```
## # A tibble: 8 x 8
##   symbol date        open  high   low close    volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1 AAPL   2016-12-30  29.2  29.3  28.9  29.0 122345200     27.3
## 2 AAPL   2017-01-03  29.0  29.1  28.7  29.0 115127600     27.4
## 3 AAPL   2017-01-04  29.0  29.1  28.9  29.0  84472400     27.4
## 4 AAPL   2017-01-05  29.0  29.2  29.0  29.2  88774400     27.5
```

```
## 5 AAPL   2017-01-06  29.2  29.5  29.1  29.5 127007600     27.8
## 6 AAPL   2017-01-09  29.5  29.9  29.5  29.7 134247600     28.1
## 7 AAPL   2017-01-10  29.7  29.8  29.6  29.8  97848400     28.1
## 8 AAPL   2017-01-11  29.7  30.0  29.6  29.9 110354400     28.3
```

The Bollinger Bands are envelopes plotted at a standard deviation level above and below a simple moving average of the price. Because the distance of the bands is based on standard deviation, they are supposed to adjust to volatility swings in the underlying price. This indicator can help us to understand the size of the expected change in the future. If the Bollinger Bands become wider then we should expect drastic changes in the stock price.

```
# Bollinger bands and simple moving average.
AAPL |>
    ggplot(aes(x = date, y = close, open = open,
               high = high, low = low, close = close)) +
    geom_line() +
    geom_bbands(ma_fun = SMA, sd = 2, n = 30,
                linetype = 2, size = 1, alpha = 0.2,
                fill        = palette_light()[[1]],
                color_bands = palette_light()[[1]],
                color_ma    = palette_light()[[2]]) +
    labs(title = "BBands with SMA", y = "Closing Price", x = "") +
    theme_tq()
```

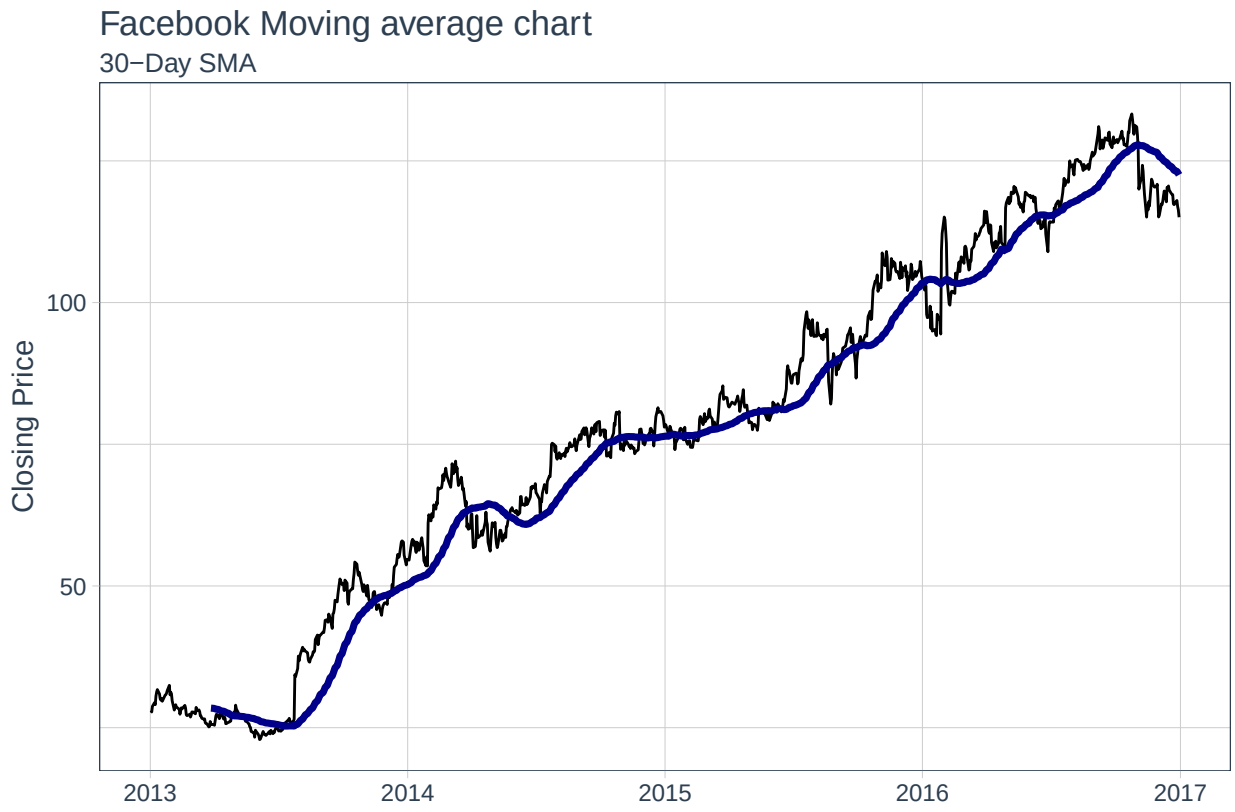


Let's look at the very last observation. The price line is above the moving average (now in red, dotted line). This means that we should not expect a drop in the stock price in the following periods. However, the Bollinger Bands suggest that the price is currently very close to the upper bound, plus the upper bounds is also close to the historical maximum price at least in this plot. In sum, the last price is 115.82 USD and we do not expect a fall in the stock price according to the moving average and the expected price change is in the range of 107 to 119 USD approximately.

In case you may have a different view or concerns about the previous interpretation then you now realize the pitfalls and the disadvantages of this kind of analysis when it is conducted as a simple chart read. Although, the technical analysis is popular, it could lead to different interpretations easily when we interpret one or two indicators by eye. This does not mean we have to ignore this kind of analysis because we can always incorporate formal techniques to try to validate buy and sell signals to help traders and conduct short-run investment strategies. In fact, there are very serious developments like the *quantstrat* package, which provides a generic infrastructure to model and backtest signal-based quantitative strategies.

We can show the simple moving average for Facebook.

```
# The simple moving average (SMA) plot.
FANG |>
  filter(symbol == "FB") |>
  ggplot(aes(x = date, y = close)) +
  geom_line(aes(open = open, high = high, low = low,
                close = close))  +
  geom_ma(ma_fun = SMA, n = 60, linetype = 1, size = 1.25) +
  labs(title = "Facebook Moving average chart",
       subtitle = "30-Day SMA",
       y = "Closing Price", x = "") + theme_tq()
```



This suggests that the stock price will decrease in the near future. Let's see if the moving average signal is correct.

```
FANG |>
  filter(symbol == "FB") |>
  filter(date >= "2014-01-01" & date <= "2014-02-01")
```

```
## # A tibble: 21 x 8
```

```
##    symbol date        open  high   low close   volume adjusted
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
##  1 FB     2014-01-02  54.8  55.2  54.2  54.7 43195500     54.7
##  2 FB     2014-01-03  55.0  55.7  54.5  54.6 38246200     54.6
##  3 FB     2014-01-06  54.4  57.3  54.0  57.2 68852600     57.2
##  4 FB     2014-01-07  57.7  58.5  57.2  57.9 77207400     57.9
##  5 FB     2014-01-08  57.6  58.4  57.2  58.2 56682400     58.2
##  6 FB     2014-01-09  58.7  59.0  56.7  57.2 92253300     57.2
##  7 FB     2014-01-10  57.1  58.3  57.1  57.9 42449500     57.9
##  8 FB     2014-01-13  57.9  58.2  55.4  55.9 63010900     55.9
##  9 FB     2014-01-14  56.5  57.8  56.1  57.7 37503600     57.7
## 10 FB     2014-01-15  58.0  58.6  57.3  57.6 33663400     57.6
## # ... with 11 more rows
```

The moving average sent the right stock price movement signal.

3 Asset returns.

An asset is any resource owned by a business or an economic entity. It is anything that can produce value (or returns) in the future. Here, we frequently use firm stocks as financial assets, but we also show other examples of assets such as commodities (oil), currencies, bonds, derivatives, etc.

A return is the percentage change of an asset price. In finance, we are very interested in asset returns because it allows us to measure asset (like stocks) performance in relative terms. Consider the following example. In (1) I buy at 100 USD and sell at 110 USD; and in (2) I buy at 10 USD and sell at 12 USD. I make money in both situations, that is for sure because $110 > 100$ and $12 > 10$. But which situation was the best deal? In absolute terms, I made 10 USD in (1) and only 2 USD in (2). However, in relative terms I made 10% in (1) and a great 20% in (2).

The previous returns were calculated as a simple percentage change: $(110 - 100)/100$ and $(12 - 10)/10$ respectively. We can calculate log-returns instead and these are equivalent, and sometimes we prefer to calculate log-returns especially when we deal with short periods of time. In this case the log-returns are $log(110/100)$ and $log(12/10)$ which are 9.53% and 18.23% respectively.

Both absolute and relative valuations (10 USD versus 2 USD, and 10% versus 20%) are important and valid depending on what we want to report. However, the good thing about the relative approach (percentages) is that we can compare among different alternatives more easily. Returns also have some relevant statistical properties (like stationarity) that are needed to implement some quantitative financial and econometrics models.

Even returns can be expressed in relative terms with respect to the associated asset risk. Consider another example. In (1) I have an asset with an expected return of 5% with a risk of 5%; and in (2) a different asset with an expected return of 5% with a risk of 10%. So, in relative terms (1) is an asset with 1 unit of return per unit of risk and (2) is an asset with 0.5 unit of return per unit of risk. Now, it is clear we should prefer alternative (1).

There are more technical reasons to work with returns, but we can proceed with our analysis. Also, there are other ways to measure and report asset returns including different frequencies, cumulative returns, compounded returns, etc., but that is something we can discuss later.

3.1 From prices to returns.

We need time series of asset prices to calculate time series of asset returns. Let's begin with a visual representation of the FANG database. Here, we have daily stock prices per stock.

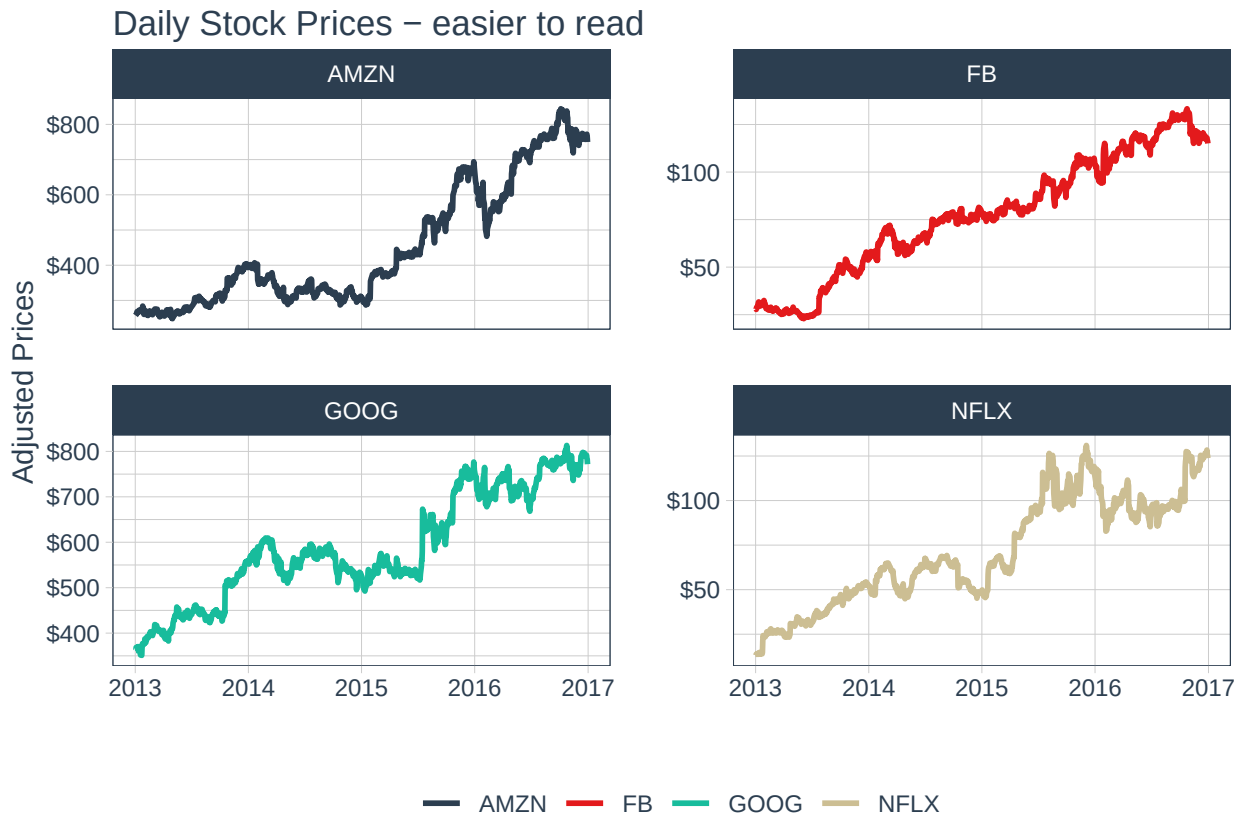
```
# Plot the data.
FANG_daily_all <- FANG |>
    group_by(symbol) |>
    ggplot(aes(x = date, y = adjusted, color = symbol)) +
    geom_line(size = 1) +
    labs(title = "Daily stock prices - not so easy to read.",
         x = "", y = "Adjusted prices", color = "") +
    scale_y_continuous(labels = scales::dollar) +
    theme_tq() + scale_color_tq()
FANG_daily_all
```

See how stock prices fluctuate in different ranges. It would be a mistake to prefer Amazon or Google simply because they experience larger price changes compared with Facebook or Netflix as we discuss in the previous section. Taking investment decisions based on prices

could be misleading because investors are most of the time looking for returns, and in particular for an attractive risk-return combination. The vertical axis is also problematic in the plot as it is difficult to see the price evolution clearly for each stock. In sum, we do not take investment decisions based on this kind of visual representation. If we are interested to evaluate the stock performance, we need a risk-return analysis, not a price chart.

Let's add one line of code to see the price time-series more clearly.

```
# Facet makes it easier to read.
FANG_daily <- FANG |>
    group_by(symbol) |>
    ggplot(aes(x = date, y = adjusted, color = symbol)) +
    geom_line(size = 1) +
    labs(title = "Daily Stock Prices - easier to read",
         x = "", y = "Adjusted Prices", color = "") +
    facet_wrap(~ symbol, ncol = 2, scales = "free_y") +
    scale_y_continuous(labels = scales::dollar) +
    theme_tq() + scale_color_tq()
FANG_daily
```



Now it is clear that all stock prices have a positive trend over time. It is also clear that 2014 was a bad year for Amazon, Google and even Netflix. This is because the stock price decreased in the period of 2014-2015. The year 2015 was good for Amazon and Google. The main point is that this new visualization of the data can help us to compare the evolution of price per stock. Remember we still have prices and not returns. Stock returns are easily calculated in R.

Here, we first select adjusted prices, then transform into yearly returns.

```
# Create yearly returns.
FANG_annual_returns <- FANG |>
    group_by(symbol) |>
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "yearly",
                 type       = "arithmetic")
FANG_annual_returns
```

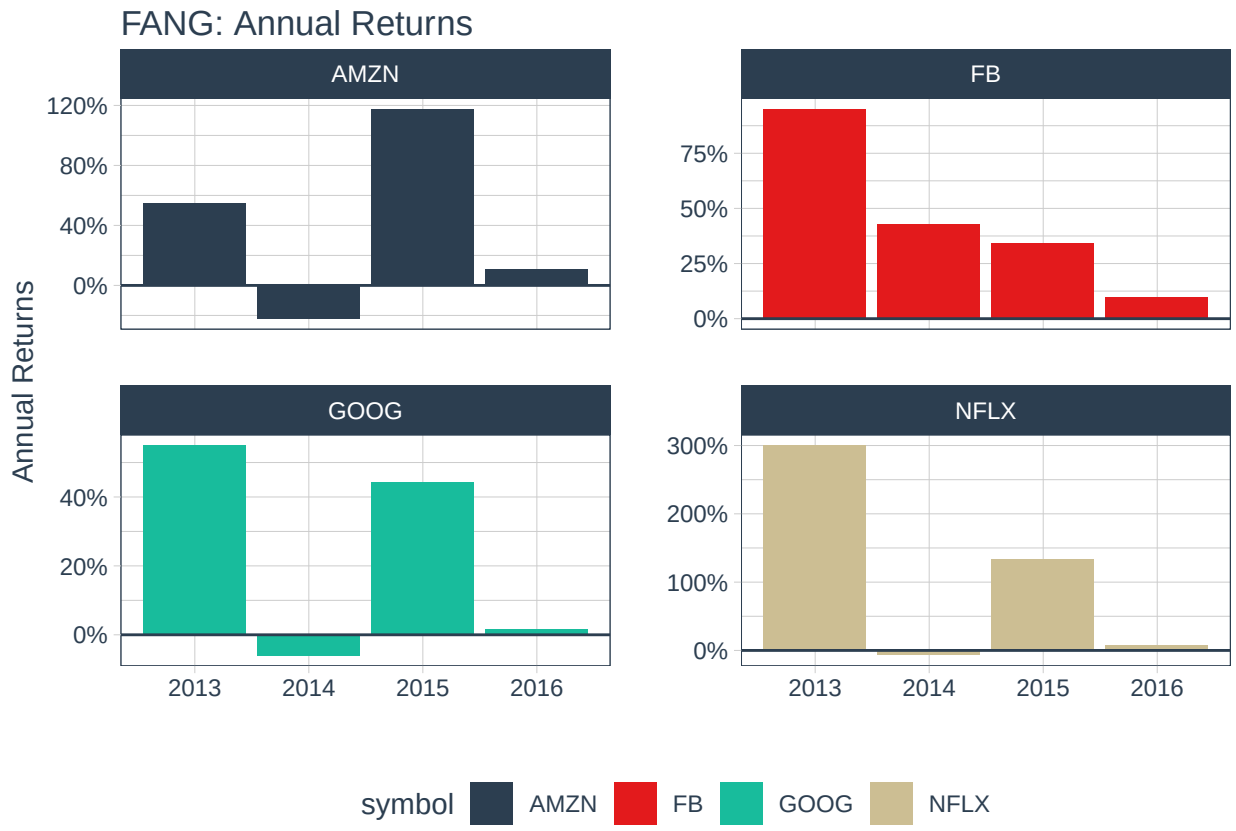
```
## # A tibble: 16 x 3
```

```
## # Groups:   symbol [4]
##    symbol date       yearly.returns
##    <chr>  <date>              <dbl>
##  1 FB     2013-12-31         0.952
##  2 FB     2014-12-31         0.428
##  3 FB     2015-12-31         0.341
##  4 FB     2016-12-30         0.0993
##  5 AMZN   2013-12-31         0.550
##  6 AMZN   2014-12-31        -0.222
##  7 AMZN   2015-12-31         1.18
##  8 AMZN   2016-12-30         0.109
##  9 NFLX   2013-12-31         3.00
## 10 NFLX   2014-12-31        -0.0721
## 11 NFLX   2015-12-31         1.34
## 12 NFLX   2016-12-30         0.0824
## 13 GOOG   2013-12-31         0.550
## 14 GOOG   2014-12-31        -0.0597
## 15 GOOG   2015-12-31         0.442
## 16 GOOG   2016-12-30         0.0171
```

This is what we call tidy data because (1) each variable forms a column; (2) each observation forms a row; and (3) each type of observational unit forms a table. These yearly returns are expressed in decimal notation. This means that 0.95 is in reality 95%.

The next chunk of code shows how to visualize the annual returns in a nice plot.

```
# Plot the annual returns.
FANG_annual_returns |>
    ggplot(aes(x = date-365, y = yearly.returns, fill = symbol)) +
    geom_col() +
    geom_hline(yintercept = 0, color = palette_light()[[1]]) +
    scale_y_continuous(labels = scales::percent) +
    labs(title = "FANG: Annual Returns",
         y = "Annual Returns", x = "") +
    facet_wrap(~ symbol, ncol = 2, scales = "free_y") +
    theme_tq() + scale_fill_tq()
```



Risky stocks like these are risky because the returns change drastically from time to time. Note that here, it is easy to see that 2014 was indeed a bad year for Amazon, Google and Netflix as we anticipate when looking at the price chart. These three assets confirm what we anticipate before because we see negative returns in 2014. The year 2015 was good for Amazon and Google as we said before.

The question that arises now is: What would be the result of investing money in each stock during these years (starting the first day of 2013 and finishing the last day of 2016)? Yearly returns show what happens from one year to another, but we cannot tell which stock would represent the best investment strategy in these years looking at this past information. Cumulative returns can be useful to answer this question.

3.2 Cumulative returns.

Let's start with a brief note about working with quantities affected by percentage changes. My salary is 100 USD and I manage to get a 5% increase. Then my new salary is $100 \times (1 + 0.05) = 105$. If I do $100 \times 0.05 = 5$ I only get the net increase, but what I care about is my new salary. Now I negotiate to change from full-time to part-time and agree to a salary

decrease of 20%. Then, my current salary is $100 \times (1 + 0.05) \times (1 - 0.2) = 84$.

Imagine we invest 100 USD in Amazon for this period. Let's calculate the value of our investment considering that we receive the four yearly returns listed and shown above. Remember the Amazon's yearly returns from 2013 to 2016 are: 54.98%, −22.18%, 117.78%, 10.95% in this specific order.

```
# Investment in USD.
investment = 100
# Amazon's cumulative return in percentage.
AMZN_cum_percentage = (1+0.54984265)*(1-0.22177086)*
  (1+1.17783149)*(1+0.10945565) - 1
# Amazon's cumulative return in USD.
AMZN_cum_usd = investment * (1 + AMZN_cum_percentage)
# Show results.
AMZN_cum_percentage
```

```
## [1] 1.914267
```

```
AMZN_cum_usd
```

```
## [1] 291.4267
```

The dollar value of 100 USD invested in Amazon at the end of the 4 years is 291.4267 USD. The cumulative return is 191.4267%. For the sake of clarity, we can check these values are correct by applying this return over the 100 USD: $100 \times (1 + 1.914267) = 291.4267$.

The 191.4267% cumulative return is usually not reported as it is here because it accumulates 4 years of returns. We normally express mean returns, in this case mean yearly return. The arithmetic mean of the returns, calculated by taking the sum of the returns and dividing by 4 is:

```
# Arithmetic mean return.
AMZN_arith_mean_retun = (0.54984265 - 0.22177086 +
                           1.17783149 + 0.10945565) / 4
AMZN_arith_mean_retun
```

```
## [1] 0.4038397
```

Let's reveal a problem with returns calculated as an arithmetic mean that it is sometimes disregarded. In our example, the arithmetic mean of returns is 40.38397% per year. Note

that this could be misleading if we calculate the growth of a 100 USD investment for four years:

```
# The long way.
investment * (1 + AMZN_arith_mean_retun) *
  (1 + AMZN_arith_mean_retun) *
  (1 + AMZN_arith_mean_retun) * (1 + AMZN_arith_mean_retun)
```

```
## [1] 388.3919
```

```
# The short way.
investment * (1 + AMZN_arith_mean_retun) ^ 4
```

```
## [1] 388.3919
```

See something strange? We know that the dollar value of 100 USD invested in Amazon at the end of the 4 years is 291.4267 USD. However, taking this mean arithmetic return we have 388.3919 USD. See how problematic this could be. I can say my 100 USD investment in Amazon during the past 4 years was great because I get a mean return of 40.38397% per year. People might think that my money grew from 100 USD to 388.3919 USD by the end of the fourth year, whereas in reality my money grew from 100 USD to 291.4267 USD.

The way we reconcile this difference is by calculating a geometric mean return rather than an arithmetic mean return. A geometric mean is the nth root of the product of n numbers. In particular:

```
# Geometric mean return.
AMZN_geom_mean_retun = ((1+0.54984265)*(1-0.22177086)*
  (1+1.17783149)*(1+0.10945565)) ^ (1/4) - 1
AMZN_geom_mean_retun
```

```
## [1] 0.3065689
```

I can say my 100 USD investment in Amazon during the past 4 years was great because I get a mean (geometric) return of 30.65689% per year. People might think that my money grew from 100 USD to 291.4267 USD by the end of the fourth year, and this is correct. Let's verify the procedure.

```
# The long way.
investment * (1 + AMZN_geom_mean_retun) *
  (1 + AMZN_geom_mean_retun) *
  (1 + AMZN_geom_mean_retun) * (1 + AMZN_geom_mean_retun)
```

```
## [1] 291.4267
```

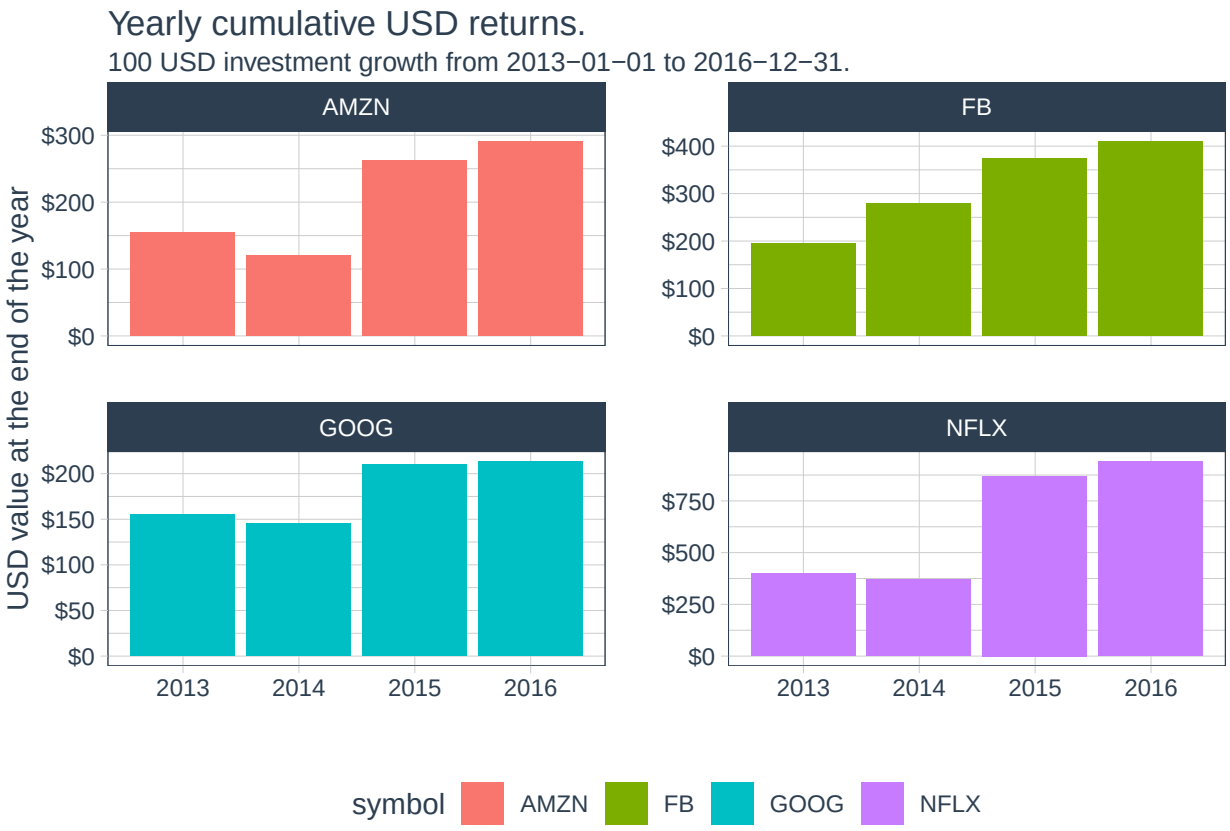
```
# The short way.
investment * (1 + AMZN_geom_mean_retun) ^ 4
```

```
## [1] 291.4267
```

This phenomenon is an example of a result that is well known in mathematics. The geometric mean of a set of numbers is always less than the arithmetic mean ($30.6 < 40.3$). Although our arithmetic mean return of 40.38397% can lead to an overestimation of the investment dollar returns, we usually report mean returns. This is a common issue when reporting mutual funds returns as it is tempting to report arithmetic mean returns because they are higher. This is why in some jurisdictions, regulations require fund managers to report geometric mean returns instead (30.65689% in this case).

For now, we will work with arithmetic returns. Let's see How 100 USD investment growth each year per stock. Note that the plotted values are in USD.

```
# Yearly returns.
FANG_annual_returns <- FANG |>
    group_by(symbol) |>
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "yearly",
                 type       = "arithmetic")
# Cumulative returns.
FANG_annual_cum_returns <- FANG_annual_returns |>
  mutate(cr = 100*cumprod(1 + yearly.returns)) |>
# Plot the results.
  ggplot(aes(x = date-365, y = cr, fill = symbol)) + geom_col() +
  labs(title = "Yearly cumulative USD returns.",
  subtitle = "100 USD investment growth from 2013-01-01 to 2016-12-31.",
  x = "", y = "USD value at the end of the year", color = "") +
  scale_y_continuous(labels = scales::dollar) +
  facet_wrap(~ symbol, ncol = 2, scales = "free_y") +
  theme_tq() + scale_color_tq()
FANG_annual_cum_returns
```

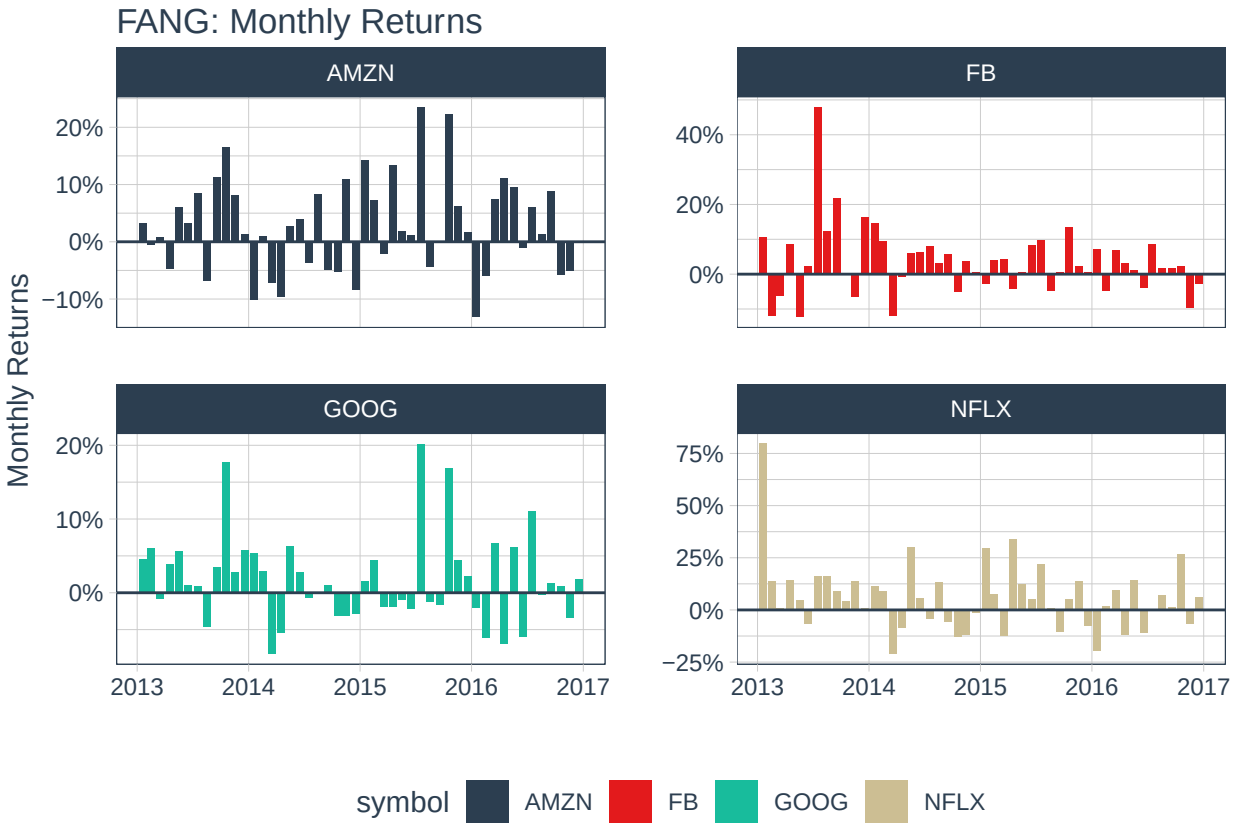


Please verify that the dollar value of 100 USD invested in Amazon at the end of the 4 years is 291.4267 USD, as we discussed before.

We can also modify the code above to show monthly returns (in percentage) instead of yearly.

```
# Monthly returns.
FANG_monthly_returns <- FANG |>
    group_by(symbol) |>
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "monthly",
                 type       = "arithmetic")
# Plot the results.
ggplot(FANG_monthly_returns, aes(x = date-12,
       y = monthly.returns, fill = symbol)) +
  geom_col() +
  geom_hline(yintercept = 0, color = palette_light()[[1]]) +
  scale_y_continuous(labels = scales::percent) +
```

```
  labs(title = "FANG: Monthly Returns", y = "Monthly Returns",
       x = "") +
  facet_wrap(~ symbol, ncol = 2, scales = "free_y") +
  theme_tq() + scale_fill_tq()
```

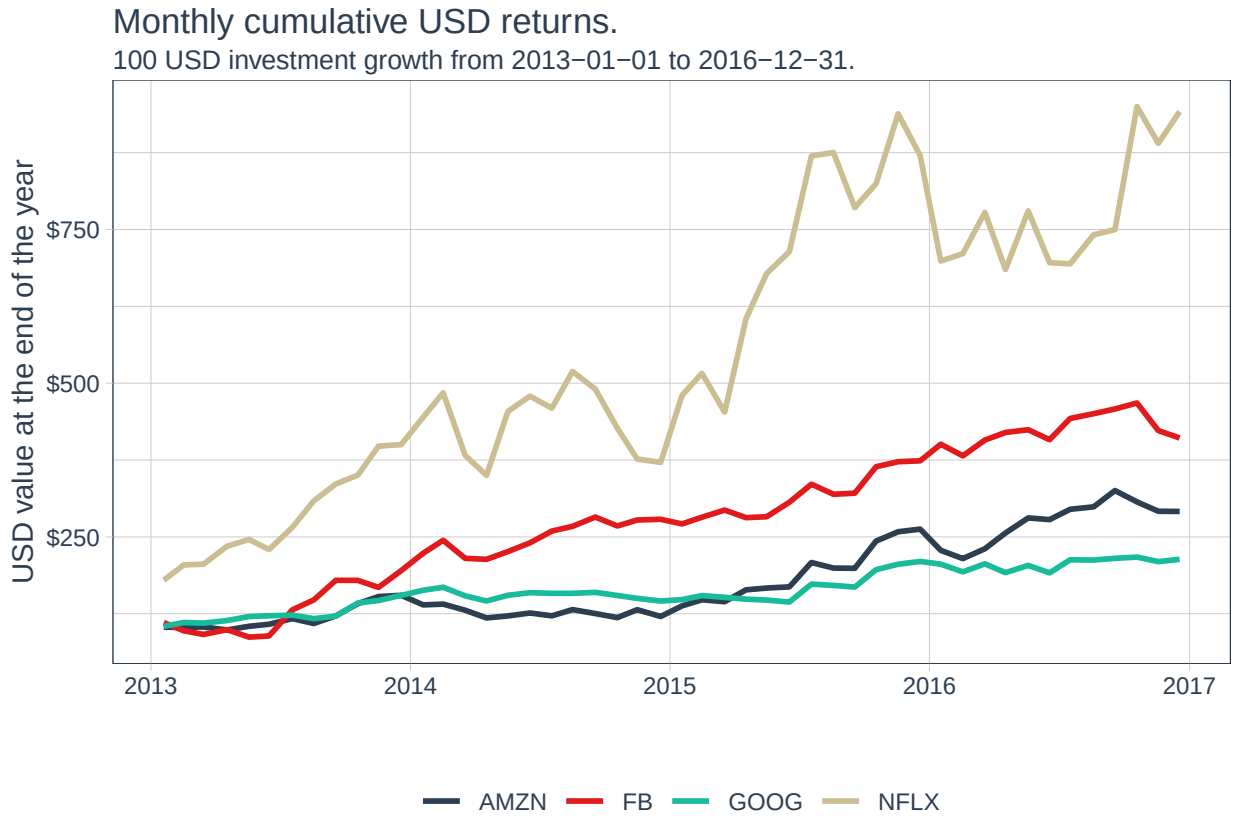


By increasing the frequency from yearly to monthly we now have 12 observations per month instead of 1. Note that this new plot reveals how volatile these stock returns are. There is a negative return just the month after GOOG reached a positive 20% return. It looks hard to anticipate what will happen the next month since there is evidence that these returns change from positive to negative quite frequently.

Let's see cumulative monthly returns per stock to extend our 100 USD investment example.

```
# Calculate monthly cumulative returns.
FANG_monthly_cum_returns <- FANG_monthly_returns |>
  mutate(cr = 100 * cumprod(1 + monthly.returns))
# Plot results.
ggplot(FANG_monthly_cum_returns, aes(x = date-12, y = cr,
                                     color = symbol)) +
```

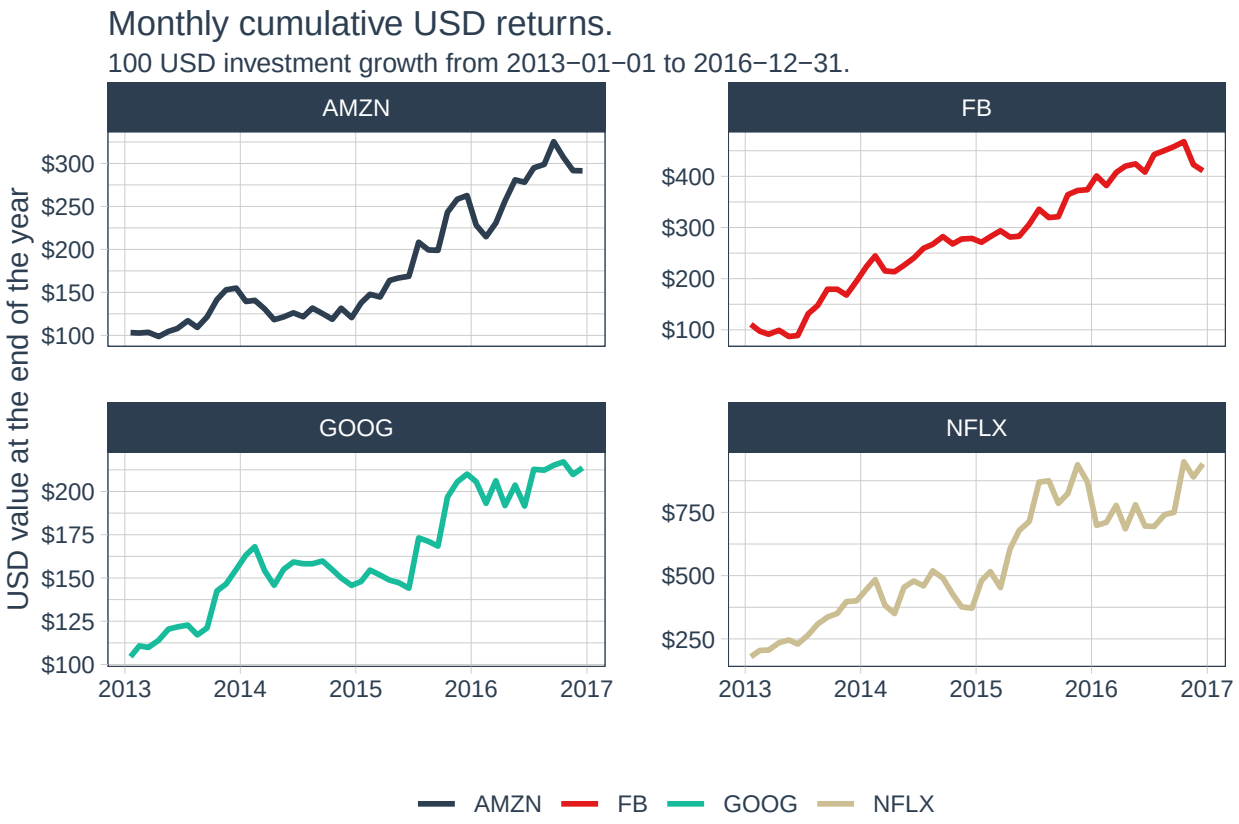
```
  geom_line(size = 1) +
  labs(title = "Monthly cumulative USD returns.",
subtitle = "100 USD investment growth from 2013-01-01 to 2016-12-31.",
         x = "", y = "USD value at the end of the year",
color = "") +
  scale_y_continuous(labels = scales::dollar) +
  theme_tq() + scale_color_tq()
```



Alternatively, we can split the plot in four panels. This facilitates the reading of the plot.

```
# Plot results.
ggplot(FANG_monthly_cum_returns, aes(x = date-12, y = cr,
                                     color = symbol)) +
  geom_line(size = 1) +
  labs(title = "Monthly cumulative USD returns.",
subtitle = "100 USD investment growth from 2013-01-01 to 2016-12-31.",
         x = "", y = "USD value at the end of the year",
color = "") +
```

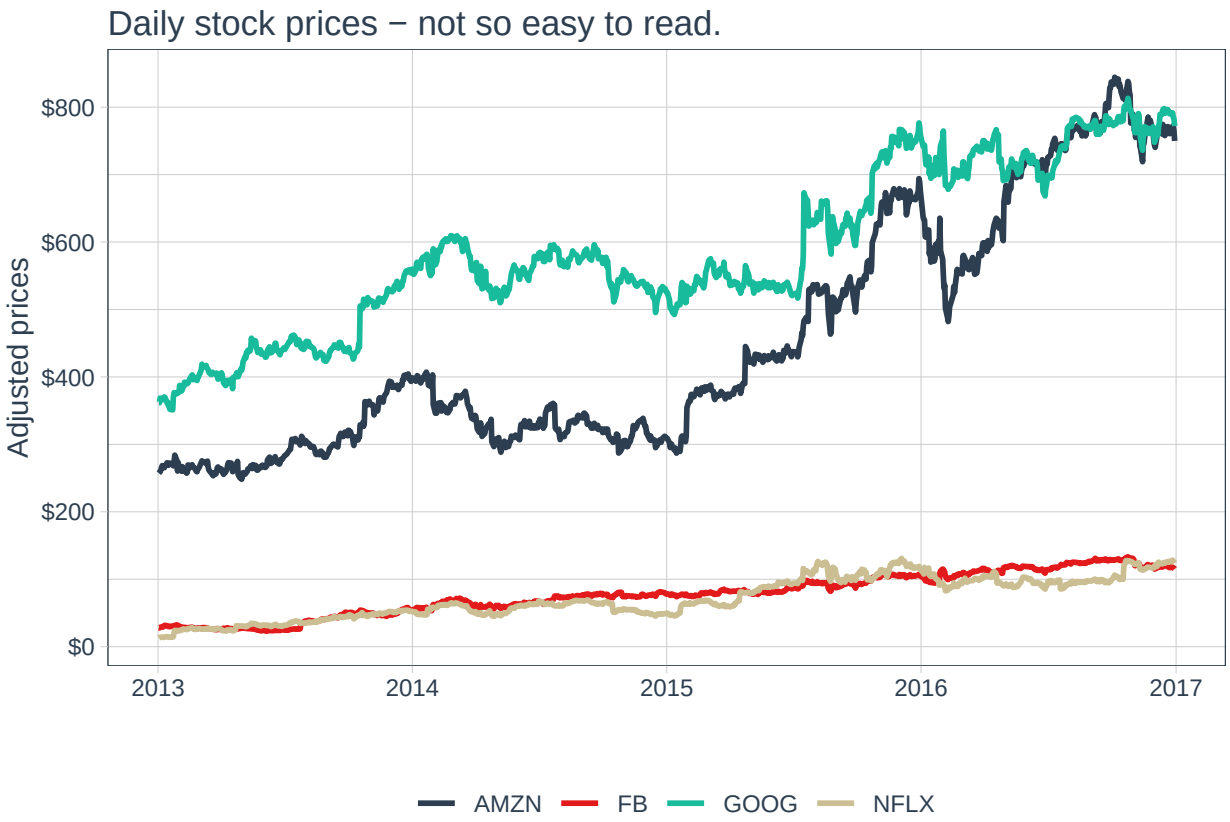
```
  scale_y_continuous(labels = scales::dollar) +
  facet_wrap(~ symbol, ncol = 2, scales = "free_y") +
  theme_tq() + scale_color_tq()
```



Again, please verify that the dollar value of 100 USD invested in Amazon at the end of the 4 years is 291.4267 USD, as we discussed before. Here, we can see the USD evolution of my investment every month.

Remember this plot?

```
# FANG stock prices.
FANG_daily_all
```



Netflix stock price remains down all the time. However, 100 USD invested in Netflix during these 4 year would lead to the highest value of almost 1000 USD. This is why we should conduct a return analysis to make investment decisions.

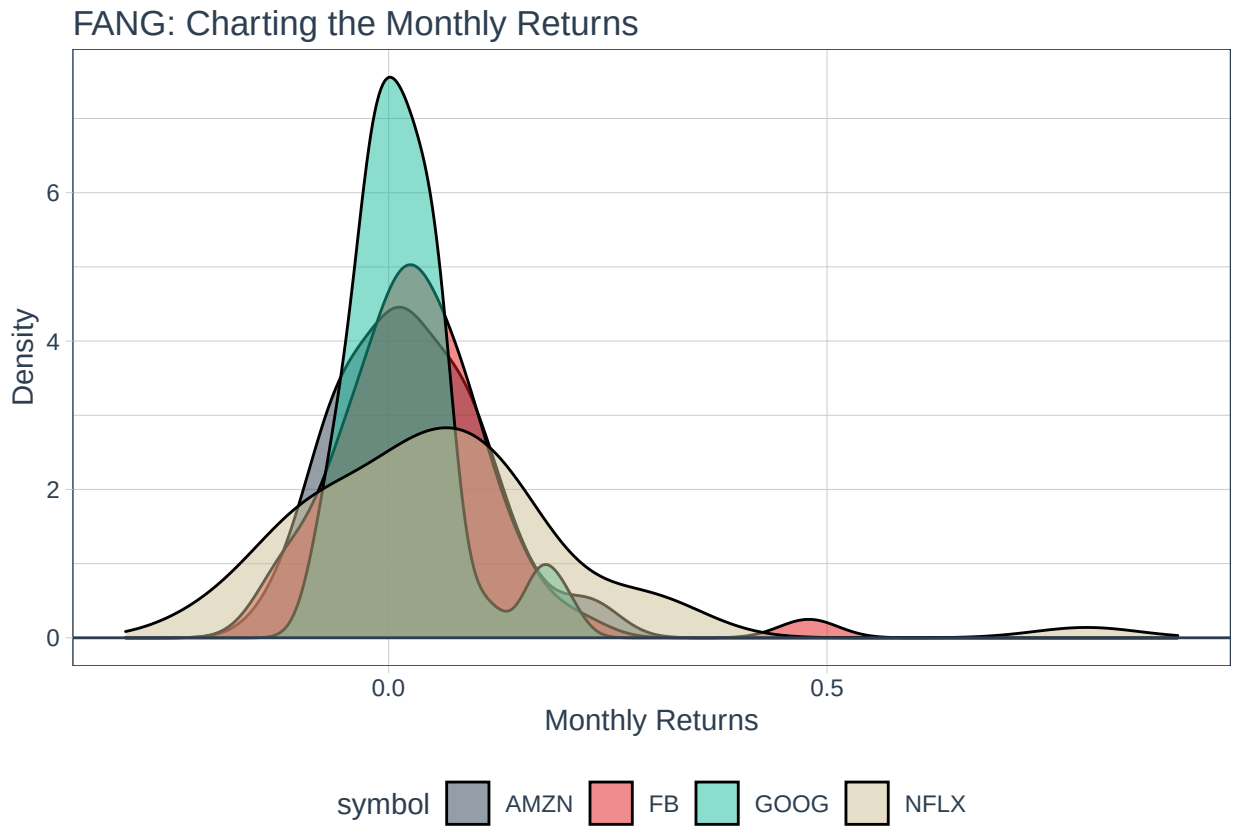
Plotting prices and returns over time is revealing because we can see the evolution of the stock performance over time. It is useful because we could analyze what happens on specific dates to better understand the behaviour of stocks. Remember 2014 was a bad year for Amazon and Google, it would be interesting to investigate what went wrong then so we can better understand the determinants of the firm's value.

3.3 Distribution of returns.

Although time series plots represent a useful tool for financial analysis, it is not the only way we can show stock returns. Density plots forget about time and show the distribution of values. The height (vertical axis) represents how frequent a given return (horizontal axis) is. This approach is useful when we are interested to know the most likely return that the stock can have, how unlikely is to expect determined return values, how risky a specific stock return is.

Let's visualize a density plot for the FENG data.

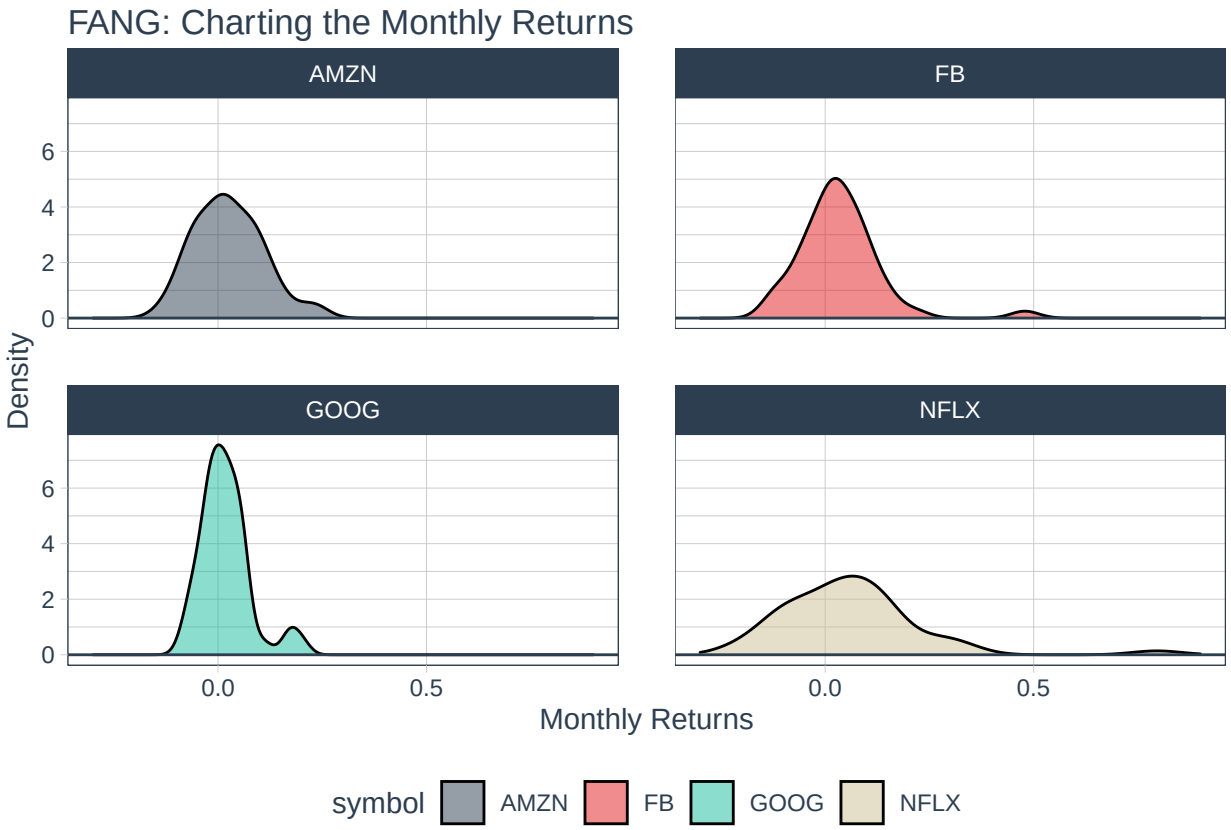
```
# Density plots.
ggplot(FANG_monthly_returns, aes(x = monthly.returns,
                                 fill = symbol)) +
  geom_density(alpha = 0.5) +
  geom_hline(yintercept = 0, color = palette_light()[[1]]) +
  labs(title = "FANG: Charting the Monthly Returns",
       x = "Monthly Returns", y = "Density") + xlim(-0.3, 0.9) +
    theme_tq() + scale_fill_tq()
```



Note that monthly returns around zero are frequent, whereas monthly returns above 50% are very infrequent. Apparently, this 50% value is red, so this should be the case of Facebook. It is also noteworthy to mention that Netflix has the higher monthly return dispersion of all. If this is not clear enough, we can add one line to the code to make it clearer.

```
# Density plots.
ggplot(FANG_monthly_returns, aes(x = monthly.returns,
                                 fill = symbol)) +
```

```
  geom_density(alpha = 0.5) +
  geom_hline(yintercept = 0, color = palette_light()[[1]]) +
  labs(title = "FANG: Charting the Monthly Returns",
       x = "Monthly Returns", y = "Density") + xlim(-0.3, 0.9) +
  theme_tq() + scale_fill_tq() + facet_wrap(~ symbol, ncol = 2)
```



Now, we have one density plot for each stock. Let's analyze each plot.

- AMZN. Note the highest value is zero, this suggests that the most frequent return value is around zero. The majority of observations are within -25% and 25%, and there are only a very few returns higher than 25%.
- GOOG. As in the case of AMZN, the most frequent return value is around zero. Contrary to AMZN, there are no monthly returns higher than 25%. GOOG shows a less dispersed distribution so it is less risky than AMZN.
- FB. The highest value is slightly above zero, this means that the average return should be higher compared with AMZN and GOOG. However, FB exhibits a greater dispersion of returns as there are a few around 50%.
- NFLX. This is the most riskiest asset compared with the rest as the monthly returns

have a wide dispersion.

Density distributions are useful to have an idea about the overall risk and return of assets as they summarize 48 observations (in this case 12 observations per year, 12×4). Some may argue 48 observations are not enough to conduct a financial analysis, but we prefer to keep it simply for now.

Now, let's propose some statistical indicators to complement our previous analysis.

```
# Calculate relevant statistics.
FANG_stats <- FANG_monthly_returns |>
summarise(mean = mean(monthly.returns), sd = sd(monthly.returns),
          sr = mean/sd, iqr = IQR(monthly.returns))
FANG_stats
```

```
## # A tibble: 4 x 5
##   symbol   mean     sd    sr    iqr
##   <chr>   <dbl>  <dbl> <dbl>  <dbl>
## 1 AMZN   0.0257 0.0817 0.314 0.126
## 2 FB     0.0341 0.0989 0.345 0.108
## 3 GOOG   0.0176 0.0594 0.296 0.0646
## 4 NFLX   0.0592 0.167  0.355 0.193
```

The mean is basically a measure of an expected return of the stock. The stock with the lowest mean is GOOG and the highest is NFLX. This is consistent with our previous graphical analysis. GOOG expected return is very close to zero, whereas NFLX expected return is definitely higher as the highest value corresponds to a positive return. The standard deviation (sd) is a measure of how disperse are the returns or in financial terms, how risky are the returns. The stock with less risk is GOOG and the stock with highest risk is NFLX. This is also consistent with our previous graphical analysis. IQR is also a measure of risk although we are not going to discuss this one right now.

Comparing return and risk, or mean and standard deviation of stock returns can be troublesome. This is because the relationship between risk and return is not perfectly proportional.

Let's see the risk and return in a plot.

```
# Mean variance plot.
FANG_stats |>
  ggplot(aes(x = sd, y = mean, color = symbol)) +
  geom_point(size = 5) +
```

```
  geom_text(aes(label = paste0(round(sr, 3))),
            vjust = 2, color = "black", size = 3.5) +
  xlim(0.04, 0.18) + ylim(0.01, 0.06) +
  labs(title = "The higher the risk, the higher the return",
subtitle = "Numerical values represent return per unit of risk.",
       x = "Risk", y = "Return") + theme_tq()
```

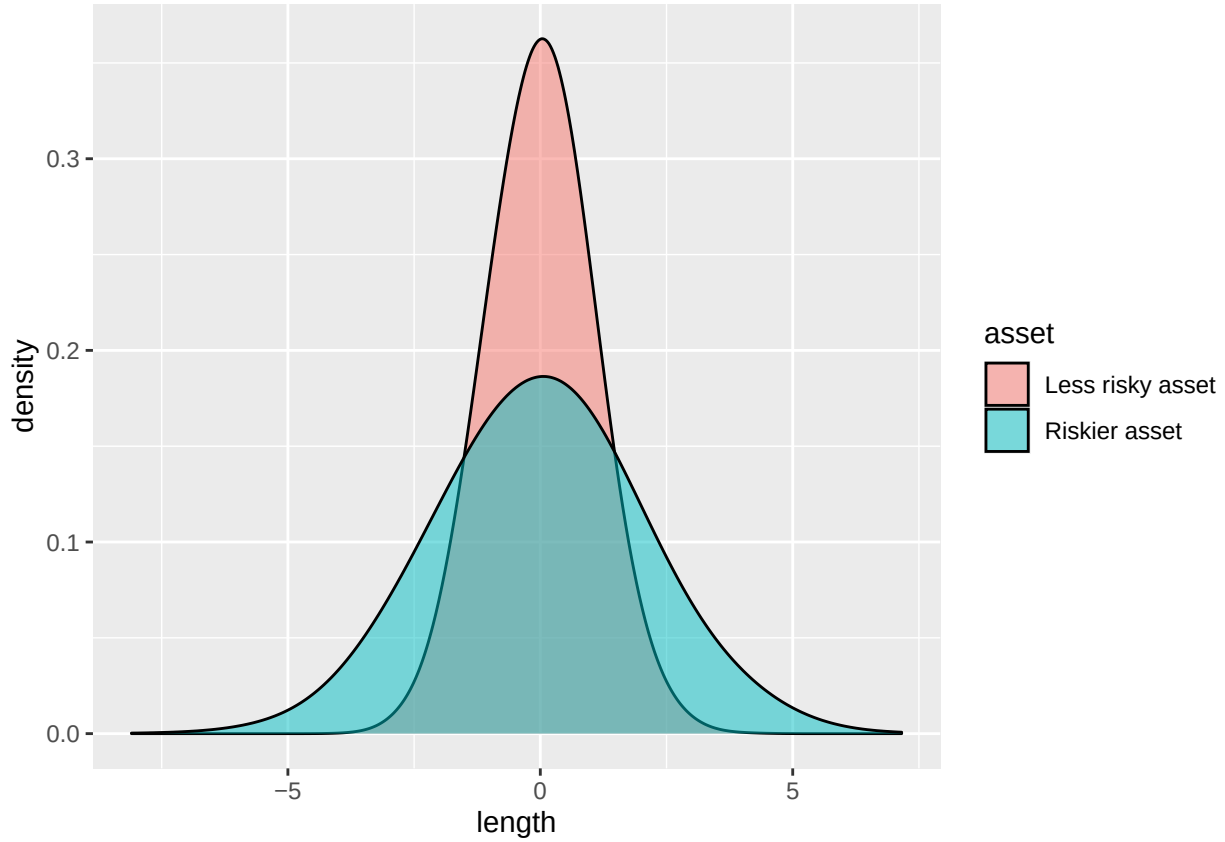
According to our results, Netflix is the stock with the highest return and risk. Does this mean Netflix has the highest return per unit of risk? This is easy to find out by dividing the mean by the standard deviation. This indicator is called *sr* in the table above. According to this, Netflix has the highest return per unit of risk compared with the rest of the stocks. This is clearly consistent with our previous analysis of cumulative returns.

The plot above measures returns in y-axis and risk in x-axes. In finance, we call this a mean-variance approach as mean is the measure of return and variance (or its square root the standard deviation) is the measure of risk. The mean-variance approach was a breakthrough in finance because we can clearly compare any kind of assets under the basis of risk and return.

Imagine stock 1 has a return of 5% and a risk of 4%, whereas stock 2 has a return of 6% and a risk of 8%. Although stock 2 has a higher return, this return does not compensate for the increase in the risk. In particular stock 1 has a 1.25 return per unit of risk (5/4), and stock 2 has a 0.75 return per unit of risk. In this case, we would prefer to invest our money in stock 1. Again, the relative values are useful to make decisions. Here the relative value is stock return per unit of risk.

Note how we can differentiate between a less risky and a riskier asset returns by looking at their correspondent distributions. Let's highlight these principles by using a brief example. Below we simulate a couple of asset returns by generating random values. Both artificially created assets have the same return (0%) and one is riskier than the other.

```
# Create random assets.
less_risky <- data.frame(length = rnorm(10000, 0, 1))
more_risky <- data.frame(length = rnorm(10000, 0, 2))
# Name random assets.
less_risky$asset <- 'Less risky asset'
more_risky$asset <- 'Riskier asset'
assets <- rbind(less_risky, more_risky)
# Plot the assets.
ggplot(assets, aes(length, fill = asset)) +
  geom_density(alpha = 0.5, adjust = 3)
```



The risky asset can take a wider range of returns. In particular, the risky asset can take values ranging from (about) -7% to $+7\%$, whereas the less risky asset about -3% to $+3\%$. In the code above we ask R to generate 10,000 random values following a normal distribution with mean zero (return), and a standard deviation of 1% and 2% for the less and riskier asset respectively.

4 Asset pricing.

The aim of asset pricing theories is to determine the fundamental value of an asset. There is a close relation between this fundamental value and an appropriate return. Then, the main focus of asset pricing theories and models is to determine this appropriate return. Most models relate expected returns to risks investors have to bear and have to be compensated for. They differ mainly in the risk factors they allow to enter into the model.

Although a large number of risk factors have been proposed in the literature and many models are used, none of the presented models or any other models used by practitioners are able to explain the observed asset prices or returns sufficiently. While most models work quite well for time horizons of more than a year, they fail in explaining short term movements.

4.1 What drives asset return changes?

What drives asset return changes? This is a big question in finance. Stock returns change as expectations of the stock market participants (buyers and sellers) change. Good expectations can be interpreted as an opportunity for investors to buy now looking to generate a profit in the future. Good expectations could increase the demand for this stock, the stock price increases and the return increases as well. Bad expectations could increase the supply for this stock as stock market participants are now willing to sell the stock before the stock price falls. An increase in the supply decreases the stock price and the return. In practice supply and demand of a given stock change at the same time and at different magnitudes most of the time, so it is not easy to explain and anticipate changes in returns. Note that we said "not easy", we are not saying "impossible".

Expectations are fueled by information, and information comes from data analysis in many forms including business news, macroeconomic news, new financial information of the firm available, market conditions, rumors, fake news, tweets, and many more. This data is subject to a wide variety of financial analysis and models, so it is not uncommon that one investor anticipates a price fall whereas others anticipate a price increase. Again, it is not easy to explain and anticipate changes in returns as measuring expectations seems like a complicated task. We can argue that changes in risk factors change expectations so risk factors drive changes in asset returns. Let's elaborate more on these risk factors.

Consider an extreme example about what we mean by explaining and anticipating changes in stock returns. Let's say someone has been awake for the last 24 hours and is driving after drinking too much alcohol. Here, we have two well known risk factors that can anticipate and explain a potential car accident. Anybody under any circumstances is exposed to a car

accident, but here there are two very clear risk factors that explain and anticipate a potential car accident. Knowing the relevant and most important risk factors is important because we can teach young people how to avoid car accidents and if we do it very well, we could even decrease the rate of car accidents. There might be many risk factors but surely there are some more significant than others. Consider driving while listening to music at a high volume. I think it is not crazy to assume that driving drunk is more dangerous than driving while listening to music at a high volume. It sounds reasonable to assume that adding two risk factors like lack of sleep and alcohol increases the chances of a car accident significantly.

Asset pricing and financial econometrics are usually combined in order to find out the risk factors of stock return changes (among other things). A typical risk factor is firm size. In particular, there is some evidence that suggests that small firms have higher returns than big firms. The industry is also considered as a risk factor as there are some industries which exhibit consistently lower returns than others. There are also some financial ratios that can be interpreted as risk factors for stock returns. In sum, we have some understanding of the risk factors of stock returns but we still need to learn more about this topic.

Why are we interested to explain and anticipate changes in stock returns? Remember the value of stocks are directly related with firm's equity, and firm's equity with firm size and performance. In finance and economics, we are interested that firms can grow and perform well because they represent not only a supply for goods and services but they also represent new job opportunities. The population in most countries is increasing so we need more and better job opportunities. As long as more families can get more and better job conditions they could (if other conditions hold) increase their living standards. Firms are also a source of innovation, research and knowledge so they play a very important role in the economy. Therefore, as long as we understand what drives stock returns can help us to understand the evolution of the whole economy and this is closely related with the people's standards of living. At the end, we should care about the firm performance not because we are interested in having money machines, but because we should aim to transform this value increase into better living conditions for families.

Firms are not always good for the economy. Firms are run by individuals and individuals can be greedy sometimes. Firms can also make wrong decisions because they simply fail to conduct a correct analysis. Just before the US credit crunch 2007-2008 crisis, banks were interested in allocate sub-prime mortgages in order to transform them and sell them as "new" assets called asset back securities. This securitization process allowed banks to sell the credit risk to others in a not so transparent process and mechanism. Let me explain. Every mortgage represents an account payable for the bank because a family is expected to

pay for that loan. Accounts payable are boring in the sense that you have to wait until you get paid, plus they are risky because families may default. By implementing a securitization, the bank can literally collect plenty of accounts payable or sub-prime mortgages and create one single new asset that can be sold. If you buy this asset you will have to pay the bank an amount of money today, in exchange you will get future payments for a number of years in the future. You might be interested in this deal because the future payments are attractive compared with the price of this new asset.

What happened then was that the borrowers default, sub-prime mortgages were not paid, asset back securities investors lost their money, and we fall into a global recession. This is relevant for us because credit rating agencies failed to assign a good estimate of the risk of asset back securities. In particular, they reported that these assets had a very high credit rating whereas in reality they were basically junk (sub-prime mortgages). In some cases, they reported the same credit rating as bonds, and this is just crazy. This missvaluation (or overvaluation) of the asset back securities motivated people to invest in these assets as they believed they were a bargain (high return and low risk assets), but in reality, they were simply high return and very high risky assets. This is why we need to understand what drives asset returns. And this is also why we argue that firms are not always good for the economy.

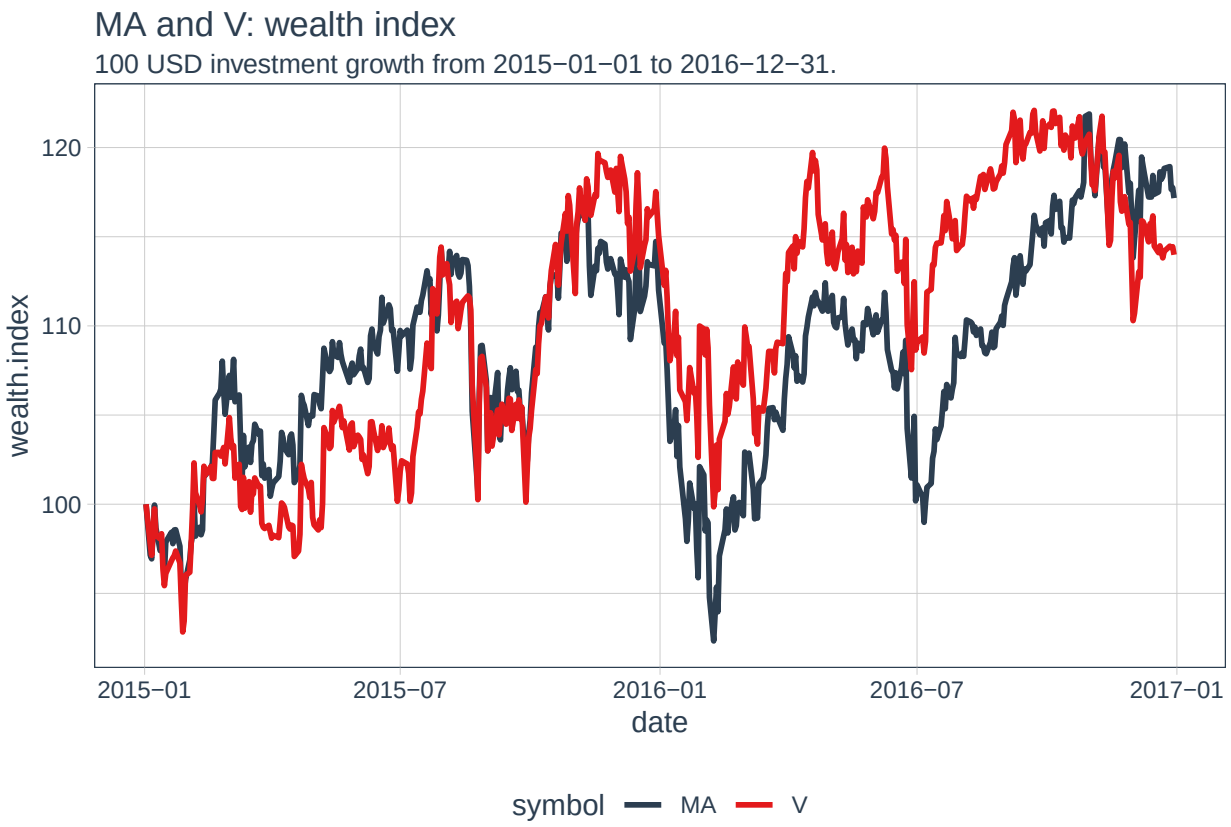
Let's come back to our main objective (drivers of stock returns) and apply a simple approach to start. We know that there are certain stocks that behave similarly. Consider Mastercard and Visa. Visa and MasterCard are the two largest payment processing networks in the world, they do not issue cards directly to the public, as do Discover and American Express, but rather through member financial institutions. Therefore, it is easy to understand that these two firms belong to the same industry, and they are exposed to very similar risk factors. In principle, if these firms are similar, their respective stock returns might behave similar as well. We may argue that the change of the returns of one can be explained by the change of the returns of the other.

Let's see how similar these two firms are by looking at their respective cumulative returns in a similar way as we did with FANG stocks. We first download the price data.

```
# Download the price data.
stock_prices_MA_V <- c("MA", "V") |>
    tq_get(get  = "stock.prices", from = "2015-01-01",
           to   = "2016-12-31") |>
    group_by(symbol)
```

Then transform prices into cumulative returns and plot them.

```
# Transform prices into log returns.
stock_cumret_MA_V <- stock_prices_MA_V |>
    tq_transmute(adjusted,
                 periodReturn,
                 period = "daily",
                 type = "log",
                 col_rename = "returns") |>
# Create the wealth index.
    mutate(wealth.index = 100 * cumprod(1 + returns))
# Visualize the wealth index.
ggplot(stock_cumret_MA_V, aes(x = date, y = wealth.index,
                              color = symbol)) +
    geom_line(size = 1) + labs(title = "MA and V: wealth index",
    subtitle = "100 USD investment growth from 2015-01-01 to 2016-12-31.") +
    theme_tq() + scale_color_tq()
```



Apparently, these stock returns are indeed closely related. Ups and downs are very similar in this time-series. These are the exact values:

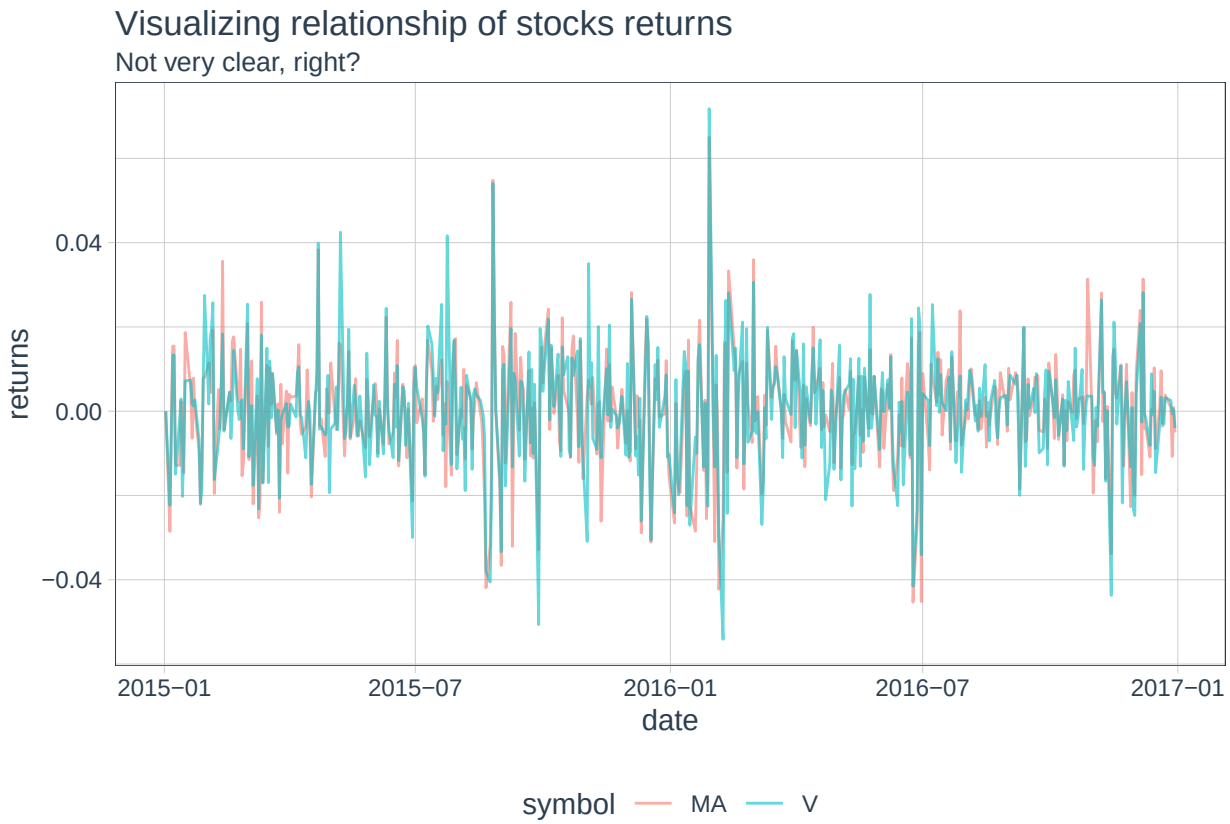
```
# Last values of the wealth index.
stock_cumret_MA_V |>
  select(-returns) |>
  spread(symbol, wealth.index) |>
  tail()
```

```
## # A tibble: 6 x 3
##   date          MA     V
##   <date>     <dbl> <dbl>
## 1 2016-12-22  118.  114.
## 2 2016-12-23  119.  114.
## 3 2016-12-27  119.  114.
## 4 2016-12-28  118.  114.
## 5 2016-12-29  118.  114.
## 6 2016-12-30  117.  114.
```

The 100 USD investment growth is basically the same in both assets.

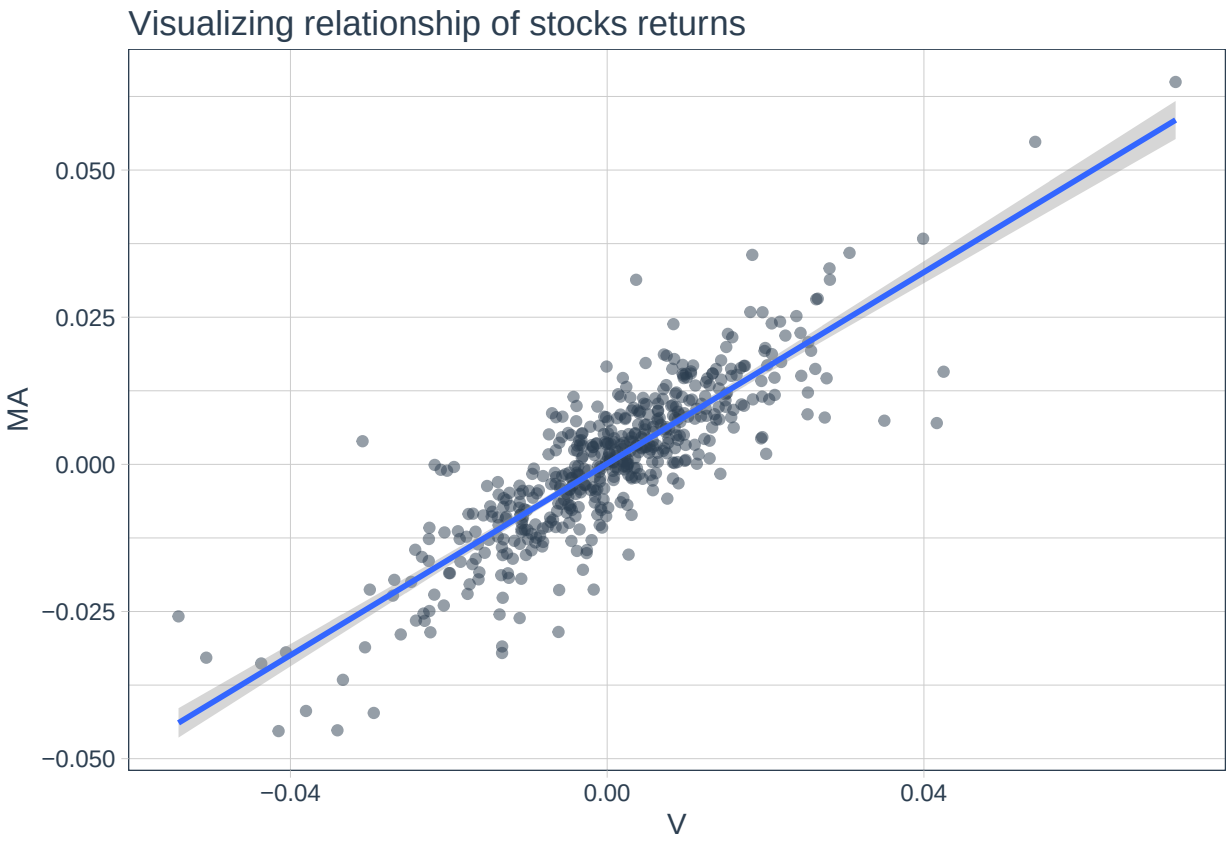
Let's see the daily returns of both stocks in a scatter plot now. This allows us to confirm if both stock returns are related.

```
# Show the return relationship.
stock_ret_MA_V <- stock_prices_MA_V |>
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "daily",
                 type       = "log",
                 col_rename = "returns")

ggplot(stock_ret_MA_V, aes(x = date, y = returns,
                           group = symbol)) +
  geom_line(aes(color = symbol), alpha = 0.6) +
  labs(title = "Visualizing relationship of stocks returns",
       subtitle = "Not very clear, right?") + theme_tq()
```



The line plot was not a good idea to show our main point. You cannot see anything clear here. Only ups and downs with no clear visual pattern. Let's propose a different way to illustrate the relationship of stock returns.

```
# Show the return relationship.
stock_ret_MA_V  |>
    spread(key = symbol, value = returns) |>
    ggplot(aes(x = V, y = MA)) +
    geom_point(color = palette_light()[[1]], alpha = 0.5) +
    geom_smooth(method = "lm") +
    labs(title = "Visualizing relationship of stocks returns") + theme_tq()
```



Dropping the time and plotting coordinates looks better. The scatter plot above suggests that if Visa stock returns are low, Mastercard stock returns are low as well. By the same token, if Visa sock returns are high, Mastercard stock returns are high as well. We have 504 points; every point represents a pair of daily returns of both firms. Stock returns are then positively related, and this relationship seems to be strong and linear. We can add one line to the above plot to summarize this linear relationship between the 504 stock return pairs.

Note that this straight line summarizes quite well the relationship between the stock returns. This is, most of the points lie very close to the straight line. The general equation of a straight line is $y = a + bx$, the value a is called intercept and b is the line slope. The slope measures how steep the line is, the higher the slope the steeper the line. We care about the slope because it measures how related are x and y. In particular, a slope of zero means that there is no relationship between x and y, whereas a slope of 1 means that if x increases 2, then we should expect y increases by 2 as well. The slope is interesting because it is the magnitude of the relationship between x and y. The sign of b reveals whether the relationship is positive or negative. According to the above plot, we can anticipate that the slope b is positive and it should be close to 1.

In this case the general model is $MA_i = \alpha + \beta V_i$, where $i = 1, ..., 504$ as we have 504 observations of MA and V. It is easy to estimate the values of α and β by conducting a simple regression analysis. The regression analysis basically finds out the values of α and β that fits better the observed relationship between MA and V.

```
# Prepare the data.
reg <- stock_ret_MA_V |>
  spread(key = symbol, value = returns)
# Estimate the model.
lm(MA ~ V, data = reg) |>
summary()
```

```
##
## Call:
## lm(formula = MA ~ V, data = reg)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0269573 -0.0039656  0.0002151  0.0039650  0.0289458
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0001130  0.0003097   0.365    0.715
## V           0.8133655  0.0226393  35.927   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00695 on 502 degrees of freedom
## Multiple R-squared:   0.72,  Adjusted R-squared:  0.7194
## F-statistic:  1291 on 1 and 502 DF,  p-value: < 2.2e-16
```

In general our model is $MA_i = \alpha + \beta V_i$, and in particular we have our model estimation result as $MA_i = 0.0001130 + 0.8133658 V_i$. Without going into details, our linear regression analysis suggests that a 1% return on V is associated with a 0.8133658% return in MA. This is a good fit of the observations.

Let's evaluate one example.

```
filter(reg, date=="2015-04-22")
```

```
## # A tibble: 1 x 3
##   date            MA      V
##   <date>       <dbl>  <dbl>
## 1 2015-04-22 0.0383 0.0399
```

Consider the observation pair $V = 0.03989731$, $MA = 0.03833513$. According to our regression line, the value of MA at $V = 0.03989731$ should be $MA = 0.0001130 + 0.03989731 \times 0.8133658 = 0.03256411$. The difference between the observed 3.833513% and the estimated 3.256411% is the estimation error. Note that the observed value is higher than the estimated value, this is the observation is slightly above the blue regression line.

Our results suggests that Visa represent a risk factor of Mastercard. This is because according to our model Mastercard return changes can be explained by changes in Visa returns. The relationship between this pair of stock returns according to beta is almost perfect as 0.8133658 is close to 1. We can also calculate the correlation of both stock returns. The correlation (or Pearson correlation) is a statistic that measures linear correlation between two variables. It has a value between $+1$ and -1. A value of $+1$ is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. The beta is similar but it could be higher than $+1$ or lower than -1.

```
cor(reg$MA, reg$V)
```

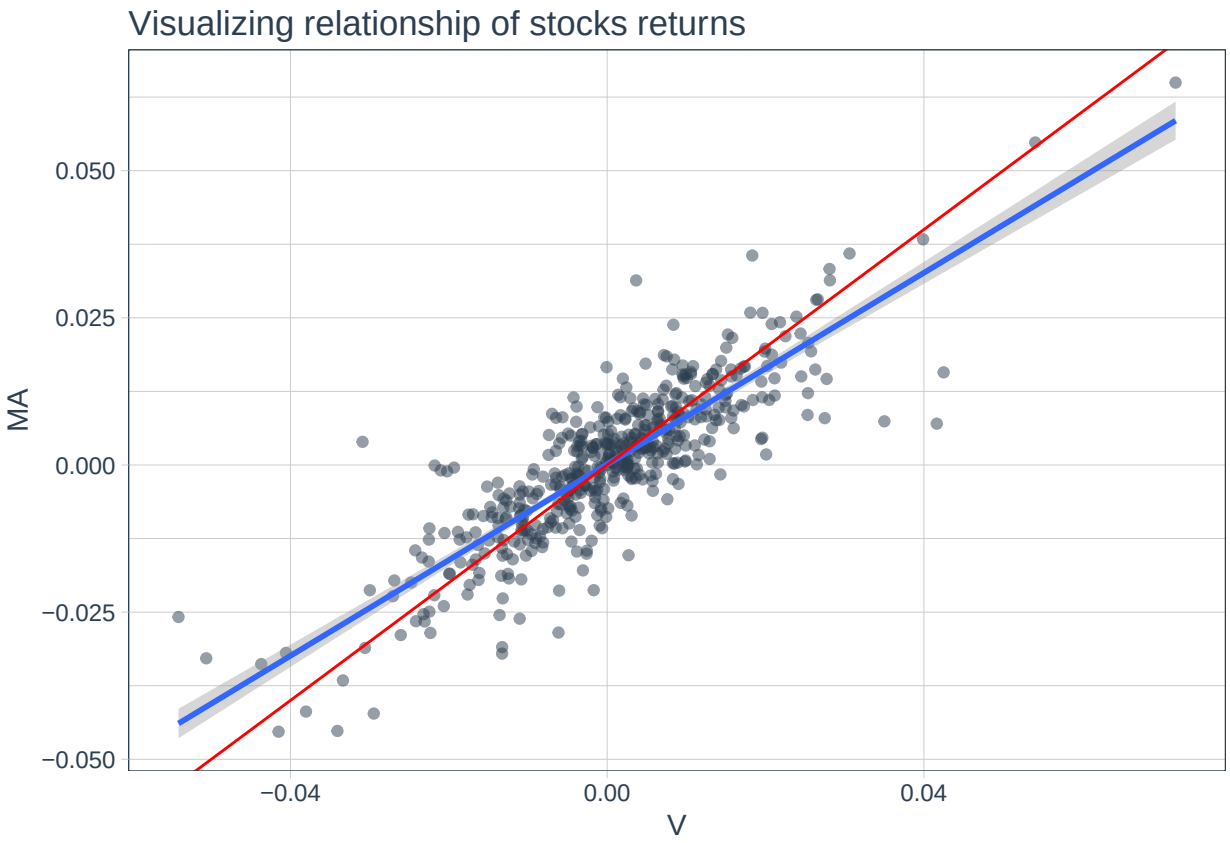
```
## [1] 0.8485193
```

In this case, the correlation of Visa and Mastercard stock returns is 0.8485187. This correlation value is very close to 1, so these variables are strongly linear correlated. This is important because we can argue that these two stocks are very similar, if one increases the other increases and if one decreases the other decreases. This behaviour makes sense because as we said before these firms are exposed to the same (or very similar) risk factors.

Let's visualize a 1:1 relationship just as a reference in the red line below.

```
# Add a reference red line representing beta=1.
stock_ret_MA_V  |>
    spread(key = symbol, value = returns) |>
    ggplot(aes(x = V, y = MA)) +
    geom_point(color = palette_light()[[1]], alpha = 0.5) +
    geom_smooth(method = "lm") +
```

```
    geom_abline(intercept = 0, color = "red") +
    labs(title = "Visualizing relationship of stocks returns") +
    theme_tq()
```



The strength of this fit can be measured by the R-squared. The R-squared value can be interpreted as the variation of MA explained by variations of V. In particular, almost 72% of the variations of MA is explained by variations in V. The rest (about 28%) remains unexplained according to the model. This 28% could be explained by adding more risk factors to the asset pricing model, or proposing an alternative estimation method.

Our example assumed that MA can be explained by V. We have not introduced a formal model so we are free to relax this assumption and turn it around. What if we assume Visa depends on Mastercard? Note that the results are not so different.

```
# Estimate the model.
lm(V ~ MA, data = reg) |>
    summary()
```

```
##
```

```
## Call:
## lm(formula = V ~ MA, data = reg)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.034372 -0.003965 -0.000033  0.003423  0.035421
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.104e-06  3.231e-04  -0.003    0.997
## MA           8.852e-01  2.464e-02  35.927   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00725 on 502 degrees of freedom
## Multiple R-squared:   0.72,  Adjusted R-squared:  0.7194
## F-statistic:  1291 on 1 and 502 DF,  p-value: < 2.2e-16
```

Here, we estimate $V_i = \alpha + \beta MA_i$, and in particular we have $V_i = -1.104 \times 10^6 + 0.8852 MA_i$.

```
# Create yearly returns.
FANG_daily_returns <- FANG |>
    group_by(symbol) |>
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "daily",
                 type       = "arithmetic")
# Transform data.
library(tbl2xts)
f <- FANG_daily_returns |> tbl_xts(spread_by = "symbol")
# Correlation.
cor(f)
```

```
##              FB      AMZN      NFLX      GOOG
## FB   1.0000000 0.4147219 0.2521936 0.4229816
## AMZN 0.4147219 1.0000000 0.3132691 0.5212696
## NFLX 0.2521936 0.3132691 1.0000000 0.3141388
## GOOG 0.4229816 0.5212696 0.3141388 1.0000000
```

```r
# Model.
lm(AMZN ~ FB + GOOG + NFLX, data = f) |>
    summary()
```

```
##
## Call:
## lm(formula = AMZN ~ FB + GOOG + NFLX, data = f)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.136211 -0.006983 -0.000131  0.006947  0.126424
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0002764  0.0005012   0.551    0.581
## FB          0.1909035  0.0252878   7.549 9.81e-14 ***
## GOOG        0.5072017  0.0383769  13.216  < 2e-16 ***
## NFLX        0.0819768  0.0163624   5.010 6.43e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01583 on 1004 degrees of freedom
## Multiple R-squared:  0.3343, Adjusted R-squared:  0.3323
## F-statistic: 168.1 on 3 and 1004 DF,  p-value: < 2.2e-16
```

4.2 Single-index model.

The relationship of the risk-return trade-off is the heart of equilibrium asset pricing theories and models. The asset pricing theories are fascinating as they allow us to go deeper in the understanding of the determinants of asset returns. Here, we will skip many aspects regarding the theory, derivation, assumptions and comparison of asset pricing models, theories and estimation approaches. For a formal approach you should look for the references by the end of this document. Here, we will motivate the main ideas underlying the asset pricing models, the underlying programming approach, and what can we learn from them.

What if we propose that the return of an individual asset is partially explained by the market return? This proposal is similar as if we assume that the education level of one individual de-

pends on the education level of the whole country. This proposal seems reasonable although not perfect. The market return is an aggregation of all individual asset returns traded in the stock market. There are several ways we can estimate the market return. One of them is to calculate the return of a stock index like the S&P500. The S&P500, or simply the S&P, is a stock market index that measures the stock performance of about 500 large companies listed on stock exchanges in the United States. It is one of the most commonly followed equity indexes, and many consider it to be one of the best representations of the US stock market.

Market indexes as the S&P500 are important for several reasons. We have discussed that stock price evolution is a good way to track the performance of a single company. However, we are usually interested in evaluating the performance of all the companies in the market, not only one or two. Stock market indexes are formed by several firms or in some cases many participants known as constituents. There is a wide variety of stock market indexes, some are country specific, or based on industries and other firm characteristics. By looking at the price or index evolution of these indexes we can track what happens with the firms in an aggregate approach. Then, an index value change in a day show what happened with the firms in average in that specific day. If the S&P500 increases in one day then we interpret this as if most of the firms that participate in the S&P500 increased its value in that particular day.

These are the S&P500 constituents ticker symbols, or in other words the 500 large companies that belongs to the S&P 500 index.

```
# Load the package.
library(BatchGetSymbols)
sp500 <- GetSP500Stocks()
sp500$Tickers
```

```
##   [1] "MMM"   "ABT"   "ABBV"  "ABMD"  "ACN"   "ATVI"  "ADBE"  "AMD"   "AAP"
##  [10] "AES"   "AFL"   "A"     "APD"   "AKAM"  "ALK"   "ALB"   "ARE"   "ALXN"
##  [19] "ALGN"  "ALLE"  "LNT"   "ALL"   "GOOGL" "GOOG"  "MO"    "AMZN"  "AMCR"
##  [28] "AEE"   "AAL"   "AEP"   "AXP"   "AIG"   "AMT"   "AWK"   "AMP"   "ABC"
##  [37] "AME"   "AMGN"  "APH"   "ADI"   "ANSS"  "ANTM"  "AON"   "AOS"   "APA"
##  [46] "AAPL"  "AMAT"  "APTV"  "ADM"   "ANET"  "AJG"   "AIZ"   "T"     "ATO"
##  [55] "ADSK"  "ADP"   "AZO"   "AVB"   "AVY"   "BKR"   "BLL"   "BAC"   "BK"
##  [64] "BAX"   "BDX"   "BRK.B" "BBY"   "BIO"   "BIIB"  "BLK"   "BA"    "BKNG"
##  [73] "BWA"   "BXP"   "BSX"   "BMY"   "AVGO"  "BR"    "BF.B"  "CHRW"  "COG"
##  [82] "CDNS"  "CZR"   "CPB"   "COF"   "CAH"   "KMX"   "CCL"   "CARR"  "CTLT"
```

```
##  [91] "CAT"   "CBOE"  "CBRE"  "CDW"   "CE"    "CNC"   "CNP"   "CERN"  "CF"
## [100] "CRL"   "SCHW"  "CHTR"  "CVX"   "CMG"   "CB"    "CHD"   "CI"    "CINF"
## [109] "CTAS"  "CSCO"  "C"     "CFG"   "CTXS"  "CLX"   "CME"   "CMS"   "KO"
## [118] "CTSH"  "CL"    "CMCSA" "CMA"   "CAG"   "COP"   "ED"    "STZ"   "COO"
## [127] "CPRT"  "GLW"   "CTVA"  "COST"  "CCI"   "CSX"   "CMI"   "CVS"   "DHI"
## [136] "DHR"   "DRI"   "DVA"   "DE"    "DAL"   "XRAY"  "DVN"   "DXCM"  "FANG"
## [145] "DLR"   "DFS"   "DISCA" "DISCK" "DISH"  "DG"    "DLTR"  "D"     "DPZ"
## [154] "DOV"   "DOW"   "DTE"   "DUK"   "DRE"   "DD"    "DXC"   "EMN"   "ETN"
## [163] "EBAY"  "ECL"   "EIX"   "EW"    "EA"    "EMR"   "ENPH"  "ETR"   "EOG"
## [172] "EFX"   "EQIX"  "EQR"   "ESS"   "EL"    "ETSY"  "EVRG"  "ES"    "RE"
## [181] "EXC"   "EXPE"  "EXPD"  "EXR"   "XOM"   "FFIV"  "FB"    "FAST"  "FRT"
## [190] "FDX"   "FIS"   "FITB"  "FE"    "FRC"   "FISV"  "FLT"   "FMC"   "F"
## [199] "FTNT"  "FTV"   "FBHS"  "FOXA"  "FOX"   "BEN"   "FCX"   "GPS"   "GRMN"
## [208] "IT"    "GNRC"  "GD"    "GE"    "GIS"   "GM"    "GPC"   "GILD"  "GL"
## [217] "GPN"   "GS"    "GWW"   "HAL"   "HBI"   "HIG"   "HAS"   "HCA"   "PEAK"
## [226] "HSIC"  "HSY"   "HES"   "HPE"   "HLT"   "HOLX"  "HD"    "HON"   "HRL"
## [235] "HST"   "HWM"   "HPQ"   "HUM"   "HBAN"  "HII"   "IEX"   "IDXX"  "INFO"
## [244] "ITW"   "ILMN"  "INCY"  "IR"    "INTC"  "ICE"   "IBM"   "IP"    "IPG"
## [253] "IFF"   "INTU"  "ISRG"  "IVZ"   "IPGP"  "IQV"   "IRM"   "JKHY"  "J"
## [262] "JBHT"  "SJM"   "JNJ"   "JCI"   "JPM"   "JNPR"  "KSU"   "K"     "KEY"
## [271] "KEYS"  "KMB"   "KIM"   "KMI"   "KLAC"  "KHC"   "KR"    "LB"    "LHX"
## [280] "LH"    "LRCX"  "LW"    "LVS"   "LEG"   "LDOS"  "LEN"   "LLY"   "LNC"
## [289] "LIN"   "LYV"   "LKQ"   "LMT"   "L"     "LOW"   "LUMN"  "LYB"   "MTB"
## [298] "MRO"   "MPC"   "MKTX"  "MAR"   "MMC"   "MLM"   "MAS"   "MA"    "MKC"
## [307] "MXIM"  "MCD"   "MCK"   "MDT"   "MRK"   "MET"   "MTD"   "MGM"   "MCHP"
## [316] "MU"    "MSFT"  "MAA"   "MHK"   "TAP"   "MDLZ"  "MPWR"  "MNST"  "MCO"
## [325] "MS"    "MOS"   "MSI"   "MSCI"  "NDAQ"  "NTAP"  "NFLX"  "NWL"   "NEM"
## [334] "NWSA"  "NWS"   "NEE"   "NLSN"  "NKE"   "NI"    "NSC"   "NTRS"  "NOC"
## [343] "NLOK"  "NCLH"  "NOV"   "NRG"   "NUE"   "NVDA"  "NVR"   "NXPI"  "ORLY"
## [352] "OXY"   "ODFL"  "OMC"   "OKE"   "ORCL"  "OGN"   "OTIS"  "PCAR"  "PKG"
## [361] "PH"    "PAYX"  "PAYC"  "PYPL"  "PENN"  "PNR"   "PBCT"  "PEP"   "PKI"
## [370] "PRGO"  "PFE"   "PM"    "PSX"   "PNW"   "PXD"   "PNC"   "POOL"  "PPG"
## [379] "PPL"   "PFG"   "PG"    "PGR"   "PLD"   "PRU"   "PTC"   "PEG"   "PSA"
## [388] "PHM"   "PVH"   "QRVO"  "PWR"   "QCOM"  "DGX"   "RL"    "RJF"   "RTX"
## [397] "O"     "REG"   "REGN"  "RF"    "RSG"   "RMD"   "RHI"   "ROK"   "ROL"
## [406] "ROP"   "ROST"  "RCL"   "SPGI"  "CRM"   "SBAC"  "SLB"   "STX"   "SEE"
```

```
## [415] "SRE"   "NOW"   "SHW"   "SPG"   "SWKS"  "SNA"   "SO"    "LUV"   "SWK"
## [424] "SBUX"  "STT"   "STE"   "SYK"   "SIVB"  "SYF"   "SNPS"  "SYY"   "TMUS"
## [433] "TROW"  "TTWO"  "TPR"   "TGT"   "TEL"   "TDY"   "TFX"   "TER"   "TSLA"
## [442] "TXN"   "TXT"   "TMO"   "TJX"   "TSCO"  "TT"    "TDG"   "TRV"   "TRMB"
## [451] "TFC"   "TWTR"  "TYL"   "TSN"   "UDR"   "ULTA"  "USB"   "UAA"   "UA"
## [460] "UNP"   "UAL"   "UNH"   "UPS"   "URI"   "UHS"   "UNM"   "VLO"   "VTR"
## [469] "VRSN"  "VRSK"  "VZ"    "VRTX"  "VFC"   "VIAC"  "VTRS"  "V"     "VNO"
## [478] "VMC"   "WRB"   "WAB"   "WMT"   "WBA"   "DIS"   "WM"    "WAT"   "WEC"
## [487] "WFC"   "WELL"  "WST"   "WDC"   "WU"    "WRK"   "WY"    "WHR"   "WMB"
## [496] "WLTW"  "WYNN"  "XEL"   "XLNX"  "XYL"   "YUM"   "ZBRA"  "ZBH"   "ZION"
## [505] "ZTS"
```

These are the S&P500 constituents. Have you seen that there are more than 500? Companies listed in the index may have issued multiple types of common stock, this is why we do not have exactly 500.

Let's conduct a regression analysis as before. Now the model is $stock_{i,j} = \alpha_i + \beta_i SP500_j + \epsilon_i$, where sub-index i represents a given stock, and j the historical observations. Note that this model is very similar as the general equation of a straight line is $y = a + bx$. The model $stock_{i,j} = \alpha_i + \beta_i SP500_j + \epsilon_i$ is basically the single-index-model. It is called single because we assume there is only one risk factor which in this case is the market return. There are other multi-factor models that adds more factors (like factor F for example) and they look like this: $stock_{i,j} = \alpha_i + \beta_i SP500_j + \delta_i F_j + \epsilon_i$.

According to the single index model, the stock return is decomposed in three parts: a constant return α_i, a component proportional to the market index $\beta_i SP500_j$, and a random and unpredictable component ϵ_i. The intercept term is the expected value of the component of security i's return that is independent of the market's performance. The beta coefficient is specific for each security and measures the security's sensitivity to the market. The random component represents the deviation of the return on the security from its expected value. The single-index model says that risks of individual securities arise from two sources: market or systematic risk reflected in $\beta_i SP500_j$ and firm-specific risk reflected in ϵ_i. This simple dichotomy may oversimplify factors of real world uncertainty. For example it ignores industry events which affect many firms within a single industry but do not influence the macroeconomy as a whole.

Let's estimate the model. Here, we have selected 10 stocks, $i = 1, ...10$, and 72 monthly returns observations $j = 1, ..., 72$. Our main estimation objective is to find α_i and β_i for

these 10 stocks.

First, we download the corresponding 10 stock returns.

```
# Download individual asset returns.
R_stocks <- c("NEM", "AMCR", "CLX", "PEAK", "KR", "TXN", "F", "TXT",
              "KLAC", "TEF") |>
  tq_get(get  = "stock.prices", from = "2010-01-01",
         to   = "2015-12-31") |>
  group_by(symbol) |>
  tq_transmute(select     = adjusted,
               mutate_fun = periodReturn,
               period     = "monthly",
               col_rename = "R_stocks")
R_stocks
```

```
## # A tibble: 692 x 3
## # Groups:   symbol [10]
##    symbol date       R_stocks
##    <chr>  <date>        <dbl>
##  1 NEM    2010-01-29  -0.115
##  2 NEM    2010-02-26   0.150
##  3 NEM    2010-03-31   0.0355
##  4 NEM    2010-04-30   0.101
##  5 NEM    2010-05-28  -0.0403
##  6 NEM    2010-06-30   0.149
##  7 NEM    2010-07-30  -0.0946
##  8 NEM    2010-08-31   0.0970
##  9 NEM    2010-09-30   0.0268
## 10 NEM    2010-10-29  -0.0310
## # ... with 682 more rows
```

Then, we download the S&P500 monthly returns, the symbol is ^GSPC.

```
# Download the market index return.
R_market <- "^GSPC" |>
  tq_get(get  = "stock.prices", from = "2010-01-01",
         to   = "2015-12-31") |>
    tq_transmute(select     = adjusted,
```

```
                 mutate_fun = periodReturn,
                 period     = "monthly",
                 col_rename = "R_market")
R_market
```

```
## # A tibble: 72 x 2
##    date       R_market
##    <date>        <dbl>
##  1 2010-01-29  -0.0522
##  2 2010-02-26   0.0285
##  3 2010-03-31   0.0588
##  4 2010-04-30   0.0148
##  5 2010-05-28  -0.0820
##  6 2010-06-30  -0.0539
##  7 2010-07-30   0.0688
##  8 2010-08-31  -0.0474
##  9 2010-09-30   0.0876
## 10 2010-10-29   0.0369
## # ... with 62 more rows
```

Now, we join them in the same variable or R object. This is convenient in the estimation procedure.

```
# Prepare the database.
R_stocks_market <- left_join(R_stocks, R_market, by = c("date" = "date"))
R_stocks_market
```

```
## # A tibble: 692 x 4
## # Groups:   symbol [10]
##    symbol date       R_stocks R_market
##    <chr>  <date>        <dbl>    <dbl>
##  1 NEM    2010-01-29  -0.115   -0.0522
##  2 NEM    2010-02-26   0.150    0.0285
##  3 NEM    2010-03-31   0.0355   0.0588
##  4 NEM    2010-04-30   0.101    0.0148
##  5 NEM    2010-05-28  -0.0403  -0.0820
##  6 NEM    2010-06-30   0.149   -0.0539
##  7 NEM    2010-07-30  -0.0946   0.0688
```

```
##  8 NEM    2010-08-31   0.0970  -0.0474
##  9 NEM    2010-09-30   0.0268   0.0876
## 10 NEM    2010-10-29  -0.0310   0.0369
## # ... with 682 more rows
```

We can use the *performance_fun* function to facilitate the regression analysis estimation.

```
# 10 models estimated at once.
R_capm <- R_stocks_market |>
  tq_performance(Ra = R_stocks,
                 Rb = R_market,
                 performance_fun = table.CAPM) |>
select(symbol, Alpha, Beta, `R-squared`)
R_capm
```

```
## # A tibble: 10 x 4
## # Groups:   symbol [10]
##    symbol   Alpha  Beta `R-squared`
##    <chr>    <dbl> <dbl>       <dbl>
##  1 NEM    -0.0084 0.155      0.0032
##  2 AMCR    0.0035 0.415      0.0748
##  3 CLX     0.0097 0.434      0.195
##  4 PEAK    0.0052 0.448      0.0965
##  5 KR      0.0165 0.713      0.215
##  6 TXN     0.0032 1.27       0.575
##  7 F      -0.0032 1.39       0.400
##  8 TXT     0.0009 1.64       0.431
##  9 KLAC    0.0024 1.74       0.541
## 10 TEF    -0.0172 1.50       0.446
```

Here, we have the results for the single index model estimation for the 10 stocks.

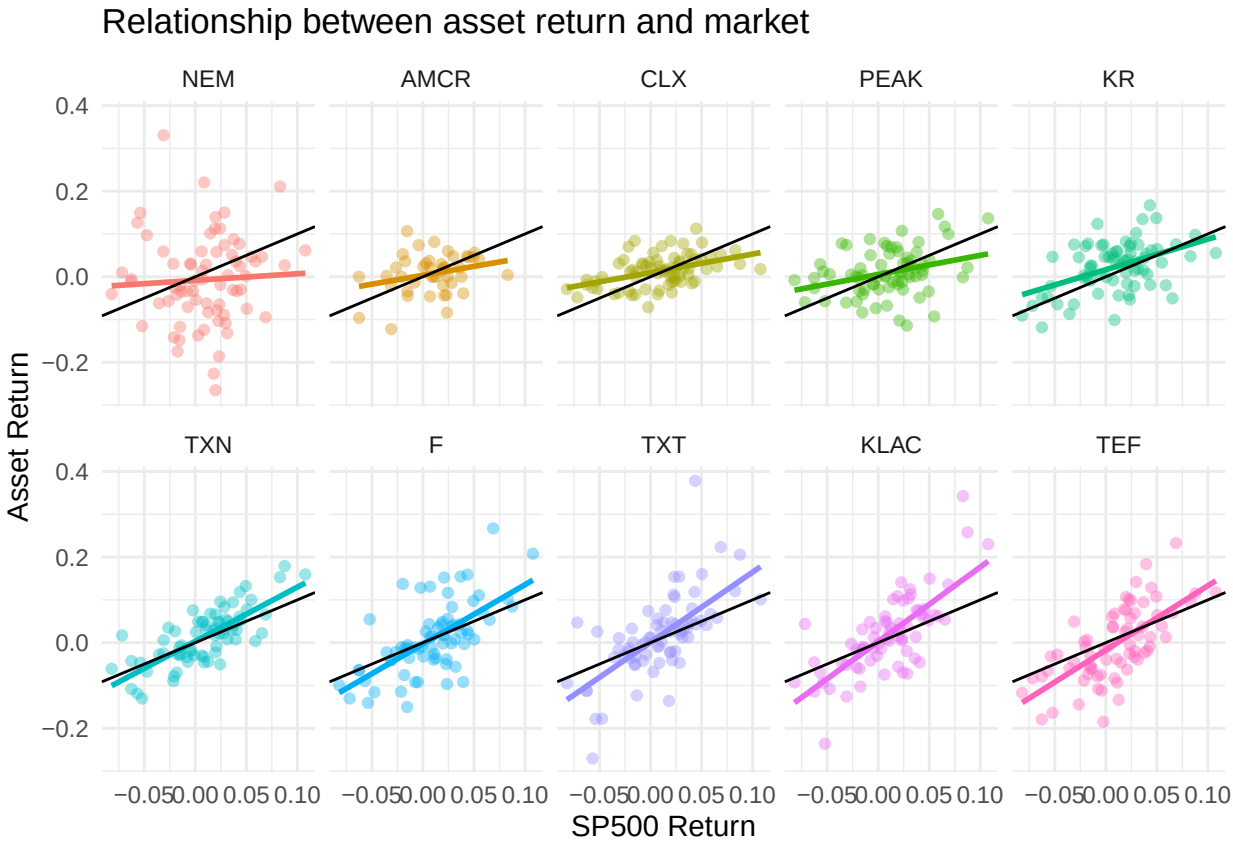
Please note how we arrange the stocks according to the beta. We have stocks β from 0.15 to 1.74. We interpret this as the riskiness of the stock with respect to the market. The stock NEM has the lowest risk with respect to the market. In other words, stocks with low betas are less exposed to changes in the S&P500 this is why we argue that are not risky with respect to the market. Stocks with low betas might be exposed to other risk factors, but not the market. On the other hand, stocks with high betas like TEF are highly exposed to changes in the S&P500 this is why we argue that are risky with respect to the market. The

beta of TEF is actually higher than 1, which means that TEF react more than proportional with respect to changes in the stock market.

We need a deeper econometric analysis to validate our interpretations. Here, we are not going to deal with a formal econometric interpretation but we can propose some arguments to better understand the stock behaviour. Alphas are all very close to zero. The R-squared is also relevant as it shows what proportion of changes in the stock returns are explained by changes in the stock market. Note that 57% of TXN stock return changes are explained by changes in the S&P500.

We can illustrate the previous results in a graphical way.

```
R_stocks_market$symbol <-
  factor(R_stocks_market$symbol, levels =
           unique(R_stocks_market$symbol))
# Plot all results.
R_stocks_market |>
  ggplot(aes(x = R_market, y = R_stocks, color = symbol)) +
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~symbol, ncol = 5) +
  geom_abline(intercept = 0, color = "black", linetype = 1) +
  theme_minimal() +
  labs(x = "SP500 Return", y = "Asset Return",
       title = "Relationship between asset return and market") +
  theme(legend.position = "none", legend.title = element_blank())
```



The stocks are sorted in such a way that the slope is increasing. I added a black line which represents a $\beta = 1$ to illustrate the cases in which the stock is less risky than the market and riskier than the market. The y-axis represents the asset returns and the x-axis the S&P500 returns.

We are interested in these results for a variety of reasons. Consider the following example. Imagine I am currently investing in the stock market and I anticipate a fall in the stock market. Some investors would prefer to sell their positions but imagine I need or want to stay for any reason. If so, then I might consider rearranging my portfolio investment to include more stocks with low betas (probably even negative betas). By doing so and if I was right with respect to the S&P500, then my position would not be severely affected because my portfolio is now more independent to the evolution of the stock market. On the other hand, if I anticipate given my analysis that the stock market will rise, then I could rearrange my portfolio to include more stocks with high betas (even higher than 1). By doing so and if I was right with respect to the S&P500, then my position would improve because my portfolio would generate more return, even more than the S&P500 itself.

In practice, we have to conduct a series of formal statistical tests to rely on our estimated

betas. In particular, we are interested to have a statistically significant estimator among other things. Things might become complicated when you know that there are many ways in which we can estimate betas. Even if we take a different historical dataset length, we can get different betas. Is we take a different market benchmark we can get different betas. Financial sites usually report betas of the stock, and sometimes it is difficult to find out what was the process of estimating the model. Then, these kinds of estimations have to be done by a professional in the area. My recommendation is to study asset pricing theory and relevant financial econometric techniques to propose these kind of investment recommendations. Our previous interpretation assumes we have correctly estimated betas.

The Capital Asset Pricing Model (CAPM) was created by William Sharpe in 1964. He won the 1990 Nobel Prize in Economic Sciences, along with Harry Markowitz and Merton Miller, for developing models to assist with investment decision making like the CAPM. This model is similar to the single index model as the CAPM estimates the return of an asset based on the return of the market and the asset's linear relationship to the return of the market. This linear relationship is the stock's β (beta) coefficient. The CAPM beta captures the linear relationship between the asset or portfolio and the market. This model is simple, but it can serve as a good base for the building of more complex models.

We can extend our analysis further. Given our 10 set of assets, we can calculate annualized returns and annualized Sharpe ratios (return per unit of risk).

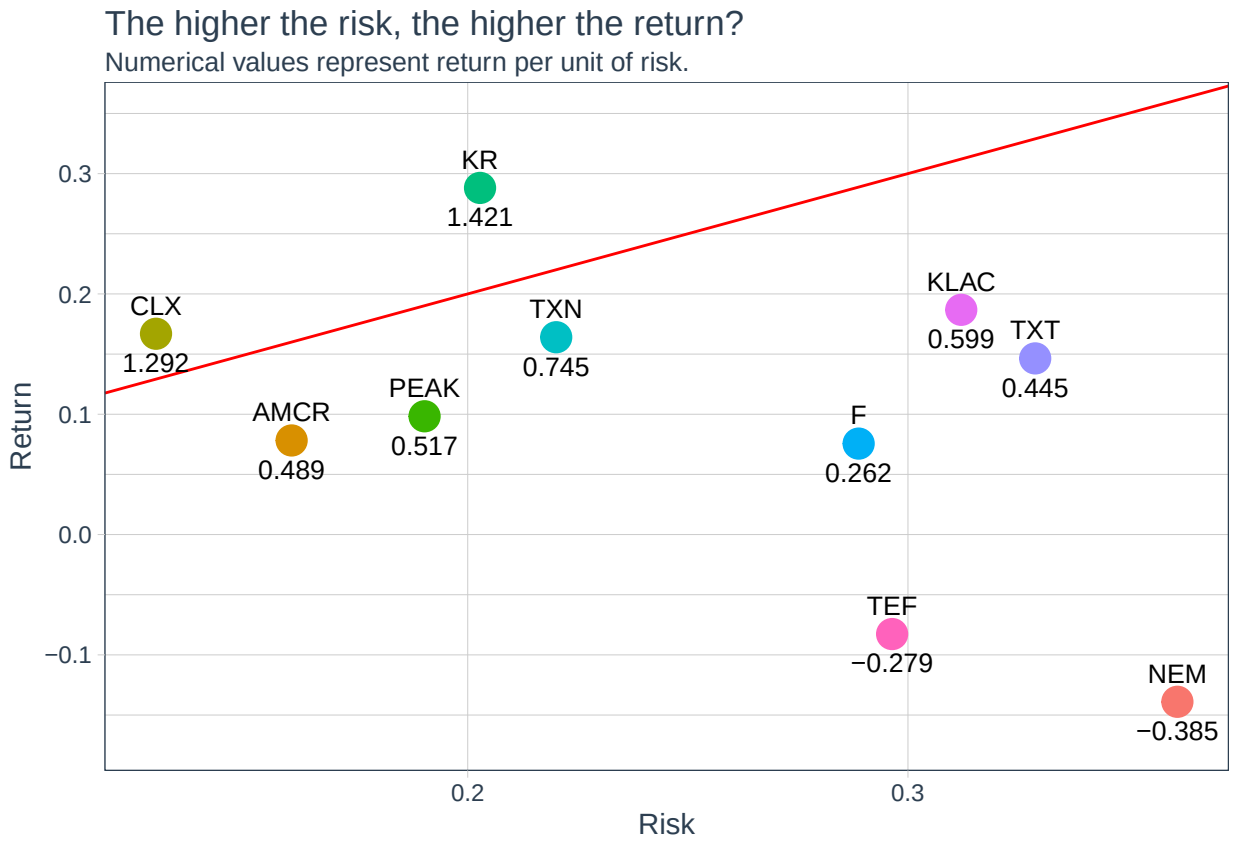
```
# Calculate annualized returns.
R_stocks_market |>
    tq_performance(Ra = R_stocks, Rb = NULL,
                   performance_fun = table.AnnualizedReturns) |>
arrange(`AnnualizedSharpe(Rf=0%)`)
```

```
## # A tibble: 10 x 4
## # Groups:   symbol [10]
##    symbol AnnualizedReturn `AnnualizedSharpe(Rf=0%)` AnnualizedStdDev
##    <fct>             <dbl>                     <dbl>             <dbl>
##  1 NEM             -0.139                    -0.385             0.361
##  2 TEF             -0.0827                   -0.279             0.296
##  3 F                0.0756                    0.262             0.289
##  4 TXT              0.146                     0.445             0.329
##  5 AMCR             0.0782                    0.489             0.16
##  6 PEAK             0.0983                    0.517             0.190
```

```
##  7 KLAC            0.187                  0.598          0.312
##  8 TXN             0.164                  0.745          0.220
##  9 CLX             0.167                  1.29           0.129
## 10 KR              0.288                  1.42           0.203
```

A stock beta is a measure of the individual asset return risk with respect to the market. The annualized Sharpe ratio above is a measure of the individual asset return per unit of risk. Let's visualize the previous table.

```
# Calculate annualized returns.
R_stocks_market_stats <- R_stocks_market |>
    tq_performance(Ra = R_stocks, Rb = NULL,
                   performance_fun = table.AnnualizedReturns) |>
# Mean variance plot.
  ggplot(aes(x = AnnualizedStdDev, y = AnnualizedReturn, color = symbol)) +
  geom_point(size = 5) +
  geom_abline(intercept = 0, color = "red") +
  geom_text(aes(label = paste0(round(`AnnualizedSharpe(Rf=0%)`, 3))),
            vjust = 2, color = "black", size = 3.5) +
  geom_text(aes(label = paste0(symbol)),
            vjust = -1, color = "black", size = 3.5) + ylim(-0.17, 0.35) +
  labs(title = "The higher the risk, the higher the return?",
subtitle = "Numerical values represent return per unit of risk.",
       x = "Risk", y = "Return") + theme_tq() +
  theme(legend.position = "none", legend.title = element_blank())
R_stocks_market_stats
```



I added a straight line of 45 degrees so any asset above the red line means a return per unit of risk above 1. By the same token, any asset below the red line means a return per unit of risk below 1.

```
# Download individual asset returns.
R_stocks_mkt <- c("NEM", "AMCR", "CLX", "PEAK", "KR", "TXN", "F", "TXT",
                  "KLAC", "TEF", "^GSPC") |>
  tq_get(get  = "stock.prices", from = "2010-01-01",
         to   = "2015-12-31") |>
  group_by(symbol) |>
  tq_transmute(select     = adjusted,
               mutate_fun = periodReturn,
               period     = "monthly",
               col_rename = "r")
R_stocks_mkt
```

```
## # A tibble: 764 x 3
## # Groups:   symbol [11]
```

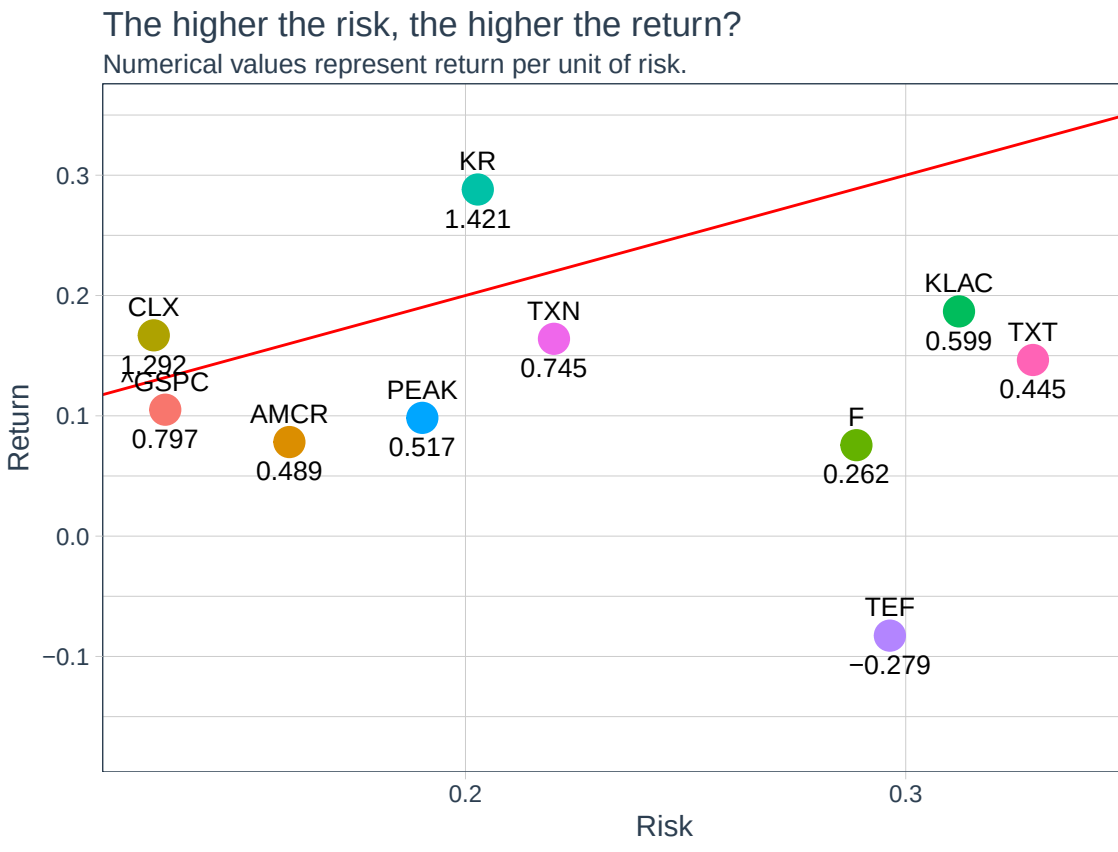
```
##    symbol date             r
##    <chr>  <date>       <dbl>
##  1 NEM    2010-01-29 -0.115
##  2 NEM    2010-02-26  0.150
##  3 NEM    2010-03-31  0.0355
##  4 NEM    2010-04-30  0.101
##  5 NEM    2010-05-28 -0.0403
##  6 NEM    2010-06-30  0.149
##  7 NEM    2010-07-30 -0.0946
##  8 NEM    2010-08-31  0.0970
##  9 NEM    2010-09-30  0.0268
## 10 NEM    2010-10-29 -0.0310
## # ... with 754 more rows
```

```
R_stocks_mkt |>
    tq_performance(Ra = r, Rb = NULL,
                   performance_fun = table.AnnualizedReturns) |>
arrange(`AnnualizedSharpe(Rf=0%)`)
```

```
## # A tibble: 11 x 4
## # Groups:   symbol [11]
##    symbol AnnualizedReturn `AnnualizedSharpe(Rf=0%)` AnnualizedStdDev
##    <chr>             <dbl>                     <dbl>             <dbl>
##  1 NEM             -0.139                    -0.385             0.361
##  2 TEF             -0.0827                   -0.279             0.296
##  3 F                0.0756                    0.262             0.289
##  4 TXT              0.146                     0.445             0.329
##  5 AMCR             0.0782                    0.489             0.16
##  6 PEAK             0.0983                    0.517             0.190
##  7 KLAC             0.187                     0.598             0.312
##  8 TXN              0.164                     0.745             0.220
##  9 ^GSPC            0.105                     0.797             0.132
## 10 CLX              0.167                     1.29              0.129
## 11 KR               0.288                     1.42              0.203
```

```
# Calculate annualized returns.
R_stocks_mkt_stats <- R_stocks_mkt |>
    tq_performance(Ra = r, Rb = NULL,
```

```
                    performance_fun = table.AnnualizedReturns) |>
# Mean variance plot.
  ggplot(aes(x = AnnualizedStdDev, y = AnnualizedReturn, color = symbol)) +
  geom_point(size = 5) +
  geom_abline(intercept = 0, color = "red") +
  geom_text(aes(label = paste0(round(`AnnualizedSharpe(Rf=0%)`, 3))),
            vjust = 2, color = "black", size = 3.5) +
  geom_text(aes(label = paste0(symbol)),
            vjust = -1, color = "black", size = 3.5) +
  ylim(-.17, .35) +
  labs(title = "The higher the risk, the higher the return?",
subtitle = "Numerical values represent return per unit of risk.",
       x = "Risk", y = "Return") + theme_tq() +
  theme(legend.position = "none", legend.title = element_blank())
R_stocks_mkt_stats
```

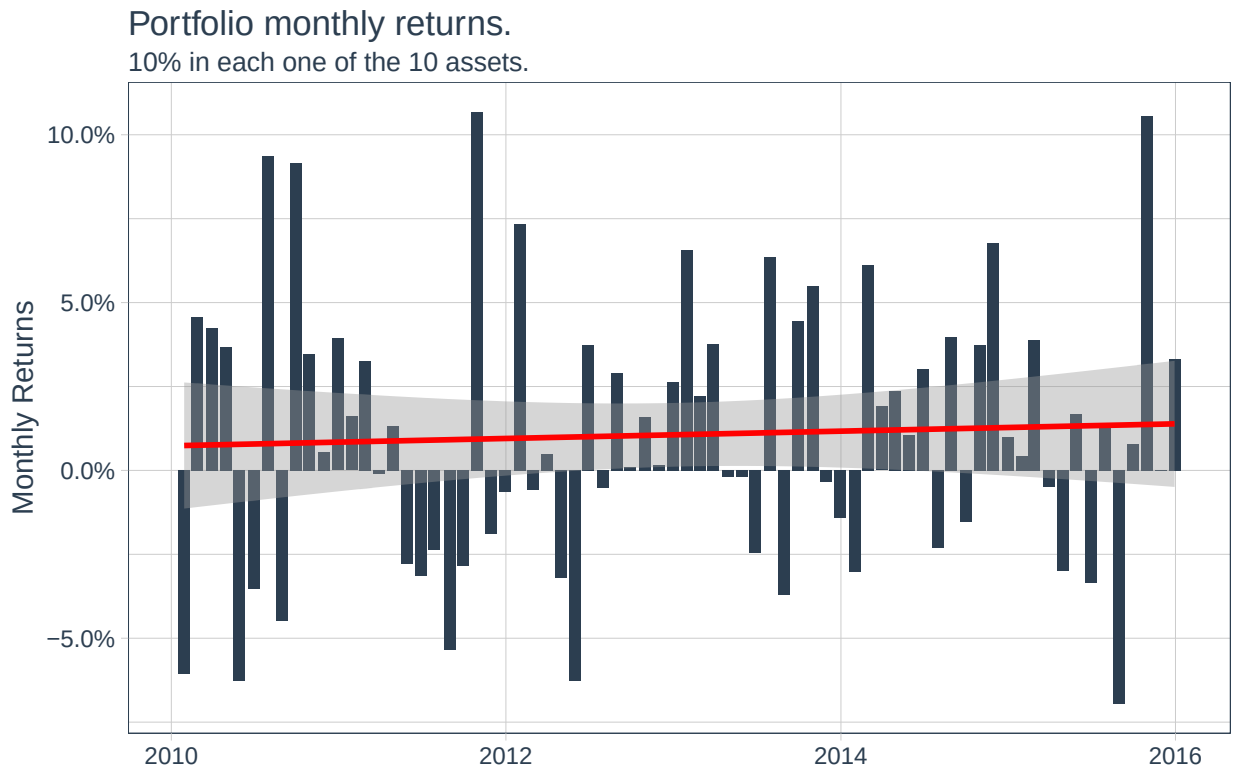


E2 q3-1.pdf E2 q3-1.bb

What if we use the values of betas and Sharpe ratios to form investment portfolios with these

10 assets? We call a portfolio to the new asset formed by several (smaller) investments in single assets. Let's start with a visualization of monthly return of a naive portfolio. A naive portfolio is basically formed by a 10% investment in each of the 10 assets. Here, we do not invest more in higher stock betas or in higher Sharpe ratio stocks. It is simply an equally weighted portfolio.

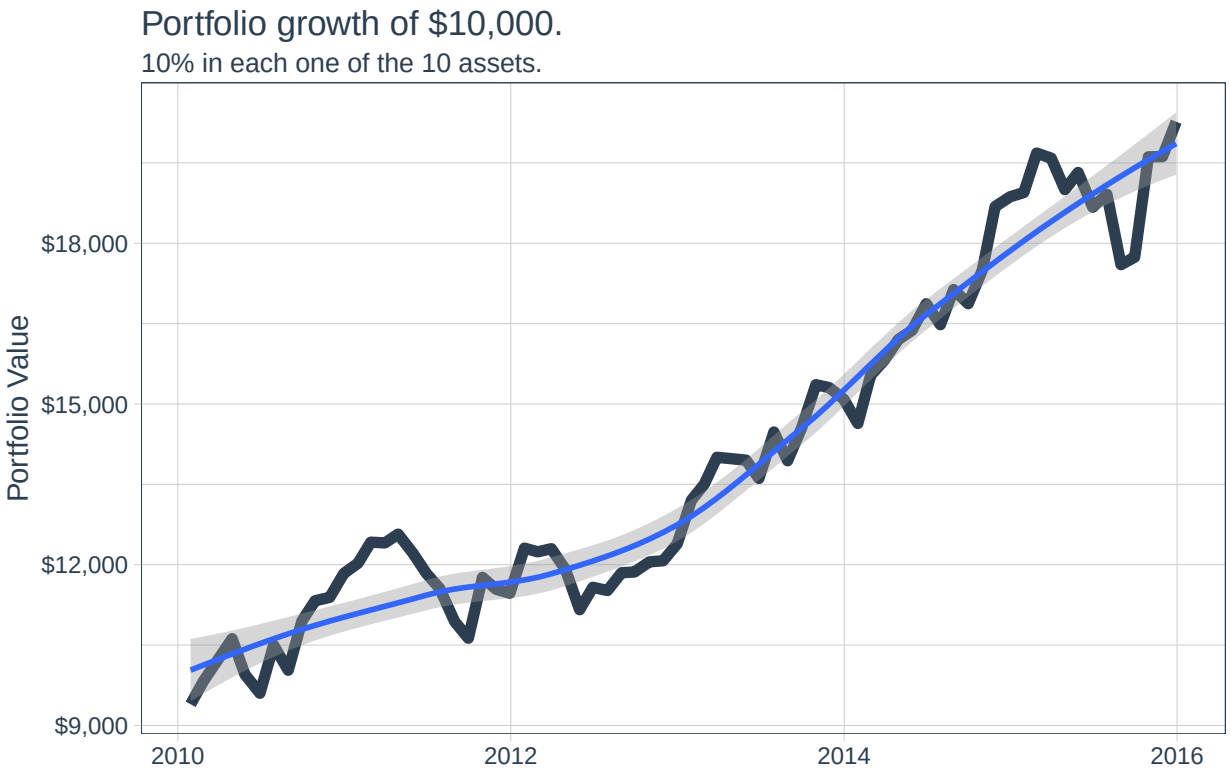
```
# Weights.
wts <- c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
# Portfolio creation.
portfolio_returns_monthly <- R_stocks_market |>
    tq_portfolio(assets_col  = symbol,
                 returns_col = R_stocks,
                 weights     = wts,
                 col_rename  = "Ra")
portfolio_returns_monthly |>
  # Visualization.
    ggplot(aes(x = date, y = Ra)) +
    geom_bar(stat = "identity", fill = palette_light()[[1]]) +
    labs(title = "Portfolio monthly returns.",
         subtitle = "10% in each one of the 10 assets.",
caption = "Shows an above-zero trend meaning positive returns.",
         x = "", y = "Monthly Returns") +
    geom_smooth(method = "lm", color = "red") +
    theme_tq() + scale_color_tq() +
    scale_y_continuous(labels = scales::percent)
```



A monthly returns plot is a good representation, but we normally show the evolution of an investment's growth. In this case we consider an initial investment of 10,000 USD.

```
# Cumulative returns.
portfolio_growth_monthly <- R_stocks_market |>
    tq_portfolio(assets_col   = symbol,
                 returns_col  = R_stocks,
                 weights      = wts,
                 col_rename   = "investment.growth",
                 wealth.index = TRUE) |>
    mutate(investment.growth = investment.growth * 10000)
portfolio_growth_monthly |>
    ggplot(aes(x = date, y = investment.growth)) +
    geom_line(size = 2, color = palette_light()[[1]]) +
    labs(title = "Portfolio growth of $10,000.",
         subtitle = "10% in each one of the 10 assets.",
         caption = "Now we can really visualize performance!",
         x = "", y = "Portfolio Value") +
```

```
    geom_smooth(method = "loess") +
    theme_tq() +
    scale_color_tq() +
    scale_y_continuous(labels = scales::dollar)
```



Looks OK. However, we are not sure if this is the best combination of assets. This is, what if the equally weighted portfolio is in fact the worst alternative? This is why we are interested in comparing this equally weighted portfolio with some other portfolios. In particular, a beta increasing portfolio and a Sharpe ratio increasing portfolio. These are basically incremental weights of 10%, 20%, 30% and 40% for the highest four betas and for the highest four Sharpe ratios.

```
# Calculate annualized returns.
R_stocks_market |>
  tq_performance(Ra = R_stocks, Rb = NULL,
                 performance_fun = table.AnnualizedReturns) |>
  arrange(`AnnualizedSharpe(Rf=0%)`) |>
  left_join(R_capm,by = 'symbol')  |>
```

```
  select(symbol, `AnnualizedSharpe(Rf=0%)`, Beta)
```

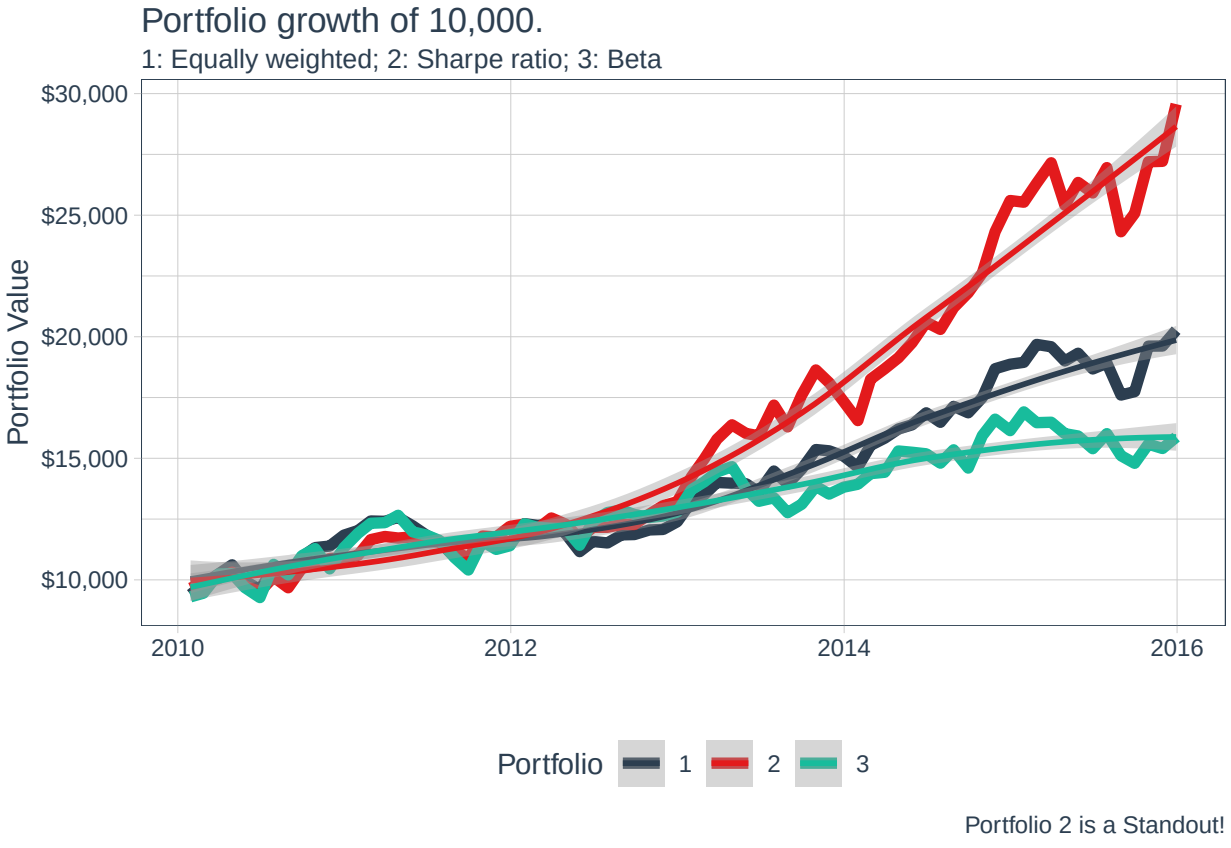
```
## # A tibble: 10 x 3
## # Groups:   symbol [10]
##    symbol `AnnualizedSharpe(Rf=0%)`  Beta
##    <chr>                       <dbl> <dbl>
##  1 NEM                        -0.385 0.155
##  2 TEF                        -0.279 1.50
##  3 F                           0.262 1.39
##  4 TXT                         0.445 1.64
##  5 AMCR                        0.489 0.415
##  6 PEAK                        0.517 0.448
##  7 KLAC                        0.598 1.74
##  8 TXN                         0.745 1.27
##  9 CLX                         1.29  0.434
## 10 KR                          1.42  0.713
```

```
# Three portfolios.
weights <- c(
    0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1, # equally weighted
    0, 0, 0, 0, 0, 0, 0.1, 0.2, 0.3, 0.4, # sr increasing
    0, 0.2, 0.1, 0.3, 0, 0, 0.4, 0, 0, 0 # beta increasing
)

stocks <- c("NEM", "TEF", "F", "TXT", "AMCR", "PEAK",
            "KLAC", "TXN", "CLX", "KR")
weights_table <-  tibble(stocks) |>
    tq_repeat_df(n = 3) |>
    bind_cols(tibble(weights)) |>
    group_by(portfolio)
```

See the results.

```
# See the evolution of three portfolios.
stock_returns_monthly_multi <- R_stocks_market |>
    tq_repeat_df(n = 3)

portfolio_growth_monthly_multi <- stock_returns_monthly_multi |>
```

```
    tq_portfolio(assets_col   = symbol,
                 returns_col  = R_stocks,
                 weights      = weights_table,
                 col_rename   = "investment.growth",
                 wealth.index = TRUE) |>
    mutate(investment.growth = investment.growth * 10000)
portfolio_growth_monthly_multi |>
  ggplot(aes(x = date, y = investment.growth, color = factor(portfolio))) +
    geom_line(size = 2) +
    labs(title = "Portfolio growth of 10,000.",
         subtitle = "1: Equally weighted; 2: Sharpe ratio; 3: Beta",
         caption = "Portfolio 2 is a Standout!",
         x = "", y = "Portfolio Value",
         color = "Portfolio") +
    geom_smooth(method = "loess") +
    theme_tq() + scale_color_tq() +
    scale_y_continuous(labels = scales::dollar)
```



In this case the increasing Sharpe ratio portfolio is the best one. In particular, the portfolio is: 10% in KLAC, 20% in TXN, 30% in CLX, 40% in KR, and 0% in the rest.

4.3 Predict asset prices.

In the previous section we were interested to explain what drives the changes of asset returns. By knowing these drivers, we could identify significant risk factors of individual assets. In the example above we propose that the risk factor was the market index (although there are more). We see that some assets are more related with the market than others. Up to this point, we know that NEM barely reacts to the changes in the S&P500 whereas TIF reacts more than proportional than the changes in the S&P500. The question that arises is, can we anticipate the evolution of the S&P500? Can we anticipate the changes in the risk factors? Because if we can, then we could use this information to anticipate the impact over individual assets. In fact, the kind of techniques and examples that we will explain in this section are as flexible that can be easily applied to any kind of asset prices.

In finance we care about the future so it makes sense to introduce some forecasting techniques in R. The forecasting techniques shown here do not depend on other external factors (as in the case of asset pricing models). You will realize that the only information we need to do these specific forecasts is the information of the past prices of the asset.

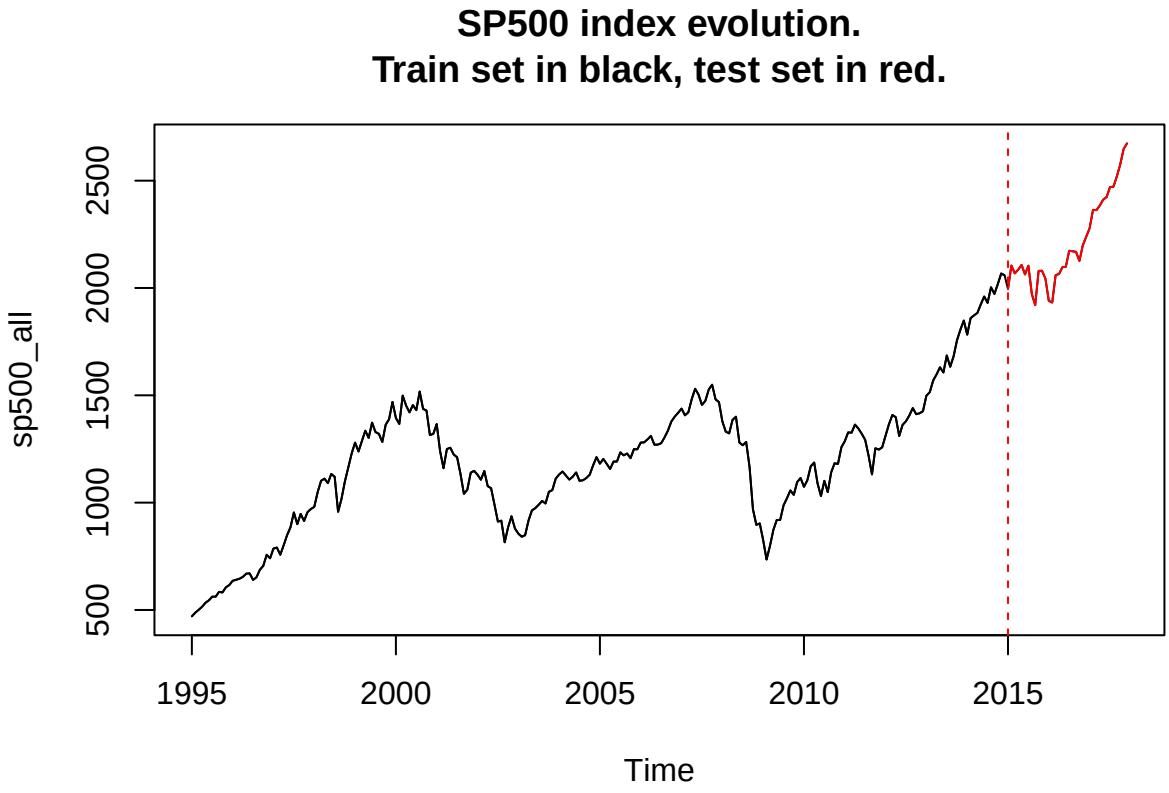
Let's download the data. In this case, the S&P500 index.

```
# Download the data.
sp_500 <- "^GSPC" |>
  tq_get(get  = "stock.prices", from = "1995-01-01",
         to   = "2017-12-31") |>
  tq_transmute(select     = adjusted,
               mutate_fun = to.monthly,
               col_rename = "sp_500")
```

Now, let's see the data.

```
# Split the data into training and test.
sp500_all <- ts(sp_500$sp_500, start = c(1995, 1), freq = 12)
sp500_training <- window(sp500_all, start = 1995, end = c(2014, 12))
sp500_test <- window(sp500_all, start = 2015)
# See the data before forecast.
plot(sp500_all, type = "l", main = "SP500 index evolution.
```

```
Train set in black, test set in red.")
lines(sp500_training, col = "black")
lines(sp500_test, col ="red")
abline(v = 2015, lty = 2, col = "red")
```



Most of us hear the weather forecast every morning and I think we agree it is not always accurate. In econometrics we care about this and we are interested in evaluating our forecasts. This is problematic because my current forecast in $t = 0$ will be validated in $t = 1$, but I need to know how good my forecast is in $t = 0$, not in $t = 1$. That is tricky. Since we do not have the Infinity Stone known as the Time Stone and we do not have Dr. Strange's skills, then we have to find an alternative solution. The solution is the following. Consider we assume that we have information until Dec 2014, so we are going to estimate our model up to Dec 2014 and forecast the following 36 months: 2015, 2016 and 2017. The train set is then from Jan 1995 to Dec 2014, and the test set from 2015 to 2017. The trick is that we actually know what happened in the test set, so we could be able to compare different forecasts and see which one was closer to what really happened. By doing this, we could have a better idea about how good is my today's forecast.

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).

Without going into more details, we can estimate an ARIMA model to conduct our forecast.

```
# Load the package.
library(forecast)
# Estimation of ARIMA model.
fit <- auto.arima(sp500_training)
# See results.
summary(fit)
```

```
## Series: sp500_training
## ARIMA(0,1,0) with drift
##
## Coefficients:
##         drift
##       6.6464
## s.e.  3.2327
##
## sigma^2 estimated as 2508:  log likelihood=-1273.99
## AIC=2551.98   AICc=2552.03   BIC=2558.93
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE    MAPE      MASE
## Training set 0.001932389 49.87271 38.14212 -0.08826945 3.32142 0.1996575
##                     ACF1
## Training set 0.05717067
```

This is the model estimation.

Now, let's see the forecast.

```
# 36-month forecast.
for_sp500_all <- forecast(fit, h = 36)
# Evaluate percentage error.
cbind(arima_forecast = tail(for_sp500_all$mean, 36),
```

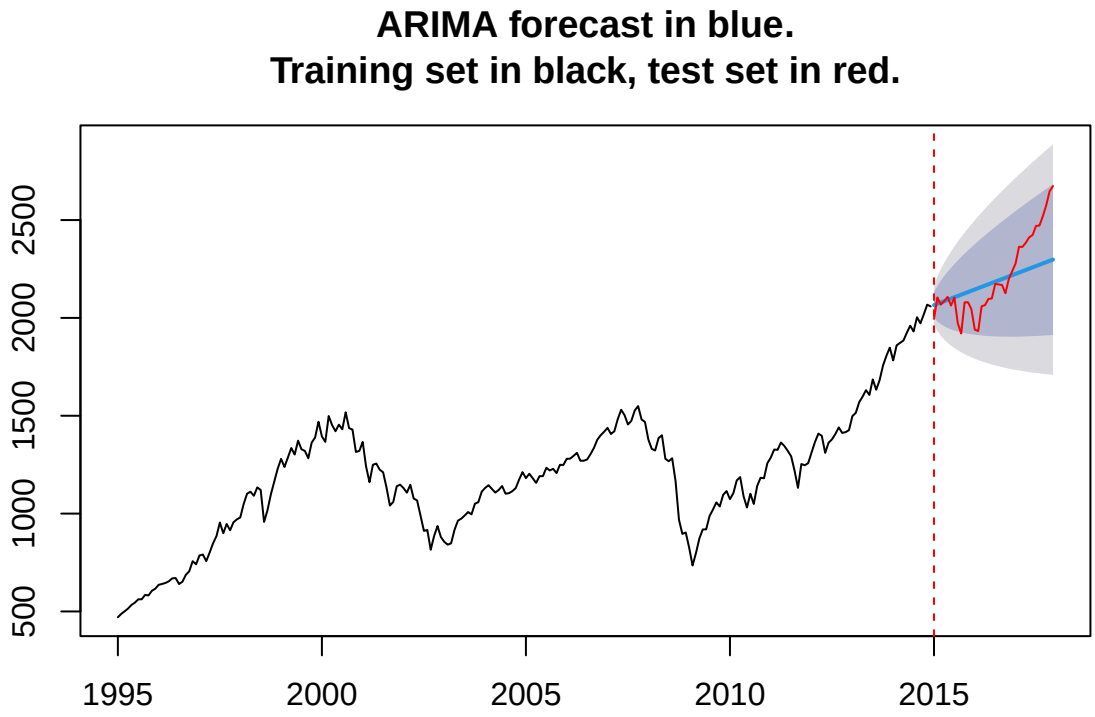
```
      test_set = tail(sp500_test, 36),
      percentage_error = 100*(tail(for_sp500_all$mean, 36) -
      tail(sp500_test, 36))/(tail(for_sp500_all$mean, 36)))
```

```
##          arima_forecast test_set percentage_error
## Jan 2015       2065.546  1994.99      3.415864979
## Feb 2015       2072.193  2104.50     -1.559091512
## Mar 2015       2078.839  2067.89      0.526692409
## Apr 2015       2085.485  2085.51     -0.001182963
## May 2015       2092.132  2107.39     -0.729313273
## Jun 2015       2098.778  2063.11      1.699462745
## Jul 2015       2105.424  2103.84      0.075249891
## Aug 2015       2112.071  1972.18      6.623391814
## Sep 2015       2118.717  1920.03      9.377708044
## Oct 2015       2125.363  2079.36      2.164495098
## Nov 2015       2132.010  2080.41      2.420248806
## Dec 2015       2138.656  2043.94      4.428775078
## Jan 2016       2145.303  1940.24      9.558678869
## Feb 2016       2151.949  1932.23     10.210230818
## Mar 2016       2158.595  2059.74      4.579612624
## Apr 2016       2165.242  2065.30      4.615725123
## May 2016       2171.888  2096.95      3.450364838
## Jun 2016       2178.534  2098.86      3.657241524
## Jul 2016       2185.181  2173.60      0.529962206
## Aug 2016       2191.827  2170.95      0.952499331
## Sep 2016       2198.473  2168.27      1.373836411
## Oct 2016       2205.120  2126.15      3.581207053
## Nov 2016       2211.766  2198.81      0.585781113
## Dec 2016       2218.413  2238.83     -0.920367647
## Jan 2017       2225.059  2278.87     -2.418418278
## Feb 2017       2231.705  2363.64     -5.911831219
## Mar 2017       2238.352  2362.72     -5.556247990
## Apr 2017       2244.998  2384.20     -6.200539551
## May 2017       2251.644  2411.80     -7.112834103
## Jun 2017       2258.291  2423.41     -7.311690660
## Jul 2017       2264.937  2470.30     -9.067051434
```

```
## Aug 2017       2271.583  2471.65     -8.807358782
## Sep 2017       2278.230  2519.36    -10.584110082
## Oct 2017       2284.876  2575.26    -12.708955504
## Nov 2017       2291.522  2647.58    -15.538036447
## Dec 2017       2298.169  2673.61    -16.336539828
```

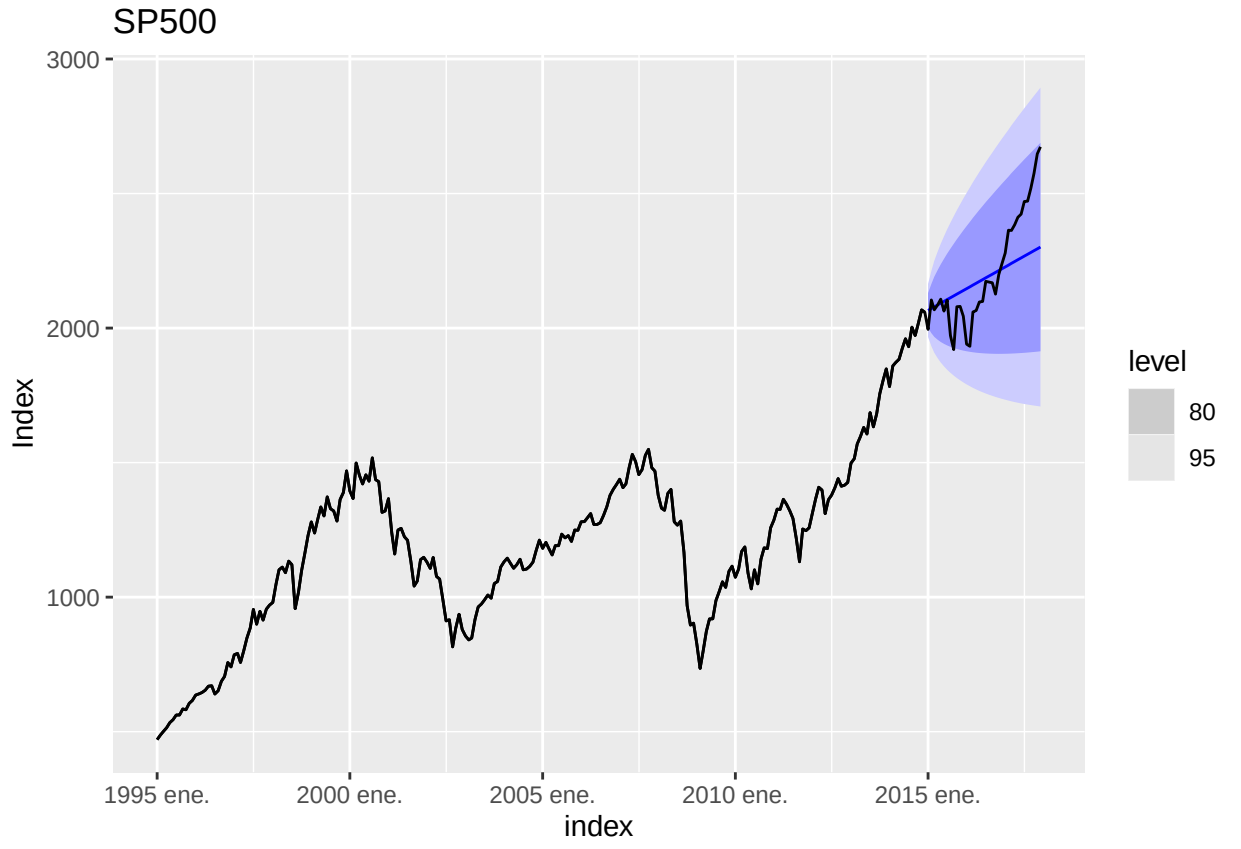
There are months in which the forecast looks OK, but in others the difference is more than 10%. It's easier if we see the forecast.

```
plot(for_sp500_all, main = "ARIMA forecast in blue.
Training set in black, test set in red.")
lines(sp500_test, col = "red")
abline(v = 2015, lty = 2, col = "red")
```



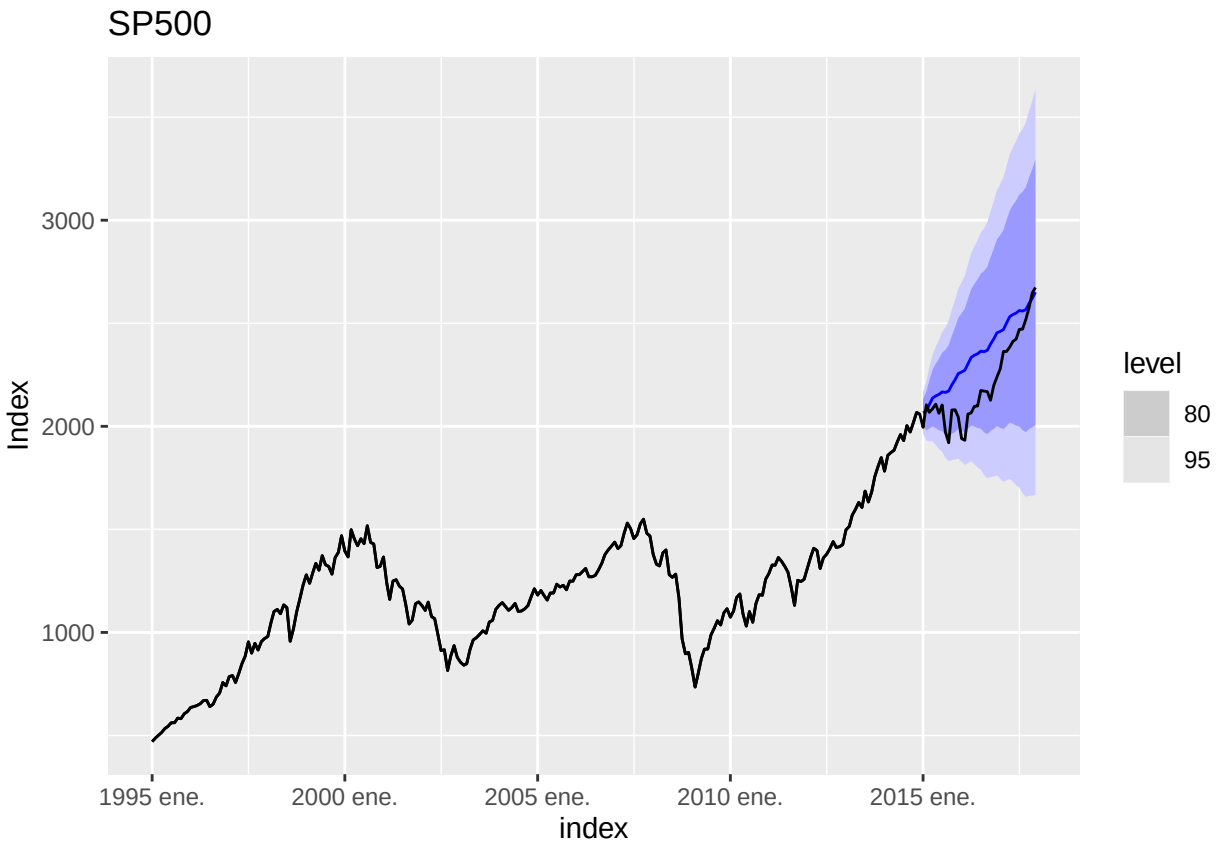
The ARIMA model overestimate the S&P500 at the first months, and then underestimate it. As a rule of thumb, long term forecasts are hard, they are usually not very accurate. This makes sense as it is easier to forecast tomorrow weather compared to 3 month forecast weather.

```
library(fpp3)
sp500_all.tsibble <- as_tsibble(sp500_all)

sp500_training.tsibble <- as_tsibble(sp500_training)

sp500_training.tsibble |>
model(AAN = ETS(sp500_training ~ error("A") + trend("A") + season("N"))) |>
  forecast(h = 36) |>
  autoplot(sp500_training.tsibble) +
  labs(title = "SP500",
       y = "Index") +
  guides(colour = guide_legend(title = "Forecast")) +
autolayer(sp500_all.tsibble)
```



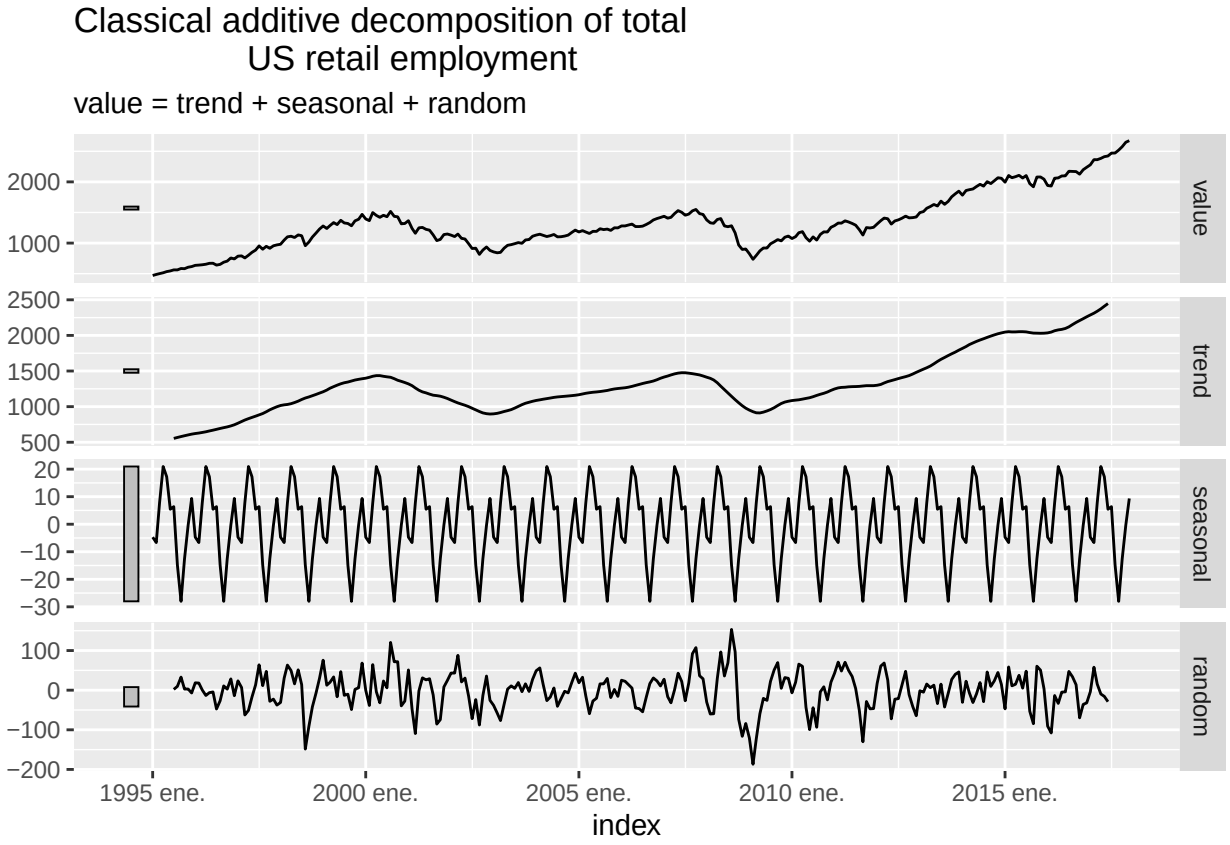

```
sp500_all <- ts(sp_500$sp_500, start = c(1995, 1), freq = 12)
sp500_all.tsibble <- as_tsibble(sp500_all)
```

```
sp500_training.tsibble |>
model(hw = ETS(sp500_training ~ error("A") + trend("A") + season("A")))|>
  forecast(h = 36) |>
  autoplot(sp500_training.tsibble) +
  labs(title = "SP500",
       y = "Index") +
  guides(colour = guide_legend(title = "Forecast")) +
autolayer(sp500_all.tsibble)
```




```
library(fpp3)
sp500_all <- ts(sp_500$sp_500, start = c(1995, 1), freq = 12)
sp500_all.tsibble <- as_tsibble(sp500_all)

sp500_all.tsibble |>
  model(
    classical_decomposition(value, type = "additive")
  ) |>
```

```
  components() |>
  autoplot() +
  labs(title = "Classical additive decomposition of total
                  US retail employment")
```



We can compare ARIMA forecast with TBATS forecast. The TBATS model combines several components, making them a very good choice for forecasting. It constitutes the following elements:

- T: Trigonometric terms for seasonality.
- B: Box-Cox transformations for heterogeneity.
- A: ARMA errors for short-term dynamics.
- T: Trend.
- S: Seasonal (including multiple and non-integer periods).

This is the implementation in R.

```
# 36-month forecast.
for_tbats <- forecast::forecast(sp500_training, h = 36)
```

```
df_tbats = as.data.frame(for_tbats)
df_tbats <- ts(df_tbats$`Point Forecast`, start = c(2015, 1), freq = 12)
# Evaluate percentage error.
cbind(TBATS_forecast = df_tbats,
      test_set=tail(sp500_test, 36),
      percentage_error = 100*((df_tbats) -
      tail(sp500_test, 36))/(df_tbats))
```

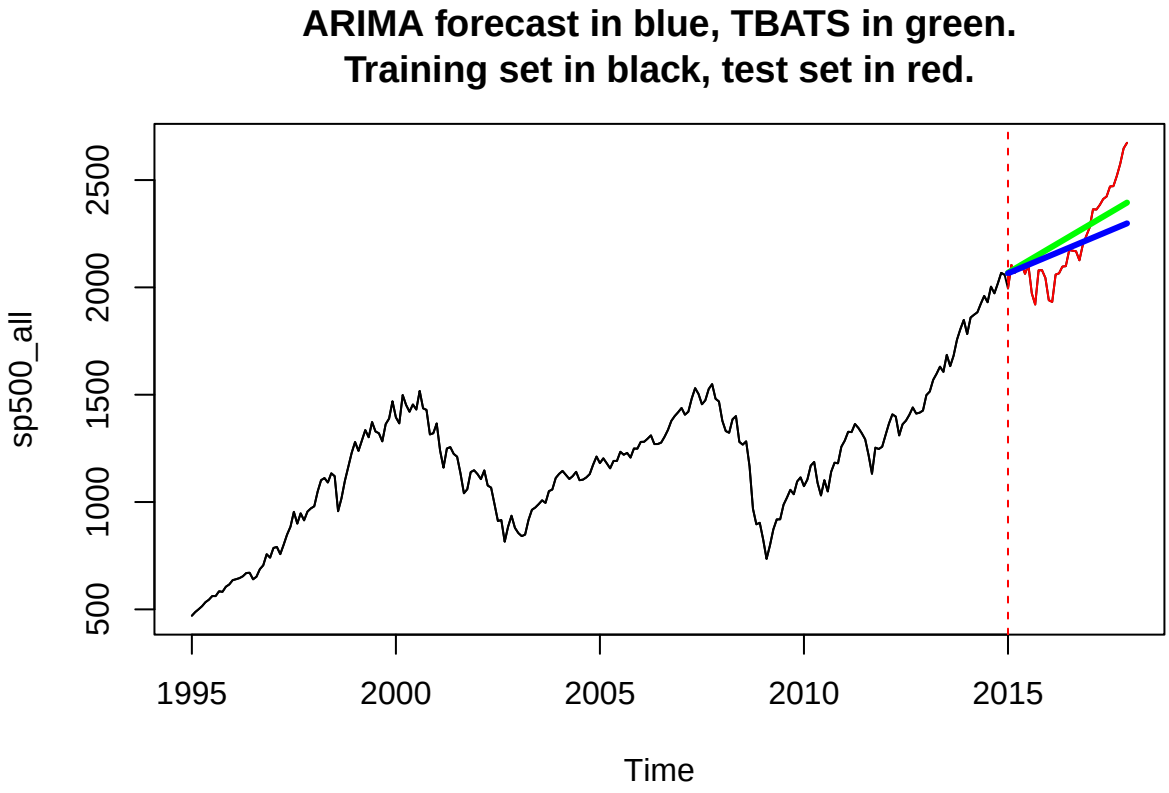
```
##          TBATS_forecast test_set percentage_error
## Jan 2015       2068.237  1994.99        3.5414982
## Feb 2015       2077.571  2104.50       -1.2961700
## Mar 2015       2086.906  2067.89        0.9111982
## Apr 2015       2096.240  2085.51        0.5118843
## May 2015       2105.575  2107.39       -0.0861980
## Jun 2015       2114.910  2063.11        2.4492498
## Jul 2015       2124.244  2103.84        0.9605314
## Aug 2015       2133.579  1972.18        7.5646921
## Sep 2015       2142.913  1920.03       10.4009471
## Oct 2015       2152.248  2079.36        3.3865893
## Nov 2015       2161.582  2080.41        3.7552389
## Dec 2015       2170.917  2043.94        5.8490097
## Jan 2016       2180.252  1940.24       11.0084399
## Feb 2016       2189.586  1932.23       11.7536497
## Mar 2016       2198.921  2059.74        6.3295087
## Apr 2016       2208.255  2065.30        6.4736814
## May 2016       2217.590  2096.95        5.4401450
## Jun 2016       2226.925  2098.86        5.7507358
## Jul 2016       2236.259  2173.60        2.8019631
## Aug 2016       2245.594  2170.95        3.3240160
## Sep 2016       2254.928  2168.27        3.8430677
## Oct 2016       2264.263  2126.15        6.0996950
## Nov 2016       2273.598  2198.81        3.2893937
## Dec 2016       2282.932  2238.83        1.9318205
## Jan 2017       2292.267  2278.87        0.5844306
## Feb 2017       2301.601  2363.64       -2.6954482
## Mar 2017       2310.936  2362.72       -2.2408217
```

```
## Apr 2017       2320.271  2384.20       -2.7552537
## May 2017       2329.605  2411.80       -3.5282736
## Jun 2017       2338.940  2423.41       -3.6114702
## Jul 2017       2348.274  2470.30       -5.1963968
## Aug 2017       2357.609  2471.65       -4.8371428
## Sep 2017       2366.944  2519.36       -6.4393814
## Oct 2017       2376.278  2575.26       -8.3736759
## Nov 2017       2385.613  2647.58      -10.9811328
## Dec 2017       2394.947  2673.61      -11.6354432
```

After 36 months (Dec 2017), ARIMA error is -16.33%, while TBATS is -11.6%. These might not be very impressive values. And in fact, we can do better by implementing more elaborated techniques, including machine learning. However, a 36 month forecast with a percentage error of -11.6% looks not so bad.

We can see all results together.

```
# Plot all previous results.
plot(sp500_all, main = "ARIMA forecast in blue, TBATS in green.
Training set in black, test set in red.")
lines(sp500_training)
lines(sp500_test, col = "red")
lines(df_tbats, col = "green", lwd = 3)
lines(for_sp500_all$mean, col = "blue", lwd = 3)
abline(v = 2015, lty = 2, col = "red")
```



In finance we are interested in forecasting because these techniques allow us to have some certainty about the future. Since we take investment decisions today looking for a return in the future, we need to have some tools to anticipate the future the best way we can.

The story so far is the following. We care about firm performance, and the evolution of the stock return is a good indicator. We care about the determinants of stock returns and we propose that the market and other risk factors determine the stock returns. Then we care about the evolution of the market and risk factors as these anticipate changes in the stock returns. Forecasting techniques are only one alternative to have some idea about the evolution of asset prices, including a stock market index like the S&P500. We know that the S&P500 was 2,058.90 in December 2014, and our best forecast for Dec 2017 (36 months in the future), is 2,394.947 and in reality, it was 2,673.61 (a percentage error of -11.6%). This is how we can anticipate firm performance.

The next section deals with the actual investment problem. In which assets should we invest? How much should we invest in each asset? Those questions can be addressed by learning asset allocation or portfolio selection techniques.

5 Asset allocation.

In finance, we are expected to take good financing and investment decisions. Asset allocation and portfolio theory are areas that show how we can take optimal investment decisions. In general, optimization is the act of achieving the best possible result under given circumstances. In our context, the best possible result is the highest return given a risk level, or alternatively the lowest risk given a return level. The "given circumstances" refer to a set of restrictions or constraints that we can face in the market. These restrictions are in the form of a maximum level of one asset in a portfolio, constraints about short sales, etc. Computers can solve complex optimization problems so it makes sense to use R and conduct asset allocation and tasks and estimate portfolio optimization models.

In principle, we all know that the popular wisdom suggests *not to put all the eggs in the same basket*. This suggestion makes sense as we should diversify as diversification is a way to manage the risk of losing all the eggs if something bad happens. For example, imagine I invest in two firms, one that sells ice cream and another that sells hot chocolate. Assume that as an exchange of my investment, I get some returns based on these firms' sales. My diversification in these two firms seems reasonable as the weather (relevant risk factor) affects the sales of these two companies in an opposite way. When it is hot, sales increases in the ice cream shop and decreases in the hot chocolate shop; and when it is cold, sales increases in the hot chocolate shop and decreases in the ice cream shop. Then, my revenues will be fairly constant every year, and this is desirable as I reduce the volatility of my revenues, they become less risky, the standard deviation of my revenues is small. You may remember our previous discussion about the correlation of Mastercard and Visa, the correlation of these firms' stock returns is 0.8480487 and we argue that this mean that they not only share the same risk factors, but they also react almost the same to changes in these risk factors, and this is why they behave similar. In the ice cream and hot chocolate shop example, we could guess that the sales' correlation of both shops have a strong negative correlation let's say -0.9. This means that the sales of both shops behave opposite and they are affected differently by the risk factor called weather.

If the popular wisdom suggests *not to put all the eggs in the same basket*, then, the interesting question is, how many eggs should we put in each basket? This is the equivalent to an investment recommendation in the form of how much to invest in each individual asset. We call portfolio to the new asset formed by several (smaller) investments in single assets. And we call a portfolio weight to the percentage invested in single assets. Asset allocation models allow us to estimate these optimal portfolio weights. Let's see an extreme example first. Imagine my current investment is 1% in the ice cream shop and 99% in the hot chocolate

shop. This does not look like a good idea because these portfolio weights make the portfolio highly vulnerable to hot weather. After implementing an asset allocation optimization, the portfolio weights could be 60% of my money invested in the ice cream shop and 40% in the hot chocolate shop.

5.1 The single-period problem.

The single-period problem is the simplest framework to implement asset allocation models. Here, we assume that the investor has some historical information about the assets in $t = 0$ (today), she makes an investment decision today, and she expects a return in $t = 1$, end of story. Let's see how we should distribute the 100% of available money to invest in four stocks using the *PortfolioAnalytics* R package.

We need data to start. For convenience, we use the monthly returns of the FANG database.

```
FANG_monthly_returns <- FANG |>
    group_by(symbol) |>
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "monthly",
                 type       = "arithmetic")
FANG_monthly_returns
```

```
## # A tibble: 192 x 3
## # Groups:   symbol [4]
##    symbol date       monthly.returns
##    <chr>  <date>               <dbl>
##  1 FB     2013-01-31         0.106
##  2 FB     2013-02-28        -0.120
##  3 FB     2013-03-28        -0.0613
##  4 FB     2013-04-30         0.0856
##  5 FB     2013-05-31        -0.123
##  6 FB     2013-06-28         0.0218
##  7 FB     2013-07-31         0.479
##  8 FB     2013-08-30         0.122
##  9 FB     2013-09-30         0.217
## 10 FB     2013-10-31        -0.000398
## # ... with 182 more rows
```

The format of *FANG_monthly_returns* is not compatible with the *PortfolioAnalytics* package as the data is currently tidy. So, we have to transform the way our database looks. Originally, we have a column called "symbol" and we need each stock to have their corresponding name. We can implement this change easily.

```
# Load the relevant packages.
library(PortfolioAnalytics)
library(ROI)
library(ROI.plugin.glpk)
library(ROI.plugin.quadprog)
library(dplyr)
library(tbl2xts)

fang <-
  FANG_monthly_returns |> tbl_xts(spread_by = "symbol")
head(fang)
```

```
##                    FB         AMZN        NFLX        GOOG
## 2013-01-31  0.10642857  0.031829319  0.79589177  0.04485307
## 2013-02-28 -0.12040026 -0.004632810  0.13822318  0.06022315
## 2013-03-28 -0.06128440  0.008400504  0.00638028 -0.00874938
## 2013-04-30  0.08561376 -0.047581495  0.14153635  0.03825280
## 2013-05-31 -0.12315448  0.060635964  0.04711437  0.05657498
## 2013-06-28  0.02176587  0.031537851 -0.06700557  0.01050246
```

As you can confirm, the database *fang* is the same as we only changed the format. Following our discussion about the correlation, we calculate the correlation matrix for these 4 stocks.

```
cor(fang)
```

```
##             FB      AMZN      NFLX      GOOG
## FB   1.0000000 0.1846197 0.2182079 0.2468989
## AMZN 0.1846197 1.0000000 0.3118020 0.6171376
## NFLX 0.2182079 0.3118020 1.0000000 0.3586214
## GOOG 0.2468989 0.6171376 0.3586214 1.0000000
```

The lowest correlation is between FB and AMZN (0.1846197). This means that both firms exhibit a weak linear relationship. On the other hand, we have GOOG and AMZN with a correlation of 0.6171376, which suggest a stronger linear relationship between these stock

returns. In principle, the lower the correlation of our assets, the greater the diversification possibilities when forming a portfolio. Remember the ice cream and the hot chocolate shop example, we assumed a correlation of -0.9 and we were supposed to decrease the volatility of our sales. In practice, it is not very easy to find negative correlated assets. However, we can achieve diversification gains as long as the correlation value is less than $+1$.

Let's continue with our main objective. We are interested in an investment recommendation (how much to invest in each of the four assets). We first define a portfolio specification.

```
# Create the portfolio specification
port_spec <- portfolio.spec(colnames(fang))
# Add a full investment constraint such that the weights sum to 1
port_spec <- add.constraint(portfolio =
                              port_spec, type = "full_investment")
# Add a long only constraint such that the
# weight of an asset is between 0 and 1
port_spec <- add.constraint(portfolio = port_spec, type = "long_only")
# Add an objective to minimize portfolio standard deviation
port_spec <- add.objective(portfolio = port_spec,
                           type = "risk",
                           name = "StdDev")
# Add an objective to minimize portfolio standard deviation
port_spec <- add.objective(portfolio = port_spec,
                           type = "return",
                           name = "mean")
port_spec
```

```
## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## portfolio.spec(assets = colnames(fang))
##
## Number of assets: 4
## Asset Names
## [1] "FB"   "AMZN" "NFLX" "GOOG"
##
```

```
## Constraints
## Enabled constraint types
##       - full_investment
##       - long_only
##
## Objectives:
## Enabled objective names
##       - StdDev
##       - mean
```

The portfolio specification indicates that we have four assets available, these are: FB, AMZN, NFLX and GOOG. Constraints indicates that we are expected to spend 100% of the funds available (full investment). This is, the sum of the portfolio weights of the four assets should be 100%. The constraint long refers that individual portfolio weights have to be positive. The StdDev is an objective because we are interested in minimizing the standard deviation (risk), and we want to maximize the mean.

The alternative of the constraint *long_only* is to allow the model to suggest negative portfolio weights. Negative portfolio weights represent short sales. Short selling is an investment strategy that speculates on the decline in a stock. Sometimes short selling is not allowed so the *long_only* constraint will deliver only positive portfolio weights. The *optimize.portfolio* function allows us to solve our portfolio problem by taking a variety of methods, constraints and objectives. Let's run two of them (random and ROI) and then we can discuss their differences.

```
# Solve the optimization problem.
set.seed(13)
opt_rand <- optimize.portfolio(fang, portfolio = port_spec,
                               optimize_method = "random", trace = TRUE)
opt_roi <- optimize.portfolio(fang, portfolio = port_spec,
                                  optimize_method = "ROI", trace = TRUE)
# Show optimization results.
opt_rand
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
```

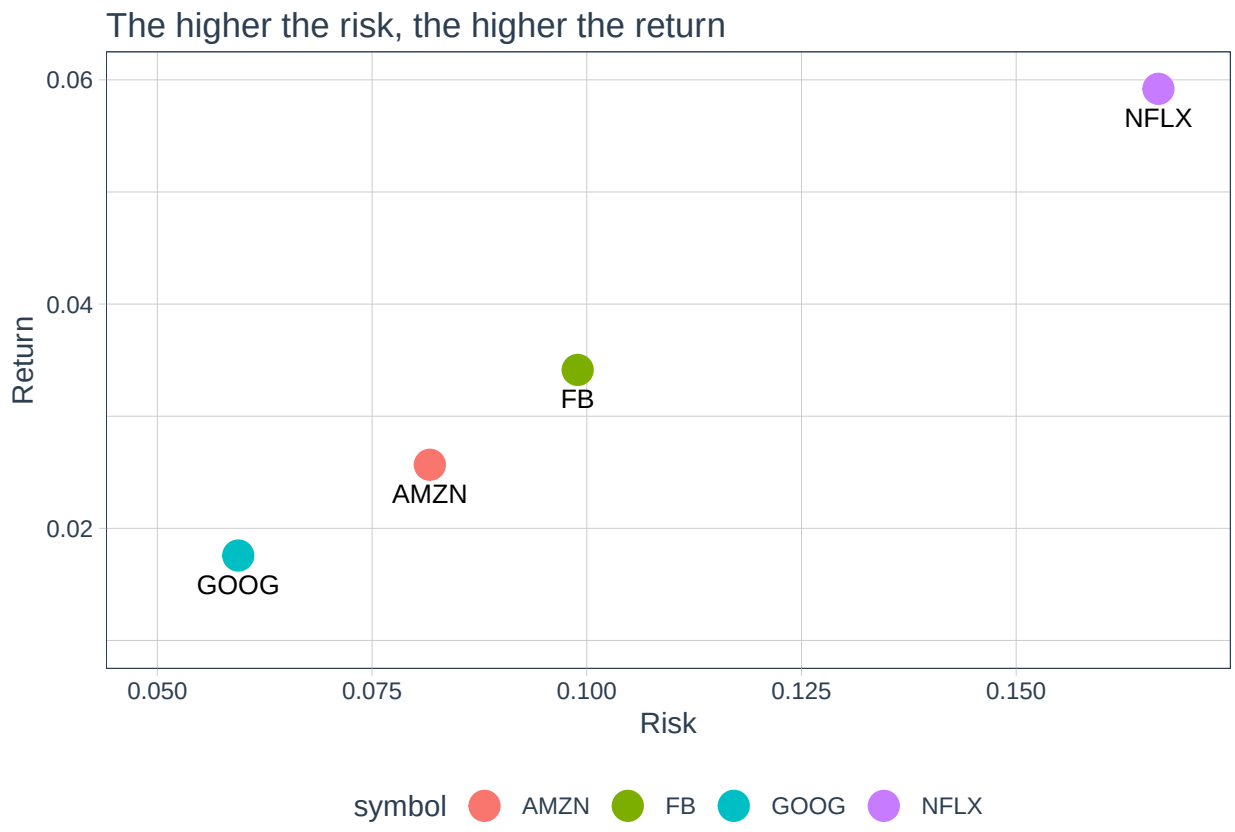
```
## Call:
## optimize.portfolio(R = fang, portfolio = port_spec, optimize_method = "random",
##     trace = TRUE)
##
## Optimal Weights:
##    FB  AMZN  NFLX  GOOG
## 0.298 0.182 0.050 0.470
##
## Objective Measures:
##  StdDev
## 0.05807
##
##
##    mean
## 0.02606
```

```
opt_roi
```

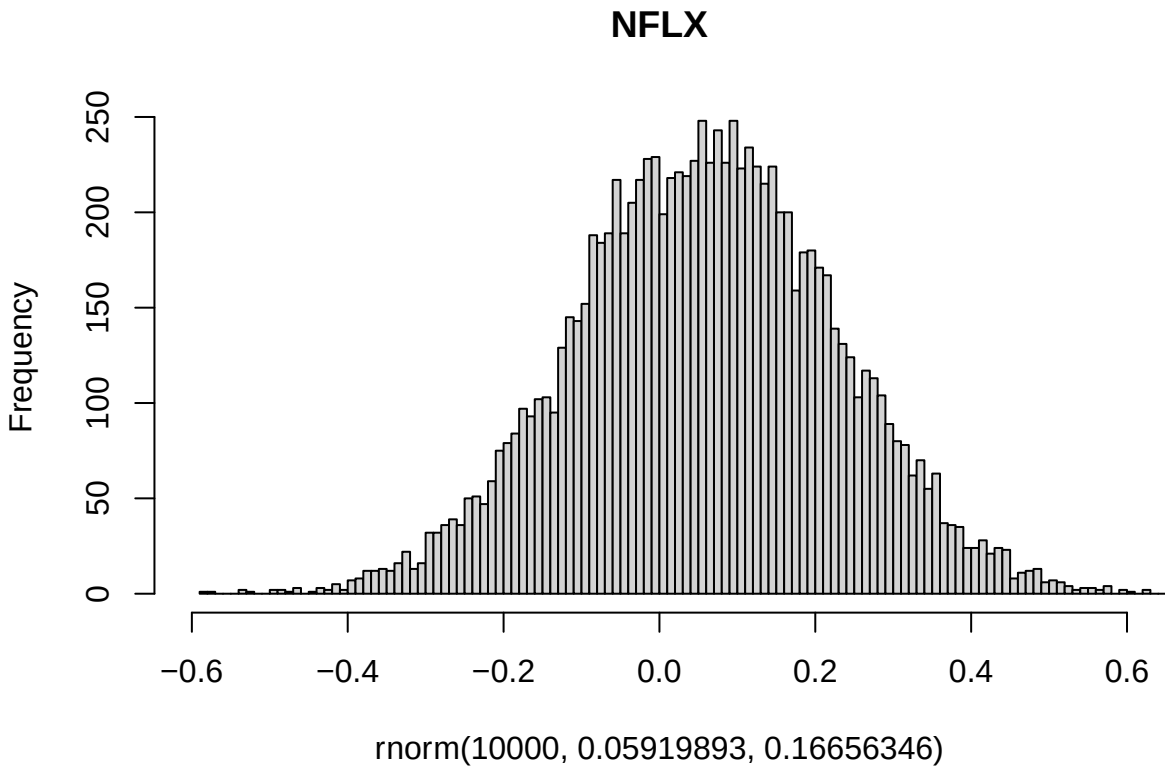
```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = fang, portfolio = port_spec, optimize_method = "ROI",
##     trace = TRUE)
##
## Optimal Weights:
##     FB   AMZN   NFLX   GOOG
## 0.3827 0.0000 0.6173 0.0000
##
## Objective Measure:
##   mean
## 0.0496
##
##
## StdDev
## 0.1171
```

The output above shows the portfolio weights, return and risk of the two optimal portfolios. Before showing the results of this optimization in a plot, let's look at the initial situation. In particular, the individual assets before adding any portfolio.

```
ggplot(FANG_stats, aes(x = sd, y = mean, color = symbol)) +
  geom_point(size = 5) + geom_text(aes(label = paste0(symbol)),
  vjust = 2, color = "black", size = 3.5) +
  xlim(0.05, 0.169) + ylim(0.01, 0.06) +
  labs(title = "The higher the risk, the higher the return",
       x = "Risk", y = "Return") + theme_tq()
```



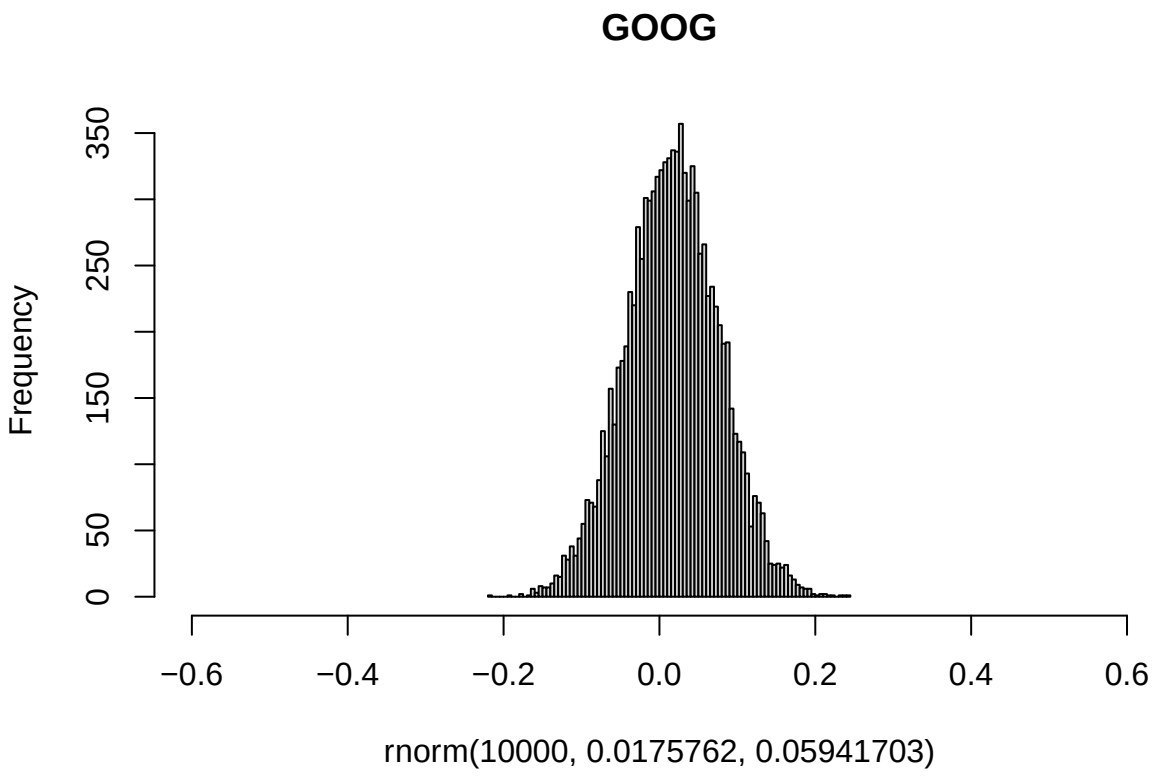

```
hist(rnorm(10000, 0.05919893, 0.16656346), 100, xlim = c(-0.6, 0.6),
     main = "NFLX")
```




```
sum(rnorm(10000, 0.05919893, 0.16656346) < -0.1)/10000
```

```
## [1] 0.1641
```

```
hist(rnorm(10000, 0.01757620, 0.05941703), 100, xlim = c(-0.6, 0.6),
     main = "GOOG")
```




```
sum(rnorm(10000, 0.01757620, 0.05941703) < -0.1)/10000
```

```
## [1] 0.0239
```

Do you remember this plot above from previous sections? This is basically the four individual assets in the mean-variance space. The proposal of asset allocation is that instead of choosing one individual asset to invest in, we can do better by creating a portfolio (investing some specific quantities in each individual asset). As we argue before, the role of the optimization process is precisely to determine which allocation is the most efficient.

For example, I do not think you will be happy with a portfolio whose return is 0.035 (3.5% monthly return) and a risk of 0.15 (15% standard deviation). This is because you could do better than that. For example, Facebook has a similar return with a 10% risk. But what if I propose a portfolio with a return of 4% and a risk of 5%? That would be great compared with the individual assets alternatives as it is a 0.8 return per unit of risk.

Remember these are the return per unit of risk of the individual assets.

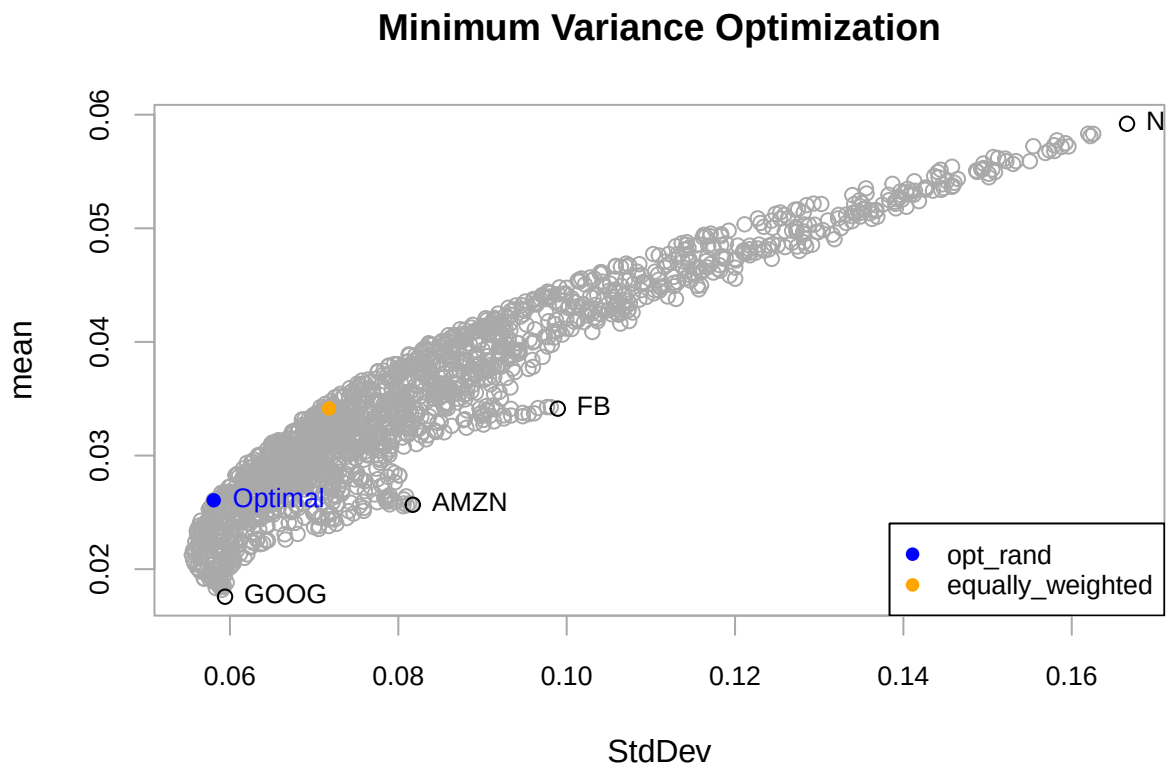
```
# List of sr, individual assets.
FANG_monthly_sr <- FANG_stats |>
```

```
  select(symbol, sr)
FANG_monthly_sr
```

```
## # A tibble: 4 x 2
##   symbol    sr
##   <chr>  <dbl>
## 1 AMZN   0.314
## 2 FB     0.345
## 3 GOOG   0.296
## 4 NFLX   0.355
```

The random technique (Random Portfolios Optimization) evaluates many investment recommendation alternatives, each one is one portfolio. In principle, the technique should recommend a portfolio which leads to a more attractive return per unit of risk alternative than investing in individual assets.

```
# Plot results.
chart.RiskReward(opt_rand, risk.col = "StdDev",
                 main = "Minimum Variance Optimization",
                 xlim = c(0, 10),
                 return.col = "mean", chart.assets = TRUE)
legend("bottomright", legend = c("opt_rand", "equally_weighted"),
       col = c("blue", "orange"), pch = 19, cex = 0.8)
```



According to our portfolio specification and assets, gray circles represent available portfolios or feasible portfolios in the sense that they are possibilities given the individual assets. Note that there are some available portfolios with a lower risk than GOOG. In fact, the algorithm suggests the blue portfolio as the optimal portfolio. This blue portfolio is attractive because it has the same risk than GOOG, but it has higher expected return. Moreover, our optimal blue portfolio has a similar return as AMZN, but with less risk. So, apparently the blue alternative is an attractive investment recommendation. Note that the gray portfolios form a kind of frontier in this mean-variance plot. This frontier suggests that it is not possible to invest in a portfolio at the left of this frontier. The optimal portfolio lies just in the frontier.

See the blue optimal portfolio again. We would prefer any other portfolio located in the top (higher return) or at the left (lower risk), but that is impossible given the data. On the other hand, it is possible to achieve a portfolio located below (low return) or at the right (high risk), but that is not optimal. This is why optimal portfolios are those that lie in the frontier, those are the portfolios located in the extreme high and left of this mean-variance plot.

Then, what can we do to invest in the blue optimal portfolio?

```
# Extract weight, risk and return.
opt_rand$weights
```

```
##    FB  AMZN  NFLX  GOOG
## 0.298 0.182 0.050 0.470
```

```
sum(opt_rand$weights)
```

```
## [1] 1
```

```
opt_rand$opt_values
```

```
## $StdDev
##            [,1]
## [1,] 0.05807126
## attr(,"names")
## [1] "StdDev"
##
## $mean
##       mean
## 0.02606294
```

```
opt_rand$opt_values$mean / opt_rand$opt_values$StdDev
```

```
##           [,1]
## [1,] 0.4488096
```

According to the opt_rand portfolio, we have to invest 29.8% in FB, 18.2% in AMZN, 0.5% in NFLX, and 47% in GOOG. The optimal portfolio has a 0.4488096 return per unit of risk. This is clearly a better alternative compared with investing in individual assets as we show below.

```
# Add the new portfolio sr to the list.
FANG_monthly_sr <-
  add_row(FANG_monthly_sr, symbol = "opt_rand",
          sr = opt_rand$opt_values$mean / opt_rand$opt_values$StdDev) |>
  arrange(sr)
FANG_monthly_sr
```

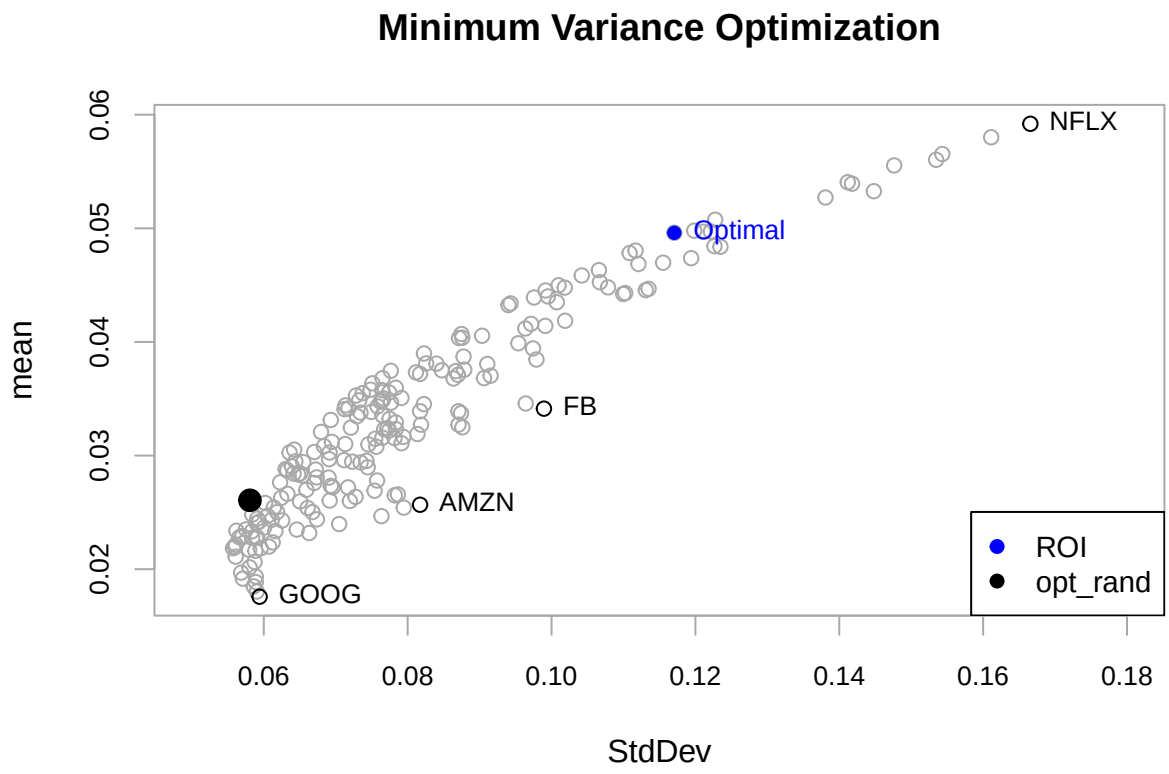
```
## # A tibble: 5 x 2
##   symbol   sr[,1]
```

```
##   <chr>     <dbl>
## 1 GOOG      0.296
## 2 AMZN      0.314
## 3 FB        0.345
## 4 NFLX      0.355
## 5 opt_rand  0.449
```

The yellow portfolio is also interesting as it represents an equally weighted portfolio. This is, 25% in each individual asset. We usually take this as a benchmark portfolio. You do not have to implement any optimization process to find out the 25% as it is basically 1/4 or 1 over the number of assets. Interestingly, this equally weighted portfolio is close to being optimal but if you look closer, it is not optimal.

A second criterion is to optimize according to ROI (R Optimization Infrastructure for linear and quadratic programming solvers).

```
# Plot results.
chart.RiskReward(opt_roi, risk.col = "StdDev",
                 main = "Minimum Variance Optimization",
                 xlim = c(0.05, 0.18),
                 return.col = "mean", rp = TRUE, chart.assets = TRUE)
points(0.05807126, 0.02606294, pch = 19, cex = 1.5, col = "black")
legend("bottomright", legend = c("ROI", "opt_rand"),
       col = c("blue", "black"), pch = 19, cex = 0.9)
```



A different optimization criterion leads to a different optimal portfolio. However, they are both optimal as they rely just on the efficient frontier.

```
# Extract weight, risk and return.
opt_roi$weights
```

```
##           FB         AMZN         NFLX         GOOG
## 3.827196e-01 0.000000e+00 6.172804e-01 2.807585e-16
```

```
sum(opt_roi$weights)
```

```
## [1] 1
```

```
opt_roi$opt_values
```

```
## $mean
##       mean
## 0.04960399
##
## $StdDev
```

```
##    StdDev
## 0.1170653
```

```
opt_roi$opt_values$mean / opt_roi$opt_values$StdDev
```

```
##      mean
## 0.4237291
```

The ROI criterion recommends investing 38.3% in FB and 61.7% in NFLX. This led to a return per unit of risk of 0.4237292. Let's compare the alternatives.

```
# Add the new portfolio sr to the list.
FANG_monthly_sr <-
  add_row(FANG_monthly_sr, symbol = "opt_roi",
          sr = opt_roi$opt_values$mean / opt_roi$opt_values$StdDev) |>
arrange(sr)
FANG_monthly_sr
```

```
## # A tibble: 6 x 2
##   symbol   sr[,1]
##   <chr>     <dbl>
## 1 GOOG      0.296
## 2 AMZN      0.314
## 3 FB        0.345
## 4 NFLX      0.355
## 5 opt_roi   0.424
## 6 opt_rand  0.449
```

According to the return per unit of risk, we should choose the opt_rand portfolio because it has a 0.449 return per unit of risk. The optimization process succeeds at proposing a better investment strategy compared with the individual assets. In practice, if you were interested to form a diversified portfolio, you could implement an analysis like this one, with some more extensive tests, but very similar to this analysis. As an individual, you could contact your broker or your financial institution at $t = 0$ to ask them to invest your money according to the opt_rand portfolio: 29.8% in FB, 18.2% in AMZN, 0.5% in NFLX, and 47% in GOOG, and you should expect to get the opt_rand portfolio risky return at $t = 1$.

The expected return in $t = 0$ of your opt_rand portfolio is 0.02606294. The value of 0.02606294 is simply the sum of the opt_rand portfolio weights multiplied by the individual asset returns. But, what will be your realized return at $t = 1$? It is hard to wait until

$t = 1$ to know your return (or loss). When we introduced the forecasts methods, we used the training and test periods to evaluate what might happen in the future. We can do the same here, but I propose a different approach now. In particular, I propose to simulate the future.

In this case we invest in $t = 0$ or 2016-12-30 and the future is $t = 1$ or 2017-01-31. There are many ways to conduct the simulation. Let's keep it simple and use the historical mean and standard deviation information we have.

```
FANG_stats_all <- FANG_stats |>
  select(symbol,mean, sd) |>
  add_row(symbol = "opt_rand",
          mean = opt_rand$opt_values$mean, sd = opt_rand$opt_values$StdDev) |>
  add_row(symbol = "opt_roi",
          mean = opt_roi$opt_values$mean, sd = opt_roi$opt_values$StdDev)
FANG_stats_all
```

```
## # A tibble: 6 x 3
##   symbol     mean sd[,1]
##   <chr>     <dbl>  <dbl>
## 1 AMZN     0.0257 0.0817
## 2 FB       0.0341 0.0989
## 3 GOOG     0.0176 0.0594
## 4 NFLX     0.0592 0.167
## 5 opt_rand 0.0261 0.0581
## 6 opt_roi  0.0496 0.117
```

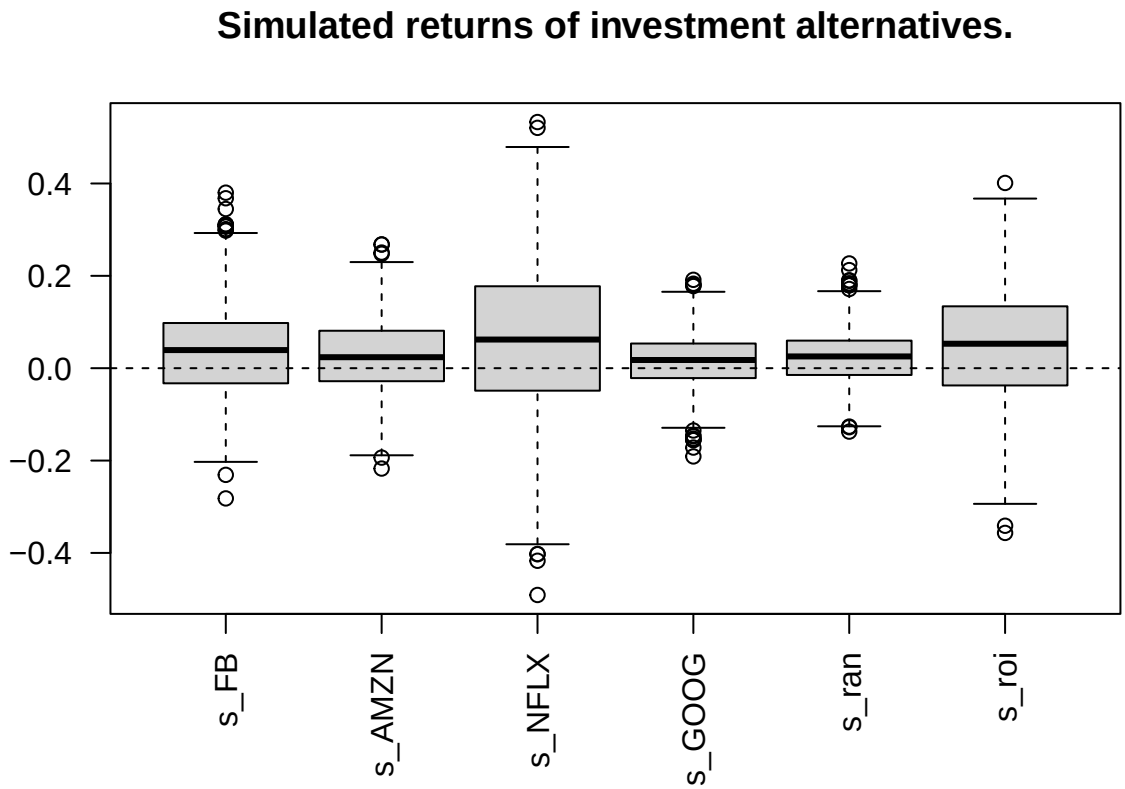
Let's simulate 1,000 observations of each individual asset and portfolios. We assume that the assets behave as a normal with mean and standard deviation as we show in the table above. This approach can be interpreted as if we were simulating 1,000 alternative values for 2017-01-31. By doing this, we could have a sense about what will be the most likely value at $t = 1$.

```
# Number of simulations.
sim = 1000
set.seed (7)
# Simulation per stock and portfolio.
s_AMZN <- rnorm(sim, 0.02566966, 0.08172605)
s_FB <- rnorm(sim, 0.03412852, 0.09894324)
```

```
s_GOOG <- rnorm(sim, 0.01757620, 0.05941703)
s_NFLX <- rnorm(sim, 0.05919893, 0.16656346)
s_ran <- rnorm(sim, opt_rand$opt_values$mean, opt_rand$opt_values$StdDev)
s_roi <- rnorm(sim, opt_roi$opt_values$mean, opt_roi$opt_values$StdDev)
```

The simulation is done. Now let's visualize the results in a boxplot. Boxplots show the distribution of the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set.

```
# The boxplot.
b = cbind(s_FB, s_AMZN, s_NFLX, s_GOOG, s_ran, s_roi)
boxplot(b, las = 2, main = "Simulated returns of investment alternatives.")
abline(h = 0, lty = 2)
```



Note that the opt_rand portfolio is well diversified and opt_roi is more risky. This plot reveals a glimpse to the future assuming that the assets follow a normal distribution. This assumption is not as bad and it is useful to simplify the analysis. Are we satisfied with this glimpse to the future? Sometimes it is the best you can have at $t = 0$. In this case, we

fortunately know what really happened with these stocks on 2017-01-31. Let's evaluate the returns that really happened at $t = 1$ for the individual and portfolios.

First, we need the 2017-01-31 actual or realized returns. We need to download the data as the FANG database ends at 2016-12-30.

```
# Download the 2017-01-31 individual asset returns.
r_stocks <- c("FB", "AMZN", "NFLX", "GOOG") |>
  tq_get(get  = "stock.prices", from = "2016-12-31",
         to   = "2017-01-31") |>
  group_by(symbol) |>
  tq_transmute(select     = adjusted,
               mutate_fun = periodReturn,
               period     = "monthly",
               col_rename = "R_stocks") |>
  tbl_xts(spread_by = "symbol")
r_stocks
```

```
##                    FB     AMZN      NFLX       GOOG
## 2017-01-30 0.1208283 0.101782 0.1076947 0.02058157
```

These are the realized monthly returns at $t = 1$. Please note that we have not introduced these returns before. Our model and portfolios ignores these returns. Now, we need to evaluate the realized return of both optimal portfolios. This is done by adding the multiplication of weights and individual realized returns. This approach is commonly known as out-of-sample evaluation.

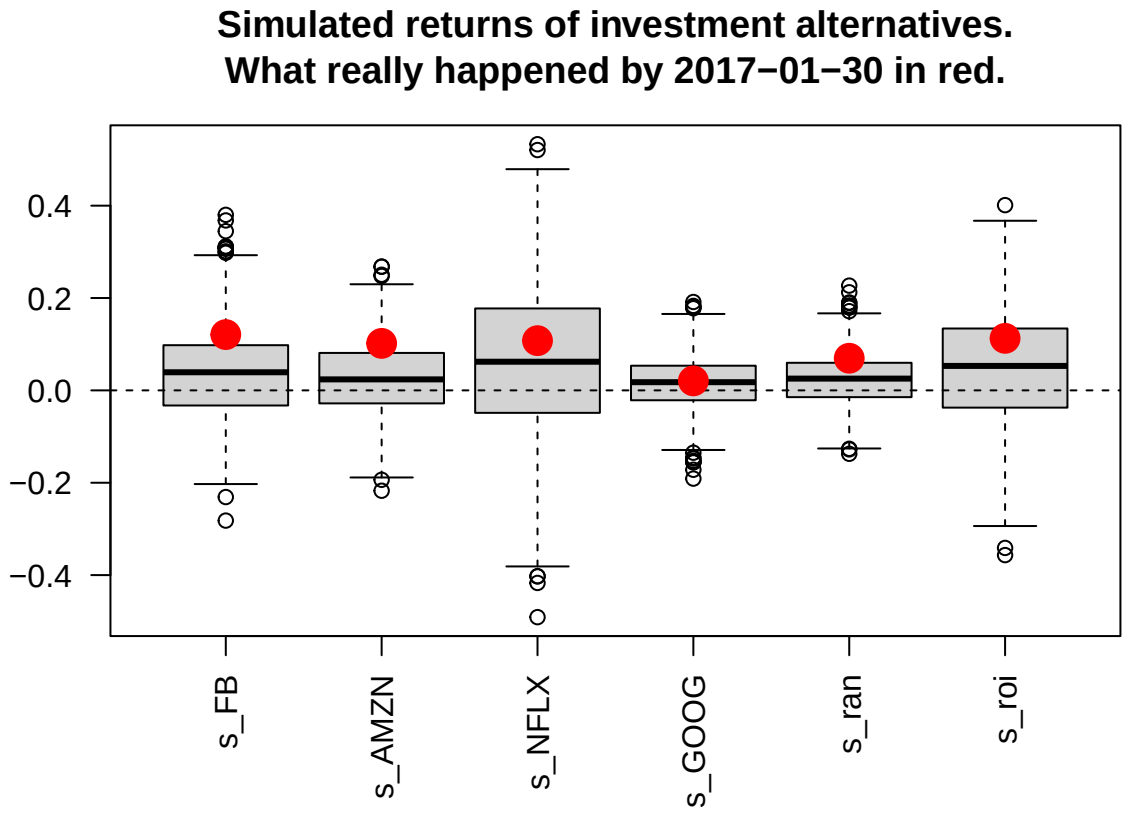
```
# Calculate realized portfolio returns.
realized_ret <- c(rand = sum(opt_rand$weights*r_stocks),
                  roi = sum(opt_roi$weights*r_stocks))

expected_ret <- c(rand = opt_rand$opt_values$mean,
                  roi = opt_roi$opt_values$mean)
data.frame(cbind(realized_ret, expected_ret))
```

```
##      realized_ret expected_ret
## rand   0.06958923   0.02606294
## roi    0.11272121   0.04960399
```

In sum, the expected monthly return in $t = 0$ of opt_rand portfolio is 0.02606294, and the

realized monthly return in $t = 1$ is 0.06958923. Not bad. Let's see the whole thing now. This is, the distribution of the simulation and the realized returns in red.

```
# The simulation.
boxplot(b, las = 2, main = "Simulated returns of investment alternatives.
What really happened by 2017-01-30 in red.")
abline(h=0, lty = 2)
# The realized returns in red.
points(1, 0.1208283,  col = "red", pch = 19, cex = 2)
points(2, 0.101782,   col = "red", pch = 19, cex = 2)
points(3, 0.1076947,  col = "red", pch = 19, cex = 2)
points(4, 0.02058157, col = "red", pch = 19, cex = 2)
points(5, sum(opt_rand$weights*r_stocks), col = "red", pch = 19, cex = 2)
points(6, sum(opt_roi$weights*r_stocks),  col = "red", pch = 19, cex = 2)
```

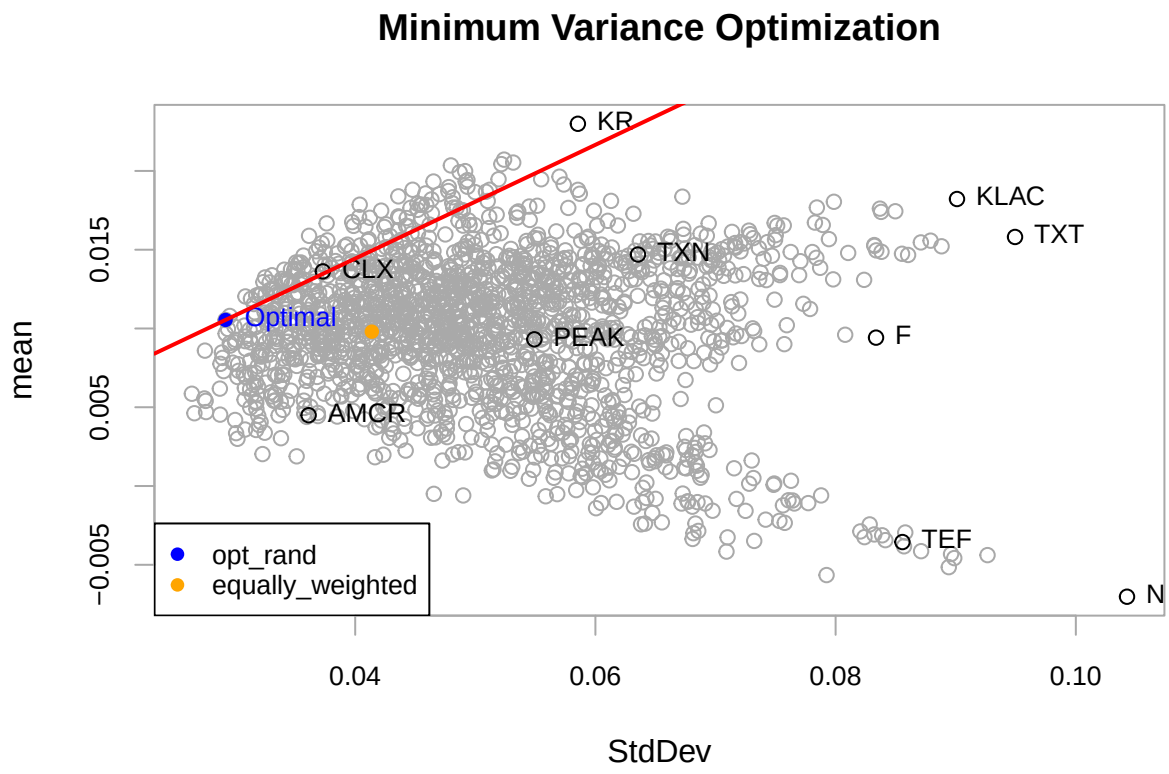


The realized returns can be quite different from the promised or expected returns. This difference depends on the method, the model, the database length, the ability of the portfolio designer, and also depends on a random component. In our FANG example, the investment

recommendations were good alternatives.

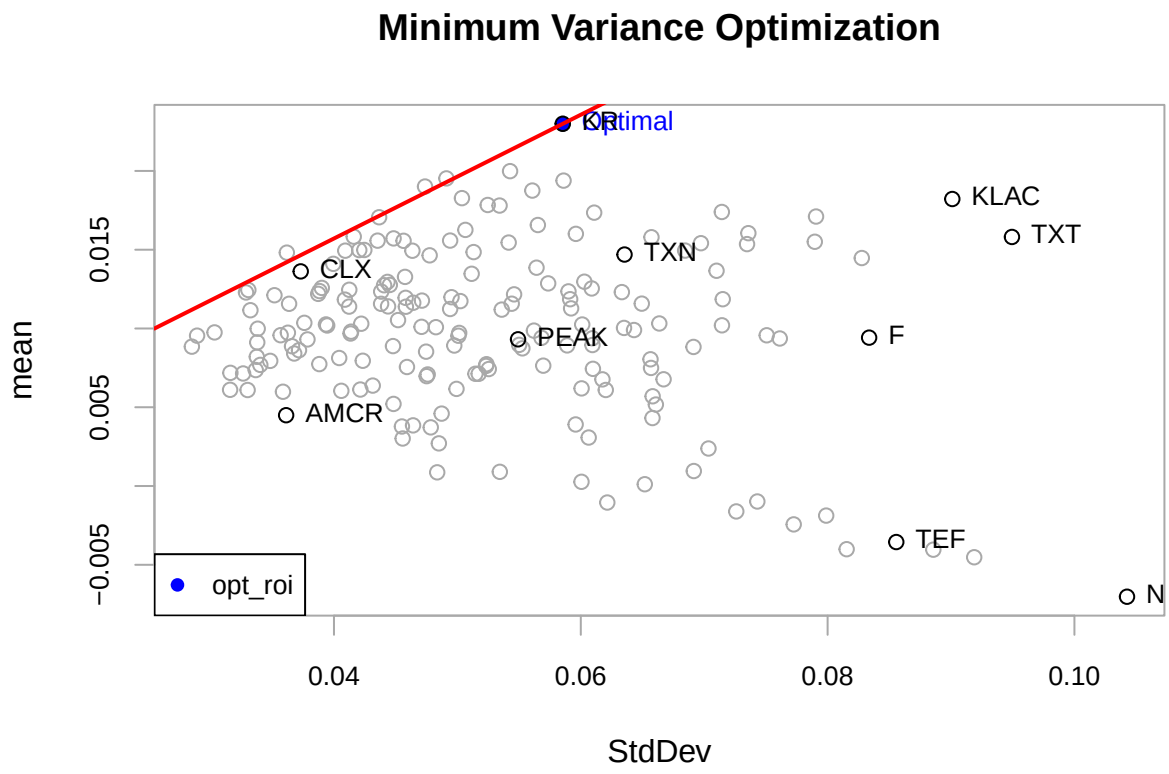
It is tempting to calculate an optimal portfolio based on our previous CAPM example. Let's do it here just to show how easy is it to replicate the analysis with a different database, which is the beauty of reproducibility as I basically did a copy paste of the previous code.

```
capm_stocks <-
  R_stocks |> tbl_xts(spread_by = "symbol") |>
  na.fill(fill = 0.00)
port_spec <- portfolio.spec(colnames(capm_stocks))
port_spec <- add.constraint(portfolio =
                              port_spec, type = "full_investment")
port_spec <- add.constraint(portfolio = port_spec, type = "long_only")
port_spec <- add.objective(portfolio = port_spec,
                           type = "risk",
                           name = "StdDev")
port_spec <- add.objective(portfolio = port_spec,
                           type = "return",
                           name = "mean")
set.seed(14)
opt_rand <- optimize.portfolio(capm_stocks, portfolio = port_spec,
                          optimize_method = "random", trace = TRUE)
chart.RiskReward(opt_rand, risk.col = "StdDev",
                 main = "Minimum Variance Optimization",
                 return.col = "mean", chart.assets = TRUE)
legend("bottomleft", legend = c("opt_rand", "equally_weighted"),
       col = c("blue", "orange"), pch = 19, cex = 0.8)
abline(0, opt_rand$objective_measures$mean /
         opt_rand$objective_measures$StdDev, lwd = 2, col = "red")
```



Here, the slope of the red line corresponds to the optimal portfolio return per unit of risk. This means that the optimal portfolio has a similar return per unit of risk compared with KR. Note how the rest of the assets have a lower return per unit of risk.

```
opt_roi <- optimize.portfolio(capm_stocks, portfolio = port_spec,
                              optimize_method = "ROI", trace = TRUE)
chart.RiskReward(opt_roi, risk.col = "StdDev",
                 main = "Minimum Variance Optimization",
                 return.col = "mean", rp = TRUE, chart.assets = TRUE)
legend("bottomleft", legend = c("opt_roi"),
       col = c("blue"), pch = 19, cex = 0.8)
abline(0, opt_roi$objective_measures$mean /
         opt_roi$objective_measures$StdDev, lwd = 2, col = "red")
```



5.2 Diversification.

In finance, diversification is the process of allocating capital (or creating a portfolio) in a way that reduces the exposure to any one particular asset or risk factor. We usually recommend to invest in a variety of assets to achieve an overall risk reduction of our portfolio. However, if those assets are as correlated as Visa and Mastercard (as discussed before), then our diversification efforts would not be so effective. In this section, we propose an experiment to illustrate the role of diversification in asset allocation. The experiment is to artificially generate two assets and add them to the FANG database. The special characteristic of these two new assets X and Y is that they both have a very extreme negative correlation value. Once we add these two assets, we will repeat the portfolio optimization and see if we could do better than before.

First, let's generate these two assets returns. Note that these two new assets X and Y do not exist in the real world, we are artificially generating them by implementing a simulation technique.

```
# Library to use the multi-variate normal random number generator.
library(MASS)
set.seed(13)
data = mvrnorm(n = 48, mu = c(0.2, 0.5),
               Sigma = matrix(c(1, -1.4, -1.4, 2), nrow = 2),
               empirical = TRUE)/10
xy = as.data.frame(data)
X = xy$V1
Y = xy$V2
# Add X and Y to the fang database.
fang_xy <- fang
fang_xy$X <- X
fang_xy$Y <- Y
head(fang_xy)
```

```
##                    FB         AMZN        NFLX        GOOG          X
## 2013-01-31  0.10642857  0.031829319  0.79589177  0.04485307 0.08235099
## 2013-02-28 -0.12040026 -0.004632810  0.13822318  0.06022315 0.11219977
## 2013-03-28 -0.06128440  0.008400504  0.00638028 -0.00874938 0.15700719
## 2013-04-30  0.08561376 -0.047581495  0.14153635  0.03825280 0.09411057
## 2013-05-31 -0.12315448  0.060635964  0.04711437  0.05657498 0.07357672
## 2013-06-28  0.02176587  0.031537851 -0.06700557  0.01050246 0.01481500
##                     Y
## 2013-01-31 -0.04241279
## 2013-02-28 -0.05184889
## 2013-03-28 -0.16791336
## 2013-04-30 -0.04495379
## 2013-05-31 -0.05019958
## 2013-06-28  0.04137669
```

Let's now verify that these two new assets are negatively correlated.

```
cor(fang_xy)
```

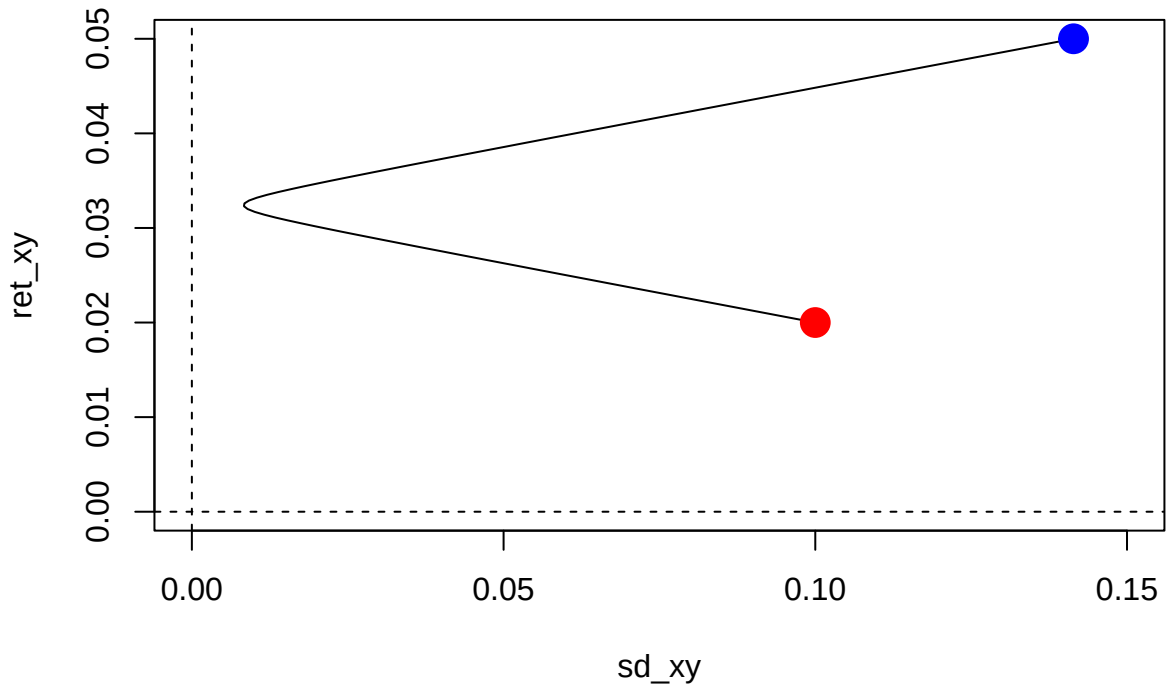
```
##                FB        AMZN        NFLX       GOOG           X           Y
## FB    1.00000000  0.18461970  0.21820791  0.2468989  0.03779375 -0.04276241
## AMZN  0.18461970  1.00000000  0.31180203  0.6171376  0.04877402 -0.05611269
## NFLX  0.21820791  0.31180203  1.00000000  0.3586214  0.06772682 -0.05943567
```

```
## GOOG  0.24689886  0.61713764  0.35862138  1.0000000  0.20682201 -0.19110508
## X     0.03779375  0.04877402  0.06772682  0.2068220  1.00000000 -0.98994949
## Y    -0.04276241 -0.05611269 -0.05943567 -0.1911051 -0.98994949  1.00000000
```

The correlation of X and Y is -0.98994949, this is very close to -1, so they are strongly negatively correlated. This is just what we wanted. In fact, the correlation of these two new assets and the rest are also negative. In principle, we should expect a new optimal portfolio with a greater return per unit of risk.

Some may argue that investing in two inversely correlated assets would lead to a zero expected return. This is not the case as shown below.

```
w_x <- seq(0, 1, 0.01)
w_y <- 1 - w_x
ret_xy <- w_x*mean(fang_xy$X) + w_y*mean(fang_xy$Y)
sd_xy <- (w_x^2*sd(fang_xy$X)^2 + w_y^2*sd(fang_xy$Y)^2 +
  2*w_x*w_y*sd(fang_xy$X)*sd(fang_xy$Y)*cor(fang_xy$X, fang_xy$Y))^0.5
plot(sd_xy, ret_xy, type = "l", xlim = c(0, 0.15), ylim = c(0, 0.05))
abline(v = 0, lty = 2)
abline(h = 0, lty = 2)
points(sd(fang_xy$X), mean(fang_xy$X), cex = 2, pch = 19, col = "red")
points(sd(fang_xy$Y), mean(fang_xy$Y), cex = 2, pch = 19, col = "blue")
```

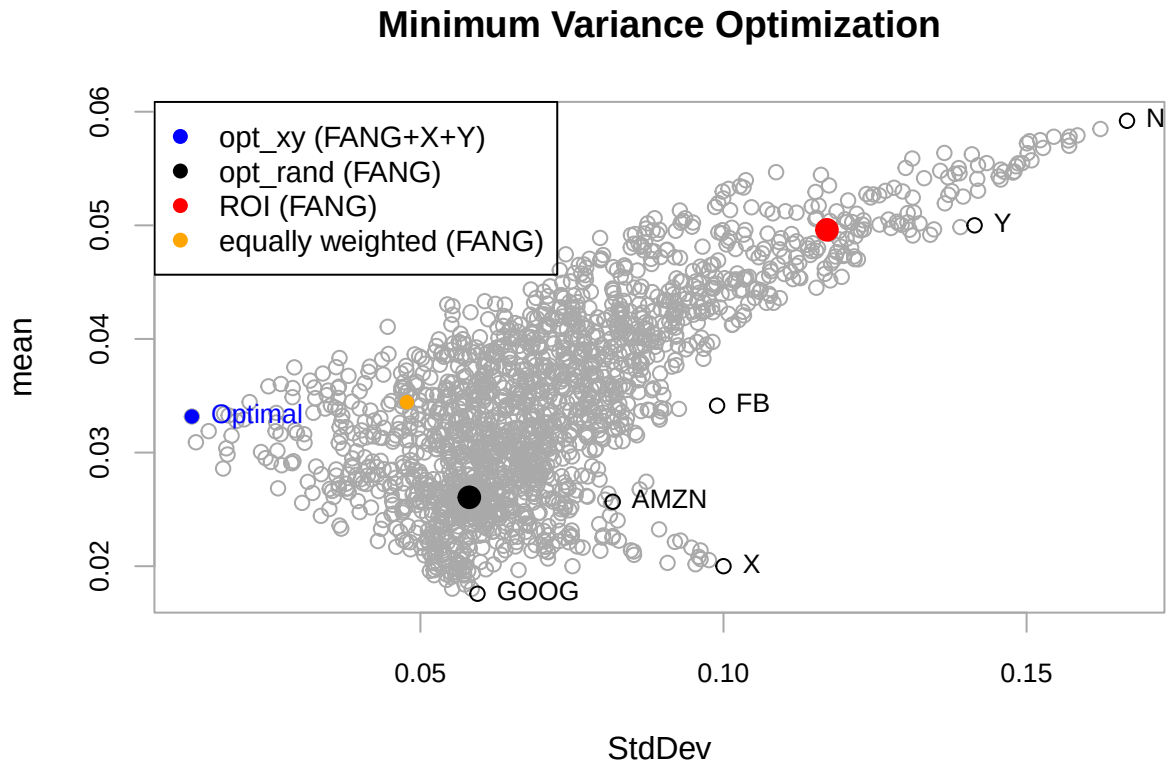


Let's generate the portfolio specification and optimize our portfolio as we did before.

```
# Create the portfolio specification
port_spec <- portfolio.spec(colnames(fang_xy))
port_spec <- add.constraint(portfolio =
                              port_spec, type = "full_investment")
port_spec <- add.constraint(portfolio = port_spec, type = "long_only")
port_spec <- add.objective(portfolio = port_spec,
                           type = "risk",
                           name = "StdDev")
port_spec <- add.objective(portfolio = port_spec,
                           type = "return",
                           name = "mean")
# Optimization.
set.seed(13)
opt_xy <- optimize.portfolio(fang_xy, portfolio = port_spec,
                          optimize_method = "random", trace = TRUE)
```

We are done. Now, let's visualize the results.

```
# Plot results.
chart.RiskReward(opt_xy, risk.col = "StdDev",
                 main = "Minimum Variance Optimization", xlim = c(0, 10),
                 return.col = "mean", chart.assets = TRUE)
points(0.05807126, 0.02606294, pch = 19, cex = 1.5, col = "black")
points(0.1170653,0.04960399, pch = 19, cex = 1.5, col = "red")
legend("topleft", legend = c("opt_xy (FANG+X+Y)",
       "opt_rand (FANG)", "ROI (FANG)", "equally weighted (FANG)"),
       col = c("blue", "black", "red", "orange"), pch = 19, cex = 0.9)
```



This looks great as the new assets contribute to the optimization process to deliver a less risky portfolio. Now, the optimal portfolio has almost the same return as Facebook but with a significant reduced risk. Then, correlation value plays a determinant role in the diversification process. This is why investors are looking for low, or even better, negatively correlated assets.

Let's see the details of the new portfolio.

```
# Extract weight, risk and return.
opt_xy$weights
```

```
##    FB  AMZN  NFLX  GOOG     X     Y
## 0.002 0.028 0.000 0.014 0.522 0.434
```

```
opt_xy$opt_values
```

```
## $StdDev
##             [,1]
## [1,] 0.01229882
## attr(,"names")
## [1] "StdDev"
##
## $mean
##       mean
## 0.03317307
```

```
opt_xy$opt_values$mean / opt_xy$opt_values$StdDev
```

```
##          [,1]
## [1,] 2.697257
```

Note that the portfolio assigns high weights to X and Y with values of 52.2% and 43.4%. The rest is 0.2% in FB, 2.8% in AMZN, 0% in NFLX, and 1.4% in GOOG.

Let's compare the return per unit of risk.

```
# Add the new portfolio sr to the list.
FANG_monthly_sr <-
  add_row(FANG_monthly_sr, symbol = "opt_xy",
          sr = opt_xy$opt_values$mean / opt_xy$opt_values$StdDev) |>
  arrange(sr)
FANG_monthly_sr
```

```
## # A tibble: 7 x 2
##   symbol   sr[,1]
##   <chr>     <dbl>
## 1 GOOG      0.296
## 2 AMZN      0.314
```

```
## 3 FB        0.345
## 4 NFLX      0.355
## 5 opt_roi   0.424
## 6 opt_rand  0.449
## 7 opt_xy    2.70
```

Impressive improvement. Informed investors are not looking for high return assets to invest in. Specially not in the context of portfolio investment. High return assets are associated with high risk so it is likely not to get the promised (or expected) return. Let me put a silly example. You buy a lottery ticket for 2 USD and you expect to win 1,000,000 USD, however the odds to win are 1 in 12,607,306. A 999,998 USD return looks quite nice but you will hardly get it. This is why we argue that we do not pick an asset with respect to its price, nor with respect to its return, but with respect to its return per unit of risk. In the context of asset allocation (invest in several assets at a time), this is partially why informed investors are not looking for high return assets to invest in (although this sounds like the popular thought). Investors can do better by selecting low correlated assets because this will allow them to form a well diversified portfolio with a more certain return at $t = 0$. In sum, contrary to popular wisdom, we can argue that in many circumstances we are more interested in risk rather than in return. In fact, this topic is called "risk management…", not "return management…".

If you are not quite happy with the 2.606294% monthly return of opt_rand portfolio, you always have the opt_roi which is 4.960399% and still with a decent return per unit of risk. If this is still not good for you, then (according to these models and assumptions) you should look to add an asset low or negatively correlated with your existing assets and conduct your optimization again. A good way to start looking at negative correlated assets is by looking at different and distant industries, or assets that belong to distant markets. This would at least improve the chances to find stocks with different responses to risk factors and this is a good way to start your search.

5.3 Rebalancing portfolio and evaluation.

In the previous section we calculate portfolio weights once. We changed the optimization criteria and we added new assets, but we only calculate the portfolio weights once. Rebalancing is something very common in finance, it means to calculate portfolio weights as time passes. This makes sense because as time passes we have access to new information (stock returns) and we should re-balance our portfolio in order to take into account this new information. Evaluation is also a very common task in finance. We are interested to know what is the

annualized return of an investment strategy in a period of time. In this section we are going to extend the asset allocation problem to incorporate rebalancing portfolio and evaluation. We are going to use a different database to illustrate our results.

Consider the following investment process. At $t = 0$ I have access to 60 months historical information of a set of individual assets. Then, I can take information from $t = -60$ to $t = 0$ to estimate optimal portfolio weights to form my portfolio at $t = 0$. At $t = 0$, $t = 1$, and $t = 2$ I simply get my returns or losses depending on the evolution of the market. Then, at $t = 3$ I calculate new portfolio weights with information from $t = -57$ to $t = 3$. At $t = 3$, $t = 4$, and $t = 5$ I simply get my returns or losses depending on the evolution of the market. Then, at $t = 6$ I calculate new portfolio weights with information from $t = -54$ to $t = 6$. And I continue with the same procedure for several years. What would be my annualized return, and my annualized return per unit of risk? How could I know whether my investment procedure is better than other alternatives?

Before answering these questions, it is convenient to think in the process above. This looks like a lot of work. Everything starts with getting the price data for the correspondent assets. Then, convert the prices to returns. Then, calculate an optimal portfolio and implement the investment recommendation. Wait for the returns, and then re-balance our portfolio and implement the investment recommendation. Wait for the returns, and do the same until the end of the investment period which could last years. After that, look back to the portfolio returns and calculate an annualized return to evaluate my investment. This process is painful without a computer and without access to a computer language like R. In a computer we can automate this process and spend our time in more strategic tasks. Automatization is very common in other industries. Have you seen how cars are manufactured nowadays? You can hardly see a human operator, most of the process is made by robots. In finance, we can design robots since our main input (or raw material) is free data. Also, most of our main technology is free (R), the most expensive input is human capital.

Let's start with the data to tackle our objectives.

```
# Get the data.
data(indexes)
returns <- indexes[, 1:4]
tail(returns)
```

```
##            US Bonds US Equities Int'l Equities Commodities
## 2009-07-31   0.0132      0.0703         0.0838      0.0044
## 2009-08-31   0.0108      0.0348         0.0517     -0.0242
```

```
## 2009-09-30   0.0108      0.0360         0.0371      0.0017
## 2009-10-31   0.0042     -0.0189        -0.0126      0.0555
## 2009-11-30   0.0134      0.0566         0.0199      0.0150
## 2009-12-31  -0.0175      0.0189         0.0143      0.0086
```

The database goes from 1980-01-31 to 2009-12-31. The set of assets are: "US Bonds", "US Equities", "Int'l Equities" and "Commodities". This means that the first investment recommendation is calculated with information from 1980-01-31 to 1985-01-31 (60 monthly observations), the second from 1980-05-31 to 1985-09-31, and so on until the last period that goes from 2009-09-31 to 2009-12-31. The first monthly return will be the one on 1985-01-31 and the last on 2009-12-31, these are 300 monthly portfolio returns.

Before calculating optimal portfolios, let's calculate a benchmark portfolio. This will allow us to compare our optimal portfolio with a benchmark. In this case the benchmark is simply an equally weighted portfolio, investing 25% in each asset for all the periods. This equally weighted portfolio implies that we will not conduct any optimization. It is like investing 25% in each asset (or index in this case) and doing nothing until the end of the investment period.

```
# Equal weight benchmark.
n <- ncol(returns)
equal_weights <- rep(1 / n, n)
benchmark_returns <- Return.portfolio(R = returns,
                                      weights = equal_weights,
                                      rebalance_on = "quarters")
colnames(benchmark_returns) <- "benchmark"
# Benchmark performance.
table.AnnualizedReturns(benchmark_returns)
```

```
##                           benchmark
## Annualized Return            0.0769
## Annualized Std Dev           0.1029
## Annualized Sharpe (Rf=0%)    0.7476
```

We are interested in a benchmark because it would facilitate our evaluation task.

Now we define the portfolio specification as we did in the section before.

```
# Base portfolio specification.
base_port_spec <- portfolio.spec(assets = colnames(returns))
```

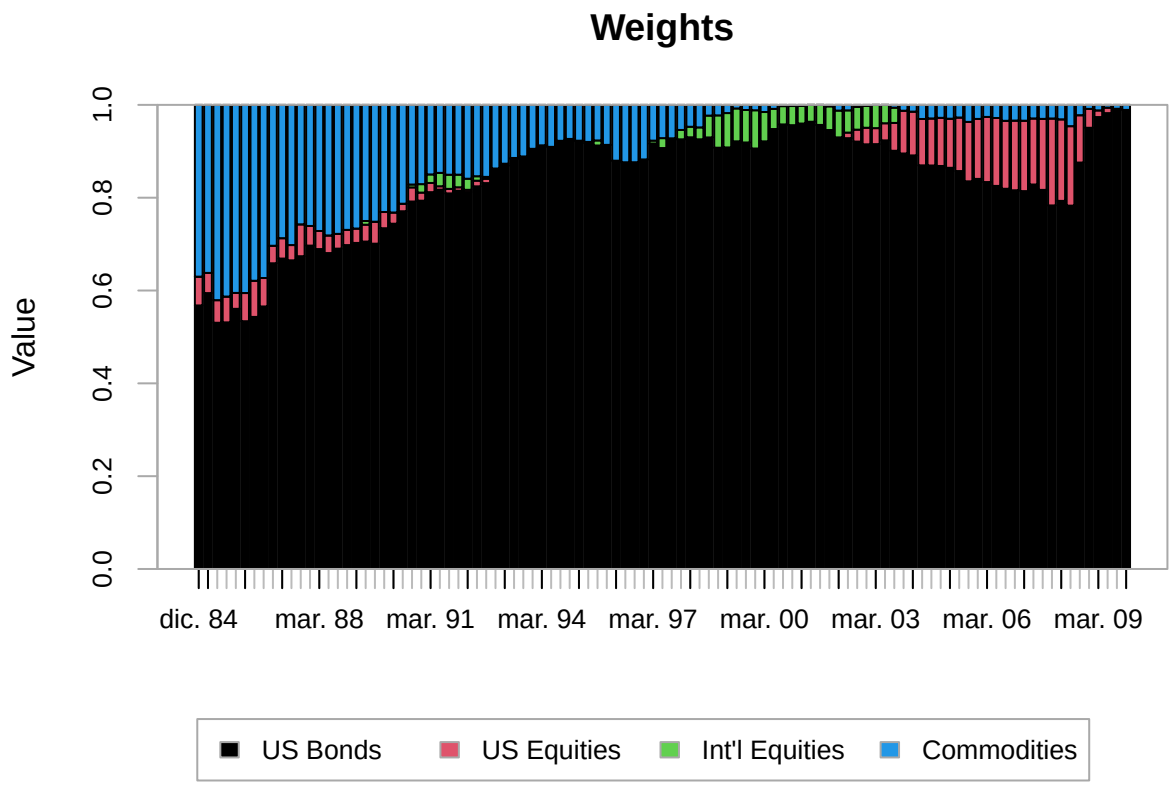
```
base_port_spec <- add.constraint(portfolio = base_port_spec,
                                 type = "full_investment")
base_port_spec <- add.constraint(portfolio = base_port_spec,
                                 type = "long_only")
base_port_spec <- add.objective(portfolio = base_port_spec,
                                type = "risk", name = "StdDev")
```

We are ready to implement the investment process described before.

```
# Run the optimization with periodic rebalancing.
opt_base <- optimize.portfolio.rebalancing(R = returns,
            optimize_method = "ROI", portfolio = base_port_spec,
            rebalance_on = "quarters", training_period = 60,
            rolling_window = 60)
# Calculate portfolio returns.
base_returns <- Return.portfolio(returns, extractWeights(opt_base))
colnames(base_returns) <- "base"
```

We are done. Rebalancing and evaluation are done. Now, let's show the results.

```
# Chart the optimal weights.
chart.Weights(opt_base)
```



This is how re-balancing looks like. It is the contribution of each asset in my portfolio. Clearly US bonds dominate my portfolio but this is what the optimization recommends. Now, let's see the annualized performance of this portfolio.

```
# Merge benchmark and portfolio returns.
ret <- cbind(benchmark_returns, base_returns)
# Annualized performance.
table.AnnualizedReturns(ret)
```

```
##                           benchmark   base
## Annualized Return            0.0769 0.0772
## Annualized Std Dev           0.1029 0.0436
## Annualized Sharpe (Rf=0%)    0.7476 1.7714
```

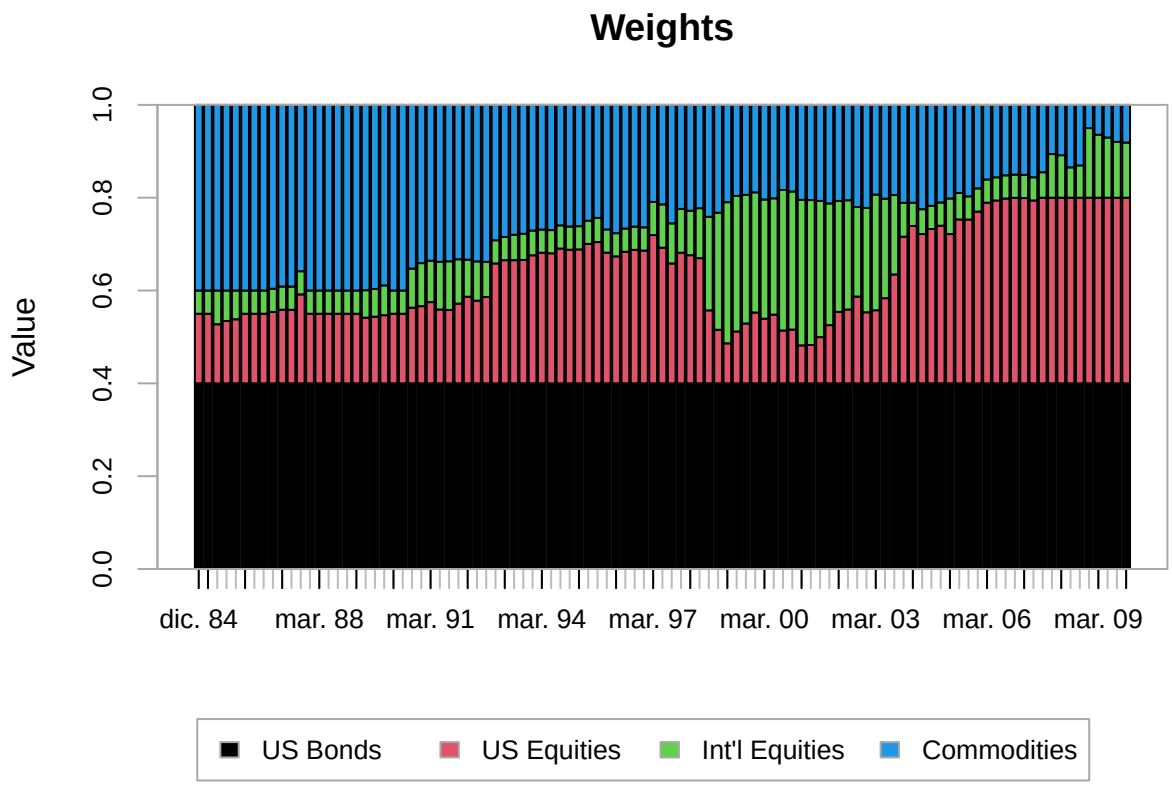
We did better than our benchmark portfolio, that is good.

Something that happens frequently is that for some reasons we face constraints about how much money to invest in an individual asset. Let's assume this is the case and that we are supposed not to invest more than 40% in US bonds. We can incorporate this constraint easily and reproduce the re-balancing chart and annualized returns.

```
# Make a copy of the portfolio specification.
box_port_spec <- base_port_spec
# Update the constraint.
box_port_spec <- add.constraint(portfolio = box_port_spec,
                                type = "box", min = 0.05, max = 0.4,
                                indexnum = 2)
# Backtest.
opt_box <- optimize.portfolio.rebalancing(R = returns,
                                          optimize_method = "ROI",
                                          portfolio = box_port_spec,
                                          rebalance_on = "quarters",
                                          training_period = 60,
                                          rolling_window = 60)
# Calculate portfolio returns.
box_returns <- Return.portfolio(returns, extractWeights(opt_box))
colnames(box_returns) <- "box"
```

In principle, more constraints will lead to a worse solution. The optimization algorithm works better as long as it can freely choose portfolio weights.

```
# Chart the optimal weights.
chart.Weights(opt_box)
```



Now, the rest of the assets have a more significant role in our portfolio.

```
# Merge box portfolio returns.
ret <- cbind(ret, box_returns)
# Annualized performance.
table.AnnualizedReturns(ret)
```

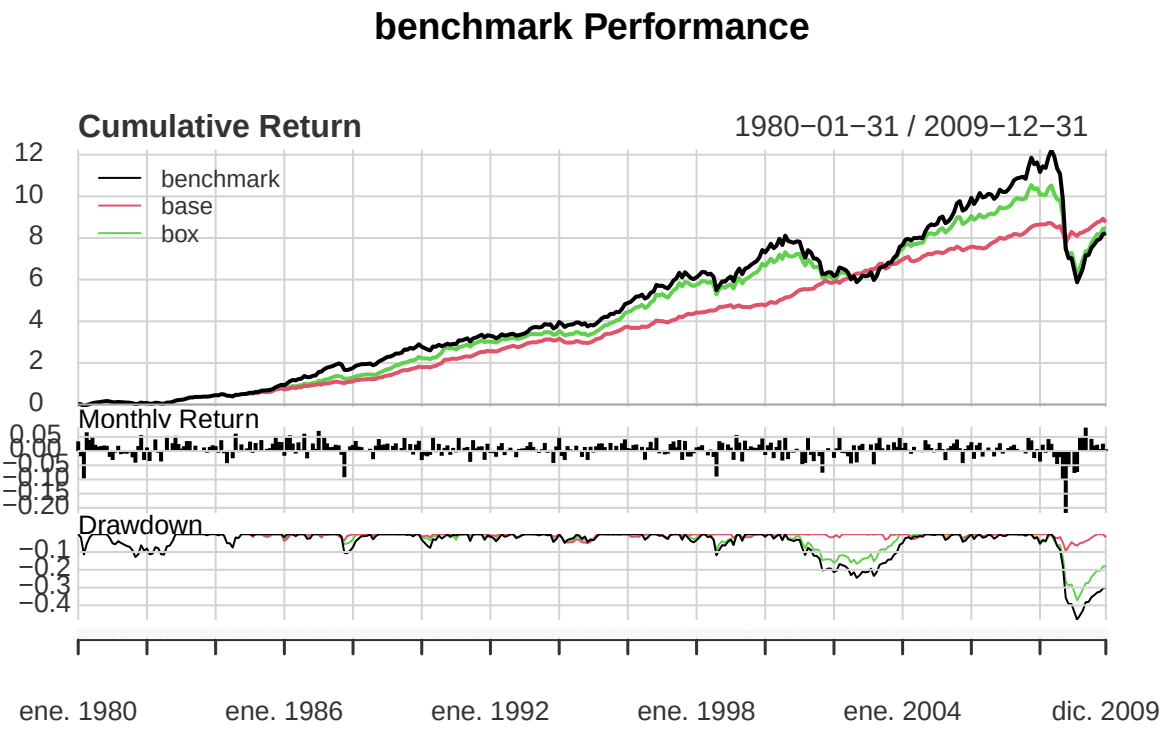
```
##                           benchmark   base    box
## Annualized Return            0.0769 0.0772 0.0760
## Annualized Std Dev           0.1029 0.0436 0.0819
## Annualized Sharpe (Rf=0%)    0.7476 1.7714 0.9282
```

As expected, more constraints negatively impact the possibilities to get a higher annualized return.

Most of these activities including providing investment recommendations in a regular manner are services offered by private firms. People looking to invest part of their money are supposed to receive assistance and guidance and many times it is in the form of an investment recommendation like the ones we calculate here. In the past, these services were provided only by those firms with access to the knowledge, capital, experience, and technology. Nowa-

days, there are an increasing number of fintechs that provide financial services because now we have access to powerful software and improved computational capabilities. In any case, functions like `optimize.portfolio` are expected to be used as a tool by professionals and not as a pure source of investment recommendation.

```
charts.PerformanceSummary(R = ret)
```



6 Data visualization and multilevel modeling.

6.1 Data visualization.

The objective of this example is to show the power of data visualization, and introduce the concept of multilevel regression given the `hsb` database.

The following is the description of the data taken from the `merTools` R package.

This is a subset of data from the 1982 High School and Beyond survey used as examples for HLM software. The data file used for this presentation is a subsample from the 1982 High School and Beyond Survey and is used extensively in Hierarchical Linear Models by Raudenbush and Bryk. It consists of 8 variables, and 7,185 students nested in 160 schools.

- `schid`. A numeric vector, 160 unique values.
- `mathach`. A numeric vector for the performance on a standardized math assessment.
- `female`. A numeric vector coded 0 for male and 1 for female.
- `ses`. A numeric measure of student socio-economic status.
- `minority`. A numeric vector coded 0 for white and 1 for non-white students.
- `schtype`. A numeric vector coded 0 for public and 1 for private schools.
- `meanses`. A numeric, the average SES for each school in the data set.
- `size`. A numeric for the number of students in the school.

Reference: Stephen W. Raudenbush and Anthony S. Bryk (2002). *Hierarchical Linear Models: Applications and Data Analysis Methods (2nd ed.)*. SAGE.

Let's see the data.

```
# Let's take a look of the data and its original structure.
data(hsb) # Load the database.
head(hsb) # Look at the first values for inspection.
```

```
##   schid minority female    ses mathach size schtype meanses
## 1  1224        0      1 -1.528   5.876  842       0  -0.428
## 2  1224        0      1 -0.588  19.708  842       0  -0.428
## 3  1224        0      0 -0.528  20.349  842       0  -0.428
## 4  1224        0      0 -0.668   8.781  842       0  -0.428
## 5  1224        0      0 -0.158  17.898  842       0  -0.428
## 6  1224        0      0  0.022   4.583  842       0  -0.428
```

```
#Now, let's inspect the data structure.
str(hsb) # See the original data structure.
```

```
## 'data.frame':    7185 obs. of  8 variables:
##  $ schid   : chr  "1224" "1224" "1224" "1224" ...
##  $ minority: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ female  : num  1 1 0 0 0 0 1 0 1 0 ...
##  $ ses     : num  -1.528 -0.588 -0.528 -0.668 -0.158 ...
##  $ mathach : num  5.88 19.71 20.35 8.78 17.9 ...
##  $ size    : num  842 842 842 842 842 842 842 842 842 842 ...
##  $ schtype : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ meanses : num  -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.4
```

Some important definitions.

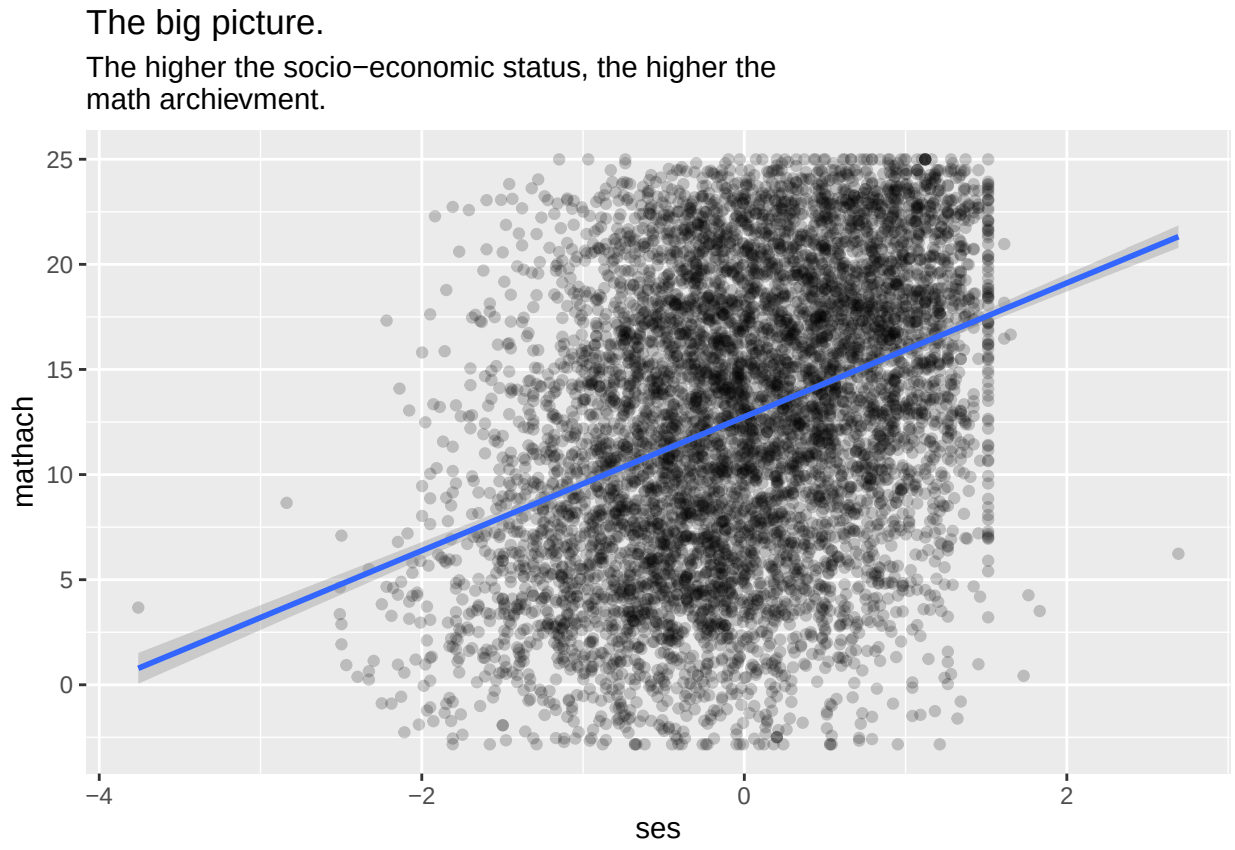
```
#This structure might not be completely appropriate as factors are not clearly  define
# From numerical to factors. This is useful for further analysis.
hsb$schid <- as.factor(hsb$schid)
hsb$minority <- as.factor(hsb$minority)
hsb$female <- as.factor(hsb$female)
hsb$schtype <- as.factor(hsb$schtype)
str(hsb) # See the new structure.
```

```
## 'data.frame':    7185 obs. of  8 variables:
##  $ schid   : Factor w/ 160 levels "1224","1288",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ minority: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ female  : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 2 1 2 1 ...
##  $ ses     : num  -1.528 -0.588 -0.528 -0.668 -0.158 ...
##  $ mathach : num  5.88 19.71 20.35 8.78 17.9 ...
##  $ size    : num  842 842 842 842 842 842 842 842 842 842 ...
##  $ schtype : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ meanses : num  -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.428 -0.4
```

There is a weak, positive and linear relationship between the performance on a standardized math assessment and the socio-economic status of students. This is supported by a very basic and preliminar data analysis taking all information available, without any filter.

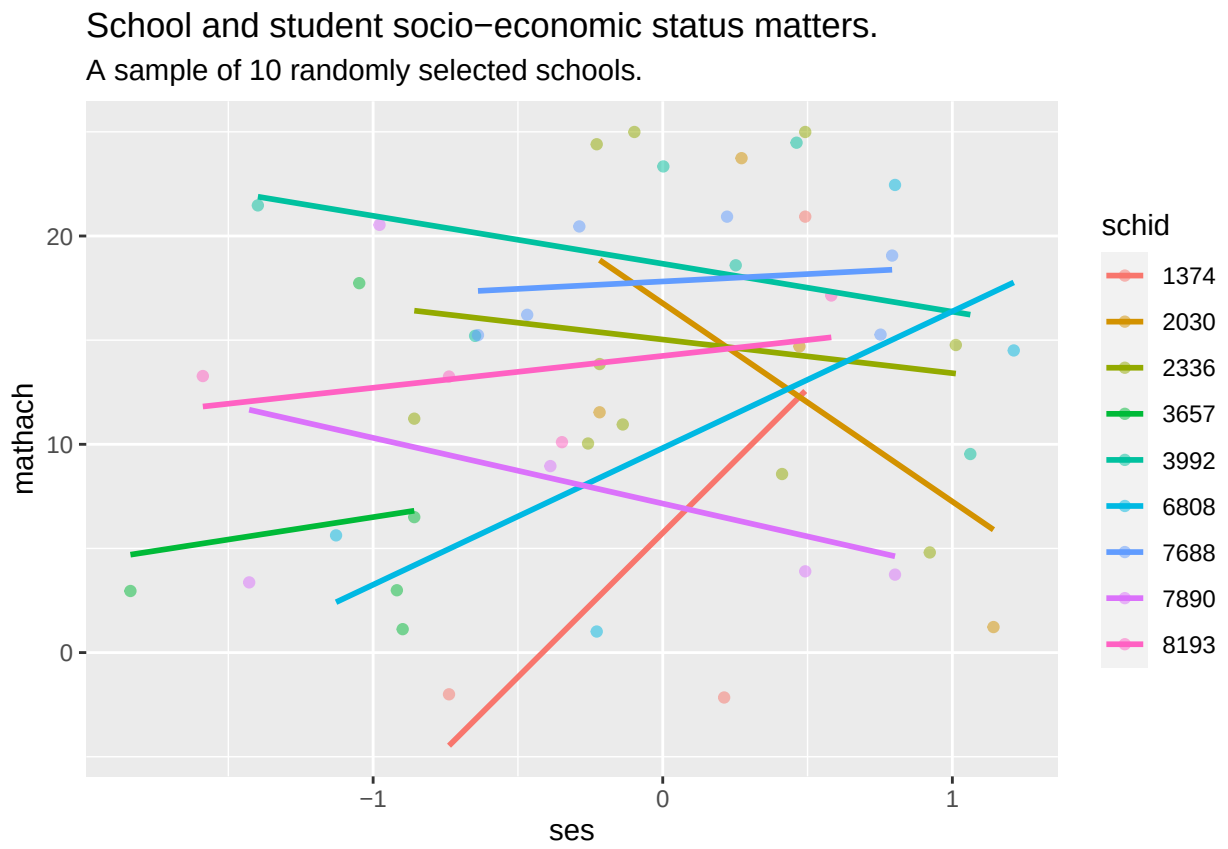
```
ggplot(hsb, aes(x = ses, y = mathach)) + geom_point(alpha = 0.2) +
  geom_smooth(method = 'lm') +
```

```
  labs(title = "The big picture.",
subtitle = "The higher the socio-economic status, the higher the
math archievment.")
```



We propose that the student and the school characteristics matters in the performance of the math assessment. We have 160 schools, let's see only a sample of 10 and see how this linear relationship changes.

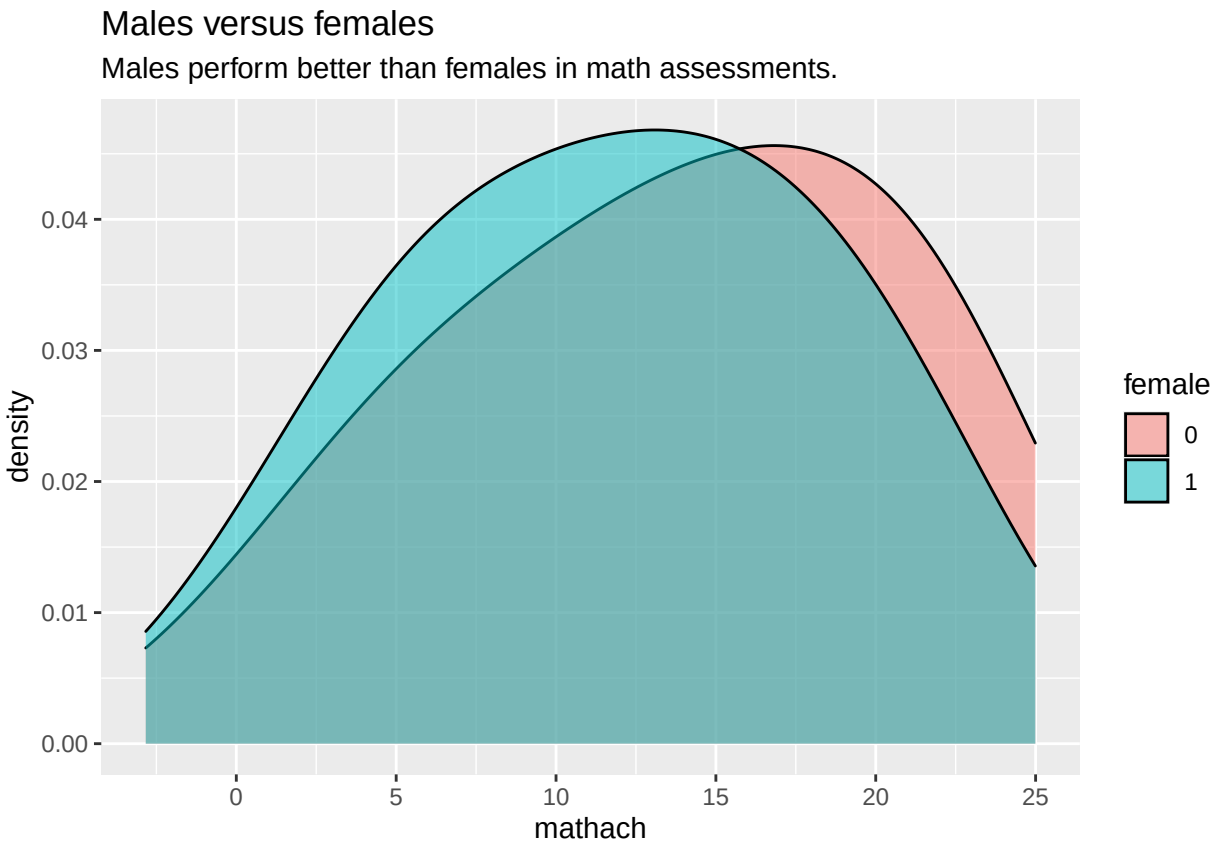
```
set.seed(13)
s = sample(hsb$schid, 10)
ggplot(hsb[hsb$schid == s, ], aes(x = ses, y = mathach, group = schid,
                    color = schid)) + geom_point(alpha = 0.5) +
  geom_smooth(method = 'lm', se = F) +
  labs(title = "School and student socio-economic status matters.",
       subtitle = "A sample of 10 randomly selected schools.")
```



Now, the previous general trend is less clear as we see what happen in 10 randomly selected schools. This suggest that the relationship between socio-economic status of students and the math achievement depends (at least) on the school characteristics.

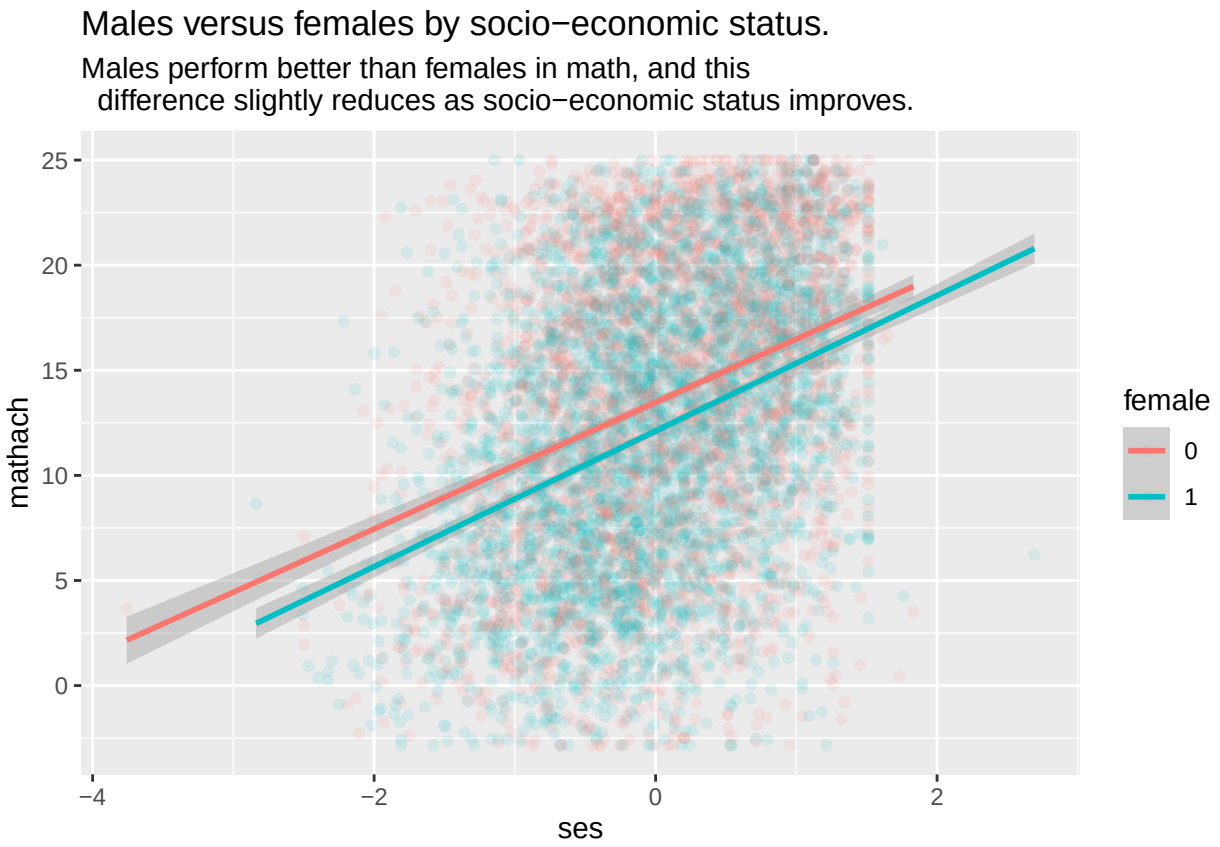
The performance on a standardized math assessment can be driven by the interaction of different factors. Here we see the differences between male and female students. According to this, males perform better than female students.

```
ggplot(hsb, aes(mathach, fill = female)) +
  geom_density(alpha = 0.5, adjust = 3) +
  labs(title = "Males versus females",
       subtitle = "Males perform better than females in math assessments.")
```



The data analysis reveals a gender issue here. Then, the math achievement seems to be a multi-factor issue that depends on the school characteristics and the student gender. We can go one step back to reveal the role of gender in our original scatterplot.

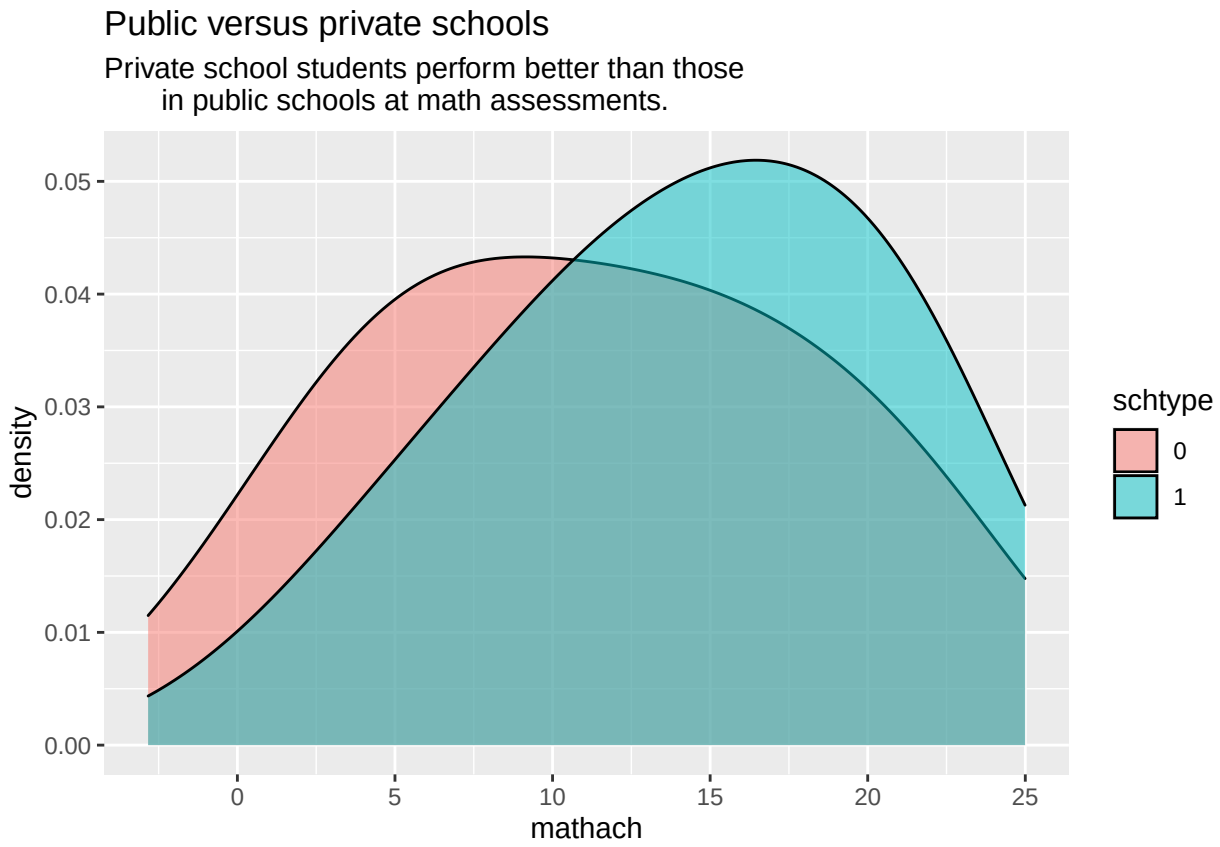
```
ggplot(hsb, aes(x = ses, y = mathach,
                color = female)) + geom_point(alpha = 0.1) +
  geom_smooth(method = 'lm') +
  labs(title = "Males versus females by socio-economic status.",
  subtitle = "Males perform better than females in math, and this
  difference slightly reduces as socio-economic status improves.")
```



This gender difference remains more or less constant across different socio-economic status. One may argue that this gap reduces as socio-economic status improves, but this reduction is minor, and the gap never closes.

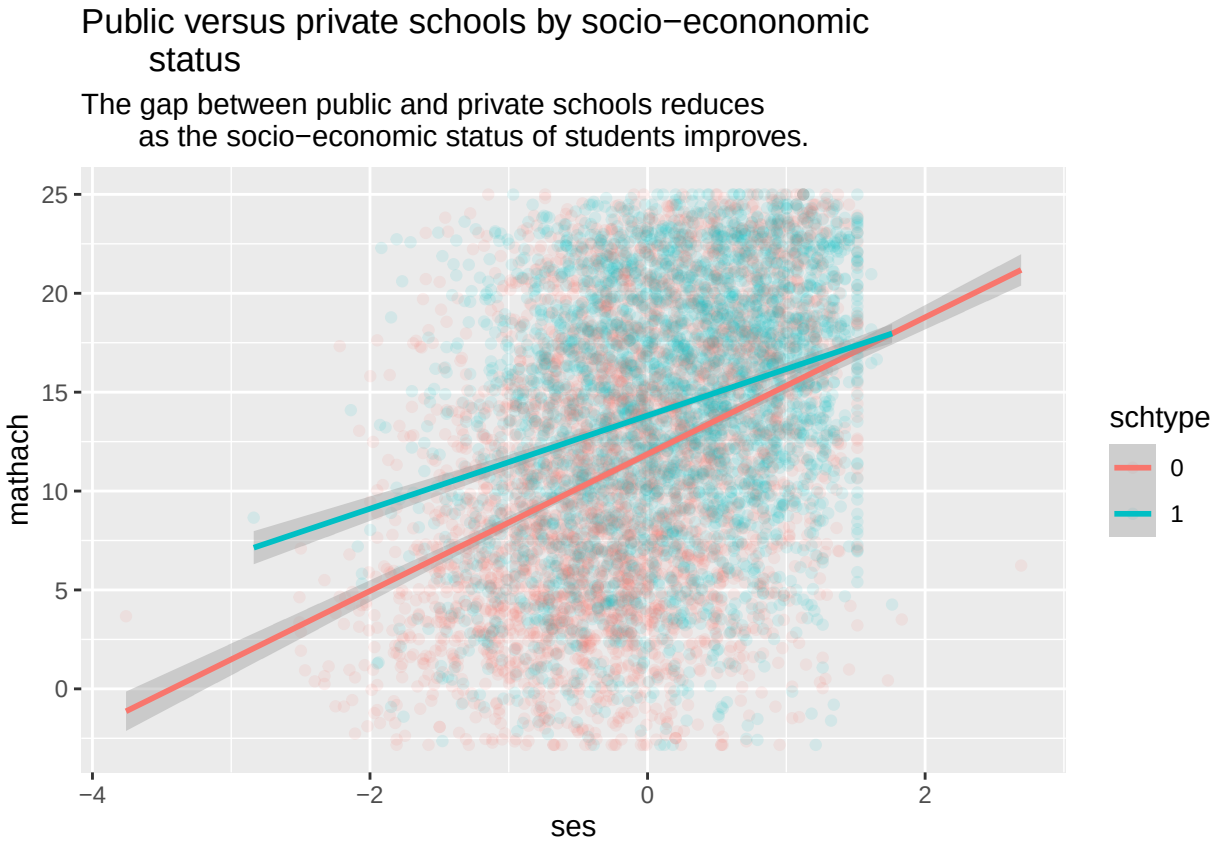
One school characteristic is whether it is private or public. Here, we illustrate that private schools show a better performance on the standardized math assessments.

```
ggplot(hsb, aes(mathach, fill = schtype)) +
  geom_density(alpha = 0.5, adjust = 3) +
  labs(title = "Public versus private schools",
       subtitle = "Private school students perform better than those
       in public schools at math assessments.")
```



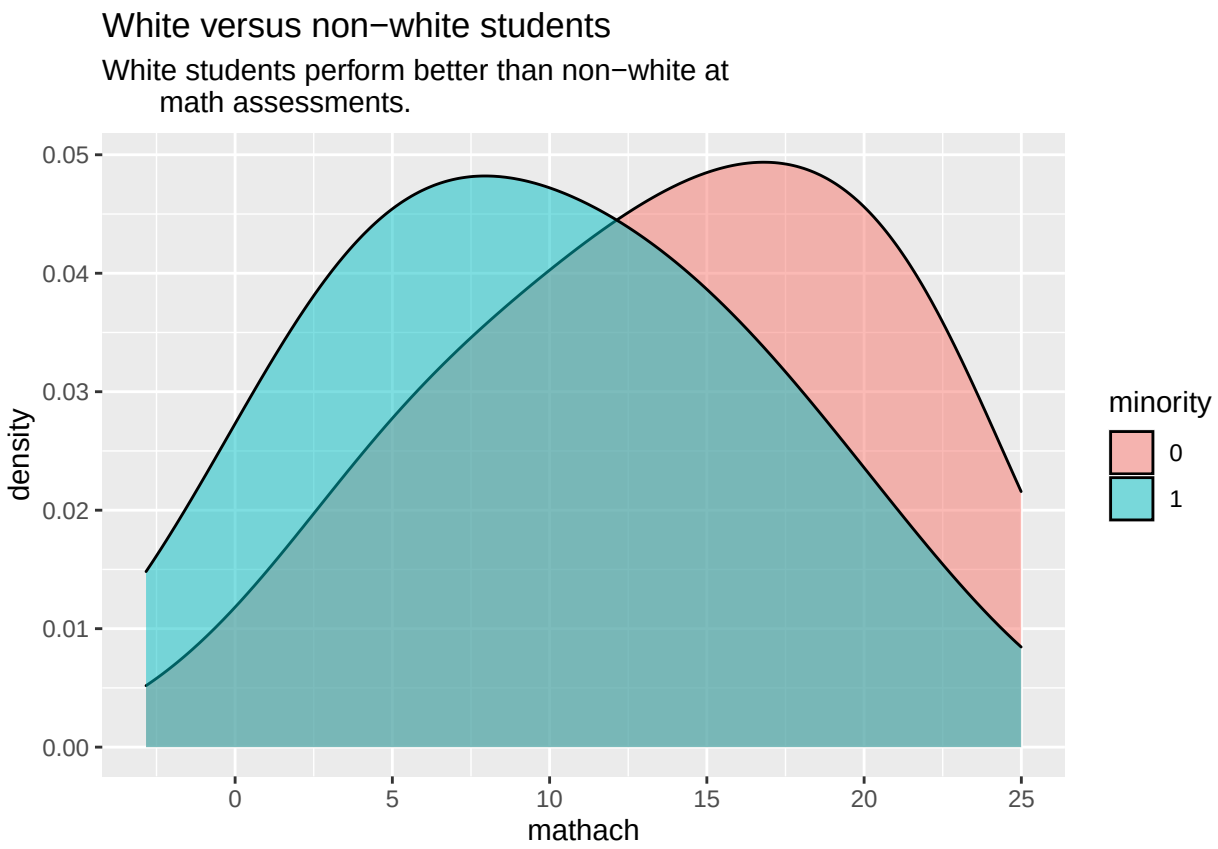
Although better math performance is achieved in private schools, the socio-economic status of the student can compensate in a way this effect. So, here, socio-economic status play an important role at making these school differences non significant.

```
ggplot(hsb, aes(x = ses, y = mathach,
                    color = schtype)) + geom_point(alpha = 0.1) +
  geom_smooth(method = 'lm') +
  labs(title = "Public versus private schools by socio-econonomic
       status",
       subtitle = "The gap between public and private schools reduces
       as the socio-economic status of students improves.")
```



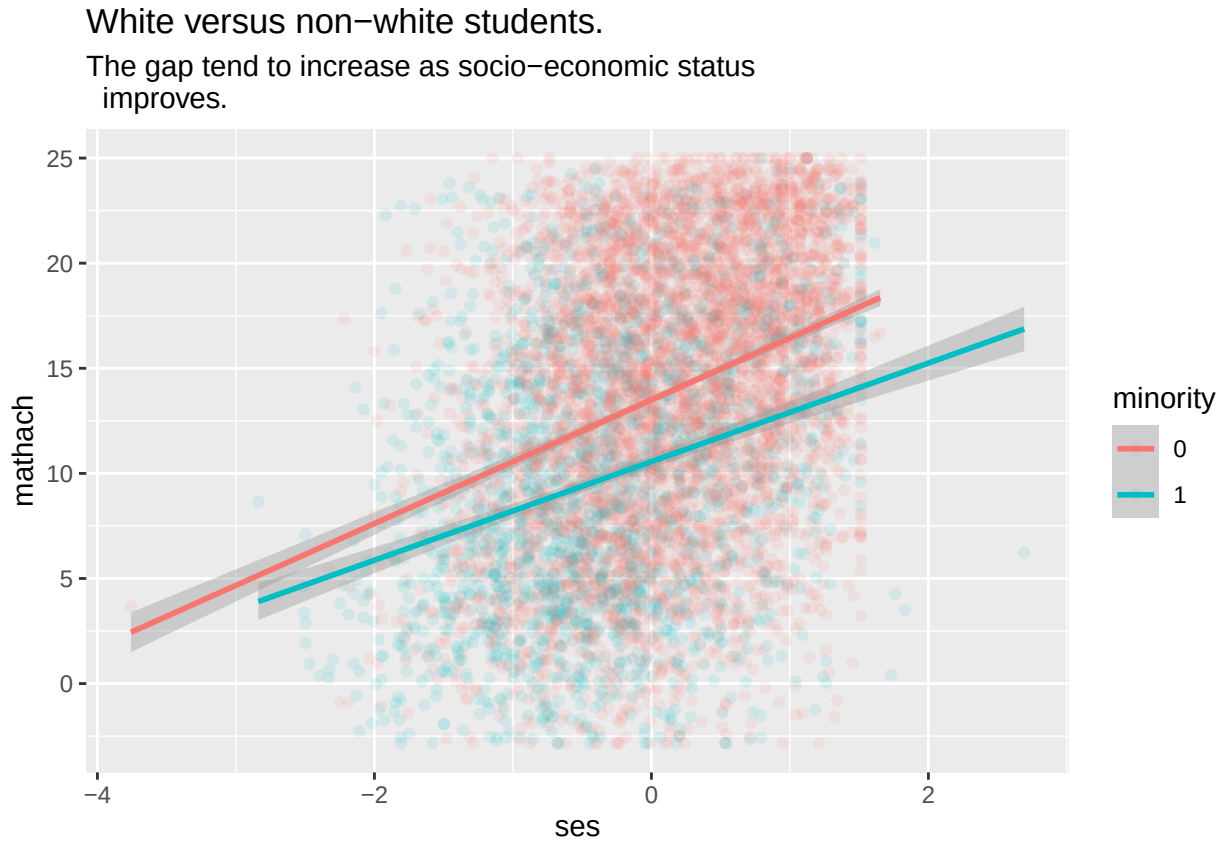
Let's see what happens with minorities. Apparently, white students show a better performance on the standardized math assessments.

```
ggplot(hsb, aes(mathach, fill = minority)) +
  geom_density(alpha = 0.5, adjust = 3) +
  labs(title = "White versus non-white students",
       subtitle = "White students perform better than non-white at
       math assessments.")
```



Now, following the logic of our previous data analysis, we show the role of the socio-economic status in this minority gap.

```
ggplot(hsb, aes(x = ses, y = mathach,
                    color = minority)) + geom_point(alpha = 0.1) +
  geom_smooth(method = 'lm') +
  labs(title = "White versus non-white students.",
  subtitle = "The gap tend to increase as socio-economic status
  improves.")
```



Minority differences increases as socio-economic status of the students improves. In other words, here the socio-economic status increases the gap between these two groups.

6.2 Multilevel modeling - estimation.

We can learn a lot by visualizing the data. However, we can learn even more by implementing an econometric technique to help us understand the determinants of the math achievement.

Two level multilevel regression. Null model: m1a.

```
# We start with the null model m1a.
m1a <- lmer(mathach ~ (1 | schid), data = hsb, REML = FALSE)
summary(m1a)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: mathach ~ (1 | schid)
##    Data: hsb
##
```

```
##      AIC      BIC   logLik deviance df.resid
##  47121.8  47142.4 -23557.9  47115.8     7182
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -3.06262 -0.75365  0.02676  0.76070  2.74184
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  schid    (Intercept)  8.553   2.925
##  Residual             39.148   6.257
## Number of obs: 7185, groups:  schid, 160
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  12.6371     0.2436 157.6209   51.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#plot(m1a, type=c("p","smooth"), col.line = 1)
m1a_AIC <- extractAIC(m1a)[2]
ans <- data.frame(m1a_AIC)
kable(t(ans), caption = "Model's AIC.", digits = 2)
```

Table 6.2.1: Model's AIC.

m1a_AIC	47121.81

Two level multilevel regression. Second model: m2.

```
# Now, the model m2.
m2 <- lmer(mathach ~ ses + female + (1 | schid), data = hsb)
summary(m2)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
```

```
## Formula: mathach ~ ses + female + (1 | schid)
##    Data: hsb
##
## REML criterion at convergence: 46595.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1769 -0.7375  0.0341  0.7658  2.8048
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  schid    (Intercept)  4.492   2.119
##  Residual             36.813   6.067
## Number of obs: 7185, groups:  schid, 160
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   13.2771     0.2025  215.6474  65.572  < 2e-16 ***
## ses            2.3564     0.1054 6795.9187  22.349  < 2e-16 ***
## female1       -1.1875     0.1654 6831.0063  -7.178  7.8e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr) ses
## ses     -0.021
## female1 -0.426  0.056
```

```
# paste0("AIC = ",extractAIC(m2)[2])
m2_AIC <- extractAIC(m2)[2]
ans <- data.frame(m1a_AIC, m2_AIC)
kable(t(ans), caption = "Model's AIC.", digits = 2)
```

Table 6.2.2: Model's AIC.

m1a_AIC	47121.81

m2_AIC	46599.74

Two level multilevel regression. Third model: m3.

```
# Now, the model called m3 in the course material.

m3 <- lmer(mathach ~ ses + female + size + meanses + schtype +
                 (1 | schid), data = hsb)
summary(m3)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: mathach ~ ses + female + size + meanses + schtype + (1 | schid)
##    Data: hsb
##
## REML criterion at convergence: 46513.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1752 -0.7330  0.0290  0.7619  2.8506
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  schid    (Intercept)  2.128   1.459
##  Residual             36.800   6.066
## Number of obs: 7185, groups:  schid, 160
##
## Fixed effects:
##               Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)  1.217e+01  3.931e-01  1.667e+02  30.957  < 2e-16 ***
## ses          2.152e+00  1.085e-01  7.023e+03  19.841  < 2e-16 ***
## female1     -1.192e+00  1.620e-01  5.697e+03  -7.360 2.10e-13 ***
## size         4.104e-04  2.467e-04  1.540e+02   1.664   0.0982 .
## meanses      3.046e+00  3.711e-01  1.791e+02   8.208 4.29e-14 ***
## schtype1     1.485e+00  3.261e-01  1.459e+02   4.555 1.10e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) ses    femal1 size   meanss
## ses      -0.009
## female1  -0.227  0.049
## size     -0.848  0.001  0.018
## meanses   0.136 -0.290  0.026 -0.034
## schtype1 -0.670  0.000 -0.003  0.432 -0.322
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
```

```
#paste0("AIC = ",extractAIC(m3)[2])

m3_AIC <- extractAIC(m3)[2]
ans <- data.frame(m1a_AIC, m2_AIC, m3_AIC)
kable(t(ans), caption = "Model's AIC.", digits = 2)
```

Table 6.2.3: Model's AIC.

m1a_AIC	47121.81
m2_AIC	46599.74
m3_AIC	46507.26

Two level multilevel regression. Optimize the model.

```
# We can propose an optimization of the model in terms of the AIC criterion.
options(na.action = "na.fail")
m_opt <- lmer(mathach ~ minority * female * ses  * schtype +
                      (1 | schid), data = hsb, REML = FALSE)
mm <- dredge(m_opt)
head(model.sel(mm, rank = AIC))
```

```
## Global model call: lmer(formula = mathach ~ minority * female * ses * schtype +
##     (1 | schid), data = hsb, REML = FALSE)
```

```
## ---
## Model selection table
##       (Int) fml mnr sch   ses fml:mnr fml:sch fml:ses mnr:sch mnr:ses sch:ses
## 9168  13.34   +   +   + 2.712                       +       +       +       +
## 9184  13.39   +   +   + 2.676       +               +       +       +       +
## 9200  13.31   +   +   + 2.706               +       +       +       +       +
## 9216  13.36   +   +   + 2.669       +       +       +       +       +       +
## 9104  13.33   +   +   + 2.912                                   +       +       +
## 11232 13.39   +   +   + 2.683       +               +       +       +       +
##       fml:mnr:ses mnr:sch:ses df    logLik     AIC delta weight
## 9168                        + 12 -23125.46 46274.9  0.00  0.331
## 9184                        + 13 -23124.76 46275.5  0.60  0.244
## 9200                        + 13 -23125.36 46276.7  1.79  0.135
## 9216                        + 14 -23124.63 46277.3  2.35  0.102
## 9104                        + 11 -23127.69 46277.4  2.45  0.097
## 11232           +           + 14 -23124.76 46277.5  2.59  0.090
## Models ranked by AIC(x)
## Random terms (all models):
## '1 | schid'
```

Two level multilevel regression. A good parsimonious model.

```
m_AIC <- lmer(mathach ~ minority + ses + minority*ses +
                      (1 | schid), data = hsb, REML = FALSE)
summary(m_AIC)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: mathach ~ minority + ses + minority * ses + (1 | schid)
##    Data: hsb
##
##      AIC      BIC   logLik deviance df.resid
##  46440.6  46481.9 -23214.3  46428.6     7179
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2104 -0.7259  0.0201  0.7520  2.9024
##
```

```
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  schid    (Intercept)  3.956   1.989
##  Residual             36.052   6.004
## Number of obs: 7185, groups:  schid, 160
##
## Fixed effects:
##                Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)     13.4175     0.1830  180.1292  73.339  < 2e-16 ***
## minority1       -3.0581     0.2091 5257.0997 -14.622  < 2e-16 ***
## ses              2.3950     0.1265 7079.1357  18.934  < 2e-16 ***
## minority1:ses   -0.8286     0.2135 7175.6539  -3.881 0.000105 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) mnrty1 ses
## minority1   -0.296
## ses         -0.085  0.079
## minorty1:ss  0.068  0.148 -0.547
```

```
#paste0("AIC = ",extractAIC(m_AIC)[2])

mpars_AIC <- extractAIC(m_AIC)[2]
ans <- data.frame(m1a_AIC, m2_AIC, m3_AIC, mpars_AIC)
kable(t(ans), caption = "Model's AIC.", digits = 2)
```

Table 6.2.4: Model's AIC.

m1a_AIC	47121.81
m2_AIC	46599.74
m3_AIC	46507.26
mpars_AIC	46440.60

Two level multilevel regression. A good not so parsimonious model.

```
m_AIC2 <- lmer(mathach ~  female + minority + schtype +
                 ses + female*ses + minority*schtype + minority*ses +
                 schtype*ses +
                      (1 | schid), data = hsb, REML = FALSE)
summary(m_AIC2)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: mathach ~ female + minority + schtype + ses + female * ses +
##     minority * schtype + minority * ses + schtype * ses + (1 |      schid)
##    Data: hsb
##
##      AIC      BIC   logLik deviance df.resid
##  46280.4  46356.1 -23129.2  46258.4     7174
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2062 -0.7185  0.0355  0.7625  2.8862
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  schid    (Intercept)  2.433   1.560
##  Residual             35.499   5.958
## Number of obs: 7185, groups:  schid, 160
##
## Fixed effects:
##                     Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)          13.3482     0.2228  245.7340  59.922  < 2e-16 ***
## female1              -1.2204     0.1600 5980.6897  -7.627 2.78e-14 ***
## minority1            -4.2766     0.3026 4333.1048 -14.132  < 2e-16 ***
## schtype1              1.7918     0.3098  184.1123   5.784 3.08e-08 ***
## ses                   2.5427     0.1831 7083.3241  13.889  < 2e-16 ***
## female1:ses           0.3901     0.1896 7172.2424   2.057   0.0397 *
## minority1:schtype1    2.1487     0.4139 3995.8561   5.191 2.20e-07 ***
## minority1:ses        -1.0927     0.2172 7163.7967  -5.030 5.01e-07 ***
## schtype1:ses         -0.8650     0.2110 6636.2888  -4.099 4.21e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) femal1 mnrty1 schty1 ses    fml1:s mnr1:1 mnrt1:
## female1     -0.377
## minority1   -0.296  0.005
## schtype1    -0.614 -0.002  0.231
## ses         -0.042  0.048  0.122 -0.016
## female1:ses  0.026  0.008 -0.024 -0.003 -0.525
## mnrty1:sch1  0.214  0.001 -0.741 -0.356 -0.075  0.014
## minorty1:ss  0.026 -0.025  0.281  0.058 -0.322 -0.041 -0.239
## schtype1:ss  0.007 -0.016 -0.193 -0.089 -0.504 -0.005  0.209 -0.054
```

```
#paste0("AIC = ",extractAIC(m_AIC2)[2])

mnopars_AIC <- extractAIC(m_AIC2)[2]
ans <- data.frame(m1a_AIC, m2_AIC, m3_AIC, mpars_AIC, mnopars_AIC)
kable(t(ans), caption = "Model's AIC.", digits = 2)
```

Table 6.2.5: Model's AIC.

m1a_AIC	47121.81
m2_AIC	46599.74
m3_AIC	46507.26
mpars_AIC	46440.60
mnopars_AIC	46280.44

6.3 Multilevel modeling - final remark.

Conclusion. Data visualization help us to better understand the data relationships. These last two models have a low AIC compared with the models shown in the course.

```
kable(t(ans), caption = "Model's AIC.", digits = 2)
```

Table 6.3.1: Model's AIC.

m1a_AIC	47121.81
m2_AIC	46599.74
m3_AIC	46507.26
mpars_AIC	46440.60
mnopars_AIC	46280.44

7 Blockchain.

Original example by Massimo Franceschet.

- https://youtu.be/J-ab9was1p0
- https://youtu.be/SSo_EIwHSd4
- https://youtu.be/AQDCe585Lnc
- https://youtu.be/XCo6yyutYAM
- https://youtu.be/M3EFi_POhps
- https://youtu.be/5I3wYAwbKMM
- https://youtu.be/OcmvMs4AMbM

A blockchain is a distributed system using cryptography to secure an evolving consensus about an economically valuable token. Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. The prefix *crypt-* means hidden or vault, and the suffix *-graphy* stands for writing.

7.1 Block.

A blockchain is a chain of blocks. Hence, we first describe a block. A block is a container for data. In its simplest form it contains:

- an identification number.
- a timestamp of block creation.
- a bunch of data.
- a reference to the previous block (parent block) in the chain.

```
# A list allow us to incorporate different kind of data.
block <- list(number = 3, # ID.
              timestamp = "2018-10-01 17:24:18 CEST", # Timestamp.
              data = "London", # Data.
              parent = 2) # Reference to the previous block in the chain.
block
```

```
## $number
## [1] 3
##
## $timestamp
## [1] "2018-10-01 17:24:18 CEST"
```

```
##
## $data
## [1] "London"
##
## $parent
## [1] 2
```

This was a single unchained block. Now, let's introduce the concept of chain.

7.2 Chain.

Blocks are concatenated into a chain. The first block of the chain is called the *genesis block* and has no parent block. The last block of a chain is the *current block* and is not parent of any other block (yet).

Here is the genesis block of the Ethereum blockchain. https://etherscan.io/block/0 https://www.ethereum.org/

Let's chain some blocks.

```
# Some blocks.
block1 <- list(number = 1,
               timestamp = "2018-10-01 17:24:00 CEST",
               data = "London",
               parent = NA)

block2 <- list(number = 2,
               timestamp = "2018-10-01 17:24:15 CEST",
               data = "Paris",
               parent = 1) # We chain the block with respect to the last one.

block3 <- list(number = 3,
               timestamp = "2018-10-01 17:24:30 CEST",
               data = "Rome",
               parent = 2) # We chain the block with respect to the last one.

# The blockchain:
blockchain <- list(block1, block2, block3)
blockchain
```

```
## [[1]]
## [[1]]$number
## [1] 1
##
## [[1]]$timestamp
## [1] "2018-10-01 17:24:00 CEST"
##
## [[1]]$data
## [1] "London"
##
## [[1]]$parent
## [1] NA
##
##
## [[2]]
## [[2]]$number
## [1] 2
##
## [[2]]$timestamp
## [1] "2018-10-01 17:24:15 CEST"
##
## [[2]]$data
## [1] "Paris"
##
## [[2]]$parent
## [1] 1
##
##
## [[3]]
## [[3]]$number
## [1] 3
##
## [[3]]$timestamp
## [1] "2018-10-01 17:24:30 CEST"
##
## [[3]]$data
```

```
## [1] "Rome"
##
## [[3]]$parent
## [1] 2
```

This is how we can get one block from the blockchain.

```
# Get the 2nd block.
blockchain[[2]]
```

```
## $number
## [1] 2
##
## $timestamp
## [1] "2018-10-01 17:24:15 CEST"
##
## $data
## [1] "Paris"
##
## $parent
## [1] 1
```

Let's also write a validation function for the blockchain. For now, we only check that the parent field of a non-genesis block references to the previous block in the chain. Therefore, we validate a blockchain as long as its sequence is correct. That is not a very secure system but it works for now. For example, first the genesis, then the first one, then the second, and finally the third.

```
# Returns false if the blockchain sequence is not correct.
validate = function(blockchain) {
  if (length(blockchain) >= 2) {
    for (i in 2:length(blockchain)) { # We skip the genesis block.
      if (blockchain[[i]]$parent != blockchain[[i-1]]$number) {
        return(FALSE)
      }
    }
  }
  return(TRUE)
}
```

```
validate(blockchain)
```

```
## [1] TRUE
```

Our blockchain is valid. At the moment the validation criterion is quite simple. The next subsection introduces a more strict validation criterion.

7.3 Hash.

The linking structure of the chain is more complex than one described above: each block in a blockchain is identified by a hash value of its content and references to the hash of the parent block, or 0 if the block is the genesis one.

Hashes of blocks are created using cryptographic hash functions, that are mathematical algorithms that maps data of arbitrary size to a bit string of a fixed size (called hash or digest). The hash is used to certify the information content of the block: by modifying even a single bit of the content, the hash completely changes. Furthermore, notice that the digest of the current block is computed in terms of the previous block digest. This means if you alter one block you need to modify not only the hash of it but that of all following block for the chain to be valid. As a result, a hash function for three blocks looks like this: $hash_3 = f_3(f_2(f_1(block_1), block_2), block_3)$.

The hash algorithm used here (SHA-256) is part of SHA-2 (Secure Hash Algorithm 2), a set of cryptographic hash functions designed by the United States National Security Agency (NSA). In particular, it uses digests of 256 bits (or 32 hexadecimal figures). See for example:

https://passwordsgenerator.net/sha256-hash-generator/

It is implemented in R package `digest`.

```
# Creates hash digests of arbitrary R objects.
library(digest)

# Hash a string.
digest("A blockchain is a chain of blocks", "sha256")
```

```
## [1] "5c2005976411a1628fabcdde3ac04be563d18943e710b7446afa2a8a5fab9abc"
```

The hash 256 has a length of 64 characters.

```
nchar("A blockchain is a chain of blocks")
```

```
## [1] 33
```

```
nchar("5c2005976411a1628fabcdde3ac04be563d18943e710b7446afa2a8a5fab9abc")
```

```
## [1] 64
```

```
# Hash blocks.
block1 <- list(number = 1,
               timestamp = "2018-10-01 17:24:00 CEST",
               data = "London",
               parent_hash = "0")
# Note that the whole block1 content is used to generate the hash.
block1$hash = digest(block1, "sha256")
# The new block2 has block1 hash as the parent hash.
block2 <- list(number = 2,
               timestamp = "2018-10-01 17:24:15 CEST",
               data = "Paris",
               parent_hash = block1$hash)
# Note that the block 2 hash has block1 hash.
block2$hash = digest(block2, "sha256")
# The new block3 has block2 hash as the parent hash.
block3 <- list(number = 3,
               timestamp = "2018-10-01 17:24:30 CEST",
               data = "Rome",
               parent_hash = block2$hash)
# Block 3 hash has block2 hash, which at the same time has block1 hash.
block3$hash = digest(block3, "sha256")
# The blockchain.
blockchain = list(block1, block2, block3)
blockchain
```

```
## [[1]]
## [[1]]$number
## [1] 1
##
## [[1]]$timestamp
```

```
## [1] "2018-10-01 17:24:00 CEST"
##
## [[1]]$data
## [1] "London"
##
## [[1]]$parent_hash
## [1] "0"
##
## [[1]]$hash
## [1] "e4064e41f246cba04e0e7d58418294c4e099aefff48f7e871f65a77aebbe8243"
##
##
## [[2]]
## [[2]]$number
## [1] 2
##
## [[2]]$timestamp
## [1] "2018-10-01 17:24:15 CEST"
##
## [[2]]$data
## [1] "Paris"
##
## [[2]]$parent_hash
## [1] "e4064e41f246cba04e0e7d58418294c4e099aefff48f7e871f65a77aebbe8243"
##
## [[2]]$hash
## [1] "12dfcc49b856a2d2c6a941b9166ee14d276811e453f11a24f7bef8aa3247966d"
##
##
## [[3]]
## [[3]]$number
## [1] 3
##
## [[3]]$timestamp
## [1] "2018-10-01 17:24:30 CEST"
##
```

```
## [[3]]$data
## [1] "Rome"
##
## [[3]]$parent_hash
## [1] "12dfcc49b856a2d2c6a941b9166ee14d276811e453f11a24f7bef8aa3247966d"
##
## [[3]]$hash
## [1] "22628b887626257faa3109dd7f786e88ea0d5ebe021a534afaef3ced3ad193d1"
```

Let's update the validation function. Now, we check that the parent field of a non-genesis block references to the correct hash of the following block.

```
validate = function(blockchain) {
  for (i in 1:length(blockchain)) {
    block = blockchain[[i]]
    hash = block$hash
    block$hash = NULL
    hash_expected = digest(block, "sha256")
    if (hash != hash_expected) {
      return(FALSE)
    }
  }
  if (length(blockchain) >= 2) {
    for (i in 2:length(blockchain)) {
      if (blockchain[[i]]$parent_hash != blockchain[[i-1]]$hash) {
        return(FALSE)
      }
    }
  }
  return(TRUE)
}

validate(blockchain)
```

```
## [1] TRUE
```

Let's test the validation algorithm.

```
blockchain[[1]]$data
```

```
## [1] "London"
```

```
# alter data of first block
blockchain[[1]]$data = "Budapest"
validate(blockchain)
```

```
## [1] FALSE
```

It works.

```
# restore data
blockchain[[1]]$data = "London"
validate(blockchain)
```

```
## [1] TRUE
```

Another test of the validation algorithm.

```
# Alter data and hash of first block.
blockchain[[1]]$data = "Budapest"
blockchain[[1]]$hash = NULL
blockchain[[1]]$hash = digest(blockchain[[1]], "sha256")

validate(blockchain)
```

```
## [1] FALSE
```

Works well.

7.4 Proof-of-Work.

Hash alone is not enough to prevent tampering, since hash values can be computed fast by computers. A Proof-of-Work (PoW) algorithm controls the difficulty of creating a new block. For blockchains like BitCoin or Ethereum blocks are created (mined) by so-called *miners*. When a new block has to be created, a hard computational problem is sent out to the network. The miner which solves the problem first creates the new block and is rewarded in crypto currency.

In the case of BitCoin the PoW problem involves the problem of finding a number (called *nonce*) that once added to the block is such that the corresponding block hash contains a

certain amount of leading zeros called difficulty (more specifically, Hashcash). The average work that a miner needs to perform in order to find a valid nonce is exponential in the difficulty, while one can verify the validity of the block by executing a single hash function.

```
proof_of_work = function(block, difficulty, print = FALSE) {
  block$nonce <- 0 # Add a nonce=0 to the block as a starting point.
  hash = digest(block, "sha256") # Generate the hash of the block.
  zero <- paste(rep("0", difficulty), collapse = "") # Number of leading 0's.
  while(substr(hash, 1, difficulty) != zero) { # Hash has not the leading 0's.
      block$nonce = block$nonce + 1 # Add 1 to the nonce value.
      hash = digest(block, "sha256")  # Generate the hash of the block.
      if(print == TRUE) {print(hash)} # Print only if TRUE.
  }
  return(list(hash = hash, nonce = block$nonce))
}

# This block will allow us to evaluate the function above.
block <- list(number = 1,
              timestamp = "2018-10-01 17:24:00 CEST",
              data = "London",
hash =
  "88e96d4537bea4d9c05d12549907b32561d3bf31f45aae734cdc119f13406cb6Parent",
              parent_hash = "d4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb
```

How difficult is difficulty 1?

```
# Evaluate a difficulty of 1 (one leading zero)
proof_of_work(block, 1, print = TRUE)
```

```
## [1] "2ac786a13832b45ccf25466aec83ef714dae4562c9573511dedc10543568562c"
## [1] "80d238e881cb74c5dd100adb12a7ba5d6c84ff1759320b1eda3fdc74eb205d52"
## [1] "d51eeaf4dc82d6c52b452c9685c7fdf8d347c3612f7cb46c9e3fe1a6baff73a2"
## [1] "495fbc7ec32f0edffc9af8344832f0a83ccf1f8ec4c650661620e38ddb5947ab"
## [1] "a7d86c47ba718439066df81bffff60a41d3cd614ba31a93faed4c609a04b9c7e"
## [1] "0a72347fbbe9ac46481b6499e3bcacd394f267d4c3d8a0544e00978fc124fe2f"

## $hash
## [1] "0a72347fbbe9ac46481b6499e3bcacd394f267d4c3d8a0544e00978fc124fe2f"
##
```

```
## $nonce
## [1] 6
```

It takes 6 iterations to solve the problem of difficulty 1. Quite easy for a computer. How difficult is difficulty 2?

```
# Evaluate a difficulty of 2 (two leading zeros)
proof_of_work(block, 2, print = TRUE)
```

```
## [1] "2ac786a13832b45ccf25466aec83ef714dae4562c9573511dedc10543568562c"
## [1] "80d238e881cb74c5dd100adb12a7ba5d6c84ff1759320b1eda3fdc74eb205d52"
## [1] "d51eeaf4dc82d6c52b452c9685c7fdf8d347c3612f7cb46c9e3fe1a6baff73a2"
## [1] "495fbc7ec32f0edffc9af8344832f0a83ccf1f8ec4c650661620e38ddb5947ab"
## [1] "a7d86c47ba718439066df81bffff60a41d3cd614ba31a93faed4c609a04b9c7e"
## [1] "0a72347fbbe9ac46481b6499e3bcacd394f267d4c3d8a0544e00978fc124fe2f"
## [1] "7d8d9144dd86cf70b24010534497970ae1e89ff7df03645d781ffbc4c5350988"
## [1] "abb9814a9680c1ba7c970eb5f8965e01f074af7547243db9d94b0a44908d5b1a"
## [1] "c61eebd36518426ac3510280eb698484856a2c9b93e1a534298cf11a3d7a62f9"
## [1] "7fe608cd3784913fed93e01263e9d013a337adbd562ee76280e4775d24adeabd"
## [1] "0e1344bc16dbbb1ee2ee2687a47925bcc82582e7fd90b281400ec9b467e81f68"
## [1] "35979c8a93866a4ec7d8e1ca1a5c9a1746c60c18d80d885d2d66fe67c773fcc0"
## [1] "3cd3f8a34fa46d01b4faedb6309a19b917fa133cede66d186ac4d5eb20685493"
## [1] "3fcab70d4a7710a16fa23b6f63024159987b3b95fb62841e60dfb824aff22d7e"
## [1] "89168037cbc585eed85f318b8063d48f350410052b47aa51a67e2a1a6a20870e"
## [1] "c233234c2ff42b73c4c633d6bd54b197a33774da062e48d02b384efe64cff332"
## [1] "75f98266dfa61fd5578f942f5140bbe58c6bbd285dab7375595b0ddaa29ce6ab"
## [1] "955d82a9f0ca7d4eaeb51ff65551b0eb5c3416c89252f6530e286d28abb22d01"
## [1] "5c5e31b26f1761d74e7409312d658be0053e5519ddfef914453a552e65832e4c"
## [1] "6cb0f90002461d203f826aa63c2abe950da14d986395b6625c03d652f3bcc15c"
## [1] "5b05120be9c2acaa937c6e8ed703d9c46c314e1bb8ec44cb8575716c7fbfc575"
## [1] "1dc89bb3f3d3168e141a78d478ef0416b9db35db49ea29d83b39d4fd39fcf137"
## [1] "11eb55ea6aa08526ea29fcbb7b2627571ce4a7ee64214f9a97bb7c11e41f65e6"
## [1] "4ed393b7603b7dcaf2c39936d8261189ac5d3fddcdd26a9c8be93da87e4f3b8a"
## [1] "1f0bb59cfc4dc2e50aa0721bc9cddc28eb1a31d64ec90c86a4579df6147342df"
## [1] "3bbf029721533141094663e2b9b2bb9ec7d69b299c1e5945bacd77dd5c736138"
## [1] "b4edd13b1f670ce9c0d66c17bb7bee6ddcb64a194f3d040d5df20b3a28c2b46e"
## [1] "b5a553d535ad26d10b4e7e8cfcc27a35696a7ce75bb1a5f1dd4322c5039877f2"
## [1] "186a2116c7fa43bb97c0573e6878b3260b57b1f854caa0412b0a45dc456f2ba2"
```

```
## [1] "ec5eb69718fb8a01bf0da8fe5320571f3671e740a8c7bc799224b8da069d7748"
## [1] "436460d156efc75f8c038c45544bbb14fc0261c3b83bdb576d36e145c759b285"
## [1] "2081ced993d0faa150a8780cff8793a871b171b775903a3e5d844c6935c8fbc7"
## [1] "332c6d6acf42dfef8b67a956367c4c318e56dca912b002ab51e62b495452eeb6"
## [1] "5193a0f62d3c57e03817228efe8c624d19e22424d529c0ceee0dcef5615c1ee7"
## [1] "6f0fc844d5c1ca51f9f64d2c34edd41fc12b3b5670cb9ae9b0f5dcb4214e90ee"
## [1] "1e399483289deafbcd0e08b5475ce984838ef09362fea3c89df8e9cd708235ae"
## [1] "49e12d2adde1c3e406a425552b676cd6b6c5006a91fb9ef02ece4f88f6dc150c"
## [1] "8d6e35c790d3e5a8d4b1b262433d81de2628a227133a036aa758b45f16b712ff"
## [1] "4a6aa4b71b76f38c063180c1506b2c0b96362f439fcb597fedf39d9bdb465ff1"
## [1] "23d12c93c530f0d7642ab4e60d0d3ec4ed71429f75c8d9b43810058a3c795a80"
## [1] "6e1912048fb80b8a6e7f4f8edde9ff79ade8b0f28c8e4f901294d4a8f7345586"
## [1] "4533c74f4933d43730495774af13db8d0d7c917522e16fcd0f55062afce09be0"
## [1] "82d16a1b0e2927216c8a241cb45fa0151cd40192b1c084e8e5b9a086f2a7c8cf"
## [1] "0cd03a81725e5133e68a693d99a8763335eb13d65f2fa1e3a2b6a1e9badca154"
## [1] "acf65bdea8bc4729749363cfe4e8ee268e8640c056c025955c0458de3b4b44ca"
## [1] "5981a2f324b01a80674f016eaa4fe44fe577842a9d825ef8e5ed06801978b865"
## [1] "8365e175af54bf134f21ef5b80c1aea66604893db27ba829118466b255d93c04"
## [1] "c98b16c8ea2928361a4ede39d38638e8927d0e6bbf5ce1f688b2fddde02e82af"
## [1] "2dafb2f5487cae69c48eedcb3bef62b6bf697a3e00cd359cc30e46e1b21bc46c"
## [1] "7aafae605f73a701ece38a1fe7e292ade217ecf99f8fcc45f759c34fcd83002d"
## [1] "eefac2bb3d27c16467125cab3be6470ea5a7d4edc749e1b3a6991fdc7f953ab7"
## [1] "706238f0d708202d409979ab444f044ed334b22e251331605bf81e8a5e5ffdd7"
## [1] "cc4aefc81929ac048f9ee988e32f5ede146afaa0016a05f8f94c9cce9324c69b"
## [1] "20c7a47dcc8bddba10358e39ea7cacf7d2a3bf783cc076117514a9b561a67700"
## [1] "1b3c6f6ec9abbad479df717fb62ceeccfe0c00371c8e4c541d68889cdd5fee98"
## [1] "07dc06fe1d907c5d018c53d6433c38f92f1d27ff7327492d2d3e0e43fbb1c837"
## [1] "70a1705abde4f30ac8e5c9ebdad79041d959de16607284e9b5add1c968686b05"
## [1] "f12a87c31615488e93f157b3b7925462a337c89778780f5a8086e5a5c0c56002"
## [1] "907ab7af7655531cb841b08db754cc790d8a6bcd3147310c66f5eae2db35401a"
## [1] "70d73d143c9e756754172d6eb9546d1f461bda31f2aa0de7e47c336368ea9286"
## [1] "a4de79e953f370022e4d6f511b928551587c42a0c849020f64c4605b33626b15"
## [1] "77b56e696d6ef39a954804412df280bfac81db42148f5e8f80127c0139c83ea6"
## [1] "f1491dc9536143365f8898d3d2adf6fc7871fa8be6bd27e89d9bb8f195dbd58c"
## [1] "ac0041cad81f86bd7625ee07e9769f6286bfeeae51e8b5c1ecd8351983d7cc9d"
## [1] "813f37737ea36c1ba97b5585f3bb16a13145b5f89e564d71248df36723165201"
```

```
## [1] "1a11c0669487cddeb9c7d2c37f89b3cd9bd0e3cf7beb9b7e2ab2eff8d826eff8"
## [1] "16e4f4a9308bb10784d212214270f1aea7d77f3320bd80427f72e40a19d5ff5c"
## [1] "1da929217ccdfd4f80e2c2c5e9fde25dea6c3980c54e5637fffe001f80bc9747"
## [1] "cbd56cca5f9ba3b0a7c7533c1c6015e719cf9e5402ea70ba7b1bd6b1c6489263"
## [1] "072fcf30ef73d861af1cdc73d29097fb0bf6482a09607e58fc79eb69a98329ad"
## [1] "53d8a89a4bee7db969c85abc5c7201b3a94751ba0a5566a3da5917a62e2a881d"
## [1] "a01522b8b24fcd160bf6b650b214ca7ce5ad66d3ee5aacdb6b7331d35d38abb8"
## [1] "4f1acdd38c7f87d99319012003483c7b3f2cd5c4b3046eba8b5d8181aee261ee"
## [1] "6e7a41a5c635cb692b6be1f180a6df8b2f35cd636df4861a80ab86448b2095c8"
## [1] "a310750d90857ed7b24527ddbc0cdf81049990becbf27004869adabc2c4b1013"
## [1] "305c739452d9a5d2f0f0b7785cc6fcbfa0d97d08ef52bc2d38c4756abd532b54"
## [1] "b59b0bb62486aea41ed1e79a9fa2a96be5cb569e25b6e4a855156ee7dad4cf31"
## [1] "4ff6199ec0734b9f7073b1ad2fb309dc42abab36dd4986d4e7aeae639affc4b4"
## [1] "d6ddaadaf7ea44af2ecc9ce19f97600391a30126ef10994c62667f710ca69090"
## [1] "2173de29b60f2275e8fac1e165676f2c276cfce19ddfe55a509dfc507f3566f9"
## [1] "41277987e6d0943ac5cdde53f398dd5a7c5ea49695437ec0a8e79aab5b5641ee"
## [1] "5c855c099e96e5327b93ba5ae16af6a98b3755cdd9c044db1632030ac0fb4764"
## [1] "fcff924a7cd4030544bc26412fc06baafae168bf9c733eedf35ca8c9e29866fe"
## [1] "84104b2db9f7de1a18c337d63f8ec1a0e7622d693b0b34b69d1f26c6fdd95b62"
## [1] "ef263baf2f39716ed981556a8ceed3f9b7d494aca0030bb08c8f7a76f568af66"
## [1] "1bc17844a4c77e70dd9c95545e6490011db36b78b9f8bae3f2d2876c1a25e72a"
## [1] "683211739c5519be35016bd7db8812e272d839c26aeace423c2786e296cbe7eb"
## [1] "049c9ae3a95d7d806cf0bdfbc3aaa8b66a370adf66d918671cc501b7eb16893c"
## [1] "98068d9988acf648dbe3768a6f4f9aaadfc4e4cacd8acacc73288bb7c3049c18"
## [1] "456fc562f458f3b0d0dfdfc148965520ec5ee12aa7c3b715bc38b1122e2b1d68"
## [1] "ac3af32e229d572a8a945a22b23a9522c2d3a121cb77a813686ef742714800a7"
## [1] "3657d923f73dca7f12fca7001cdb757c817b8697af1e2c42d45147bc84463fe8"
## [1] "3442caf8b04a69ba355b8948d4e065a03bde0120516ca3f6ea8914dce3f987bd"
## [1] "cacd037f2f9b610acfcd8ec44a3721c0c2144de59bb5450d18367d9044f91360"
## [1] "7ace1e58c1222cf7c1e0df4bca9356e238cb6a713e247a07b0735d4b07584ed9"
## [1] "9104d8a71ec23a48ee2f1dde07acbe7a8072bcfa7d20f189fbad2904dfeb78d2"
## [1] "860abeb5ed9e479c879b8d3caa0ecfb97e2c30c609d80a6f5a88d2fbe916a1fc"
## [1] "5162ec75fc2dd6b382e2f4f3c00a0f615954a11de4ecf44006b88ec44f99539a"
## [1] "1f931b0d9af7b9d580e3a163e5e86d9e012deedd22aaacd43d3103a25374f7a2"
## [1] "c8b722b23c5e40ba2f2525515b876be5ebad271060cb53f7838f2c8ed263e148"
## [1] "b5c405a0db9f29811a72a2c41ad95fbed4d6e9c415d4116f7b0b278e5f54d260"
```

```
## [1] "87b5a6962debb873e956ce00b2ccd795cbc79e28626eae87fe451a759056d269"
## [1] "46833431670435f7f6f90285075ddd5a34576ca758953c01f937d170674cacec"
## [1] "fb3d6f18a15b0a2cb56620e98edb5e9859d8b00aa3b4f4471665ba544590b065"
## [1] "39e5cef0b6fab302c47756379956c251595eab426437a69deb2235bd3e57012d"
## [1] "9252f22d653092513e7884a9683f8da284e6f0ce569319eb84a2ad76b5706cbc"
## [1] "66a709069db59ff4784db4227969a2e0e16b087363ee724dae3e078088821a14"
## [1] "3b2e9ddc6f7f4902de3a01ad59b9dc04530aa31202b91526ee2d51a2fb0c00da"
## [1] "21514f1f808593feecf52cc2b9b2f9b56619af9c4d0faaad82d48731c367b557"
## [1] "219c7d2727e44929f38d1f5e36f1bc616a31d04bd097245788c2e1d9f919b077"
## [1] "1093ed88bb41127ada9191ce767d301f53589821d69e88ea8858e55d39e793ae"
## [1] "9c07c5aaec78eacec5078e30cdeed30353d3f265a7bdafc39b596b08bb69ee57"
## [1] "c6c998cf58a382829f19e178b852db5648caa1432b022ee80a178d707654eaac"
## [1] "1823ef4d4d8d90a63e9400fc80f23f83b71f12310415706db18c9095c9795e51"
## [1] "1314d466891aab41490ffd96dcb8b75f7574f05dbe7c3f19caa76a486fb21b71"
## [1] "5d1583ef53ad165c7391e04e91375c1759808db3ca69d756c3c6c139cf6f3720"
## [1] "ae8ab2358f553a5a215690042c1e70de8b4b5c4ee508dd59245677f3989a1563"
## [1] "ad4eeb607d10505eea0509fa15bc777d24f11b1ec087a2e3d6021a05d9b60659"
## [1] "8ac87302354e7a84e69f4c21b1080e94c4bdb6e81dbe6f9416260063ced369cc"
## [1] "a8b2e8a1b078555263811dabfa5cb093ef4dcf101441af63ff95f4a1a28945c8"
## [1] "4d0e1bf41f146c6adf07123e4ab762402a77f0c3c8a006d625f30277b47affcb"
## [1] "50f556421663748be602a3b38bfe4adfbf882228320134c029b91e68214e8a68"
## [1] "5391020a77657d85a8e4c3d663d931efdb236c998e4d0f378ac0a63d3696c5f4"
## [1] "1e9840d0fffde05227c7979e464143d29302c197585fe731dd6528f72adcfa51"
## [1] "6167a09373071a5ba01fcde8272f370efff1d88abd54ca704a7126db1a523023"
## [1] "be6cda1e9261fed00192044908a4cf679e536fd9684111a61ce35c70d8be9be7"
## [1] "fa1456d8392dba2d27c5bda83e20bdb0295614a91569bb85f00c69eabb88540e"
## [1] "b9e211bbed593135b2f58e1cc412e6166702c67dfdab358fdc74183f6a945e93"
## [1] "8771dd4f9bc9da3c0744a2b733adc3dd479762085464d5f08719795d41ab5d07"
## [1] "74821c8f2a305c3eac7b9bc9c809f39811f652e88869fc03b227368f3f1b17b0"
## [1] "551301f75871d4feb21289ad44113ff8b394347ddd6d30c9c4eccd784167a41c"
## [1] "55a261ba8cfd352ca563eb9d62c1dcd6a3c3303543b2171c6b4b21dcb0b5d1bf"
## [1] "d249e152c0cabbd2d8e1914e95782c7c8e1c52b804d1a0b280d886b104a63d07"
## [1] "5940cc54f0a5433a6162bd37e47c159b19ebe04c3d2e7e89bd76e2f2e783647e"
## [1] "419b96e236b29536aae8908681df2d554317e54342f4563603cce2ce821c6e82"
## [1] "10010e32757342c7d51275cb96e1a7d78094724121d63ce1cd497c659da0a9f9"
## [1] "cac1870358b94529781cd46abff8c41b3d3a09638820e015648d4ce2e67f44d0"
```

```
## [1] "e44814b8a26051bd63c4b162c19c6b387d0174b36bf13f4bdf0f36cbaffdfec2"
## [1] "850617605c3d60ecd0145bfe4a8053fa529e9061f46824d3b902f8706161ee8c"
## [1] "4a5f0d925bc3e6641ecedf78487fc6fa23efb0ad5a00708d80e6210271f8e881"
## [1] "690bf0da1035be54725598ea0b30589bf09b2fe2298c4e021ca90b52fac634bf"
## [1] "e4aea7dcaa870a731fc2a3a04b318540a3a21fee406dfa098fb467e22f49380f"
## [1] "79215c4cc40681dfd88dedd5df0d1fd4f7b62a7fb64abde4ee2848b25c19df1a"
## [1] "13a08458d3af517b14e055f08d9852b22077844c7b372b73edaf479125ff51f7"
## [1] "5e57c85f8c9f603ddea5bbb312ff0261869be4581e582cfea5c2cab4b8f8236e"
## [1] "683c94a5d35aae72308415911e50414bf8d3f0813cca154e28e0b148524df9b7"
## [1] "cc0b40e48c3bb9ae06df848fa49873f5cf5d1f72991ba8280553ed68e4eb533e"
## [1] "80e7ddf3fee383209542a8ec40de5f923a3f81cd74d10b7a0552898b40aef5ee"
## [1] "50bbe55e82304fbcbc6a385dcf2625f8dbc6138dd2919791431c744fe5ed81e1"
## [1] "d1339007864e3dc8f59e2eae95bea59512818abadcae1ca98785cbed87b5879e"
## [1] "2eec1a0c571a37322daa637766a087db6d0c0075f5465af4e552f233bfdfa081"
## [1] "34dff9655eb63b7732184707cfef092d63082f5e661561d05a753fb21d06e5a0"
## [1] "a78f257422b7a915e3613f9fc61c099a874b71c4a515b1a5e14d5f9c8426f7ce"
## [1] "33d1915f3c2fd3ef06cf4fcb4c0341037a41ecb55d23abc0d8519da588fe13e4"
## [1] "098ae6adfee83c74b419015e4c24b6a3d0d67604852af563d7dca79b6f78f63f"
## [1] "e728212b1f5ab89604be312d5c482d4ba40db1f72a9ddb5f1f4c09b5e56b36c0"
## [1] "9a0e2179e639cc0cd57944ce35e3a2e4cec1bbbfc547059bd9da758b4f828c4f"
## [1] "8b6178e6ce14b58635357d434f7fb21d3183b6e5d396f76a923e6269d481b870"
## [1] "bb842bac82a00602cedd230e460056aa5a53ab094bbff5a681d217985b5f0309"
## [1] "929731c7b52798f066238f85423e7dc8a105ad3037f5f92da26bd065553b443f"
## [1] "98cb093b7305be1ee75f80bebf03d1661e4ffcd303b6b4438d0a37dd07200644"
## [1] "2403175b1811790a93347df0313902446e455273aac7dd6b92ba0d229483adf6"
## [1] "aa84ac653b8fb19914bb687f08b190f458ad7620253687d2bf5222bb1eb0b899"
## [1] "239a832c6b7bdd66c51a4b85a734da94c7d1adf34320834f45a2e5116244fbe4"
## [1] "005269bb674beb32c104bb415f9cd7501aace44bb3de16758b840ed08e12900a"

## $hash
## [1] "005269bb674beb32c104bb415f9cd7501aace44bb3de16758b840ed08e12900a"
##
## $nonce
## [1] 165
```

It takes 165 iterations to solve the problem of difficulty 2. How difficult is difficulty 3? This time `print = FALSE` to save space.

```
# Evaluate a difficulty of 3 (three leading zeros)
proof_of_work(block, 3)
```

```
## $hash
## [1] "0003b3433712535240d61d5c5c29c2975ec8a170e941323d489bcc5fedad55f6"
##
## $nonce
## [1] 2937
```

It takes 2937 iterations to solve the problem of difficulty 3.

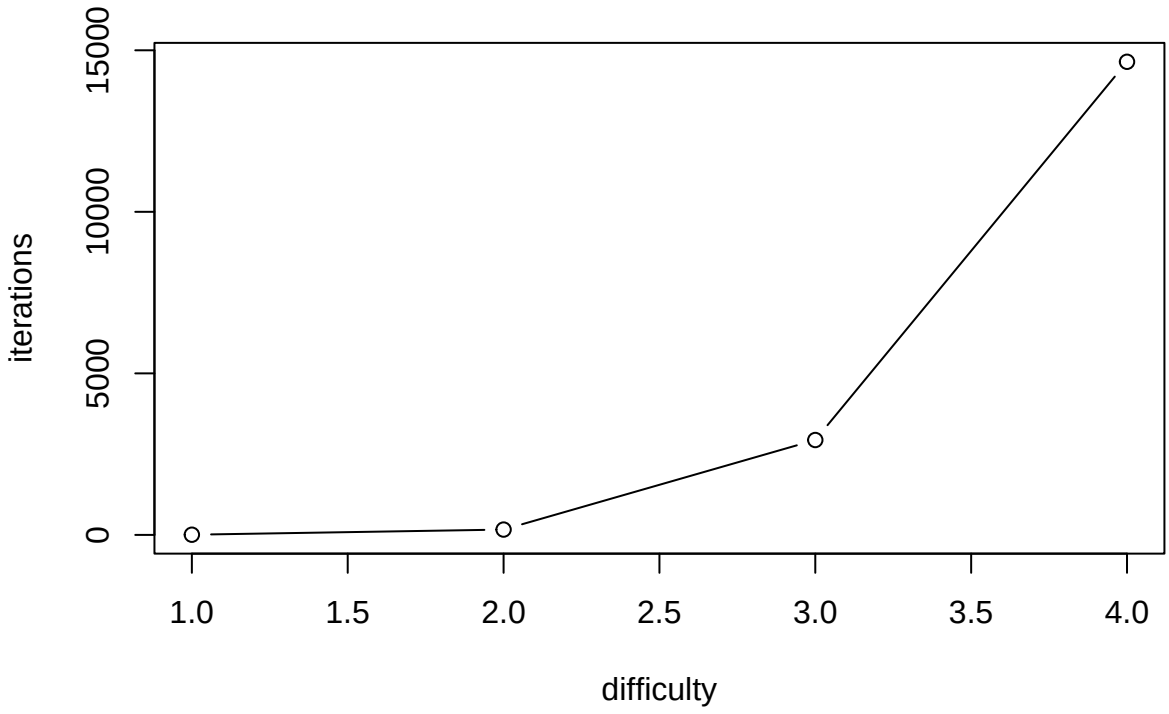
See a 19 leading zeros example: https://blockstream.info/block/000000000000000000ae7a6057c9c3d00ff21c1f8f87ca4413d89059cd9f2c6

```
n = 4 # Do not try difficulty 5!
# Collect the nonce for the first four levels of difficulty.
iterations = vector("integer", n)
for (i in 1:n) {
  iterations[i] = proof_of_work(block, i)$nonce
}

iterations
```

```
## [1]     6   165  2937 14644
```

See how difficult the problem gets as we look for more leading zeros in the hash.

```
plot(1:n, iterations, type = "b", xlab = "difficulty",
     ylab = "iterations")
```



Now let's build a blockchain using the PoW method:

```
# Mine algorithm.
mine <- function(previous_block, difficulty, genesis = FALSE){
  op <- options(digits.secs = 6) # To get millisecond timestamps.
  if (genesis) {
    # Define genesis block.
    new_block <-  list(number = 1,
                       timestamp = Sys.time(),
                       data = "I'm genesis block",
                       parent_hash = "0")
  } else {
    # Create new block.
    new_block <- list(number = previous_block$number + 1,
                      timestamp = Sys.time(),
                      data = paste0("I'm block ", previous_block$number + 1),
                      parent_hash = previous_block$hash)
  }
```

```
  # Add nonce with PoW.
  new_block$nonce <- proof_of_work(new_block, difficulty)$nonce
  # Add hash.
  new_block$hash <- digest(new_block, "sha256")
  return(new_block)
}

# Create a blockchain using the PoW method:
blockchained = function(difficulty, nblocks) {
  # Mine genesis block.
  block_genesis = mine(NULL, difficulty, TRUE)
  # First block is the genesis block.
  blockchain <- list(block_genesis)

  if (nblocks >= 2) {
    # Add new blocks to the chain.
    for (i in 2:nblocks){
      blockchain[[i]] <- mine(blockchain[[i-1]], difficulty)
    }
    }
  return(blockchain)
  }
```

Create 4 blocks with difficulty 3.

```
# 4 blocks, difficulty 3
blockchained(difficulty = 3, nblocks = 4)
```

```
## [[1]]
## [[1]]$number
## [1] 1
##
## [[1]]$timestamp
## [1] "2021-07-05 22:48:56.483855 CDT"
##
## [[1]]$data
## [1] "I'm genesis block"
```

```
##
## [[1]]$parent_hash
## [1] "0"
##
## [[1]]$nonce
## [1] 654
##
## [[1]]$hash
## [1] "000ae9e6b3a007b962febcc685838435839d5ae57f19c8d47fd0ecc44ee18f1f"
##
##
## [[2]]
## [[2]]$number
## [1] 2
##
## [[2]]$timestamp
## [1] "2021-07-05 22:48:56.519784 CDT"
##
## [[2]]$data
## [1] "I'm block 2"
##
## [[2]]$parent_hash
## [1] "000ae9e6b3a007b962febcc685838435839d5ae57f19c8d47fd0ecc44ee18f1f"
##
## [[2]]$nonce
## [1] 5001
##
## [[2]]$hash
## [1] "0004de1a323c834c43866eb686ae6da6624c9b8a4c5274f274725dea028e7ec5"
##
##
## [[3]]
## [[3]]$number
## [1] 3
##
## [[3]]$timestamp
```

```
## [1] "2021-07-05 22:48:56.816031 CDT"
##
## [[3]]$data
## [1] "I'm block 3"
##
## [[3]]$parent_hash
## [1] "0004de1a323c834c43866eb686ae6da6624c9b8a4c5274f274725dea028e7ec5"
##
## [[3]]$nonce
## [1] 457
##
## [[3]]$hash
## [1] "000e5123f6724e3565a6c422dea402616955857d2c3dfbaba829952ddde55f7d"
##
##
## [[4]]
## [[4]]$number
## [1] 4
##
## [[4]]$timestamp
## [1] "2021-07-05 22:48:56.83648 CDT"
##
## [[4]]$data
## [1] "I'm block 4"
##
## [[4]]$parent_hash
## [1] "000e5123f6724e3565a6c422dea402616955857d2c3dfbaba829952ddde55f7d"
##
## [[4]]$nonce
## [1] 2105
##
## [[4]]$hash
## [1] "000fd64d405ad34ff271a38a5d13acf81756ac021c6baf4eaadac00054326485"
```

Nice.

7.5 Transactions.

Typically, the data field of a block contains a certain number of transactions. Each transaction has a sender, a receiver and a value of crypto currency that is transferred from sender to receiver. Moreover, it contains a fee of the transaction. Transactions are stored in a pool of pending transactions and each mined block will include a (proper) subset of pending transactions. The miner of the block gets the fees of all blocked transactions plus a fixed mining reward (and this transaction is included in the following block). This is how new coins are introduced in the blockchain economy.

Let's create a pool of fictitious pending transactions. We store them in a data frame structure using the `tibble` package.

```
library(tibble)
ntrx <- 10 # Number of transactions.
sender <- sample(LETTERS, ntrx) # Sender names.
receiver <- sample(LETTERS, ntrx) # Receiver names.
value <- round(runif(n = ntrx, min = 0, max = 100), 0) # Transaction value.
fee <- round(runif(n = ntrx, min = 0, max = 1), 2) # Tansaction fee.
(transactions = tibble(sender, receiver, value, fee)) # Pending transactions.
```

```
## # A tibble: 10 x 4
##    sender receiver value   fee
##    <chr>  <chr>    <dbl> <dbl>
##  1 X      Q           15  0.15
##  2 S      K           33  0.34
##  3 Q      D           73  0.23
##  4 J      W           68  0.1
##  5 O      N           56  0.5
##  6 E      U           40  0.57
##  7 L      C            7  0.8
##  8 Z      T          100  0.63
##  9 V      E            2  0.35
## 10 Y      X           45  0.45
```

Let's update the mining function. Each block will include the most profitable transactions (the ones with the highest fees) among the pending ones. The miner of the block gets as reward the sum of the fees of transactions in the block. The reward transaction is inserted in the next block. We take advantage of `dplyr` package.

Update of the mining function.

```
library(dplyr)

mine <- function(previous_block, transactions, difficulty = 3,
                 block_size = 3, miner = "Z", genesis = FALSE){
  # Filter transactions to add.
  trans_pending = arrange(transactions, -fee)
  if (nrow(trans_pending) < block_size) {
    trans_to_add = trans_pending
    trans_pending = tibble()
  } else {
    trans_to_add =  filter(trans_pending, row_number() <= block_size)
    trans_pending = filter(trans_pending, row_number() > block_size)
  }

  if (genesis) {
    # Define genesis block.
    new_block <-  list(number = 1,
                       timestamp = Sys.time(),
                       data = trans_to_add,
                       parent_hash = "0")
  } else {
    # Create new block.
    new_block <- list(number = previous_block$number + 1,
                      timestamp = Sys.time(),
                      data = trans_to_add,
                      parent_hash = previous_block$hash)
  }

  # Add nonce with PoW.
  new_block$nonce <- proof_of_work(new_block, difficulty)$nonce
  # Add hash.
  new_block$hash <- digest(new_block, "sha256")
  # Add reward transaction.
  trans_pending = rbind(trans_pending,
                        data.frame(sender = NA, receiver = miner,
```

```
                                    value = sum(new_block$data$fee),
                                    fee = 0.01))
  return(list(block = new_block, transactions = trans_pending))
}
```

Blockchain with transactions.

```
blockchained = function(transactions, difficulty, block_size, nblocks) {
  # define genesis block
  mined = mine(NULL, transactions, difficulty, block_size, miner = "Z",
               genesis = TRUE)
  block_genesis <- mined$block
  pending = mined$transactions
  # first block is the genesis block
  blockchain <- list(block_genesis)

  if (nblocks >= 2) {
    # add blocks to the chain
    for (i in 2:nblocks){
      mined <- mine(blockchain[[i-1]], pending, difficulty, block_size,
                    miner = "Z")
      blockchain[[i]] <- mined$block
      pending = mined$transactions
    }
  }

  return(blockchain)
}
```

Build a new blockchain with transactions.

```
trans <- blockchained(transactions, difficulty = 3, block_size = 5,
                      nblocks = 3)
trans[[1]]
```

```
## $number
## [1] 1
##
```

```
## $timestamp
## [1] "2021-07-05 22:48:57.048154 CDT"
##
## $data
## # A tibble: 5 x 4
##   sender receiver value   fee
##   <chr>  <chr>    <dbl> <dbl>
## 1 L      C            7  0.8
## 2 Z      T          100  0.63
## 3 E      U           40  0.57
## 4 O      N           56  0.5
## 5 Y      X           45  0.45
##
## $parent_hash
## [1] "0"
##
## $nonce
## [1] 671
##
## $hash
## [1] "0007c3908c65e188dfc1cb5e26ab37068605790b2b0a9a4da1d5fed0a05b16e8"
```

In the first block above, we have the first half of the transactions sorted by fee.

```
trans[[2]]
```

```
## $number
## [1] 2
##
## $timestamp
## [1] "2021-07-05 22:48:57.08306 CDT"
##
## $data
## # A tibble: 5 x 4
##   sender receiver value   fee
##   <chr>  <chr>    <dbl> <dbl>
## 1 V      E            2  0.35
## 2 S      K           33  0.34
```

```
## 3 Q      D           73  0.23
## 4 X      Q           15  0.15
## 5 J      W           68  0.1
##
## $parent_hash
## [1] "0007c3908c65e188dfc1cb5e26ab37068605790b2b0a9a4da1d5fed0a05b16e8"
##
## $nonce
## [1] 5327
##
## $hash
## [1] "0001ea60075625f55d6b8e96302d3443179229ad3a727b61a7f7117841730933"
```

In the second block above, we have the second half of the transactions sorted by fee.

```
trans[[3]]
```

```
## $number
## [1] 3
##
## $timestamp
## [1] "2021-07-05 22:48:57.283524 CDT"
##
## $data
## # A tibble: 2 x 4
##   sender receiver value   fee
##   <chr>  <chr>    <dbl> <dbl>
## 1 <NA>   Z         2.95  0.01
## 2 <NA>   Z         1.17  0.01
##
## $parent_hash
## [1] "0001ea60075625f55d6b8e96302d3443179229ad3a727b61a7f7117841730933"
##
## $nonce
## [1] 5504
##
## $hash
## [1] "000508a011b39fe653aff2b1bd38027d841d8f5e2285a10610ce9cbd2a399b25"
```

In the third block above, the miner of the block Z solves the leading zeros problem and gets the fees of all blocked transactions plus a fixed mining reward (and this transaction is included in the following block).

7.6 Digital signature.

How can we be sure that transactions are authentic? Blockchain uses asymmetric cryptography to implement digital signatures of transactions. Each transaction is signed by the sender with her private key and anyone can verify the authenticity of the signature (and of the transaction) using the sender's public key.

Public-key cryptography, or asymmetric cryptography, is any cryptographic system that uses pairs of keys: public keys which may be disseminated widely, and private keys which are known only to the owner. This accomplishes two functions: authentication, where the public key verifies that a holder of the paired private key sent the message, and encryption, where only the paired private key holder can decrypt the message encrypted with the public key. The strength of a public key cryptography system relies on the computational effort required to find the private key from its paired public key.

In a public key encryption system, any person can encrypt a message using the receiver's public key. That encrypted message can only be decrypted with the receiver's private key. In a public key signature system, a person can combine a message with a private key to create a short digital signature on the message. Anyone with the corresponding public key can combine the signed message and the known public key to verify whether the signature on the message was valid, i.e. made by the owner of the corresponding private key. Changing the message, even replacing a single letter, will cause verification to fail.

To speed up the process of transmission, instead of applying the sender's digital signature to the (possibly large) message, the sender can rather hash the message using a cryptographic hash function and then digitally sign the generated hash value. The sender would then sign the newly generated hash value with their private key and encrypt the original message with the receiver's public key. The transmission would then take place securely and with confidentiality and non-repudiation still intact. The receiver would then verify the signature with sender's public key and decrypt the message with their private key.

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In RSA, the asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers.

Next is an example of encryption and digital signature using package `openssl`. First, the encryption example.

```
library(openssl)
# Encryption.
# Generate a RSA (Rivest-Shamir-Adleman) private and a public key (pubkey).
key <- rsa_keygen(512)
pubkey <- key$pubkey
key
```

```
## [512-bit rsa private key]
## md5: e2ddd1a4f803e5fbd4411f9fd73856b4
```

```
pubkey
```

```
## [512-bit rsa public key]
## md5: e2ddd1a4f803e5fbd4411f9fd73856b4
```

Looks the same, but the the private key has some more information inside the key object.

```
# Message. charToRaw converts a length-one character string to raw bytes.
msg <- charToRaw("Blockchain is terrific!")
msg
```

```
##  [1] 42 6c 6f 63 6b 63 68 61 69 6e 20 69 73 20 74 65 72 72 69 66 69 63 21
```

The message *Blockchain is terrific!* is now transformed into data to encrypt.

```
# cipher the message with public key
ciphermsg <- rsa_encrypt(msg, pubkey)
ciphermsg
```

```
##  [1] 26 44 45 57 37 80 d0 57 25 e1 0a 6f 74 fd e5 a7 9c c9 41 f9 a2 46 79 e3 b9
## [26] 66 48 38 b0 bc 79 6d 76 1a 00 20 09 47 ec de b2 f5 7f 09 5d 25 89 c2 40 f0
## [51] 9f ad 64 29 b7 17 07 4e e8 03 df ed cb 79
```

This is how it looks the encrypted message *Blockchain is terrific!*

```
# Decrypt the message with private key.
rawToChar(rsa_decrypt(ciphermsg, key))
```

```
## [1] "Blockchain is terrific!"
```

And this is how we can decrypt the message with the private key. Now, the signature example where the message is a transaction.

```
# Signature.
# Generate a private key (key) and a public key (pubkey)
key <- rsa_keygen()
pubkey <- key$pubkey
# build a transaction
trans = list(sender = "A", receiver = "B", amount = "100")
trans
```

```
## $sender
## [1] "A"
##
## $receiver
## [1] "B"
##
## $amount
## [1] "100"
```

This is the original transaction.

```
# Serialize data.
data <- serialize(trans, NULL)
data
```

```
##   [1] 58 0a 00 00 00 03 00 04 01 00 00 03 05 00 00 00 00 06 43 50 31 32 35 32 00
##  [26] 00 02 13 00 00 00 03 00 00 00 10 00 00 00 01 00 04 00 09 00 00 00 01 41 00
##  [51] 00 00 10 00 00 00 01 00 04 00 09 00 00 00 01 42 00 00 00 10 00 00 00 01 00
##  [76] 04 00 09 00 00 00 03 31 30 30 00 00 04 02 00 00 00 01 00 04 00 09 00 00 00
## [101] 05 6e 61 6d 65 73 00 00 00 10 00 00 00 03 00 04 00 09 00 00 00 06 73 65 6e
## [126] 64 65 72 00 04 00 09 00 00 00 08 72 65 63 65 69 76 65 72 00 04 00 09 00 00
## [151] 00 06 61 6d 6f 75 6e 74 00 00 00 fe
```

In computing, serialization is the process of translating a data structure or object state into a format that can be stored or transmitted and reconstructed later. Here, the transaction is translated into a format that can be reconstructed later. At the moment, this object called `data` has not been signed yet by the sender. The next code chunk sign the `data` object by the sender.

```
# Sign (a hash of) the transaction with private key.
sig <- signature_create(data, sha256, key = key)
sig
```

```
##   [1] 0a ef eb 0b a5 f0 be d3 a7 55 09 da 20 ef 5f ca 41 52 e5 3b 84 02 0a 3b c7
##  [26] be 53 f6 96 e6 58 00 92 82 0b fc d9 be d4 15 dd 97 ce bc 16 1f 09 2c 94 5a
##  [51] 78 fc 5b 71 93 b0 33 54 ef 1c 22 c5 be 95 b9 d9 4e 42 35 5b 51 ec 0e 69 e6
##  [76] 67 9c 9d 62 69 75 e0 f5 5e fd 22 a7 5a 60 d9 dc 32 d5 a6 97 1a 7e 05 f6 58
## [101] 01 11 8e 76 db 83 56 e1 e3 0c d3 db 79 df 03 f5 33 0b 73 00 63 f3 f7 92 86
## [126] 38 03 cb de 4e 4f f2 f8 f7 c8 28 9a 99 ab 46 29 da 4f 1e 8b e0 18 79 a6 1a
## [151] 3b fd 9e 88 30 ca 5a 66 90 9d 38 21 11 b6 d8 b5 ed eb cf d2 a3 cb 64 7f 3f
## [176] eb d0 c2 03 fc 4f 36 be fb 81 59 8d c5 ec 8f c2 e3 46 e8 64 fa 23 e8 85 95
## [201] e6 38 83 8c 41 d6 96 2c 08 2f ff 8a 90 fb 74 c0 0f 1c 13 7a ce 46 aa db 3c
## [226] 4f ee 3c 38 5f 04 85 04 4f a0 f4 98 6f 09 26 49 b1 1b c5 90 37 e3 cc 94 8f
## [251] ed a8 d6 02 c1 50
```

The new object `sig` has not only the translated transaction, it has been also signed by the sender. With the function `signature_verify`, anyone could verify that the sender of the transaction is the owner of the `pubkey`.

```
# verify the message with public key
signature_verify(data, sig, sha256, pubkey = pubkey)
```

```
## [1] TRUE
```

An example of encryption and digital signature together. We repeat a few steps on purpose.

```
# Signature and encryption
# Generate a private key (key) and a public key (pubkey)
key <- rsa_keygen()
pubkey <- key$pubkey
# Build a transaction
trans = list(sender = "A", receiver = "B", value = "100")
trans
```

```
## $sender
## [1] "A"
##
## $receiver
```

```
## [1] "B"
##
## $value
## [1] "100"
```

Again, we have the transaction.

```
# Serialize data
data <- serialize(trans, NULL)
data
```

```
##   [1] 58 0a 00 00 00 03 00 04 01 00 00 03 05 00 00 00 00 06 43 50 31 32 35 32 00
##  [26] 00 02 13 00 00 00 03 00 00 00 10 00 00 00 01 00 04 00 09 00 00 00 01 41 00
##  [51] 00 00 10 00 00 00 01 00 04 00 09 00 00 00 01 42 00 00 00 10 00 00 00 01 00
##  [76] 04 00 09 00 00 00 03 31 30 30 00 00 04 02 00 00 00 01 00 04 00 09 00 00 00
## [101] 05 6e 61 6d 65 73 00 00 00 10 00 00 00 03 00 04 00 09 00 00 00 06 73 65 6e
## [126] 64 65 72 00 04 00 09 00 00 00 08 72 65 63 65 69 76 65 72 00 04 00 09 00 00
## [151] 00 05 76 61 6c 75 65 00 00 00 fe
```

We serialize the transaction. This is, it is translated into a format that can be reconstructed later.

```
# Sign (a hash of) the transaction with private key
sig <- signature_create(data, sha256, key = key)
sig
```

```
##   [1] 19 13 3b eb 2f 9b 4d 07 f9 49 d8 65 66 09 f2 b2 07 ea ac 09 fc 31 2c 48 9d
##  [26] 2f 74 22 4c 85 d6 11 a7 9b 35 28 d4 e3 0d 6e 57 0b 90 37 89 f7 cc e1 92 2e
##  [51] 50 f1 1a 12 3e 23 e8 a2 53 d6 ba a8 78 19 8a 8f 41 40 f5 6d d5 85 d4 95 02
##  [76] 7a 33 e5 69 00 fb 01 f2 ae 60 77 d0 15 a3 16 d0 47 57 b7 e7 9b 3d 89 46 d5
## [101] b0 c8 cf 5a 6e b2 86 b7 21 3a b0 55 46 91 1e 6d 7f e2 ad ad 7b e1 04 09 89
## [126] 37 5c e2 1f 0c a3 b6 b0 47 54 2d de 5e 97 44 18 6a 50 dc 37 6d f4 6c 97 8c
## [151] 38 d7 12 57 9b 90 f8 94 64 b8 49 92 20 5f 05 c8 48 c3 de e0 e5 e6 8f 29 6d
## [176] 17 f9 23 fa c0 b6 c4 da 77 9d e0 be 2e 5a 5b 19 f6 49 df 19 c8 49 b2 c9 b9
## [201] 6c b3 96 61 38 a0 a5 0d c3 fc 72 02 a9 92 72 5d bf 58 a4 ee c5 28 c3 8e d9
## [226] c6 1d a4 a1 cf db b8 cd e1 40 f5 63 ff 00 29 02 5b 89 f6 25 4c 4a 7f 97 41
## [251] 02 ae 05 08 9e d7
```

The transaction is translated and signed by the sender.

```
# Verify the message with public key
signature_verify(data, sig, sha256, pubkey = pubkey)
```

```
## [1] TRUE
```

Anyone could know that the transaction belongs to the sender with the `pubkey`. Now, we encrypt the translated and signed transaction.

```
# Cipher the transaction with public key
ciphermsg <- rsa_encrypt(data, pubkey)
ciphermsg
```

```
##   [1] 8b 2b 81 34 b3 0c c8 38 35 8d 46 63 8a 4a 1b 3e 35 64 20 b2 2a af 26 2c 77
##  [26] 50 67 6f 37 23 bc a4 f6 f6 96 6e 31 43 f4 5b 44 14 91 31 c9 a6 cf 04 c6 88
##  [51] 4f ef fa 6f 67 3e 7e f4 8b ef 87 16 11 cb 33 7b 8f b9 c2 56 34 20 70 c0 a9
##  [76] 06 8d 46 a9 93 02 d2 b7 80 5d 5d a7 8e c0 26 6f 29 7a 2c 63 e9 90 b7 d1 0c
## [101] e0 c1 b3 d4 5e a1 65 de df 51 17 96 9f 47 b3 b3 b8 a7 0c 8f f9 04 1f 80 2d
## [126] 64 5c f8 f9 00 0a 95 3b c2 24 4f 59 5d 41 af 82 81 3b fd 6b cf 8c 7c 8e 25
## [151] 5b 52 e3 cd 34 16 2f ae 0e 4f a0 4c f3 51 87 59 04 9f 1b fd 46 e0 1c b3 31
## [176] 23 88 2e 41 f4 e0 ac f3 56 37 42 66 20 57 3c 16 28 b4 e4 32 7d cd 80 53 81
## [201] a9 d0 57 44 ac 2f e6 19 ee 00 71 65 82 c2 be bc dd 75 33 fe ec 52 b7 15 37
## [226] 8d d1 30 07 98 a2 70 59 83 84 05 e6 2b de bf b9 a4 9e 63 b3 7e d7 a4 83 f1
## [251] e6 74 b2 c0 f4 36
```

The transaction is translated, signed by the sender and now encrypted with the sender's `pubkey`. To recover the original transaction we would need to decrypt the serialized message with the private key `key`, and then unserialize the result.

```
# Decrypt and unserialize the transation with private key.
unserialize(rsa_decrypt(ciphermsg, key))
```

```
## $sender
## [1] "A"
##
## $receiver
## [1] "B"
##
## $value
## [1] "100"
```

The transaction details are back.

7.7 Network.

Finally, the blockchain ledger is distributed over a peer-to-peer network. The steps to run the network are as follows:

- new transactions are broadcast to all nodes;
- each node collects new transactions into a block;
- each node works on finding a difficult proof-of-work for its block;
- when a node finds a proof-of-work, it broadcasts the block to all nodes;
- nodes accept the block only if all transactions in it are valid and not already spent;
- nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. The incentive of rewards may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules (generate new coins), such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

7.8 Privacy.

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the tape, is made public, but without telling who the parties were.

8 Conclusion.

Finance is not about how to make money; it is about much more than that. Finance is about how to find and use resources, and how to assign them into projects. This is not an

easy task as resources are limited and projects are risky. Moreover, sometimes the financial markets do not work well: bad projects are financed and good projects are not. However, if we rely on finance theory, quantitative models, and data analysis, we increase the chances to make good and informed decisions. Making money is a necessary step that allows us to pursue superior objectives like making firms growth, create better jobs, stimulate innovation, economic growth, and hopefully improve the standard of living of the population.

We have made some progress at implementing financial and economic models to achieve these superior objectives but we cannot say we have succeeded. Poverty levels in some countries are high, and income inequality leads not only to economic but also social problems. Fortunately, there is an increasing and genuine interest in learning finance principles by good people around the world.

This document took 323.6 seconds to compile in Rmarkdown.

References.

Donald Ervin Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

Robert C Merton. Theory of rational option pricing. *The Bell Journal of economics and management science*, pages 141–183, 1973.

OECD. *OECD/INFE 2020 International Survey of Adult Financial Literacy*. 2020. URL https://www.oecd.org/financial/education/oecd-infe-2020-international-survey-of-adult-financial-literacy.pdf.

Myron Scholes. The pricing of options and corporate. *The journal of political economy*, 81 (3):637–654, 1973.