Fundamentals of Mathematics and Statistics with R

Dr. Priyanga D. Talagala

2023-10-17

Contents

Chapter 1

Introduction to R, Rstudio and Posit Cloud

1.1 Installing R and Rstudio

- **Step 1:** First download R freely from the Comprehensive R Archive Network (CRAN) https://cran.r-project.org/. (At the moment of writing, R 4.3.1 is the latest version. Choose the most recent one.)
- **Step 2:** Then install R Studio's IDE (stands for integrated development environment), a powerful user interface for R from [https://posit.co/download/rstudio-desktop/](https://posit.co/download/rstudio-desktop/. Get the Open Source Edition of RStudio Desktop. RStudio allows you to run R in a more user-friendly environment.
  - You need to install **both** R and Rstudio to use RStudio.
  - If you have a pre-existing installation of R and/or RStudio, I highly recommend that you reinstall both and get as current as possible.
- **Step 3:** Then open **Rstudio**.

1.1.1 Posit Cloud

- In 2022, RStudio changed its corporate name to Posit with the aim of expanding its focus beyond R to include users of Python and Visual Studio Code.
- If you don't want to download or install R and R Studio, you can use RStudio on Posit Cloud (https://posit.cloud/) for free.

1.2 RStudio layout

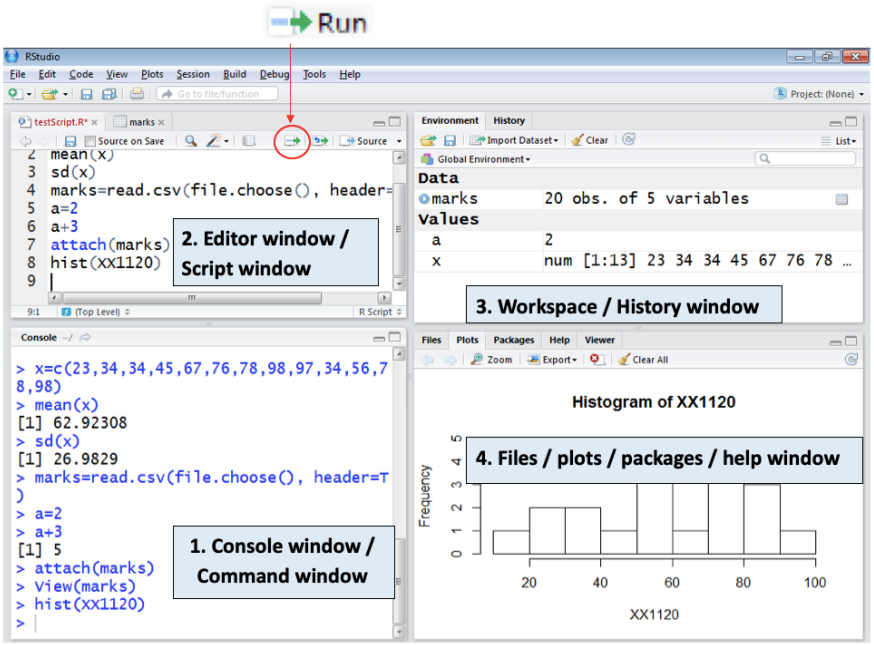
The RStudio interface consists of four windows (see Figure 1 and 2).

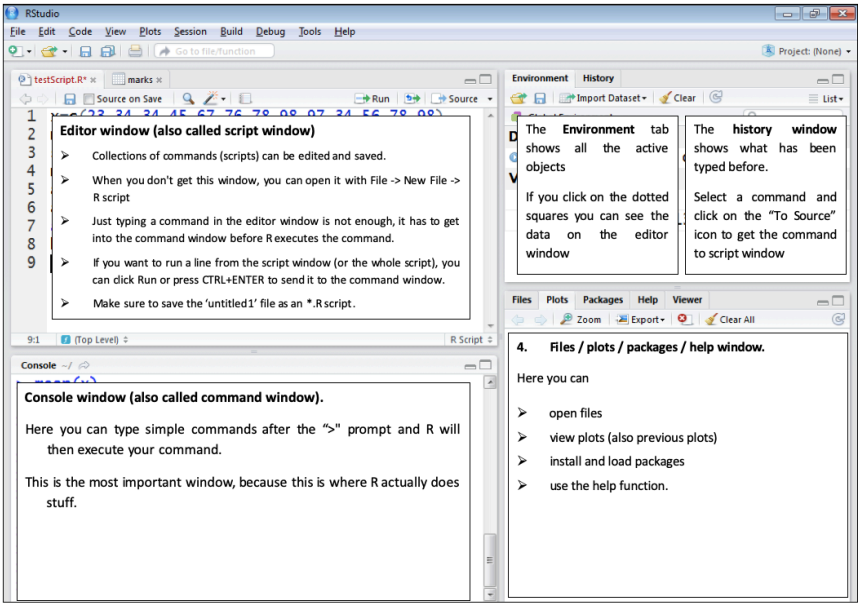
1. Bottom left: console window (also called command window). **This is where you type and run all your R commands**

2. Top left: editor window (also called script window).

3. Top right: workspace / history window.

4. Bottom right: Files / plots / packages / help window.





Now you are familiar with the layout. Let's begin with R basics.

1.3 Installing an R Package

- The primary location for obtaining R packages is CRAN
- Packages can be installed with the `install.packages()` function in R
- To install a single package, pass the name of the package to the `install.packages()` function as the first argument

The following the code installs the `tidyverse` package from CRAN

```
install.packages("tidyverse")
```

- This command downloads the `tidyverse` package from CRAN and installs it on your computer
- Any packages on which this package depends will also be downloaded and installed
- **Installing the tidyverse package could take several minutes. You only need to do this once**.

1.4 Loading an R Packages

- Installing a package does not make it immediately available to you in R; you must load the package
- The `library()` function is used to load packages into R
- The following code is used to load the tidyverse package into R
- **NOTE:** Do not put the package name in quotes!

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
```

- Some packages produce messages when they are loaded (but some don't)

1.5 Getting started with R

An Introduction to R: https://cran.r-project.org/doc/manuals/R-intro.pdf

Chapter 2

Differentiation

First, we take the equation as an expression

```
f <- expression(x^2)
```

To calculate first derivative of f, we use `D()` function and `x` to specify that derivation has to be carried out with respect to x.

```
f_1 <- D(f, "x")
print(f_1)
```

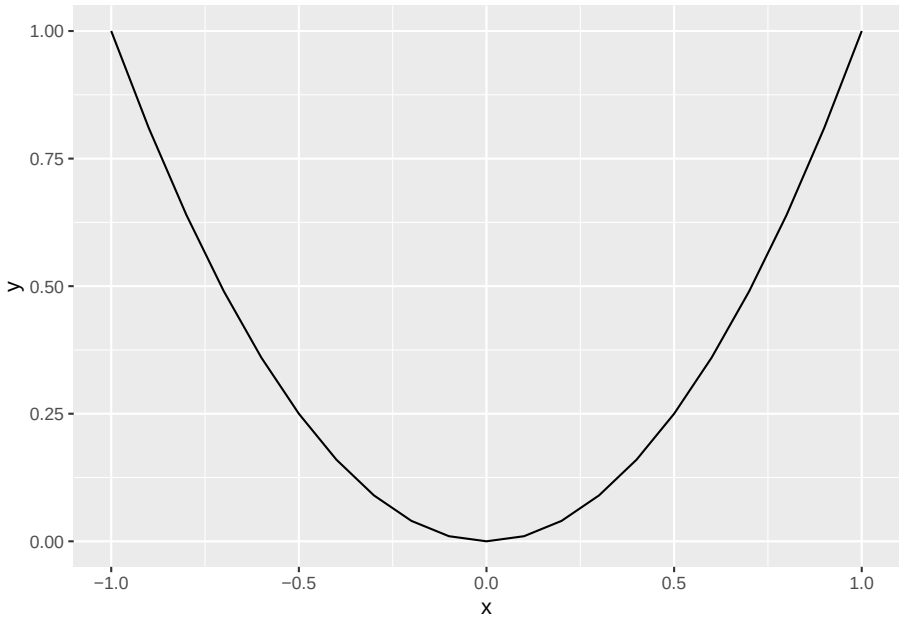
```
## 2 * x
```

Sketch the graph of f and f'

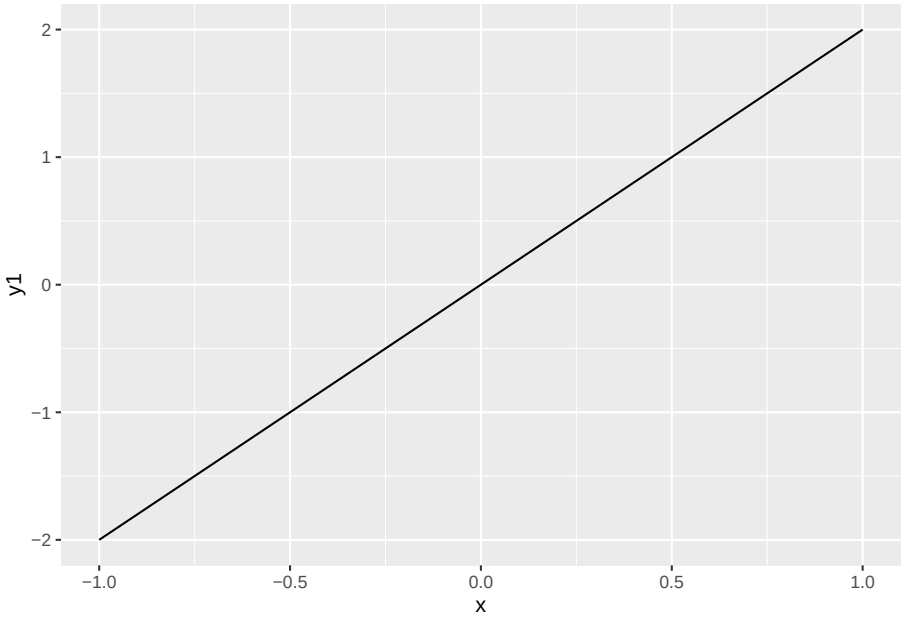
```
library(ggplot2)
x <- seq(-1, 1, by = 0.1)
y <- eval(f)
x <- seq(-1, 1, by = 0.1)
y1 <- eval(f_1)
data <- data.frame(x, y, y1)
head(data)
```

```
##      x    y   y1
## 1 -1.0 1.00 -2.0
## 2 -0.9 0.81 -1.8
## 3 -0.8 0.64 -1.6
## 4 -0.7 0.49 -1.4
## 5 -0.6 0.36 -1.2
## 6 -0.5 0.25 -1.0
```

```
p <- ggplot(data, aes(x = x, y = y)) +
  geom_line()
print(p)
```




```
q <- ggplot(data, aes(x = x, y = y1)) +
  geom_line()
print(q)
```



2.1 Higher Derivatives

The following R command can be used to find second derivative of the above f.

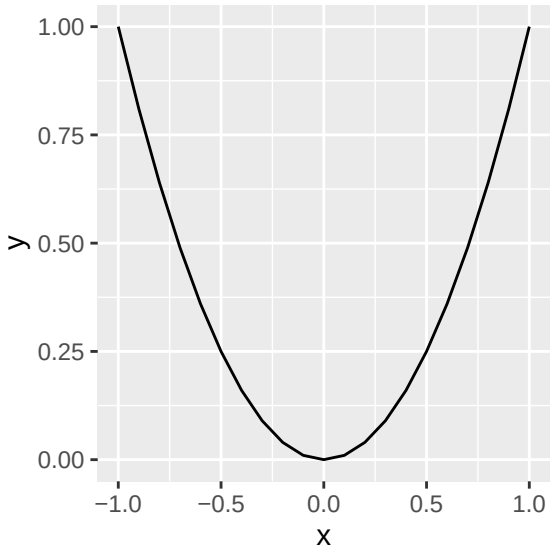
```
f_2 <- D(D(f, "x"), "x")
print(f_2)
```

```
## [1] 2
```

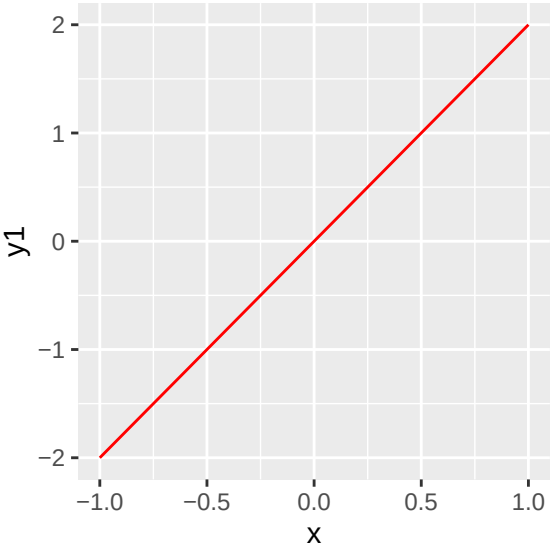
```
x <- seq(-1, 1, by = 0.1)
y2 <- eval(f_2)

data <- data.frame(x, y, y1, y2)
head(data)
```

```
##      x    y   y1 y2
## 1 -1.0 1.00 -2.0  2
## 2 -0.9 0.81 -1.8  2
## 3 -0.8 0.64 -1.6  2
## 4 -0.7 0.49 -1.4  2
## 5 -0.6 0.36 -1.2  2
## 6 -0.5 0.25 -1.0  2
```

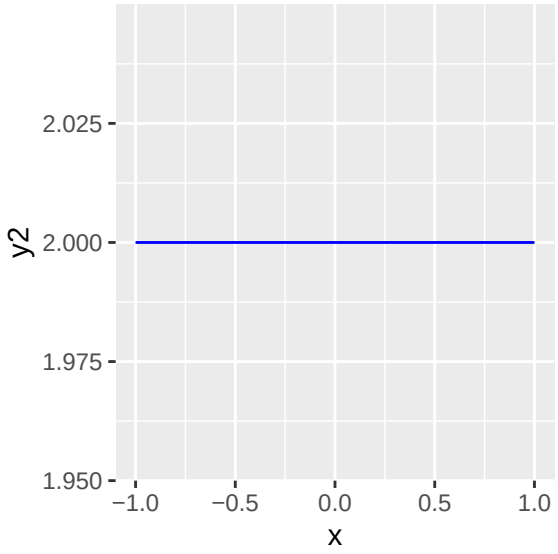
```
p <- ggplot(data, aes(x = x, y = y)) +
  geom_line()
print(p)
```




```
q <- ggplot(data, aes(x = x, y = y1)) +
  geom_line(colour = "red")
print(q)
```



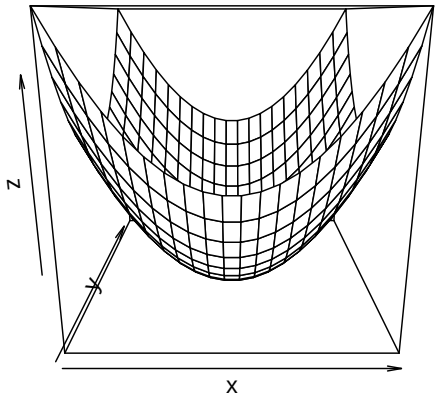
```
r <- ggplot(data, aes(x = x, y = y2)) +
  geom_line(colour = "blue")
print(r)
```



2.2 Partial Derivatives

If the expression is having more than one independent variable, we can calculate differentiation with respect to each of them.

```
f <- expression(x^2 + y^2)

x <- y <- seq(-3, 3, length = 20)
surface <- function(x, y) {
  eval(f)
}
z <- outer(x, y, surface)
persp(x, y, z)
```



Differentiate with respect to `x`

```
D(f, "x")
```

```
## 2 * x
```

Differentiate with respect to `y`

```
D(f, "y")
```

```
## 2 * y
```

Chapter 3

Statistical Distributions

- Density, cumulative distribution function, quantile function and random variate generation for many standard probability distributions are available in the stats package.

  - dxxx : functions for the density/mass function,
  - pxxx : cumulative distribution function
  - qxxx : quantile function
  - rxxx : random variable generation.

3.1 pxxx

Cumulative distribution function (lower tail probability)

Example : Standard normal distribution

- `pnorm(value-of-x-axis)` or `pnorm(quantile)`

- `pnorm(0) = 0.5` (the area under the standard normal curve to the left of zero).

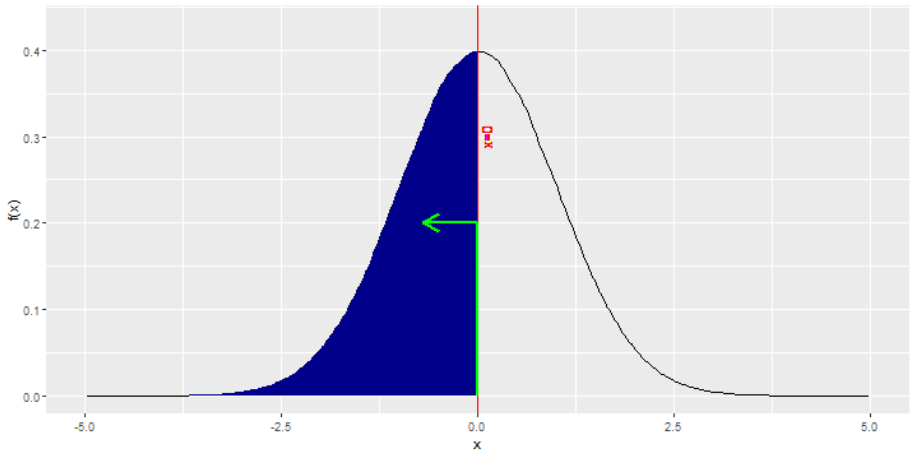


Figure 3.1: Standard normal distribution

- `pnorm(1.281552) = 0.9000` (the area under the standard normal curve to the left of 1.281).

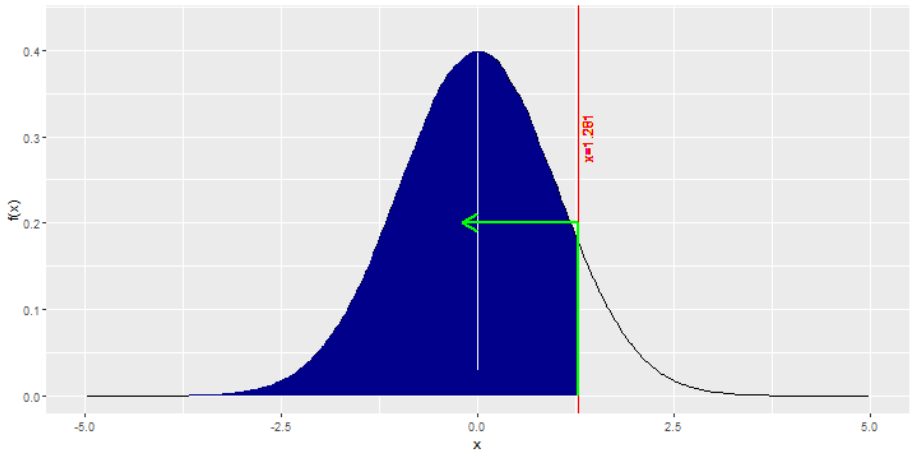


Figure 3.2: Standard normal distribution

- The `pnorm` function also takes the argument `lower.tail`. If `lower.tail` is set equal to `FALSE` then `pnorm` returns the upper tail probability (*the integral from q to ∞ of the pdf*) of the normal distribution.

- Note that

```r
pnorm(1.281552)
```

```
## [1] 0.9000001
```

```r
pnorm(1.281552, lower.tail = TRUE)
```

```
## [1] 0.9000001
```

```r
pnorm(1.281552, lower.tail = FALSE)
```

```
## [1] 0.09999992
```

```r
1-pnorm(1.281552, lower.tail = TRUE)
```

```
## [1] 0.09999992
```

3.2 qxxx

Example : Standard normal distribution

The `qnorm` function is simply the inverse of the cdf, which you can also think of as the inverse of `pnorm`!

- `qnorm(probability)`
- `qnorm(0.5) = 0` (0 is the 50th percentile of the standard normal distribution)

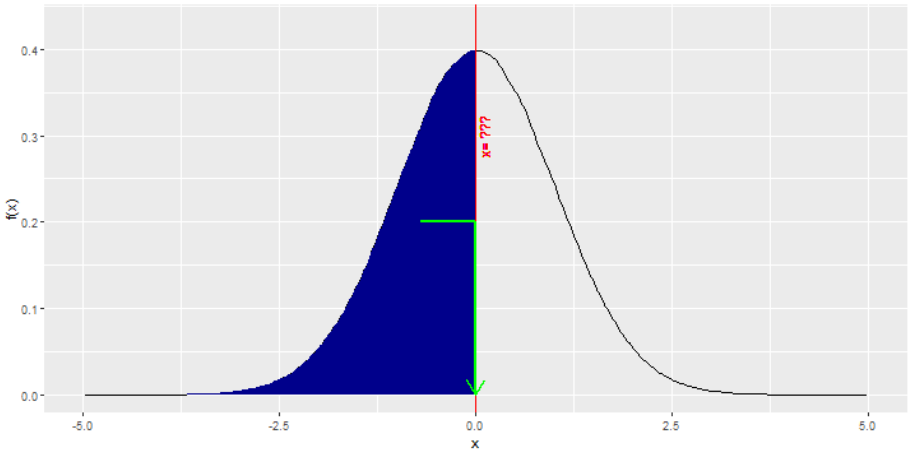


Figure 3.3: Standard normal distribution

```r
qnorm(0.5)
```

```
## [1] 0
```

```r
qnorm(0.9)
```

```
## [1] 1.281552
```

```r
qnorm(0.1, lower.tail = FALSE)
```

```
## [1] 1.281552
```

```r
qnorm(0.9,  lower.tail = FALSE)
```

```
## [1] -1.281552
```

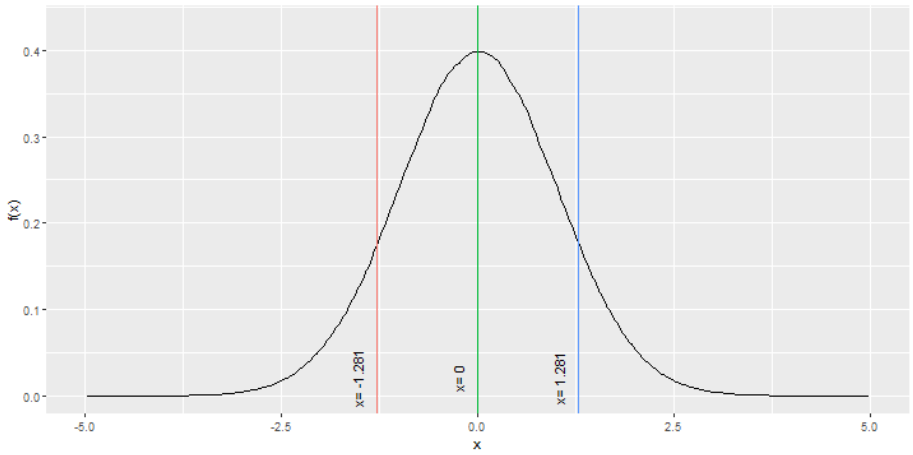


Figure 3.4: Standard normal distribution

3.3 dxxx

Example : Standard normal distribution

The function `dnorm` returns the value of the probability density function for the normal distribution given parameters for x, μ, and σ.

- `dnorm(0) == 1/sqrt(2*pi)`

If $Z \sim N(0,1)$, then

$$\phi_Z(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}; \ -\infty < z < \infty$$

$$\phi_Z(0) = \frac{1}{\sqrt{2\pi}} = 0.3989423$$

```
dnorm(0)
```

```
## [1] 0.3989423
```

```
1/sqrt(2*pi)
```

```
## [1] 0.3989423
```

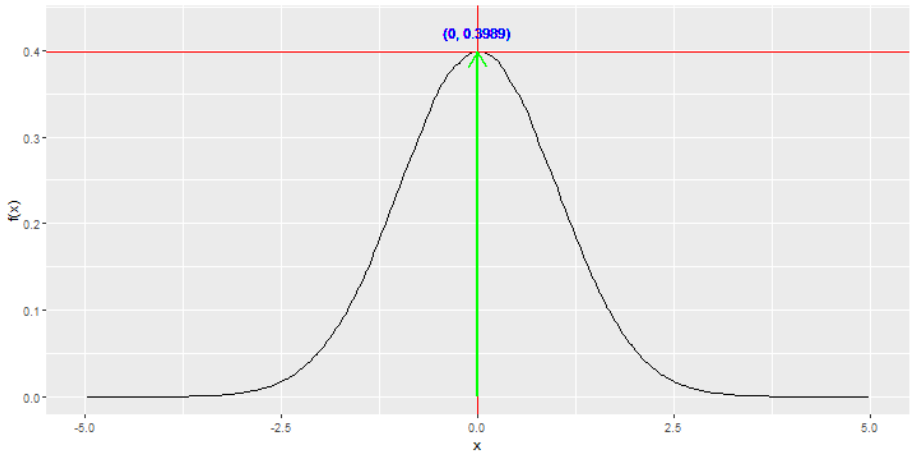


Figure 3.5: Standard normal distribution

$$\phi_Z(1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}} = 0.3989423$$

```
dnorm(1)
```

```
## [1] 0.2419707
```

```
(1/sqrt(2*pi)) * exp(-1/2)
```

```
## [1] 0.2419707
```

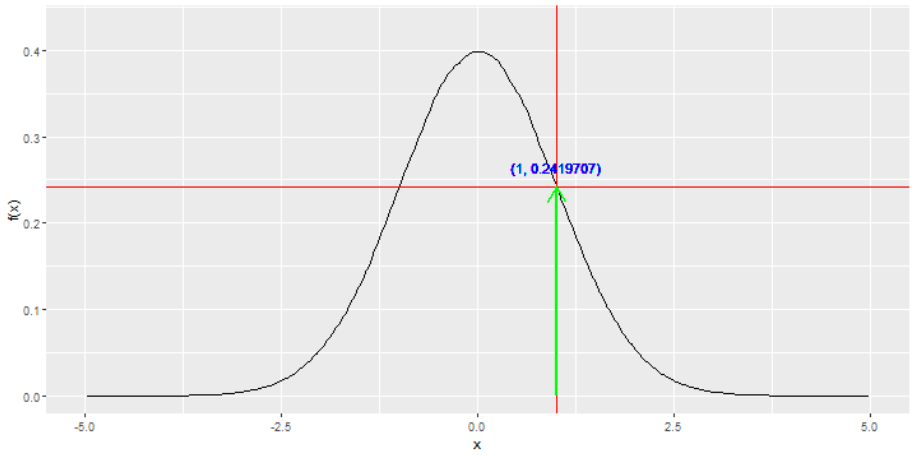


Figure 3.6: Standard normal distribution

3.4 rxxx

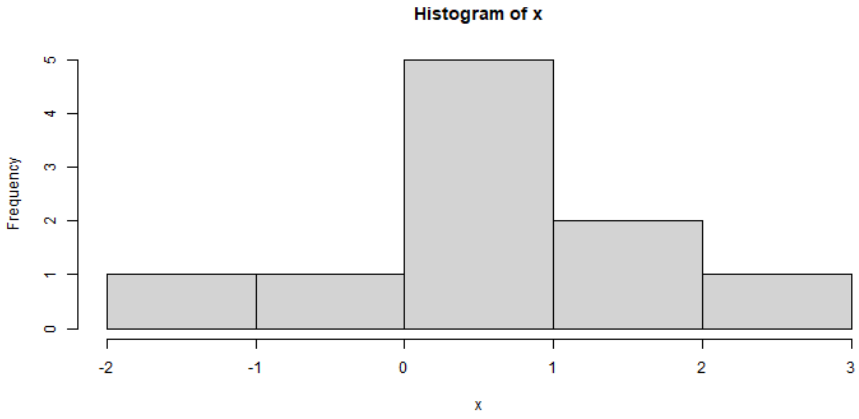
Example : Standard normal distribution

`rnorm(100)` generates 100 random deviates from a standard normal distribution.

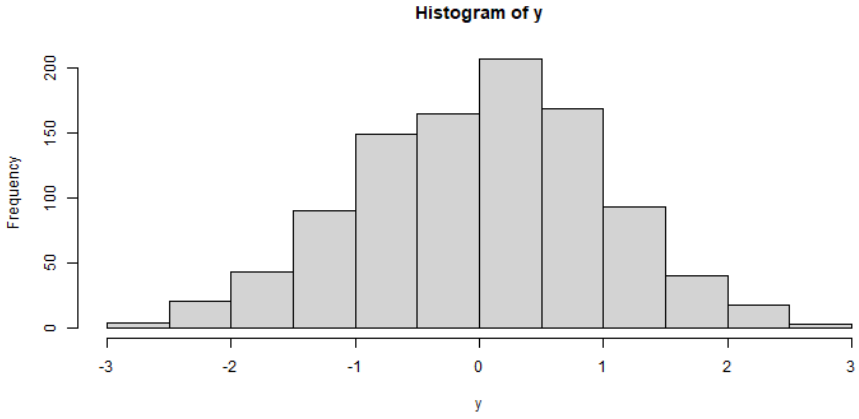
```
x <- rnorm(10)
x
```

```
##  [1] -1.0080707  1.3549394 -0.4689749  1.4681936  0.4425564  0.1462031
##  [7]  0.1715031  0.5925072  2.7647493  0.6192188
```

```
hist(x)
```




```
y<- rnorm(1000)
hist(y)
```



3.5 Distributions in R

Distribution	R name	Additional arguments
beta	`beta`	`shape1, shape 2, ncp`
binomial	`binom`	`size, prob`
Cauchy	`cauchy`	`location, scale`
chi-squared	`chisq`	`df, ncp`

Distribution	R name	Additional arguments
exponential	`exp`	`rate`
F	`f`	`df1, df2, ncp`
gamma	`gamma`	`shape, scale`
geometric	`geom`	`prob`
hypergeometric	`hyper`	`m, n, k`
log-normal	`lnorm`	`meanlog, sdlog`
logistic	`logis`	`location, scale`
negative binomial	`nbinom`	`size, prob`
normal	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
signed rank	`signrank`	`n`
Student's t	`t`	`df, ncp`
uniform	`unif`	`min, max`
Weibull	`weibull`	`shape, scale`
Wilcoxon	`wilcox`	`m, n`

3.6 Exercise

3.6.1 Normal

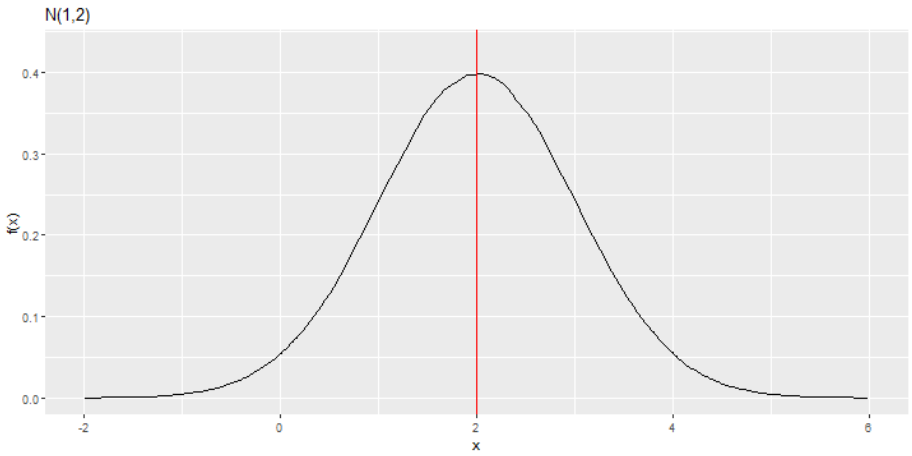


Figure 3.7: Standard normal distribution

```r
# ?dnorm - Help page

dnorm(2, mean = 2, sd = 1, log = FALSE)
```

```
pnorm(2, mean = 2, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(0.5, mean = 2, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n=10, mean = 2, sd = 1)
```

```
# ?dnorm - Help page

dnorm(2, mean = 2, sd = 1, log = FALSE)
```

```
## [1] 0.3989423
```

```
pnorm(2, mean = 2, sd = 1, lower.tail = TRUE, log.p = FALSE)
```

```
## [1] 0.5
```

```
qnorm(0.5, mean = 2, sd = 1, lower.tail = TRUE, log.p = FALSE)
```

```
## [1] 2
```

```
rnorm(n=10, mean = 2, sd = 1)
```

```
##  [1] 0.9919293 3.3549394 1.5310251 3.4681936 2.4425564 2.1462031 2.1715031
##  [8] 2.5925072 4.7647493 2.6192188
```

3.6.2 Gamma

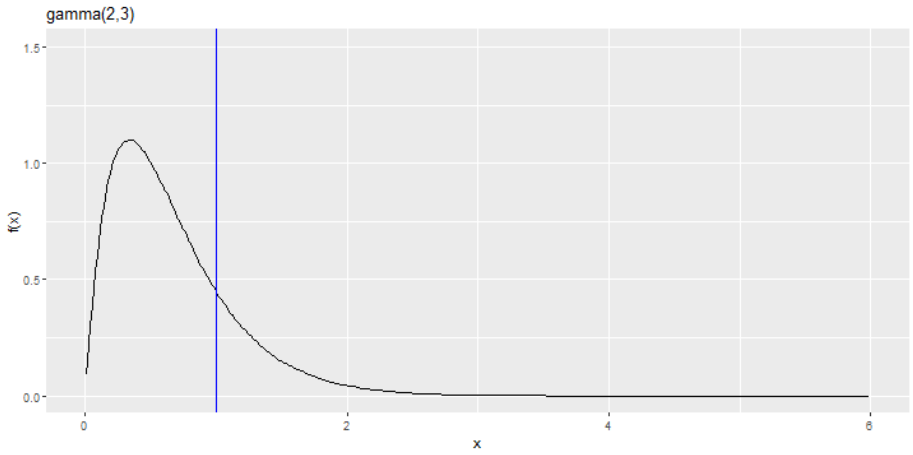


Figure 3.8: Standard normal distribution

```
dgamma(1, shape =2, scale = 1/3, log = FALSE)
pgamma(1, shape =2, scale = 1/3, lower.tail = TRUE,
       log.p = FALSE)
qgamma(0.8, shape = 2,  scale = 1/3, lower.tail = TRUE,
       log.p = FALSE)
rgamma(10, shape =2 , scale = 1/3)
```

```
dgamma(1, shape =2, scale = 1/3, log = FALSE)
```

```
## [1] 0.4480836
```

```
pgamma(1, shape =2, scale = 1/3, lower.tail = TRUE,
       log.p = FALSE)
```

```
## [1] 0.8008517
```

```
qgamma(0.8, shape = 2,  scale = 1/3, lower.tail = TRUE,
       log.p = FALSE)
```

```
## [1] 0.9981028
```

```
rgamma(10, shape =2 , scale = 1/3)
```

```
##  [1] 0.3117027 0.5663228 1.8567817 0.1403567 0.7776307 0.8252981 1.4063366
##  [8] 0.7292312 0.9434481 0.9001169
```

3.7 References

Discrete http://www.r-tutor.com/elementary-statistics/probability-distributions

https://rstudio-pubs-static.s3.amazonaws.com/100906_8e3a32dd11c14b839468db756cee7400.html

Linear Regression in R

In many practical situations we want to identify various types of relationships between variables. **Regression analysis** is a statistical technique for investigating and **modeling the relationship between variables**.

Linear regression attempts to model the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables) by fitting a **linear equation** to observed data. The case of one explanatory variable is called **simple linear regression.**

3.8 Loading required R packages

The following R packages are required for this chapter:

- `ggplot2` for data visualization
- `datarium` data bank for statistical analysis and visualization
- The `ggpairs()` function of the `GGally` package allows to build a great scatterplot matrix.

```
library(ggplot2)
library(datarium)
## To learn more about the dataset
# ?marketing

library(GGally)
```

- We are going to use the marketing data set available in `datarium` package, which contains the impact of the amount of money spent on three advertising medias (youtube, Facebook and newspaper) on sales.

3.9 Load Data set

```
data(marketing)
head(marketing)
```

```
##   youtube facebook newspaper sales
## 1  276.12    45.36     83.04 26.52
## 2   53.40    47.16     54.12 12.48
## 3   20.64    55.08     83.16 11.16
## 4  181.80    49.56     70.20 22.20
## 5  216.96    12.96     70.08 15.48
## 6   10.44    58.68     90.00  8.64
```

```
summary(marketing)
```

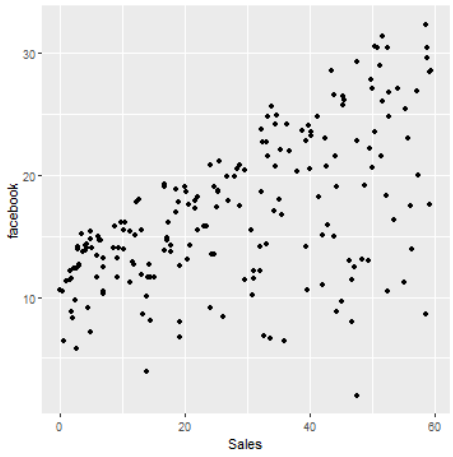
```
##     youtube          facebook       newspaper          sales      
##  Min.   :  0.84   Min.   : 0.00   Min.   :  0.36   Min.   : 1.92  
##  1st Qu.: 89.25   1st Qu.:11.97   1st Qu.: 15.30   1st Qu.:12.45  
##  Median :179.70   Median :27.48   Median : 30.90   Median :15.48  
##  Mean   :176.45   Mean   :27.92   Mean   : 36.66   Mean   :16.83  
##  3rd Qu.:262.59   3rd Qu.:43.83   3rd Qu.: 54.12   3rd Qu.:20.88  
##  Max.   :355.68   Max.   :59.52   Max.   :136.80   Max.   :32.40
```

3.10 Explore data

3.10.1 Scatterplot

- Usually, the first step in regression analysis is to construct a scatter plot (or scatter matrix).

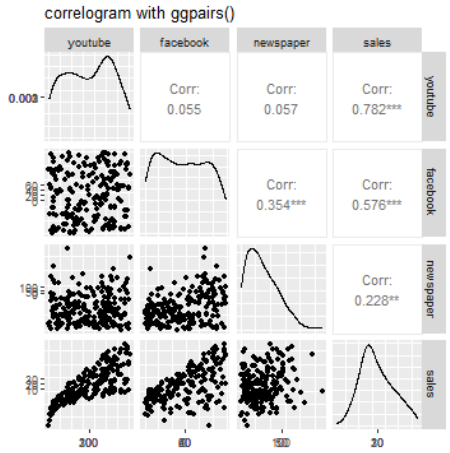
```
ggplot(data = marketing, aes(x = facebook, y = sales)) +
  geom_point() +
  theme(aspect.ratio = 1)+
  xlab("Sales")+
  ylab("facebook")
```



3.10.2 Scatterplot matrix

- The `ggpairs()` function of the `GGally` package allows to build a great scatterplot matrix.
- Scatterplots of each pair of numeric variable are drawn on the left part of the figure.
- Pearson correlation is displayed on the right.
- Variable distribution is available on the diagonal.

```
# Check correlations (as scatterplots), distribution and print corrleation coefficient

ggpairs(marketing, title="correlogram with ggpairs()") +
   theme(aspect.ratio = 1)
```



- The term *corr* is the Pearson product-moment correlation coefficient (r).
- It is a measure of the **linear** correlation of two variables.
- It is a number that ranges from -1 to 0 to +1, representing th strength of the linear relationship between the variables.
- An r value of +1 denotes a perfect **linear** positive relationship between two variables.
- An r value of -1 denotes a perfect **linear** negative relationship between two variables, which indicates an inverse relationship between two variables: as one variable gets larger, the other gets smaller.
- An r value of 0 means no **linear** relationship is present between the two variables (There can be a non-linear relationship.)

3.11 Simple Linear Regression

- The most elementary regression model is called **simple linear regression**.
- The variable to be predicted is called the *dependent variable* an is denoted by y.
- The *predictor* is called the *independent variable* or *explanatory variable* and is denoted by x
- In simple *linear* regression analysis, only a strait-line relationship between two variables is examined.

```
#lm(y ~ x)
reg <- lm(sales ~ facebook,  data = marketing)

reg
```

```
##
## Call:
## lm(formula = sales ~ facebook, data = marketing)
##
## Coefficients:
## (Intercept)     facebook
##     11.1740       0.2025
```

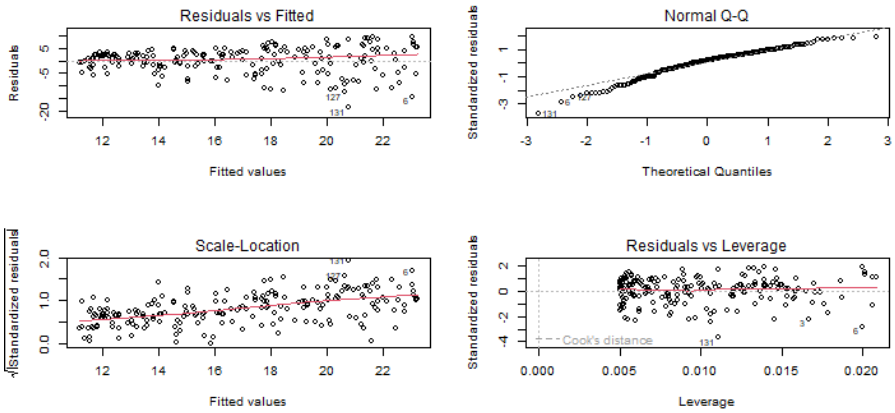
```
summary(reg)
```

```
##
## Call:
## lm(formula = sales ~ facebook, data = marketing)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.8766  -2.5589   0.9248   3.3330   9.8173
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.17397    0.67548  16.542   <2e-16 ***
## facebook     0.20250    0.02041   9.921   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.13 on 198 degrees of freedom
## Multiple R-squared:  0.332,  Adjusted R-squared:  0.3287
## F-statistic: 98.42 on 1 and 198 DF,  p-value: < 2.2e-16
```

3.11.1 Residual Analysis

```
par(mfrow = c(2,2))
plot(reg)
```



- You want these plots to display random residuals (no patterns) that are uncorrelated and uniform.
- Generally speaking, if you see patterns in the residuals, your model has a problem, and you might not be able to trust the results.

- Heteroscedasticity produces a distinctive fan or cone shape in residual plots.
- To check for heteroscedasticity, you need to assess the residuals by fitted value plots specifically.
- Typically, the telltale pattern for heteroscedasticity is that as the fitted values increases, the variance of the residuals also increases.

Read more about residual analysis:

- Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to linear regression analysis (Vol. 821). John Wiley & Sons.

3.12 Multiple Linear Regression

- Regression models with more than one independent variable can be explored by using multiple regression models.
- We want to build a model for estimating sales based on the advertising budget invested in youtube, facebook and newspaper, as follow:

$$sales = \beta_0 + \beta_1 * youtube + \beta_2 * facebook + \beta_3 * newspaper$$

You can compute the model coefficients in R as follow:

```
m_reg <- lm(sales ~ youtube + facebook + newspaper, data = marketing)
summary(m_reg)
```

```
##
## Call:
## lm(formula = sales ~ youtube + facebook + newspaper, data = marketing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5932  -1.0690   0.2902   1.4272   3.3951
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.526667   0.374290   9.422   <2e-16 ***
## youtube      0.045765   0.001395  32.809   <2e-16 ***
## facebook     0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.023 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

3.12.1 How to test if your linear model has a good fit?

- Most common value to check how good is your model is the coefficient of determinations or R^2
- As we have seen in simple linear regression, the overall quality of the model can be assessed by examining the R-squared (R^2).
- $R^2 = 0.05602$ means that the model explains only 5% of the data variability.
- R^2 represents the proportion of variance, in the outcome variable y, that may be predicted by knowing the value of the x variables.
- An R^2 value close to 1 indicates that the model explains a large portion of the variance in the outcome variable.
- The second one has an R^2 of 0.89, and the model can explain 89% of the total variability.
- In the regression summary output notice that there's two different R^2, one multiple and one adjusted.
- One problem with this R^2 is that it will always increase when more variables are added to the model, even if those variables are only weakly associated with the response (i.e. these variables don't add anything to your predictions)
- For this reason, the **adjusted** R^2 is probably better to look at if you are adding more than one variable to the model, since it only increases if it reduces the overall error of the predictions.
- The adjustment in the **Adjusted R Square** value in the summary output is a correction for the number of x variables included in the prediction model.

NOTE

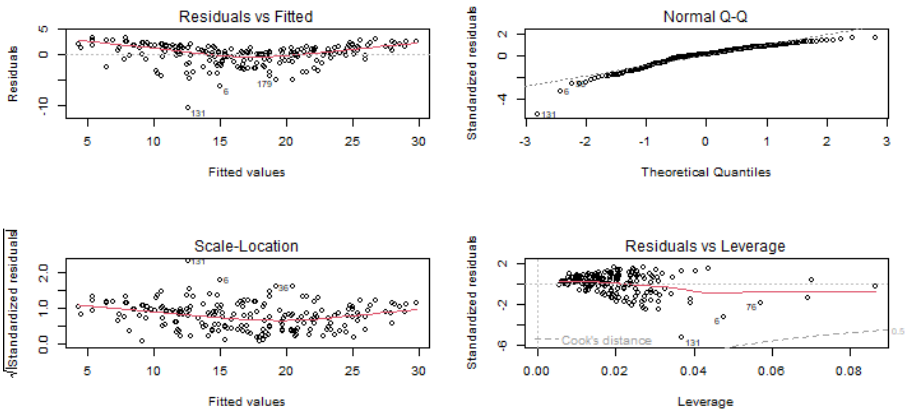
In our example, with youtube, newspaper and facebook predictor variables, the adjusted $R^2 = 0.89$, meaning that "89% of the variance in the measure of sales can be predicted by youtube, newspaper and facebook advertising budgets.

This model is better than the simple linear model with only facebook, which had an adjusted R^2 of 0.05.

3.12.2 Don't forget to look at the residuals

- You can have a pretty good R^2 in your model, but let's not rush to conclusions here.
- Ideally, when you plot the residuals, they should look random. Otherwise, it means that maybe there is a hidden pattern that the linear model is not considering.

```
par(mfrow = c(2,2))
plot(m_reg)
```



3.13 Prediction for new data set

- Using the above model, we can predict the sales for a new advertising budget.

```
new.budget <- data.frame(
  youtube = c(150, 200, 100),
  facebook = c( 150, 100, 200),
  newspaper = c(0,0,0)
)
new.budget
```

```
##   youtube facebook newspaper
## 1     150      150         0
## 2     200      100         0
## 3     100      200         0
```

```
predict(m_reg, new.budget)
```

```
##        1        2        3
## 38.67087 31.53260 45.80914
```

Simple Linear regression

```
new.facebook <- data.frame(
  facebook = c(50, 100, 200)
)

predict(reg, new.facebook)
```

```
##        1        2        3
## 21.29875 31.42354 51.67312
```

3.14 References

https://www.scribbr.com/statistics/linear-regression-in-r/

https://statisticsbyjim.com/regression/heteroscedasticity-regression/

hypothesis testing part

http://www.sthda.com/english/articles/40-regression-analysis/168-multiple-linear-regression-in-r/

Chapter 4

mtcars

Cars Dataset: https://www.rpubs.com/dksmith01/cars

mtcars descriptive analysis: https://rstudio-pubs-static.s3.amazonaws.com/481654_883a4b47c9b244d4859dd1db235f0165.html

mtcars: regression analysis: https://rstudio-pubs-static.s3.amazonaws.com/156794_e6ddaf8ca4ac4c2882a91dfa2ca8e3e8.html

ANOVA: https://medium.com/humansystemsdata/anova-with-mtcars-54dd6344c4e1

two way anova: https://www.geeksforgeeks.org/anova-test-in-r-programming/

chisquare : https://rpubs.com/daheza/assignment3

https://datascienceplus.com/chi-squared-test-in-r/

descriptive statistics: http://www.sthda.com/english/wiki/r-built-in-data-sets

http://www.sthda.com/english/wiki/r-built-in-data-sets

explore mtcars: https://cran.r-project.org/web/packages/explore/vignettes/explore_mtcars.html

descrpive statistics: https://rpubs.com/rpubsNovice/459602

mpg: https://www.sandicliffe.co.uk/blog/what-is-mpg#:~:text=MPG%20is%20an%20abbreviation%20for,a%20car%

horseower: https://www.quora.com/Why-does-horsepower-matter-Does-it-mean-better-MPG-or-something

t-test and Chi Squared: https://rpubs.com/daheza/assignment3

code mtcars data : http://psych.colorado.edu/~lharvey/P4165/P4165_2018_1_Spring/Material_2018_Spring_PSYC4165/Class_Handout_pdfs_2018_Spring_PSYC4165/R%20guide_JMS%20Revised_1_11_18.pdf

page 15

Chapter 5

Introduction to R

5.1 Installing R

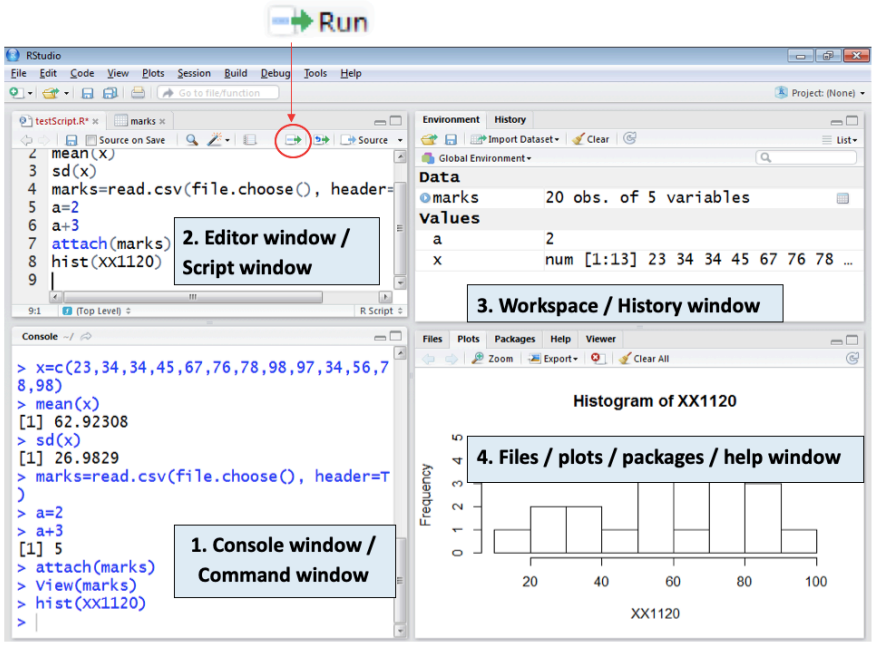
- **Step 1:** First download R freely from the Comprehensive R Archive Network (CRAN) https://cran.r-project.org/. (At the moment of writing, R 4.3.1 is the latest version. Choose the most recent one.)
- **Step 2:** Then install R Studio's IDE (stands for integrated development environment), a powerful user interface for R from https://rstudio.com/products/rstudio/download/. Get the Open Source Edition of RStudio Desktop. RStudio allows you to run R in a more user-friendly environment.
  - You need to install **both** R and Rstudio to use RStudio.
  - If you have a pre-existing installation of R and/or RStudio, I highly recommend that you re install both and get as current as possible.
- **Step 3:** Then open **Rstudio**.

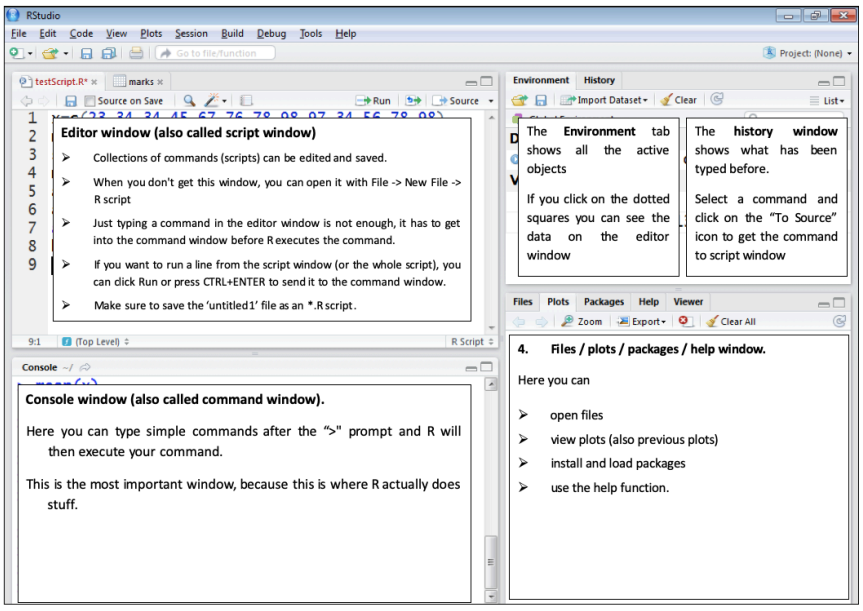
5.2 RStudio layout

The RStudio interface consists of four windows (see Figure 1 and 2).

1. Bottom left: console window (also called command window). **This is where you type and run all your R commands**
2. Top left: editor window (also called script window).
3. Top right: workspace / history window.

4. Bottom right: Files / plots / packages / help window.





Now you are familiar with the layout. Let's begin with R basics.

5.3 Installing an R Package

- The primary location for obtaining R packages is CRAN
- Packages can be installed with the `install.packages()` function in R
- To install a single package, pass the name of the package to the `install.packages()` function as the first argument

The following the code installs the `tidyverse` package from CRAN

```
install.packages("tidyverse")
```

- This command downloads the `tidyverse` package from CRAN and installs it on your computer
- Any packages on which this package depends will also be downloaded and installed
- **Installing the tidyverse package could take several minutes. You only need to do this once**.

5.4 Loading an R Packages

- Installing a package does not make it immediately available to you in R; you must load the package
- The `library()` function is used to load packages into R
- The following code is used to load the tidyverse package into R
- **NOTE:** Do not put the package name in quotes!

```
library(tidyverse)
```

- Some packages produce messages when they are loaded (but some don't)

5.5 Getting started with R

An Introduction to R: https://cran.r-project.org/doc/manuals/R-intro.pdf