# Should I Use R?

Brad Wakefield

DDSN Sept 2022

## Installing R and RStudio

### Installing R

R is a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques. Installation files and guides to install R can be found for the main three operating systems (windows, macosx, and linux) on the Comprehensive R Archive Network (CRAN) website https://cran.r-project.org/. System requirements related to running R version 4.2.1 can be found on the CRAN website. Tablets such as iPads may not be suitable to run R programming. Cloud-based R platforms may restrict the installation of required dependencies.

A rough installation guide is given below:

**On Windows:**

Go to https://cran.r-project.org/bin/windows/base/ and download R-4.2.1 for Windows.

A download should begin with the R installer program. When finished downloading, run this program and step through the installation wizard that appears. The wizard will install R into your program files folders and place a shortcut in your Start menu provided you have the appropriate administration privileges to install new software on your machine. If you do not have administration privileges, you can still install R into your personal folders.

**On Macosx:**

To install R on a Mac, go to https://cran.r-project.org/bin/macosx/. Next, click on the 4.2.1 package link. An installer will download to guide you through the installation process, which is very easy. The installer lets you customize your installation, but the defaults will be suitable for most users In order to compile some R packages on Mac, some additional development tools may be required. Please go to https://mac.r-project.org/tools/ and check that you have the appropriate development tools (**Xcode** and a **GNU Fortran compiler)** listed installed on your computer.

**On Linux:**

R comes preinstalled on many Linux systems, but you'll want newest version of R, 4.2.1 if yours is out of date. Go to https://cran.r-project.org/bin/linux/ and files to build R from source on Debian, Redhat, SUSE, and Ubuntu systems under the link "Download R for Linux" can be found. Click the link and then follow the directory trail to the version of Linux you wish to install on. The exact installation procedure will vary depending on the Linux system you use.

### Installing RStudio

The RStudio IDE is a set of integrated tools designed to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

Before installing RStudio, please ensure you have successfully installed R on your computer.

To download the **free** RStudio IDE go to https://www.rstudio.com/products/rstudio/download/#download and download the RStudio installer for your operating system. Once downloaded, run the installer wizard and follow the prompts. When completed, open R Studio and ensure it has found your previously installed R version.

## Using RStudio

### The RStudio Layout

### Open RStudio.

You should see a screen with 3 (or 4) windows.

**The top right window** has multiple tabs but take note of the `Environment` tab, which is broadly defined as the current information R has loaded and shows the objects (eg dataframes, lists) in use in the current session. The other tabs we will not worry about for the time being.

**The bottom right window** has 5 tabs:

- `File`; shows the files in the current working directory,
- `Plots`; displays plots when generated using an R command,
- `Packages`; shows a list of packages in the user and system libraries,
- `Help`; is used to search for help on packages and r commands,
- `Viewer`; this frame allows you to view data.

**The left window (or bottom left)** has the **Console**.

**This is R** - it is like a calculator and computes the things we tell it to.

Try doing a simple calculation like `1+1` and pressing enter.

To open **the top right window** (if not already) open a new a script file. You can do this by going to `File; New File >` and selecting `R script`.

A new window called the **Source** window will pop up with a new tab called `Untitled1`. This is a good place to save all your code that your put through the console so you can run it again later.

A script should only contain **executable code** - text that R understands. If you want to write non-executable text, you can do this by putting a `#` before the line you want to write. This is called adding comments, and is a good way of keeping track of what your script does.

We can add a title to our script by adding the line:

```
# Should I Learn R Script
```

If you were to copy and paste this into the **console**. R won't have an error because it knows to ignore it.

### Setting up a Working Directory

It is good to setup a **working directory** for your particular session. A working directory is a place on your computer where you put all the files you want to load in and save from R.

You can get R to create a working directory by creating a new `Project`. Or you can navigate to an existing folder on your computer which you can use as a working directory.

I'm going to add a new folder to use as a working directory on my desktop.

I'm going to call it `ShouldILearnR` and save within it, the data sets I plan to use today that are currently sitting on my desktop.

To tell R where this folder is click `Session` from the top menu bar, `Set Working Directory >` and select `Choose Directory....`

I'm going to navigate to the `ShouldILearnR` folder on my Desktop and click `Open`.

**Notice the following code appears in the R console.**

```
setwd("C:/Users/bradleyw/Desktop/ShouldILearnR")
```

Copy and pasting the line (without the `>` symbol in front of it) into your script means we can simply run it again in the next session to set the working directory.

We now have a place to save our script file.

Go to `File` and click `Save` and save your script file into you working directory.

To see what is in your working directory from R, you can select the `Files` tab in the bottom right. Select the `More` button and choose `Go to Working Directory`.

Now we are ready to start using R.

## Installing Packages

R comes with a basic set of packages (a package is a program written to perform different procedures). Many of the analyses you perform when you start using R require additional packages.

In RStudio it is easy to see the packages you already have and install new ones by clicking on the `Packages` tab in the bottom right window.

You can also use commands to do this

`library()` # command to see packages

To install packages click on the `Packages` tab in RStudio and then press `Install`. Generally we will be installing using the default settings for the CRAN repository and the installation library.

**Note: Once you've installed a package you don't have to install it again but you will have to load it each session you want to use functions from it.**

We are going to have a go at installing the package `readxl`, we need this package to import data from excel documents.

- Click the `Packages` tab.
- Click `Install`
- Make sure in the `Install from:` option, you have `Repository (CRAN)` selected.
- Type `readxl` into the `Packages (separate multiple with space or comma):` option.
- To load the package into R, select the `Packages` tab and scroll down the list until you see `readxl` and click it. You will need to do this every session.
  - Alternatively, add to your script `library(readxl)`. This will make sure the package is loaded when you run your script.

## Importing Data into R

We are now all ready to import data into R.

I want to import data from the excel spreadsheet `cholesterol.xlsx`.

The data are 5 variables (columns) from the chronic heart disease data set. It includes the cholesterol level (`chol`), the age (`age`), gender category (`gender`), chronic health disease status (`chd`), and resting blood pressure (`trestbps`).

I'm just going to open it up in excel and make sure it is in **rectangular form.**

To load into R, select `File > Import Dataset > From Excel...`

A window will open that will assist you to import the data.

I'm going to change the name of my data set to `data` in the `Name:` option.

When I click `Import` a data viewer should open and the following code should appear in the console.

```
library(readxl)
data <- read_excel("C:/Users/bradleyw/Desktop/ShouldILearnR/cholesterol.xlsx")
View(data)
```

Copy and paste the code

`data <- read_excel("cholesterol.xlsx")`

and add it to your script file.

Now if I was to make changes to my data, I just need to rerun the code and it should update `data` in R.

Note that you can import all sorts of data files into R, including text, SPSS, SAS, and STATA.

See how R displays the data by just typing `data` into the console.

```
data
```

```
## # A tibble: 297 x 5
##      age gender trestbps  chol   chd
##    <dbl> <chr>     <dbl> <dbl> <dbl>
##  1    41 Female      130  5.28     0
##  2    62 Female      140  6.93     1
##  3    57 Female      120  9.15     0
##  4    56 Female      140  7.60     0
##  5    48 Female      130  7.11     0
##  6    58 Female      150  7.32     0
##  7    50 Female      120  5.66     0
##  8    58 Female      120  8.79     0
##  9    66 Female      150  5.84     0
## 10    69 Female      140  6.18     0
## # ... with 287 more rows
## # i Use `print(n = ...)` to see more rows
```

## Summary Statistics in R

To obtain a basic summary of the cholesterol data and how it is stored in R we can use the `summary()` function.

```
summary(data)
```

```
##       age           gender             trestbps          chol
##  Min.   :29.00   Length:297         Min.   : 94.0   Min.   : 3.258
##  1st Qu.:48.00   Class :character   1st Qu.:120.0   1st Qu.: 5.456
##  Median :56.00   Mode  :character   Median :130.0   Median : 6.284
##  Mean   :54.54                      Mean   :131.7   Mean   : 6.396
##  3rd Qu.:61.00                      3rd Qu.:140.0   3rd Qu.: 7.137
##  Max.   :77.00                      Max.   :200.0   Max.   :14.585
##       chd
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.4613
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

The summary gives us information about how R "sees" the data. All the variables, except for `gender`, have been imported as numeric variables and described using a six number summary (Min, Q1, Median, Mean, Q3, Max). `gender` is not a numeric variable, it is categorical (binary) and was read in as character (string) and needs to be recoded so R will see it, and use it correctly. Distinctions between variable types are important as R will only analyse appropriately described variables.

To select only a specific variable (column) from the data set you can add a `$` sign after the data set variable `data` and typing the name of the column.

Hence to calculate the mean cholesterol (`chol`) using the `mean()` function…

```
mean(data$chol)
```

```
## [1] 6.396475
```

To calculate the standard deviation use the `sd()` function.

```
sd(data$chol)
```

```
## [1] 1.344657
```

To get the five number summary you can use the `fivenum()` function.

```
fivenum(data$chol)
```

```
## [1]  3.25836  5.45646  6.28398  7.13736 14.58504
```

These statistics are appropriate for numeric variables.

For categorical variables, calculating frequencies are a good way to describe the data.

This can be done with the `table()` function.

```
table(data$gender)
```

```
##
## Female   Male
##     96    201
```

```
table(data$chd)
```

```
##
##   0   1
## 160 137
```

To provide summary statistics describing the relationship between two variables in R, we have:

- For two continuous variables, the correlation can be calculated with the `cor()` function.

  ```
  cor(data$chol,data$age)
  ```

  ```
  ## [1] 0.2026435
  ```

- For two categorical variables, a cross tabulation can be performed with the `table()` function.

  ```
  table(data$gender,data$chd)
  ```

  ```
  ##
  ##            0   1
  ##   Female  71  25
  ##   Male    89 112
  ```

- For a continuous and categorical variable, we can compute statistics (such as mean) using the `aggregate()` function.

  ```
  aggregate(chol ~ gender,FUN = mean, data = data)
  ```

  ```
  ##   gender     chol
  ## 1 Female 6.781246
  ## 2   Male 6.212704
  ```

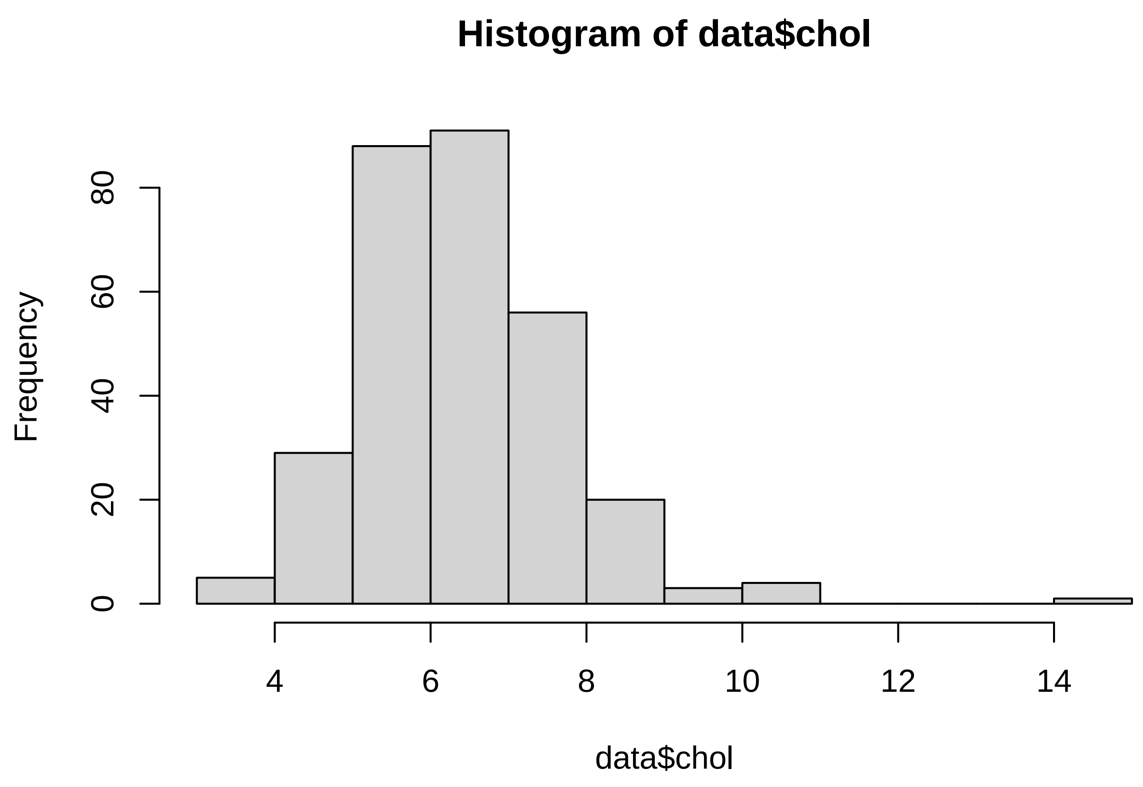
  Note the syntax `chol ~ gender`. We read this as `chol` “by” `gender`.

## Plotting in R

When performing data analysis,we must **always plot our data!**

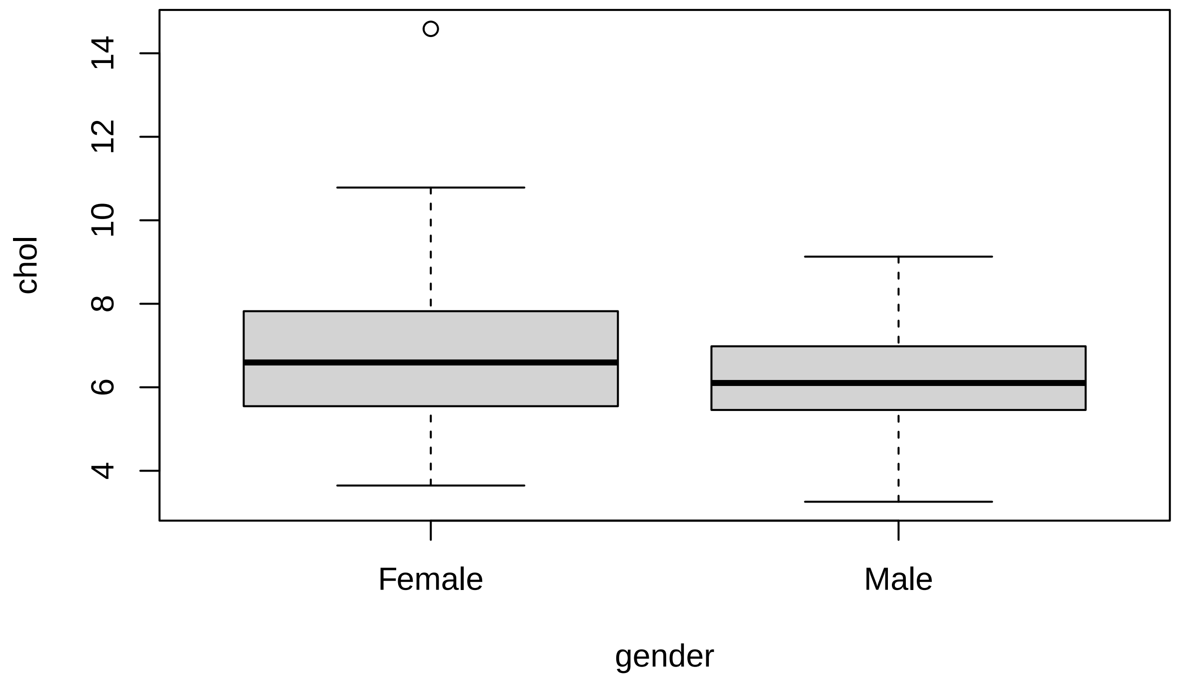
Histograms are a good way to visualise the distribution of numeric variables. We can generate a histogram in R with the `hist()` function.

```
hist(data$chol)
```



Boxplots are great for comparing numeric variables across multiple categories.

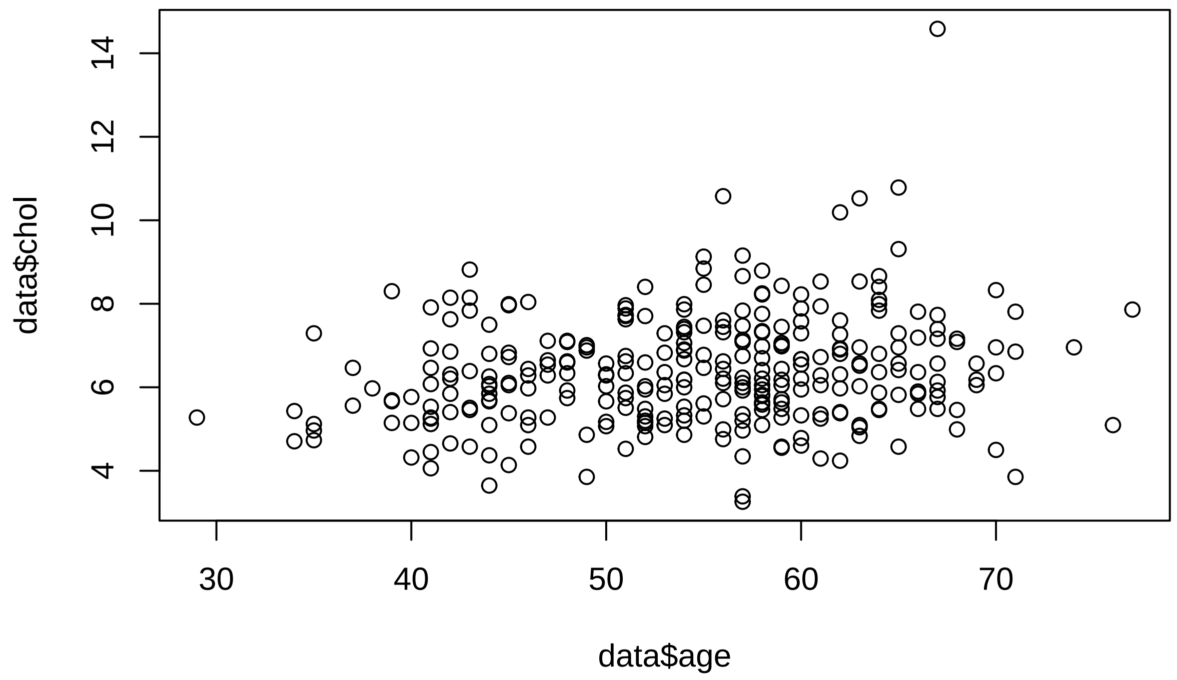
```
boxplot(chol ~ gender ,data=data)
```



Notice the "by" syntax again with `~`.

To produce a scatterplot for comparing two numeric variables, we can use the `plot()` function.

```
plot(x=data$age,y=data$chol)
```



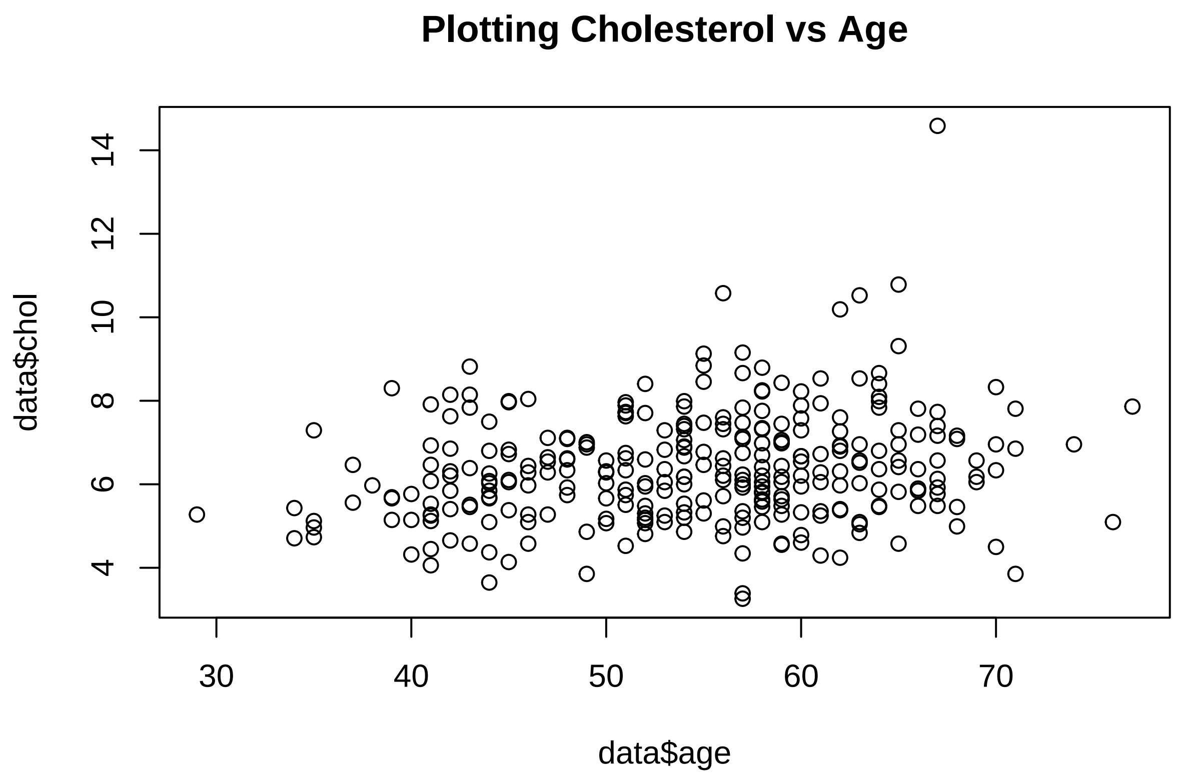
To customise plots in R, you need to specify the options with additional arguments.

**Note:** To see what options you have available to you, (for any functions) you can use the `help()` command.

```
help(plot)
```

Looking at the help document, we can see that to change the title, we can specify the `main` argument.

```
plot(x=data$age,y=data$chol,
     main="Plotting Cholesterol vs Age")
```



Another important data visualisation package in R is `ggplot2`. Part of the `tidyverse` packages, `ggplot2` implements the grammar of graphics.

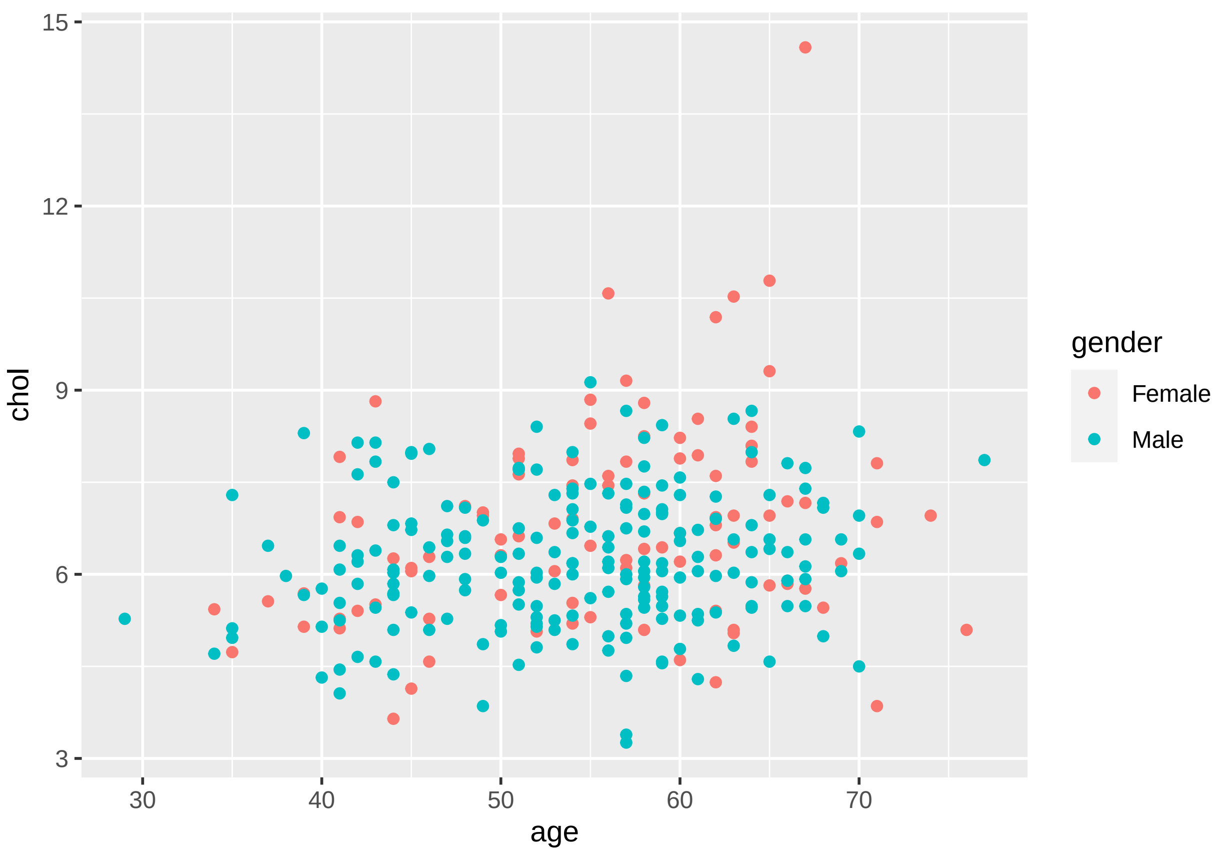
It is a little bit tricky to get used to but you can obtain some good visualisations.

Here is an example below...

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.1
```

```
ggplot(data) +
  geom_point(aes(x=age,y=chol,col=gender))
```



## A Simple Model in R

One of the key strengths in R is the ability to perform a large range of statistical analyses.

**Regression analysis** refers to the statistical process by which these coefficients are estimated based on an observed sample of the data and the process of checking assumptions of the model.

**Linear regression** refers to the application of a linear model to describe/predict the relationship between a numeric response variable (often denoted by $Y$) and one or more predictor variables (often denoted $X_1, X_2, \ldots$).

To fit a linear regression model in R we can use the `lm()` function with the same kind of syntax previously seen.

Suppose we wanted to model cholesterol (`chol`) based on the age (`age`) and gender category (`gender`) of an individual in the data set.

In this model cholesterol is our response variable ($Y$) and age and gender are our predictor variables.

To specify the model in `lm()` we write,

```
model <- lm(chol ~ age + gender, data=data)
```

We can see the key results of the analysis by returning `model` into the console.

```
model
```

```
##
## Call:
## lm(formula = chol ~ age + gender, data = data)
##
## Coefficients:
## (Intercept)          age   genderMale
##     5.24109      0.02763     -0.51923
```

To get a more detailed output we can use the `summary()` function.

```
summary(model)
```

```
##
## Call:
## lm(formula = chol ~ age + gender, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3494 -0.8865 -0.1231  0.7586  7.4930
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.241091   0.485458  10.796  < 2e-16 ***
## age          0.027626   0.008377   3.298  0.00109 **
## genderMale  -0.519234   0.161812  -3.209  0.00148 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.299 on 294 degrees of freedom
## Multiple R-squared:  0.07351,    Adjusted R-squared:  0.06721
## F-statistic: 11.66 on 2 and 294 DF,  p-value: 1.335e-05
```

You can also perform other analyses to check your assumptions (e.g. by producing diagnostic plots with the `plot()` function), but due to time constraints we wont run through those now.

Now lets try another example from start to finish...

## A Worked Example

Consider the following data from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. All patients here are females at least 21 years old of Pima Indian heritage.

The data is currently saved in a SPSS file, `diabetes.sav`.

```
## Warning: package 'haven' was built under R version 4.2.1
```

Suppose we wish to model the odds that a women from the Pima Indians diabetes study haves diabetes given their plasma glucose concentration (`gluc`), body mass index (`bmi`), and age (`age`). That is, we wish to model the probability of the `Diabetes` column taking the value `Has Diabetes`.

We also want to get a few descriptive statistics and plots on our data.

We run the script below.

```r
# Load in Data
library(haven)
diabetes <- read_sav("diabetes.sav")

# Allocate labels to categorical variable
diabetes$Diabetes <- factor(diabetes$Diabetes,levels = c(2,1))
levels(diabetes$Diabetes) <- c("No Diabetes","Has Diabetes")

# Proportion of patients with and without diabetes
table(diabetes$Diabetes)/nrow(diabetes)
```

```
##
##  No Diabetes Has Diabetes
##    0.6716867    0.3283133
```

```r
# Difference between groups
aggregate(gluc ~ Diabetes,FUN = summary,data=diabetes)
```

```
##       Diabetes gluc.Min. gluc.1st Qu. gluc.Median gluc.Mean gluc.3rd Qu.
## 1  No Diabetes  3.610000     5.110000    5.720000  6.004350     6.830000
## 2 Has Diabetes  4.330000     6.270000    7.990000  7.875963     9.550000
##   gluc.Max.
## 1 10.930000
## 2 10.930000
```

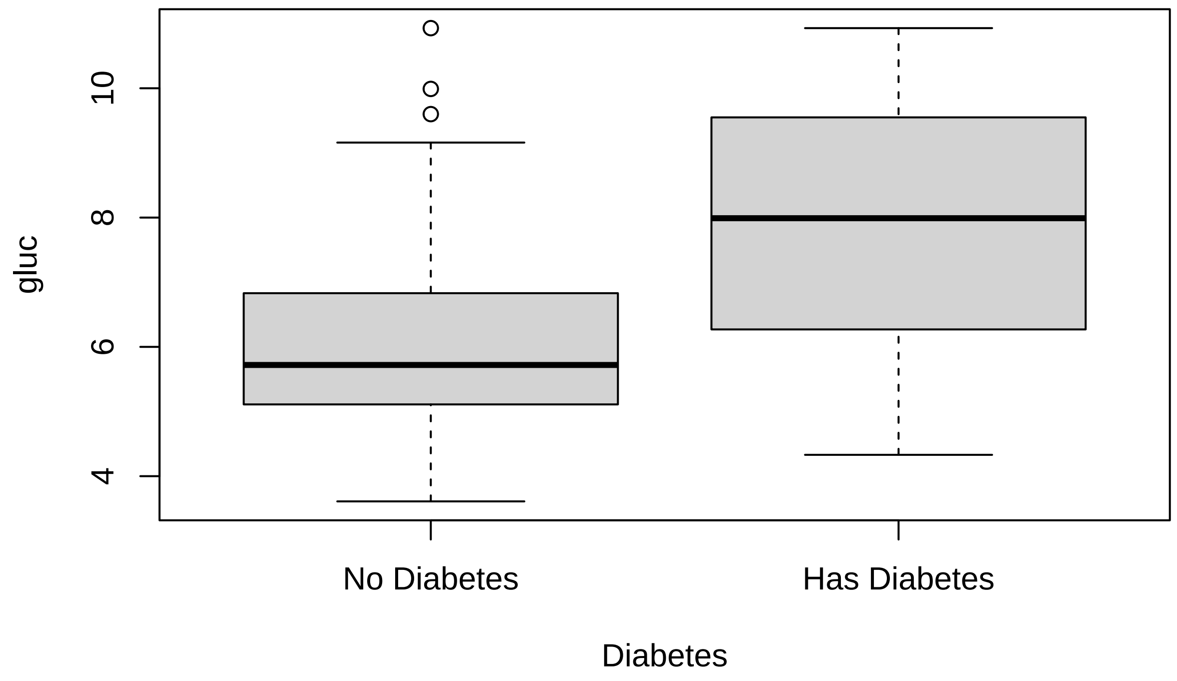
```r
aggregate(bmi ~ Diabetes,FUN = summary,data=diabetes)
```

```
##       Diabetes bmi.Min. bmi.1st Qu. bmi.Median bmi.Mean bmi.3rd Qu. bmi.Max.
## 1  No Diabetes 19.40000    27.00000   30.80000 31.63991    35.90000 57.30000
## 2 Has Diabetes 25.50000    31.60000   34.90000 36.51284    40.90000 67.10000
```

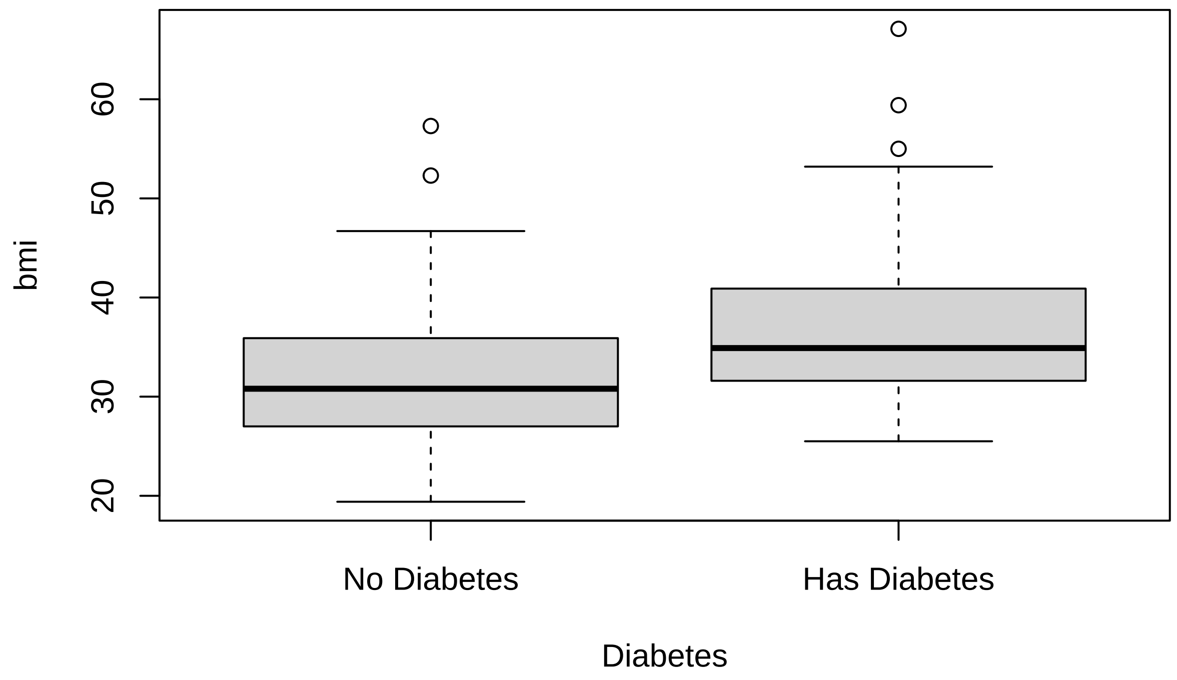
```r
aggregate(age ~ Diabetes,FUN = summary,data=diabetes)
```

```
##       Diabetes age.Min. age.1st Qu. age.Median age.Mean age.3rd Qu. age.Max.
## 1  No Diabetes 21.00000    22.00000   25.00000 29.21525    33.00000 81.00000
## 2 Has Diabetes 21.00000    27.00000   33.00000 35.61468    43.00000 70.00000
```

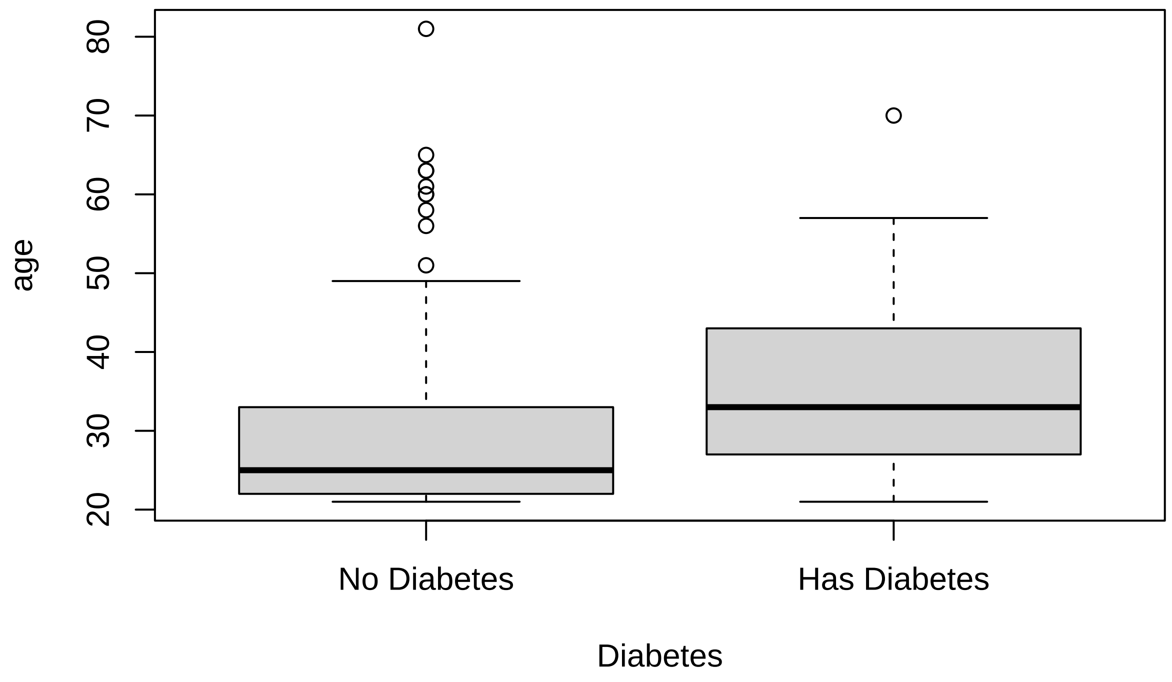
```r
boxplot(gluc ~ Diabetes,data=diabetes)
```



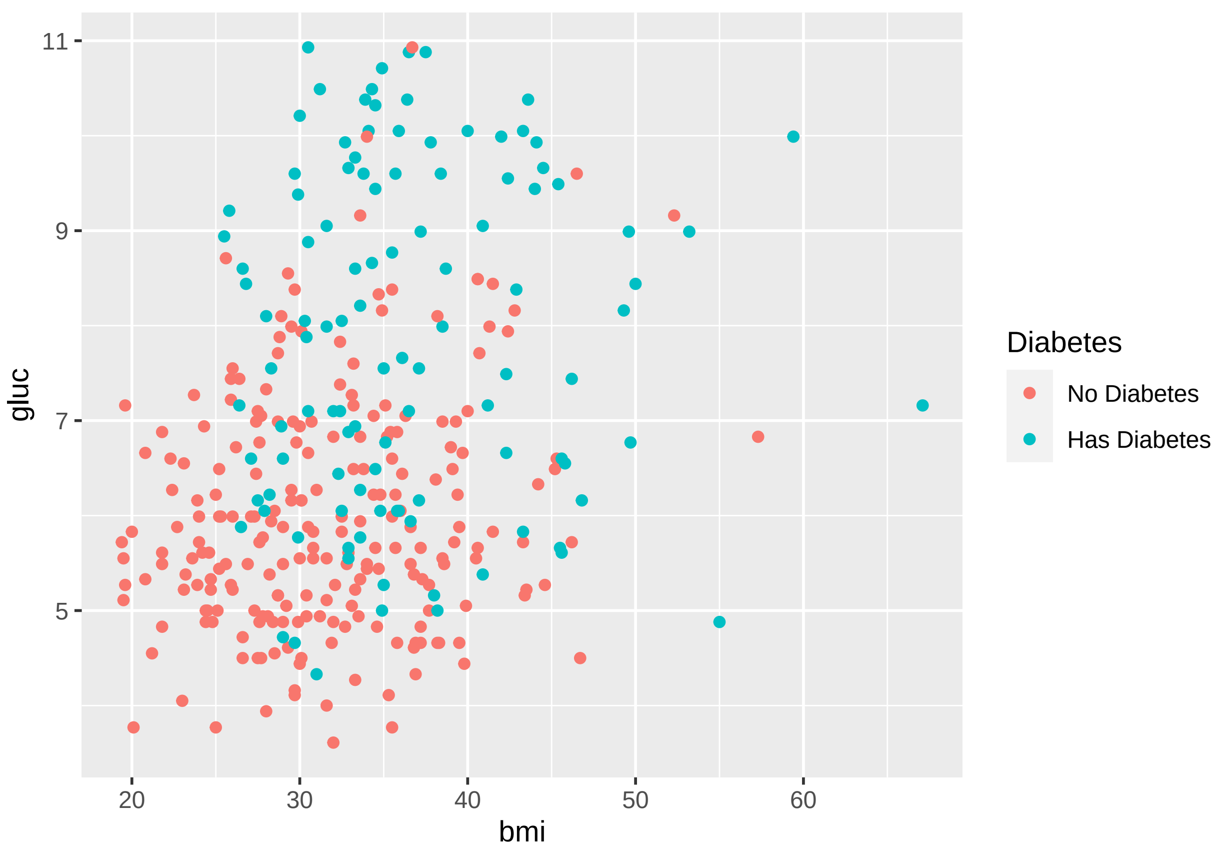
```r
boxplot(bmi ~ Diabetes,data=diabetes)
```



```
boxplot(age ~ Diabetes,data=diabetes)
```



```
# Plot of data
library(ggplot2)
ggplot(diabetes) +
  geom_point(aes(x=bmi,y=gluc,col=Diabetes))
```



```r
# Fit a model (we are after a logistic regression)

diabetes_model <- glm(Diabetes ~ gluc + bmi + age,
                      data=diabetes,
                      family = binomial(link = "logit"))
summary(diabetes_model)
```

```
##
## Call:
## glm(formula = Diabetes ~ gluc + bmi + age, family = binomial(link = "logit"),
##     data = diabetes)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5851  -0.6512  -0.3955   0.6183   2.4407
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.48147    1.08270  -8.757  < 2e-16 ***
## gluc         0.66299    0.09565   6.932 4.16e-12 ***
## bmi          0.07932    0.02105   3.768 0.000165 ***
## age          0.04713    0.01331   3.540 0.000400 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.30  on 331  degrees of freedom
## Residual deviance: 299.32  on 328  degrees of freedom
## AIC: 307.32
##
## Number of Fisher Scoring iterations: 5
```

```
# Compute Odds Ratios and 95% CI
exp(coef(diabetes_model))
```

```
##  (Intercept)         gluc          bmi          age
## 7.625142e-05 1.940581e+00 1.082553e+00 1.048261e+00
```

```
exp(confint(diabetes_model))
```

```
## Waiting for profiling to be done...
```

```
##                    2.5 %       97.5 %
## (Intercept) 8.102322e-06 0.0005717271
## gluc        1.620629e+00 2.3608336569
## bmi         1.039955e+00 1.1297447823
## age         1.021747e+00 1.0767517595
```