R Extra Credit Assignment

BEPP 2800: Applied Data Analysis

The Wharton School, University of Pennsylvania

For this assignment, you will work in teams of two to create two stock portfolios. Each portfolio will start with $100,000 to invest. For the first portfolio, you will select stocks based on your general intuition, choosing those you believe will perform well and yield the highest return on investment over a two-month period. For the second portfolio, you will select stocks based on indicators derived from stock return data, including the Sharpe ratio, Jensen's Alpha, and correlations between stocks (I will explain how to calculate these in this document).

The goal of this assignment is to determine whether using financial indicators leads to better investment outcomes than relying on intuition alone. As you might already be thinking, it is possible that one portfolio simply performs better by chance (i.e., the difference in outcomes is due to natural variation rather than a meaningful advantage). Since multiple teams will be completing this exercise, we can perform a statistical hypothesis test to assess whether the observed differences are significant or random.

This project will (i) introduce you to basic portfolio theory, (ii) demonstrate the importance of hypothesis testing, and (iii) provide a crash course in R.

Just for fun, the team with the highest performing portfolio (either one) will win a prize. But do not get too excited, I am not a millionaire and the prize will be small.

We will use `https://www.investopedia.com/simulator/portfolio` to create the portfolios. To start, one of you create an account and together pick some stocks to buy.

In the following sections, I will give you instructions on how to use R and make the indicators. After following the instructions, the other team member will make an account and together you will pick stocks for the second portfolio.

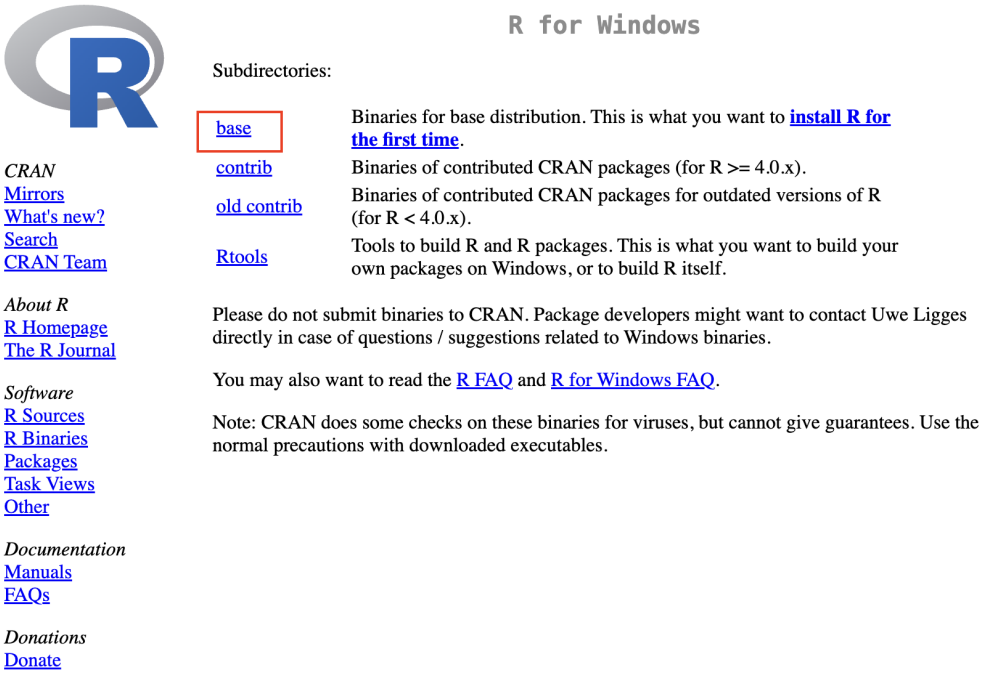
Installing R and RStudio:

R is a programming language and software environment designed for statistical computing, data analysis, and visualization. It is widely used in academia, finance, and data science because it has powerful tools for working with data.

RStudio is an integrated development environment (IDE) for R. It makes writing and running R code much easier by providing a user-friendly interface, with features like a script editor, console, environment viewer, and built-in plotting windows—all in one place.

Link for installing R: `https://cran.r-project.org/`

For Window, choose Windows and select the Base option to download:



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs

Donations
Donate

R for Windows

Subdirectories:

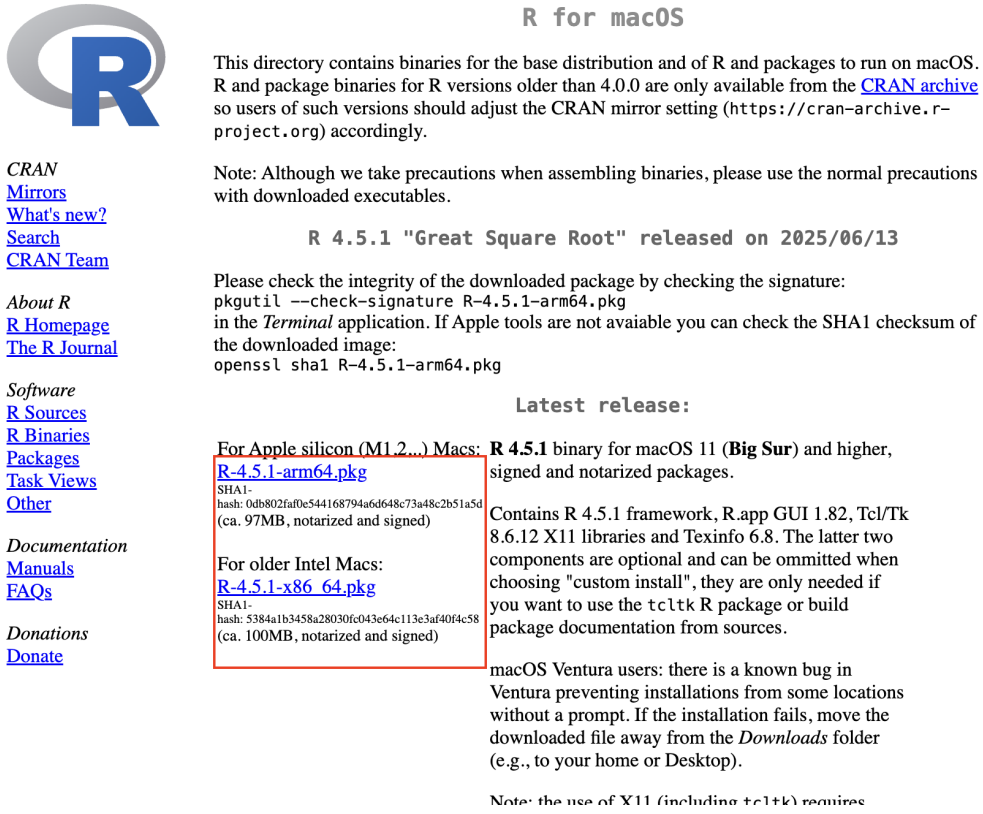
base — Binaries for base distribution. This is what you want to **install R for the first time**.
contrib — Binaries of contributed CRAN packages (for R >= 4.0.x).
old contrib — Binaries of contributed CRAN packages for outdated versions of R (for R < 4.0.x).
Rtools — Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

For Mac, select Mac the the appropriate option to download:



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs

Donations
Donate

R for macOS

This directory contains binaries for the base distribution and of R and packages to run on macOS. R and package binaries for R versions older than 4.0.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting (`https://cran-archive.r-project.org`) accordingly.

Note: Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

R 4.5.1 "Great Square Root" released on 2025/06/13

Please check the integrity of the downloaded package by checking the signature:
`pkgutil --check-signature R-4.5.1-arm64.pkg`
in the *Terminal* application. If Apple tools are not avaiable you can check the SHA1 checksum of the downloaded image:
`openssl sha1 R-4.5.1-arm64.pkg`

Latest release:

For Apple silicon (M1,2...) Macs:
R-4.5.1-arm64.pkg
SHA1-hash: 0db802faf0e544168794a6d648c73a48c2b51a5d
(ca. 97MB, notarized and signed)

For older Intel Macs:
R-4.5.1-x86_64.pkg
SHA1-hash: 5384a1b3458a28030fc043e64c113e3af40f4c58
(ca. 100MB, notarized and signed)

R 4.5.1 binary for macOS 11 (**Big Sur**) and higher, signed and notarized packages.

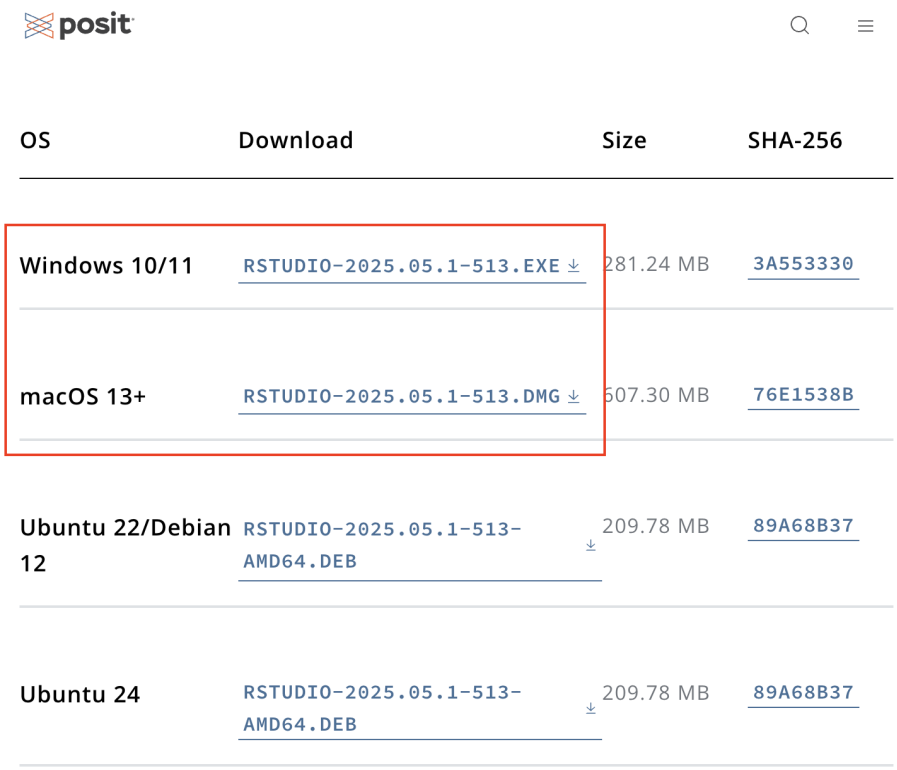
Contains R 4.5.1 framework, R.app GUI 1.82, Tcl/Tk 8.6.12 X11 libraries and Texinfo 6.8. The latter two components are optional and can be ommitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

macOS Ventura users: there is a known bug in Ventura preventing installations from some locations without a prompt. If the installation fails, move the downloaded file away from the *Downloads* folder (e.g., to your home or Desktop).

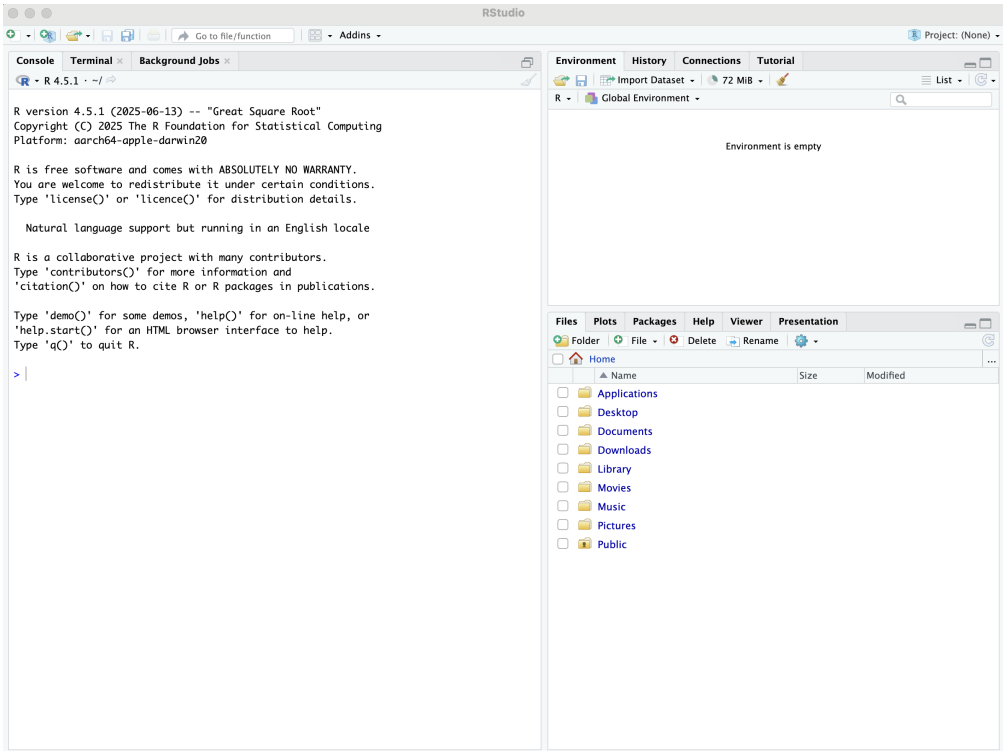
Note: the use of X11 (including `tcltk`) requires

Once you have downloaded R, you can download RStudio with the following link: `https:`

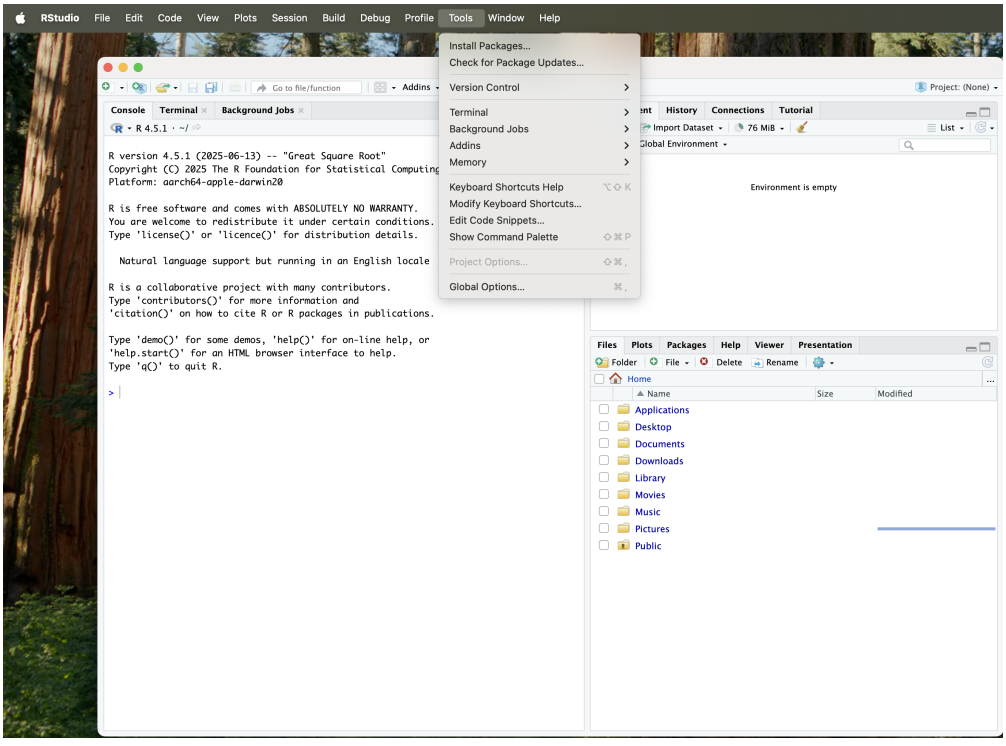
//www.rstudio.com/



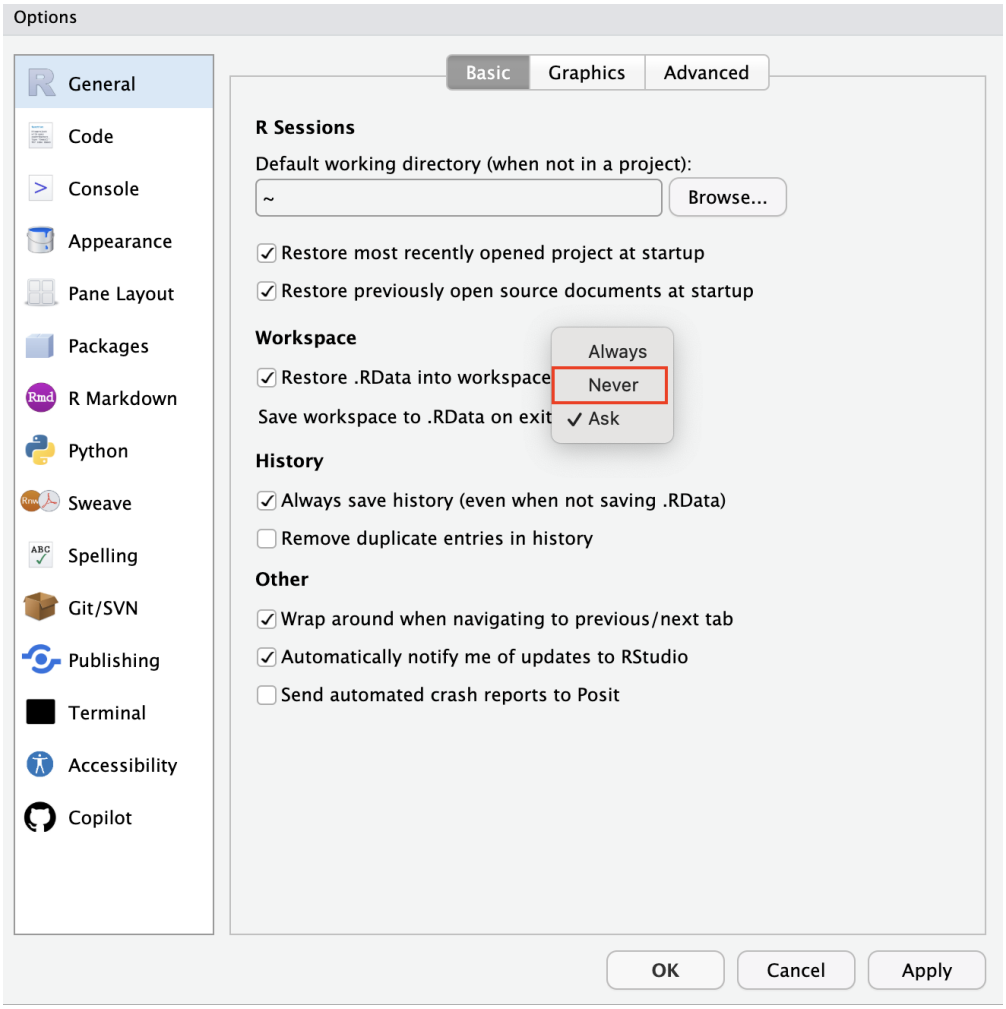
When you open RStudio, it should look like:



Let's change an option. Go to Tools>Global Options:



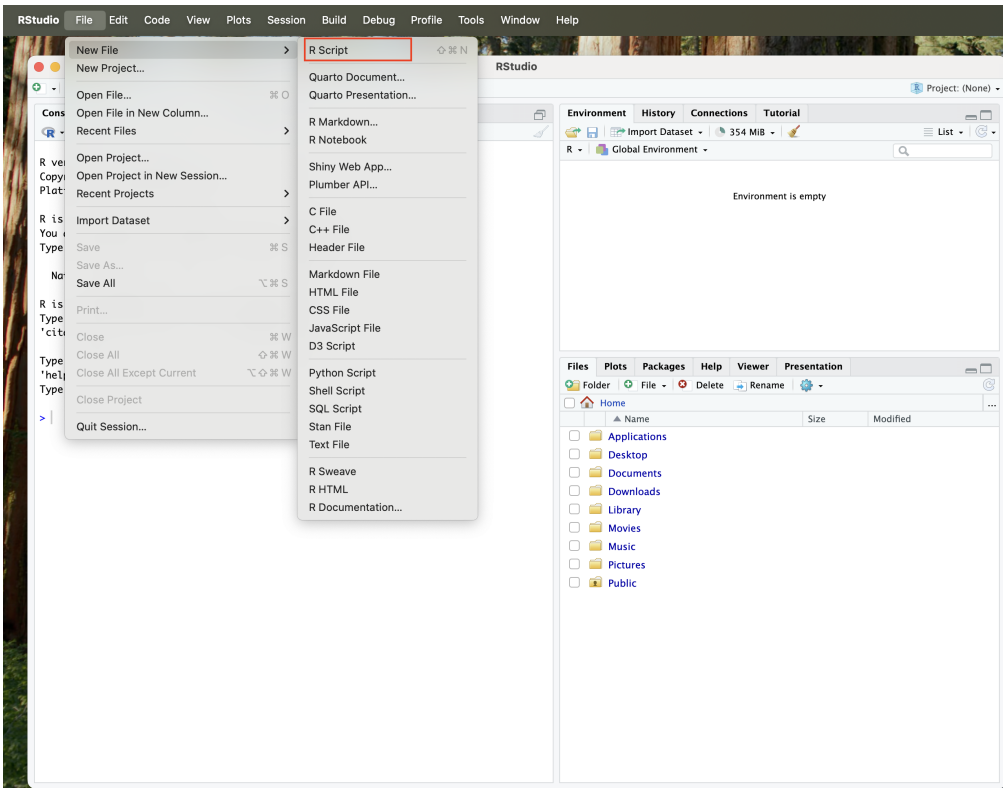
Change Save workplace to .RData on exit to Never:



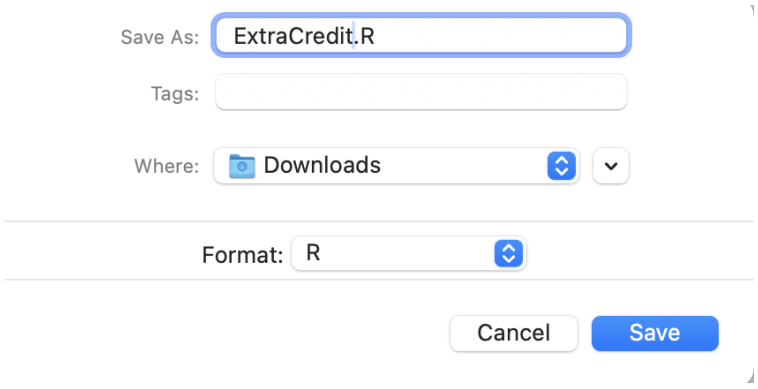
Make sure to hit Apply.

R Crash Course:

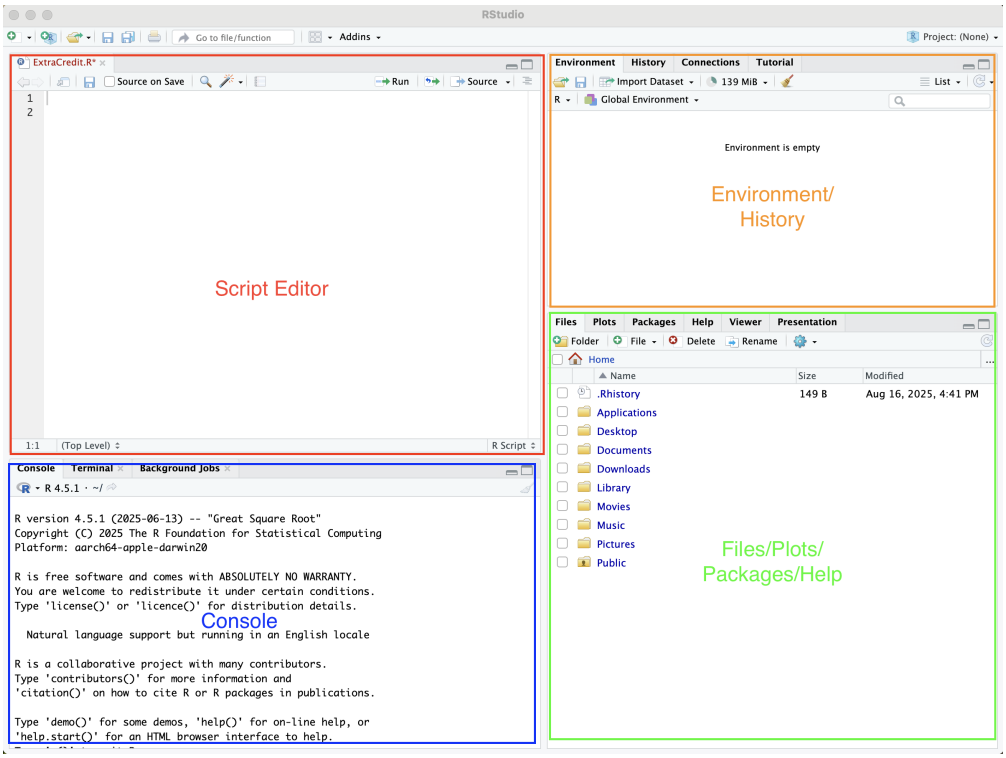
An *R script* is a plain text file that contains a sequence of R commands, making it useful for writing, saving, and re-running code. Start by creating a script with File>New File>R Script:



Save the file with `Ctrl + s` (Windows) or `Cmd + s` (Mac) and name your file:



The RStudio interface is divided into four main panes: script editor (write and edit R code), console (run code and view output), environment/history (see saved variables and command history), and files/plots/packages/help (manage files, view plots, load packages, and access help)which together make it easy to write code, run it, view results, and manage your workspace.

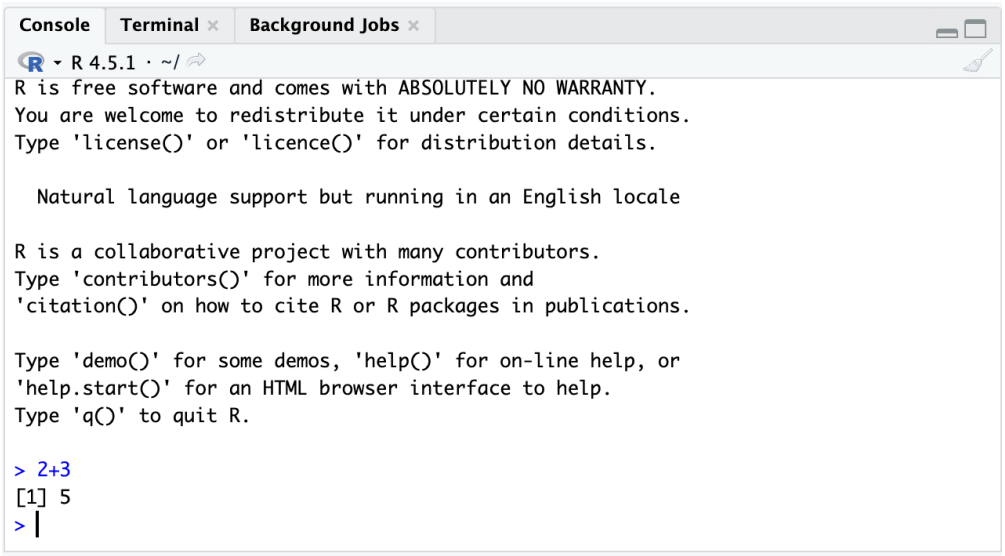


Basic Arithmetic

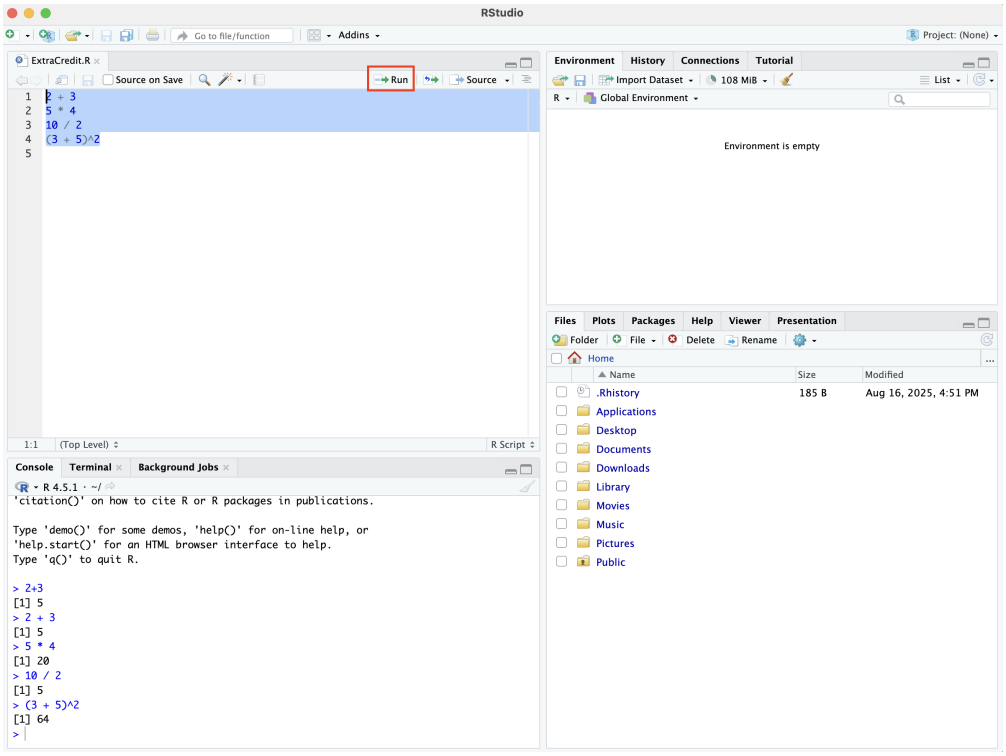
R can be used like a calculator:

```
2 + 3
5 * 4
10 / 2
(3 + 5)^2
```

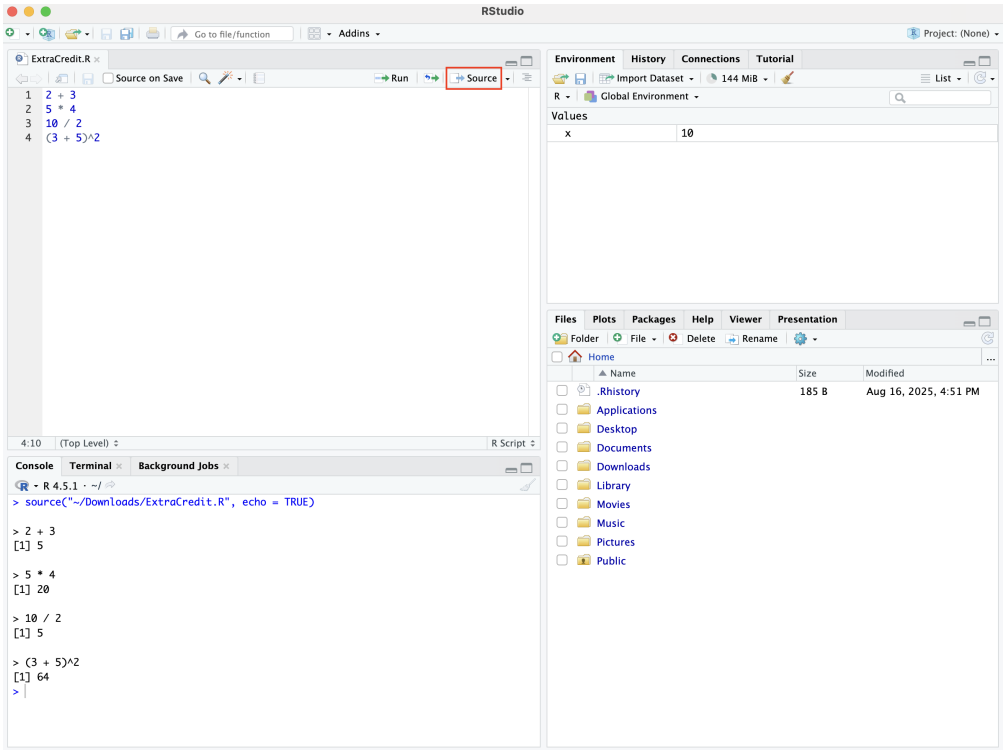
You can type the command directly into the console and press `enter`:



Or, you can highlight the code in your script and press run:



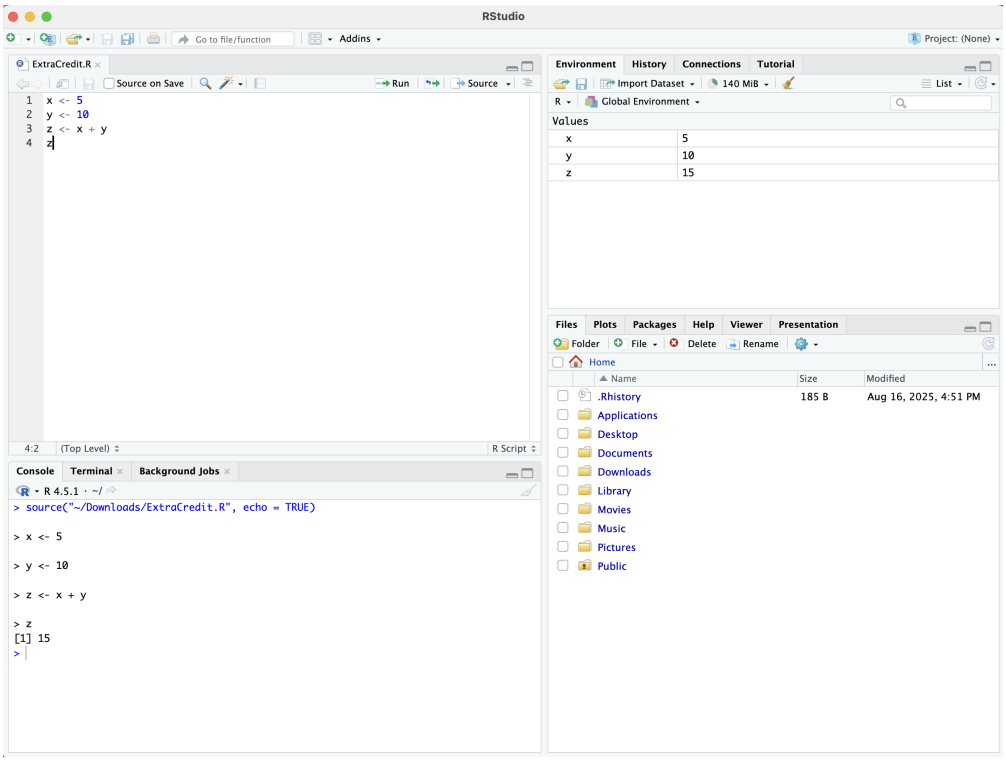
The Source button runs the entire R script:



Variables and Assignment

You can store values in variables using `<-`:

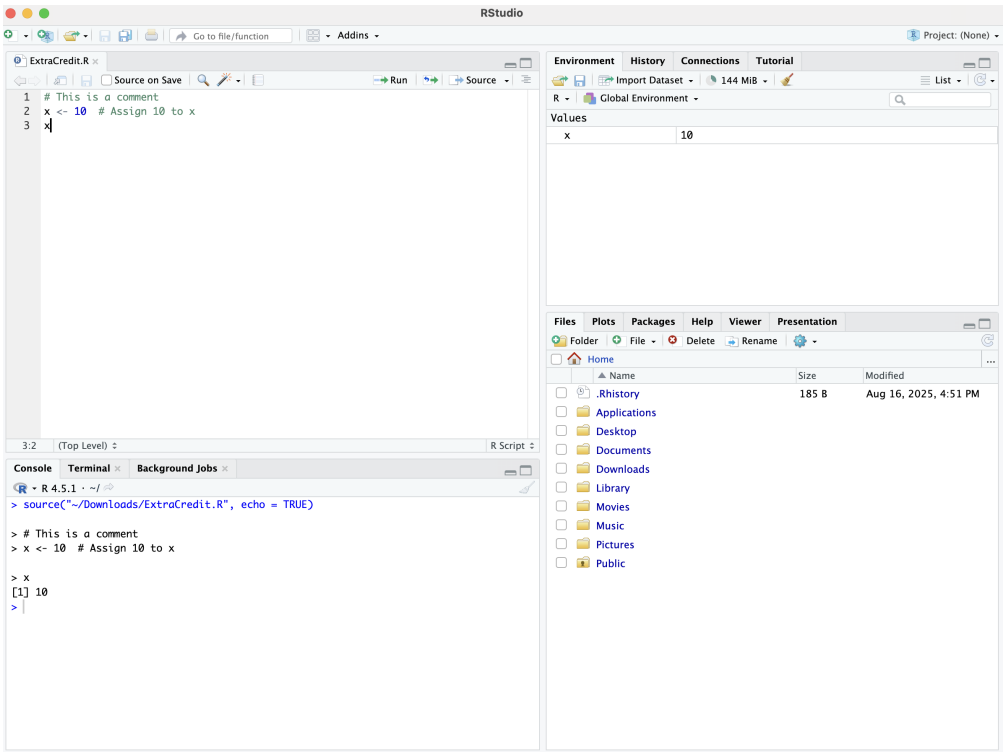
```r
x <- 5
y <- 10
z <- x + y
z
```



Comments

Comments are lines of text that R ignores when running your code. Use them to explain your code:

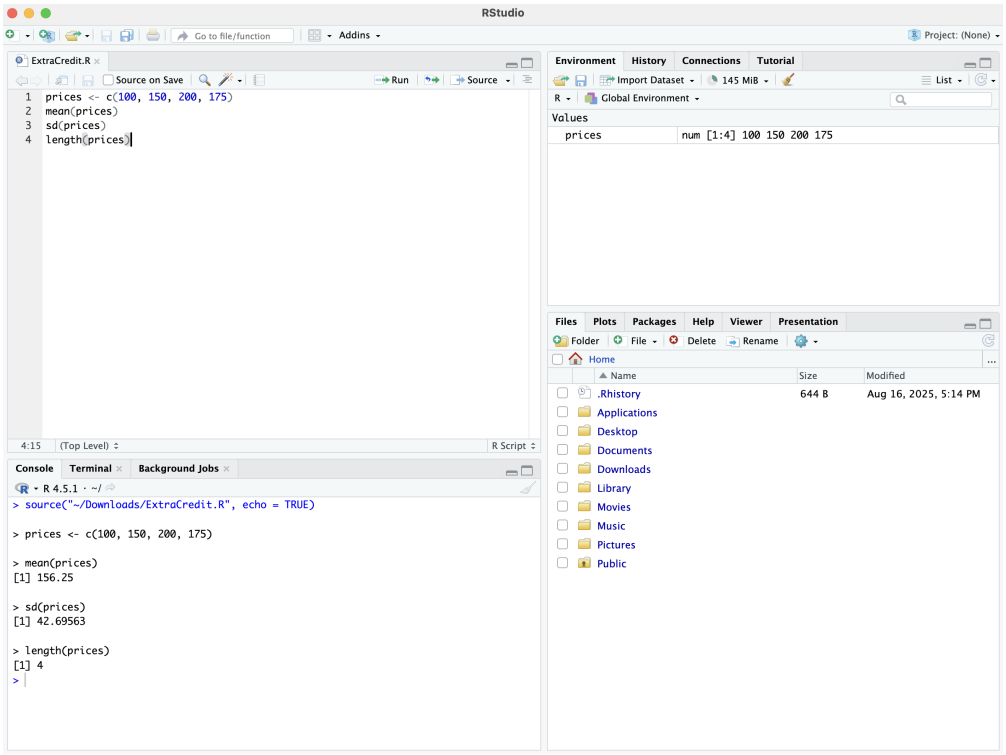
```r
# This is a comment
x <- 10  # Assign 10 to x
```



Vectors

Vectors are like lists of numbers:

```
prices <- c(100, 150, 200, 175)
mean(prices)
sd(prices)
length(prices)
```



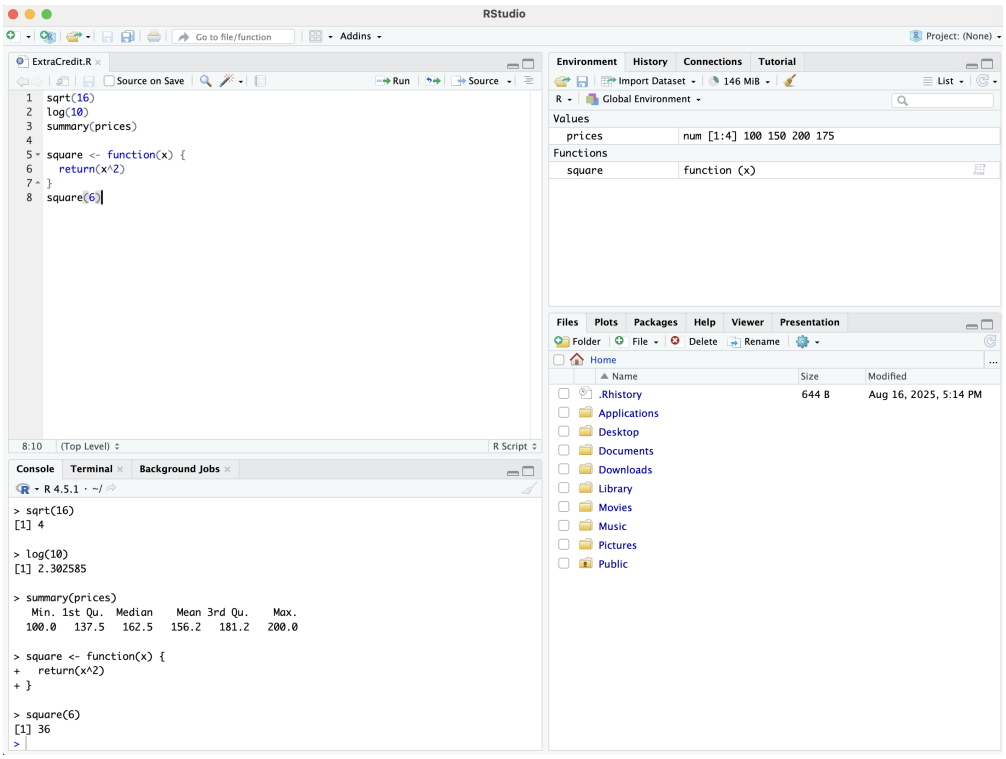
Functions

R has many built-in functions:

```
sqrt(16)
log(10)
summary(prices)
```

You can also define your own functions:

```
square <- function(x) {
  return(x^2)
}
square(6)
```

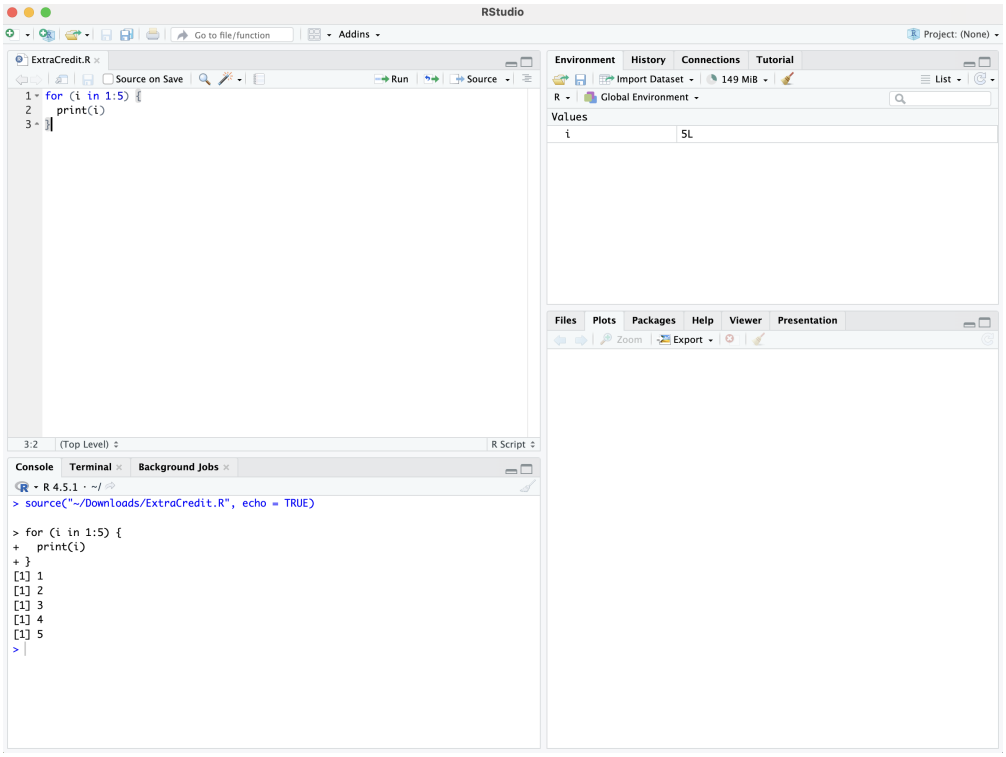


Loops

Loops allow you to repeat actions multiple times. The most common loop in R is the `for` loop, which lets you run the same code for each item in a sequence.

Example: Print numbers 1 to 5

```
for (i in 1:5) {
  print(i)
}
```

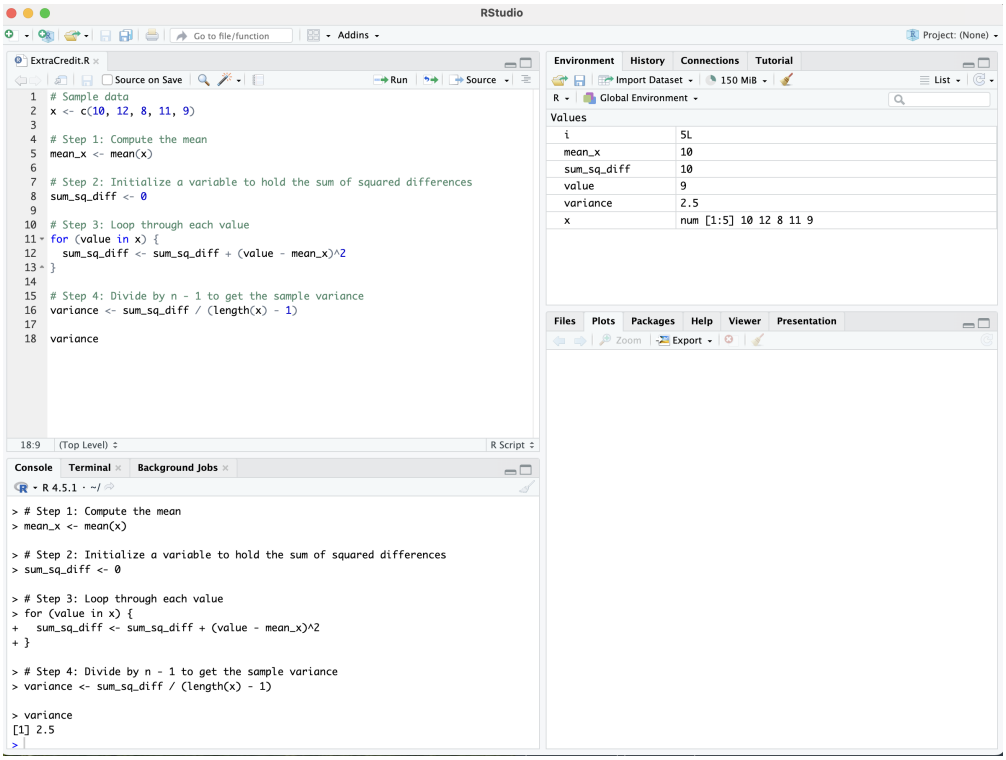


Example: Calculate the Sample Variance

This example walks through the basic idea of calculating the *sample variance*:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

```r
# Sample data
x <- c(10, 12, 8, 11, 9)

# Step 1: Compute the mean
mean_x <- mean(x)

# Step 2: Initialize a variable to hold the sum of squared differences
sum_sq_diff <- 0

# Step 3: Loop through each value
for (value in x) {
  sum_sq_diff <- sum_sq_diff + (value - mean_x)^2
}

# Step 4: Divide by n - 1 to get the sample variance
variance <- sum_sq_diff / (length(x) - 1)

variance
```



While loops are useful, in R it's often better to use built-in functions (like `apply()`, `lapply()`, or vectorized operations) for better performance.

Loading Packages

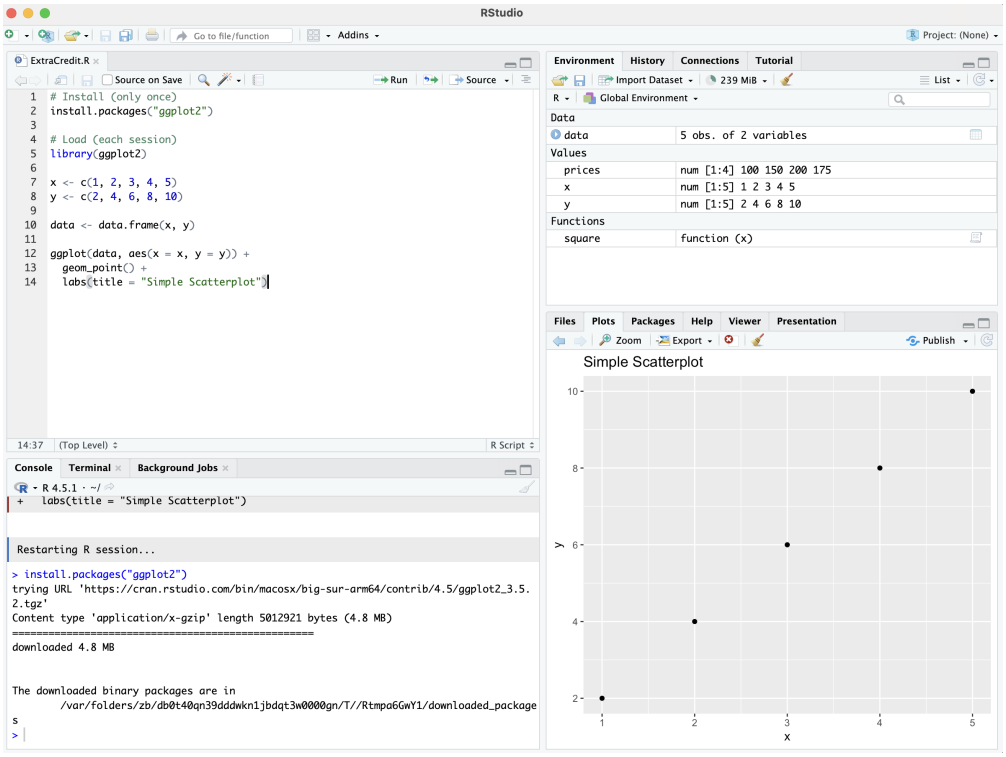
Packages add extra functionality. You only need to install a package once, but you must load it every time you use it.

```
# Install (only once)
install.packages("ggplot2")

# Load (each session)
library(ggplot2)
```

Plotting

Here's how to make a basic scatterplot using `ggplot2`:

```
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)

data <- data.frame(x, y)

ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Simple Scatterplot")
```



`ggplot2` is a plotting system based on the *grammar of graphics*, which means you build plots by adding *layers* to define what to show, how to show it, and how to style it.

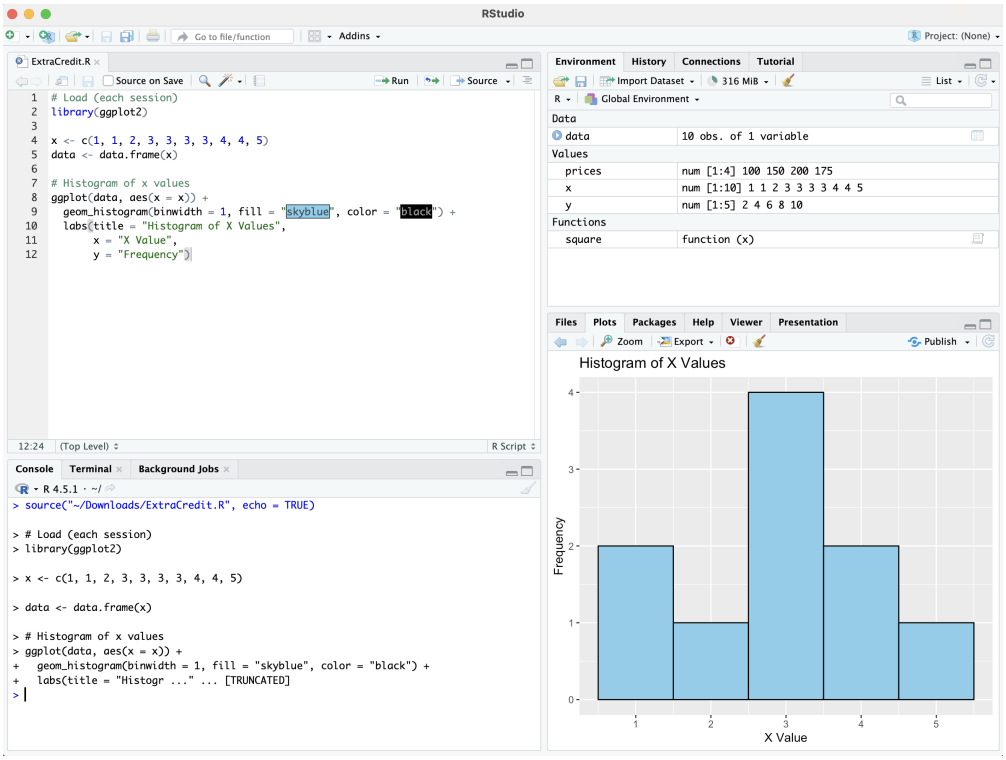
At its core, a ggplot has:

- A dataset (`data`)
- An aesthetic mapping (`aes()`) that connects data to visual features (like x and y axes)
- One or more *geoms* that specify what type of plot (points, lines, bars, etc.)

You start with a base plot using `ggplot()`, then add layers using `+`.

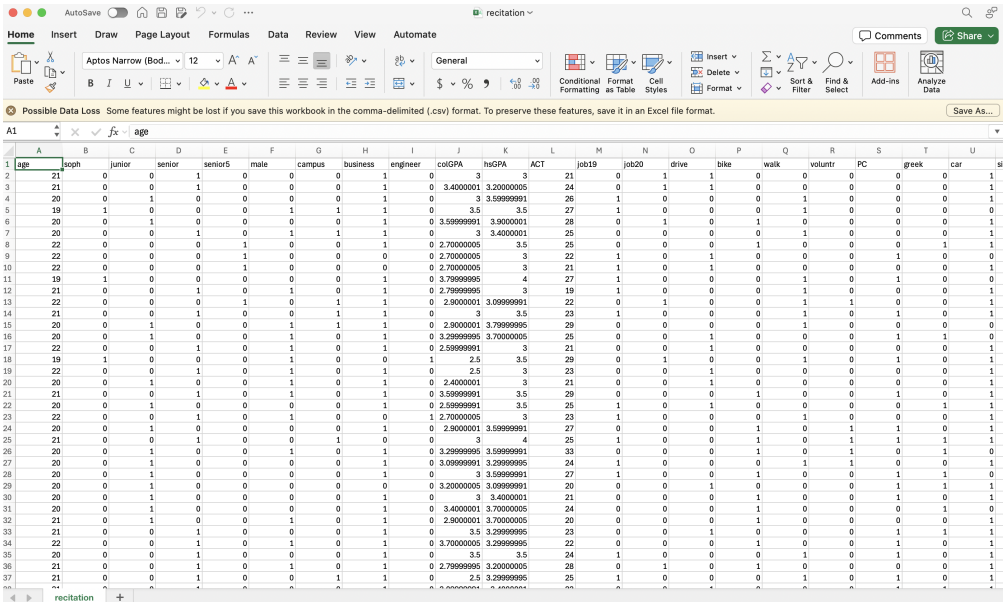
Here is another example of a ggplot graph as a histogram:

```
x <- c(1, 1, 2, 3, 3, 3, 3, 4, 4, 5)
data <- data.frame(x)

# Histogram of x values
ggplot(data, aes(x = x)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
  labs(title = "Histogram of X Values",
       x = "X Value",
       y = "Frequency")
```



Loading and Inspecting CSV Data

Suppose I have data in a CSV file:



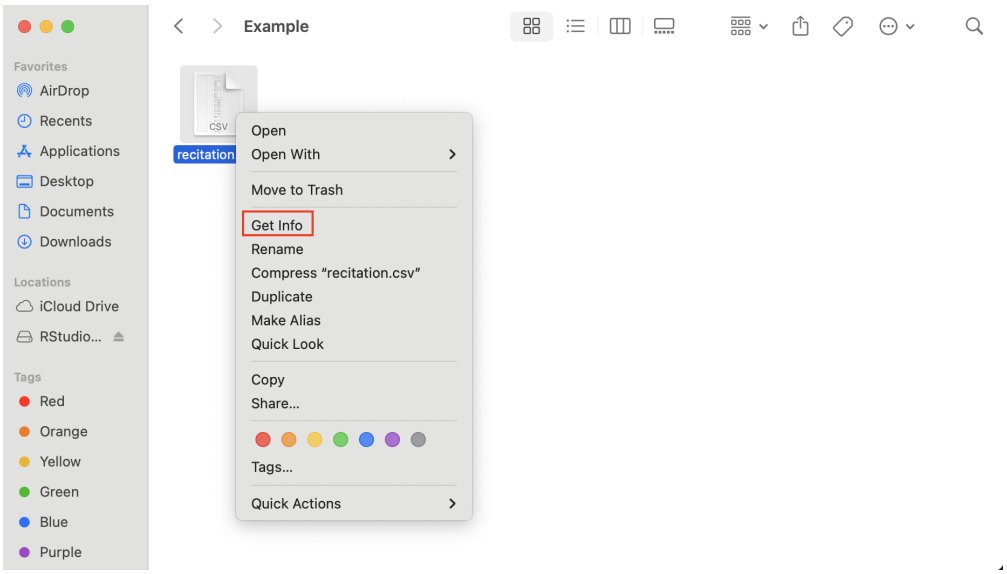
Each row in an observation and each column is a variable.
To load a CSV file, use the following commands:

```
# Load a CSV file
data <- read.csv("file_path/your_file.csv")
```

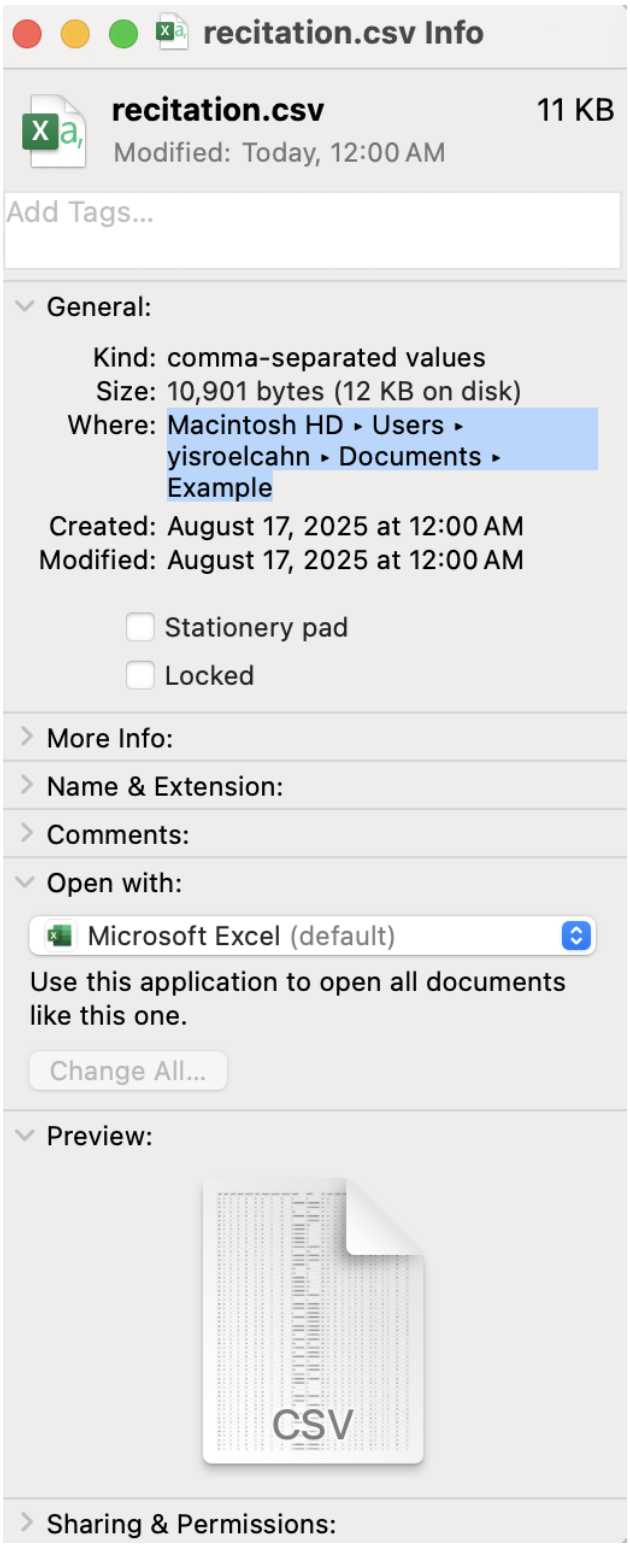
For example:

```
# Load a CSV file
data <- read.csv("/Users/yisroelcahn/Documents/Example/recitation.csv")
```

To find out the file path, left click on a file and select get info:



Then just copy the file path:



To explore the data:

```
# View the first few rows
head(data)
```

```r
# View the structure of the data
str(data)

# See summary statistics for each column
summary(data)

# View the column names
colnames(data)
```

To access a variable in the data us `$`:

```r

age <- data$age

age

mean(age)
```

Help and Documentation

Need help with a function? Use:

```r
?mean       # Help on mean()
help(sd)    # Help on sd()
```

Google and ChatGPT are your friends, do not be afraid to use them!

Useful Shortcuts in RStudio

- `Ctrl + Enter` (Windows) or `Cmd + Enter` (Mac): Run current line
- `Tab`: Auto-complete
- `Ctrl + Shift + C`: Comment/uncomment selected lines

RMarkdown

RMarkdown is a tool that lets you combine text, code, and output (like plots or tables) in a single document, making it easy to create reproducible reports, analyses, and presentations. RMarkdown will not be covered in this crash course.

Basic Portfolio Theory:

All investments come with an *opportunity cost*. By putting your money into real estate, for example, you are giving up the chance to invest in something else, that is your opportunity cost. So how do we decide what is the best investment?

It turns out, there is no free lunch. Every investment involves a trade-off between risk and reward. High-return investments (like cryptocurrencies) usually come with high risk, their value can fluctuate wildly and may drop significantly. On the other hand, low-risk investments (like U.S. Treasury bonds, which are unlikely to default and are generally considered "riskless") typically offer lower returns. As of August 15, 2025, the yield on the 10-year U.S. Treasury bond is 4.32%.

Portfolio theory is about how to combine different types of investments (like stocks, bonds, and other assets) to balance risk and return. By *diversifying*, spreading your money across multiple assets, you can reduce risk without lowering the expected return.

There are two types of risk:

- **Systematic risk** (also called *market risk*) affects the entire market and **cannot** be eliminated. Examples include recessions, inflation, or changes in interest rates.
- **Unsystematic risk** is specific to a single company or industry, such as bad management or a product failure. This type of risk **can be reduced** through diversification.

By building a well-diversified portfolio, you can reduce overall volatility (risk) while maintaining the same expected return. In other words, diversification helps you achieve a better risk-return tradeoff: the same potential reward, but with less uncertainty. While you still face a risk-return trade-off, diversification allows you to face a better one.

Correlation and Diversification

Correlation measures how two stocks move relative to each other. It ranges from -1 to 1:

- A correlation of 1 means the stocks move perfectly together.
- A correlation of 0 means the stocks move independently.
- A correlation of -1 means the stocks move in opposite directions.

When building a portfolio, combining stocks that are *uncorrelated* or *negatively correlated* helps reduce unsystematic risk, the risk specific to individual companies. This is because when one stock underperforms, another might perform well, helping to smooth out the overall portfolio's returns. This is the key idea behind diversification: spreading your investments across different assets can lower total risk without reducing expected return.

Sharpe Ratio

The *Sharpe Ratio* is a measure of how much return an investment gives you for each unit of risk you take. It compares the investment's excess return (return above the risk-free rate) to its volatility:

$$\text{Sharpe Ratio} = \frac{R_i - R_f}{\sigma_i}$$

Where:

- R_i = average return of the investment
- R_f = risk-free rate
- σ_i = standard deviation of the investment's returns (a measure of risk)

A higher Sharpe Ratio means better risk-adjusted performance—more return per unit of risk. It helps investors compare investments with different levels of risk.

CAPM (Capital Asset Pricing Model)

The *Capital Asset Pricing Model (CAPM)* is a basic but important model in finance that helps explain how much return an investor should expect from a stock, given the amount of risk they are taking. Specifically, CAPM links a stock's expected return to its systematic risk, that is, risk that comes from overall market movements, not from anything specific to the company. This kind of risk is measured by a number called *beta*.

The idea is that investors should be rewarded in two ways:

- For the *time value of money*, using the risk-free rate (the return from a very safe investment, like a U.S. Treasury bond).
- For the *risk taken*, based on how much riskier the investment is compared to the overall market.

The CAPM equation is:

$$\text{Expected Return} = R_f + \beta \underbrace{(R_m - R_f)}_{\text{market risk premium}}$$

Where:

- R_f = the risk-free rate (e.g., 10-year Treasury yield)
- R_m = the expected return of the overall market
- β = how sensitive the stock is to market movements

The term $(R_m - R_f)$ is called the *market risk premium*. It represents how much more return investors expect to earn from the market compared to a risk-free investment.

Regression and Jensen's Alpha:

We can estimate how well a stock performed relative to CAPM expectations using a *regression*, which is a way to fit a straight line to data points. Specifically, we regress the stock's excess returns (its returns minus the risk-free rate) on the market's excess returns:

$$\text{Stock Return} - R_f = \alpha + \beta(R_m - R_f) + \epsilon$$

- α is called Jensen's Alpha. It tells us whether the stock earned more or less than CAPM predicted.
- β is the same as before—how much the stock moves with the market.
- ϵ is the error term (the random variation not explained by the model).

If α is positive, the stock did better than CAPM predicted (i.e., it "beat the market" on a risk-adjusted basis). If α is negative, it underperformed after adjusting for risk.

Calculating Correlations, the Sharpe Ratio, and Jensen's Alpha in R:

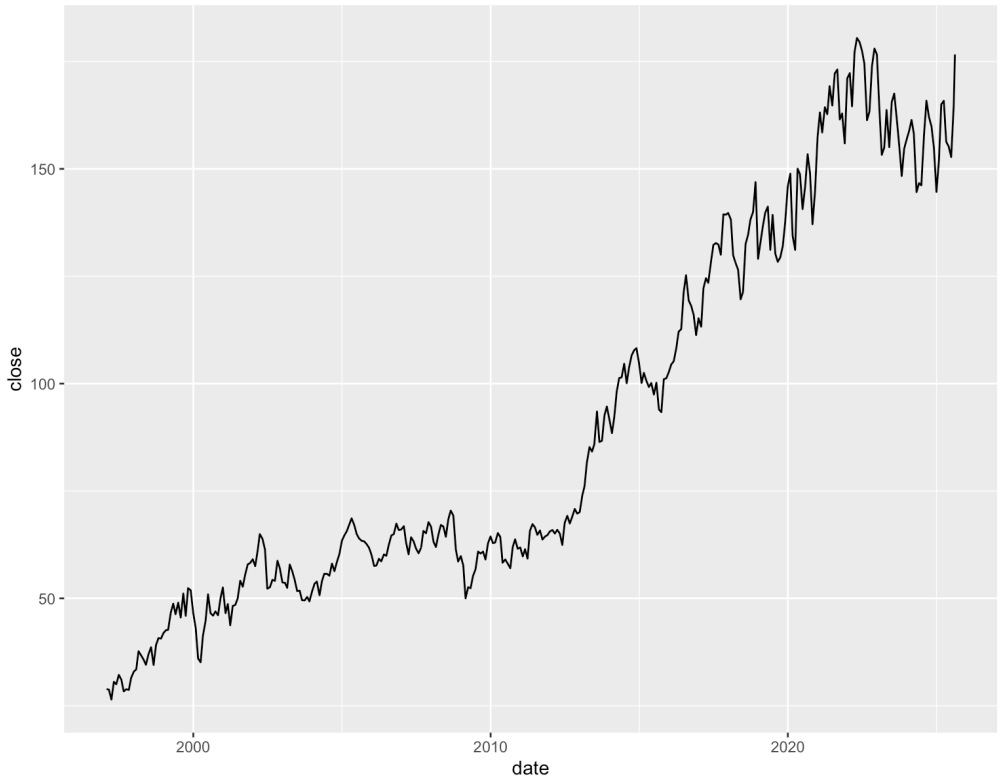
Correlations

While you could download stock price data as Excel files from Yahoo Finance and then upload them into R, R already has a built-in package called `tidyquant` that allows you to access stock price data directly.

In R, type and run:

```
#install packages (if installed before, you do not need to install them again)
install.packages("tidyquant")
install.packages("ggplot2")
install.packages("dplyr")
install.packages("GGally")

#load libraries
library(tidyquant)
library(ggplot2)
library(dplyr)
library(GGally) #for correlation matrix

#get historical data for a single stock. E.g., "TSLA" Tesla or "GOOG" Google
tsla <- tq_get("TSLA",get="stock.prices", from="2010-01-01") %>%
                tq_transmute(mutate_fun=to.period,period="months")

ggplot(jnj,aes(date,close)) + geom_line()
```

Tesla's stock price over time looks like:



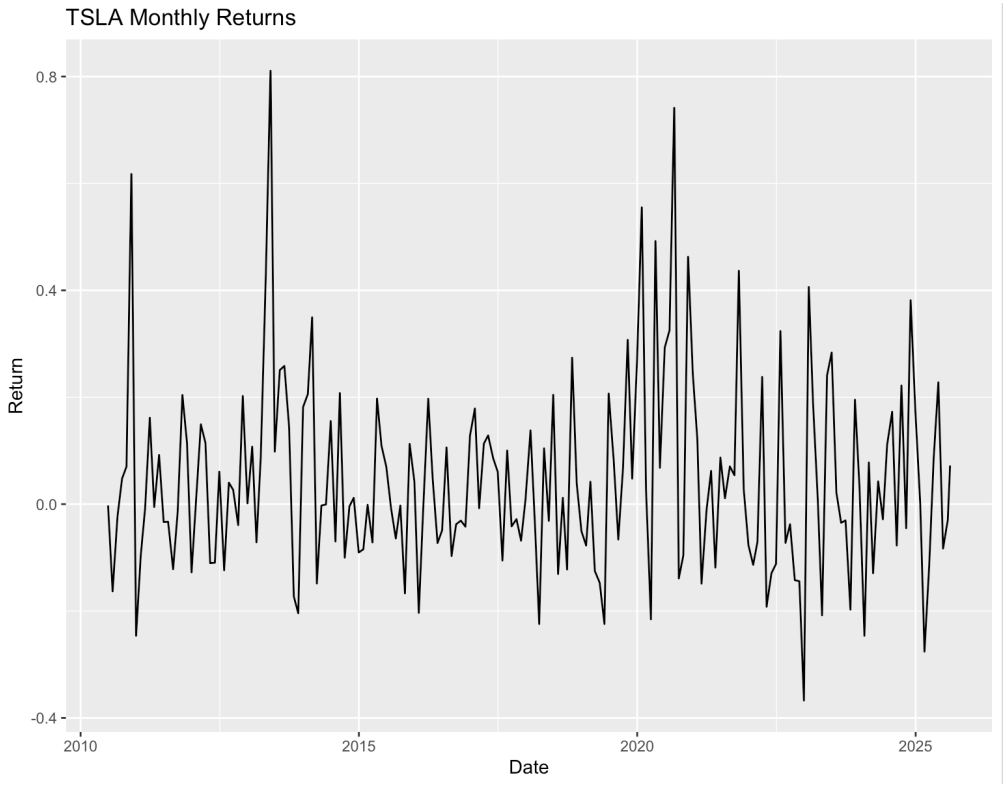
Instead of looking at prices, we can look at returns.

$$\text{Return}_t = \frac{\text{Price}_t - \text{Price}_{t-1}}{\text{Price}_{t-1}}$$

For example, if the price last month was \$500 and the price this month is \$600, then the return is $\frac{600-500}{500} = 0.2$ or 20%.

Telsa's monthly returns looks like:

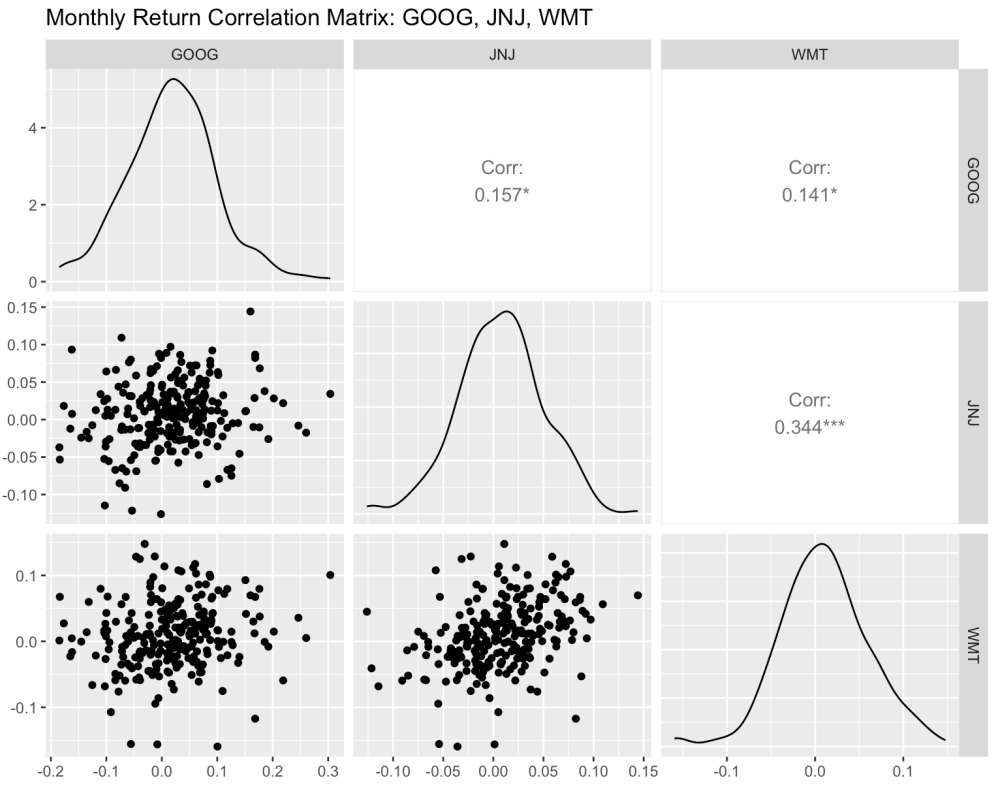
```
# Get monthly returns for TSLA from 2010
tsla_returns <- tq_get("TSLA", get = "stock.prices", from = "2010-01-01") %>%
  tq_transmute(
    select     = adjusted,
    mutate_fun = periodReturn,
    period     = "monthly",
    col_rename = "monthly_return"
  )

ggplot(tsla_returns, aes(x = date, y = monthly_return)) +
  geom_line() +
  labs(title = "TSLA Monthly Returns",
       x = "Date",
       y = "Return")
```



Now to find the correlations between different stock returns:

```
# Define tickers
tickers <- c("GOOG", "JNJ", "WMT")

# Get monthly returns for all stocks
```

```r
returns_data <- tq_get(tickers,
                       get = "stock.prices",
                       from = "2005-01-01") %>%  # GOOG IPO was in 2004
  group_by(symbol) %>%
  tq_transmute(
    select     = adjusted,
    mutate_fun = periodReturn,
    period     = "monthly",
    col_rename = "monthly_return"
  ) %>%
  ungroup() %>%
  tidyr::pivot_wider(names_from = symbol, values_from = monthly_return)

# Remove rows with missing data (e.g., early dates with missing GOOG data)
returns_data_clean <- na.omit(returns_data)

# View a correlation scatterplot matrix
ggpairs(returns_data_clean[, -1],  # exclude date column
        title = "Monthly Return Correlation Matrix: GOOG, JNJ, WMT")
```



As you can see, the stock returns of Walmart and Johnson and Johnson are more highly correlated than those of Walmart and Google. Therefore, including both Walmart and Google in your portfolio (as opposed to Walmart and Johnson and Johnson) would provide better diversification and help reduce unsystematic risk.

Now, change the tickers to at least five other stocks that you are interested in possibly including in your portfolio, and examine their correlations in the same way.

Sharpe Ratio and Jensen's Alpha

To calculate the Sharpe Ratio and Jensen's Alpha for Google:

```r
# Step 1: Get GOOG monthly returns
goog <- tq_get("GOOG", get = "stock.prices", from = "2010-01-01") %>%
  tq_transmute(
    select     = adjusted,
    mutate_fun = periodReturn,
    period     = "monthly",
    col_rename = "goog_return"
  )

# Step 2: Get S&P 500 returns (market proxy)
sp500 <- tq_get("^GSPC", get = "stock.prices", from = "2010-01-01") %>%
  tq_transmute(
    select     = adjusted,
    mutate_fun = periodReturn,
    period     = "monthly",
    col_rename = "market_return"
  )

# Step 3: Join returns
returns <- left_join(goog, sp500, by = "date")

# Step 4: Set risk-free rate (monthly)
rf_annual <- 0.0432
rf_monthly <- rf_annual / 12  # = 0.0036

# Step 5: Calculate excess returns
returns <- returns %>%
  mutate(
    excess_goog   = goog_return - rf_monthly,
    excess_market = market_return - rf_monthly
  )

# Step 6: Sharpe Ratio
sharpe_ratio <- mean(returns$excess_goog, na.rm = TRUE) / sd(returns$goog_return,
    ↪ na.rm = TRUE)
print(paste("Sharpe Ratio:", round(sharpe_ratio, 3)))

# Step 7: Jensen's Alpha via CAPM regression
model <- lm(excess_goog ~ excess_market, data = returns)
alpha <- coef(model)[1]
beta <- coef(model)[2]

# Monthly Jensen's Alpha
print(paste("Beta:", round(beta, 3)))
print(paste("Jensen's Alpha (monthly):", round(alpha, 5)))

# Annualized Jensen's Alpha
jensen_alpha_annualized <- (1 + alpha)^12 - 1
print(paste("Jensen's Alpha (annualized):", round(jensen_alpha_annualized, 4)))
```

Google's Jensen's Alpha (annualized) of 0.0719 means that, after adjusting for risk, the investment outperformed the return predicted by the CAPM model by 7.19% per year.

Google's beta of 1.059 means the stock is slightly more volatile than the overall market (which has a beta of 1). Under the CAPM model, this also implies a slightly higher expected return to compensate for the increased risk.

Google's Sharpe Ratio of 0.178 indicates that the investment delivered a very low amount of return relative to the risk taken. Generally, a Sharpe Ratio of 1 is considered acceptable, 1.5 is

good, and >2 is excellent.

Now, calculate the Jensen's Alpha and Sharpe ratio for at least five other stocks that you are interested in possibly including in your portfolio, and examine their Jensen Alphas and Sharpe Ratios in the same way.

Receiving Credit for this Assignment:

Calculate the correlation matrix, Sharpe ratios, and Jensen's Alpha for at least 5 stocks. Take a screenshot of your R output and submit it on Canvas. Then, create your team's second portfolio on Investopedia based on your analysis.

In two months, you will be asked to provide screenshots showing the final values of both of your portfolios. You will receive the extra credit at that time.