Chapter 2

Basic R Language Tools

Integrand:

```
integrand <- function(x) 1/((x + 1) * sqrt(x))

integrate(integrand, lower = 0, upper = Inf)
```

```
3.141593 with absolute error < 2.7e-05
```

Vectorization:

```
x <- c(1, 5, 10, 15, 20)
x
```

```
[1]  1  5 10 15 20
```

```
x2 <- 2 * x
x2
```

```
[1]  2 10 20 30 40
```

```
x3 <- x^2
x3
```

```
[1]   1  25 100 225 400
```

```
x4 <- x / x2
x4
```

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

```
x5 <- round(x * (x/2) ^ 3.5 + sqrt(x4), 3)
x5
```

```
[1]     0.795   124.234  2795.792 17330.991 63246.260
```

```
x6 <- round(c(x2[2:4], x3[1:2], x5[4]), 2)
x6
```

```
[1]    10.00    20.00    30.00     1.00    25.00 17330.99
```

Matrix:

```
my_matrix <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)
my_matrix
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
my_matrix <- matrix(seq(1, 6), nrow = 2, ncol = 3, byrow = T)
my_matrix
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Attributes:

```
dimnames(my_matrix) <- list(c("one", "hello"), c("column1", "column2", "c3"))

my_matrix
```

```
      column1 column2 c3
one         1       2  3
hello       4       5  6
```

```
attributes(my_matrix)
```

```
$dim
[1] 2 3

$dimnames
$dimnames[[1]]
[1] "one"   "hello"

$dimnames[[2]]
[1] "column1" "column2" "c3"
```

```
ans <- my_matrix[1, 3]

new_matrix_1 <- my_matrix * my_matrix
new_matrix_1
```

```
      column1 column2 c3
one         1       4  9
hello      16      25 36
```

```
new_matrix_2 <- sqrt(my_matrix)
new_matrix_2
```

```
      column1  column2       c3
one         1 1.414214 1.732051
hello       2 2.236068 2.449490
```

```
mat1 <- matrix(rnorm(1000), nrow = 100)
round(mat1[1:5, 2:6], 3)
```

```
       [,1]   [,2]   [,3]   [,4]   [,5]
```

```
[1,] -0.035  0.432 -0.899  0.926  1.944
[2,] -0.934  0.550 -0.721 -0.160  0.447
[3,] -0.651 -0.460  0.173 -2.141  0.212
[4,] -0.424 -0.058  0.674 -0.239 -0.444
[5,]  0.349 -0.499  1.616  0.439 -0.735
```

```
mat2 <- mat1[1:25,] ^2
mat2
```

```
            [,1]         [,2]         [,3]         [,4]         [,5]
 [1,] 2.32702384 0.0012409972 0.1867416740 0.8088082290 0.8575203465
 [2,] 1.48214615 0.8720886948 0.3027137405 0.5205266949 0.0256621437
 [3,] 2.67889817 0.4235997733 0.2112639485 0.0298391134 4.5835938463
 [4,] 2.37526725 0.1795313437 0.0033994916 0.4541919559 0.0570071167
 [5,] 0.60240947 0.1216769068 0.2490498739 2.6125430345 0.1928678249
 [6,] 0.74131636 0.2315586113 0.4362031285 0.2147558509 0.5425297975
 [7,] 2.35623876 0.0001751577 0.0026102330 0.0954432152 1.6538415728
 [8,] 0.02299588 4.6563391169 0.0110455552 4.2533799873 2.8893650782
 [9,] 0.41781075 3.0751800910 0.3450875288 1.4483114844 1.4194936702
[10,] 1.35424689 2.4805526789 0.0932965390 0.0334452911 0.4962463958
[11,] 3.45589070 0.3146283138 4.0624508192 0.5493103315 0.0575349823
[12,] 4.77300455 1.7735895861 1.5642980066 0.3988636875 0.9411470413
[13,] 1.60468214 0.1218036426 0.4701503112 0.4090860536 2.4894577733
[14,] 1.81674123 1.1230149116 1.0729431236 1.4881118783 1.5384664671
[15,] 0.14928297 0.3109956713 0.0187200919 0.8685841968 0.1166552799
[16,] 0.09623634 0.0013530686 1.4696165656 0.0098279395 0.3570726463
[17,] 1.32060774 0.4381422787 0.0020789469 0.1812285441 1.2422759807
[18,] 0.95141850 1.9446739115 2.1730648832 1.5235838247 0.0003737362
[19,] 0.07754080 0.3456154521 1.2113178087 0.2077546140 0.0451021176
[20,] 0.23521739 1.8375478760 3.7365402314 1.4703395602 0.7541992149
[21,] 2.52038308 0.0735857470 0.5664942500 4.1798952254 0.0074295408
[22,] 3.44094107 0.3021494855 0.0438107308 0.0004239004 0.0015500054
[23,] 0.07434312 0.5333252244 0.0124338245 0.7226908601 0.0933026775
[24,] 0.55194019 0.0445769677 0.0731597173 0.0115736956 0.2055763783
[25,] 0.73250090 0.3427587984 0.0006946125 0.8973704412 0.0035949450
             [,6]         [,7]         [,8]         [,9]        [,10]
 [1,] 3.780705973 1.0420511348 0.0008791289 0.0002794511 3.297291e-01
 [2,] 0.199369750 0.7561384659 2.6769959954 3.1452750926 2.112898e-01
 [3,] 0.045080381 0.3580636174 1.6687153922 1.5324578084 1.662378e-01
 [4,] 0.197526868 1.5195774142 0.0092784962 1.0017141819 2.747619e-01
 [5,] 0.539648004 0.0254886317 0.2449509746 0.2384537479 1.945603e+00
 [6,] 1.411827161 0.0935194387 5.1983197170 2.1815145182 7.087378e-01
 [7,] 0.166032180 2.1311579726 1.3366184956 4.3125264503 1.189803e-04
 [8,] 1.327074820 0.1097722742 0.2426777646 0.2015070994 4.213056e-01
 [9,] 0.220289816 0.0037342045 2.7634690920 1.2480057477 1.455305e-02
[10,] 1.765278616 0.0583017696 0.3093319113 0.6350155274 1.738019e-02
```

```
[11,] 5.770743465 0.0001492057 0.0262889904 1.4531773027 4.248264e-05
[12,] 2.693062774 1.1301176306 1.0577078042 0.0547979392 1.531757e-01
[13,] 0.002401549 0.0065016285 6.6150303810 0.0022143870 1.090037e+00
[14,] 0.001885671 0.1541352585 4.1165402685 0.0006253756 3.748754e-01
[15,] 3.275742579 0.3907939319 0.3285921460 0.0490261904 9.957565e-01
[16,] 0.009688609 7.4301363921 0.0017935930 2.5040099964 2.192945e+00
[17,] 2.279738290 0.0313769257 0.5639957411 0.1771762344 2.513435e-03
[18,] 1.471411160 0.0001465288 2.3969223664 0.1765802883 1.035325e+00
[19,] 0.462479457 0.0297905909 0.0541835146 0.9487615308 7.869901e-01
[20,] 6.772901647 1.1649848633 0.0331538402 0.4561549033 4.940424e+00
[21,] 5.359905669 0.0652074980 0.0439427263 0.1287992033 6.995775e-02
[22,] 0.353516848 0.8608651928 1.3427409373 1.9313425814 4.153003e-01
[23,] 0.096335851 0.0512632486 0.2292913902 0.0805186810 8.989041e-01
[24,] 0.015884408 1.0618791509 1.4995039117 1.5519379152 3.346021e-01
[25,] 0.095316997 3.8805029430 0.4315778849 0.1124124456 2.028451e-01
```

data.frame:

```r
df <- data.frame(price = c(89.2, 23.2, 21.2),
                 symbol = c("MOT", "AAPL", "IBM"),
                 action = c("Buy", "Sell", "Buy"))
df
```

```
  price symbol action
1  89.2    MOT    Buy
2  23.2   AAPL   Sell
3  21.2    IBM    Buy
```

```r
class(df$symbol)
```

```
[1] "factor"
```

```r
df2 <- data.frame(price = c(89.2, 23.2, 21.2),
                 symbol = c("MOT", "AAPL", "IBM"),
                 action = c("Buy", "Sell", "Buy"),
                 stringsAsFactors = F)
df2
```

```
  price symbol action
1  89.2    MOT    Buy
2  23.2   AAPL   Sell
3  21.2    IBM    Buy
```

```r
class(df2$symbol)
```

```
[1] "character"
```

```r
price <- df[1, 1]
```

```
df3 <- data.frame(col1 = c(1, 2, 3, 4),
                  col2 = c(1, 2, 3, 4))

symbols <- df$symbol

symbols
```

```
[1] MOT  AAPL IBM
Levels: AAPL IBM MOT
```

```
class(symbols)
```

```
[1] "factor"
```

list:

```
my_list <- list(a = c(1, 2, 3, 4, 5),
                b = matrix(1:10, nrow = 2, ncol = 5),
                c = data.frame(price = c(89.3, 98.2, 21.2)),
                stock = c("MOT", "IBM", "CSCO"))

my_list
```

```
$a
[1] 1 2 3 4 5

$b
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10

$c
  price
1  89.3
2  98.2
3  21.2

$stock
[1] "MOT"  "IBM"  "CSCO"
```

```
first_element <- my_list[[1]]

first_element
```

```
[1] 1 2 3 4 5
```

```
class(first_element)
```

```
[1] "numeric"
```

```r
second_element <- my_list[["b"]]

second_element
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

```r
part_of_list <- my_list[c(1, 3)]

part_of_list
```

```
$a
[1] 1 2 3 4 5

$c
  price
1  89.3
2  98.2
3  21.2
```

```r
class(part_of_list)
```

```
[1] "list"
```

```r
size_of_list <- length(my_list)

size_of_list
```

```
[1] 4
```

Env:

```r
env <- new.env()

env[["first"]] <- 5
env[["second"]] <- 6
env$third <- 7

env
```

```
<environment: 0x000000001c952c28>
```

```r
ls(env)
```

```
[1] "first"  "second" "third"
```

```r
get("first", envir = env)
```

```
[1] 5
```

```r
rm("second", envir = env)
```

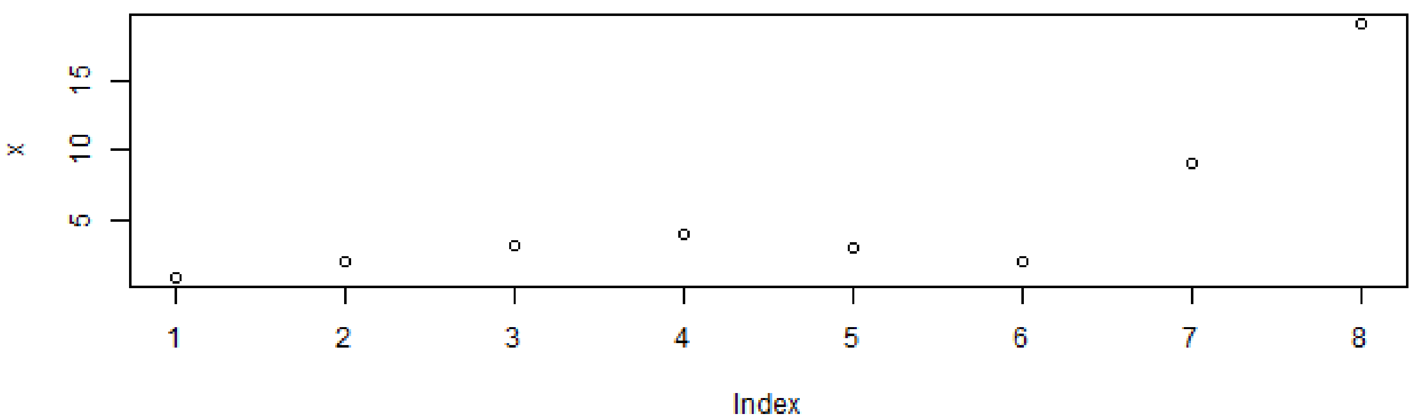
```r
ls(env)
```

```
[1] "first" "third"
```

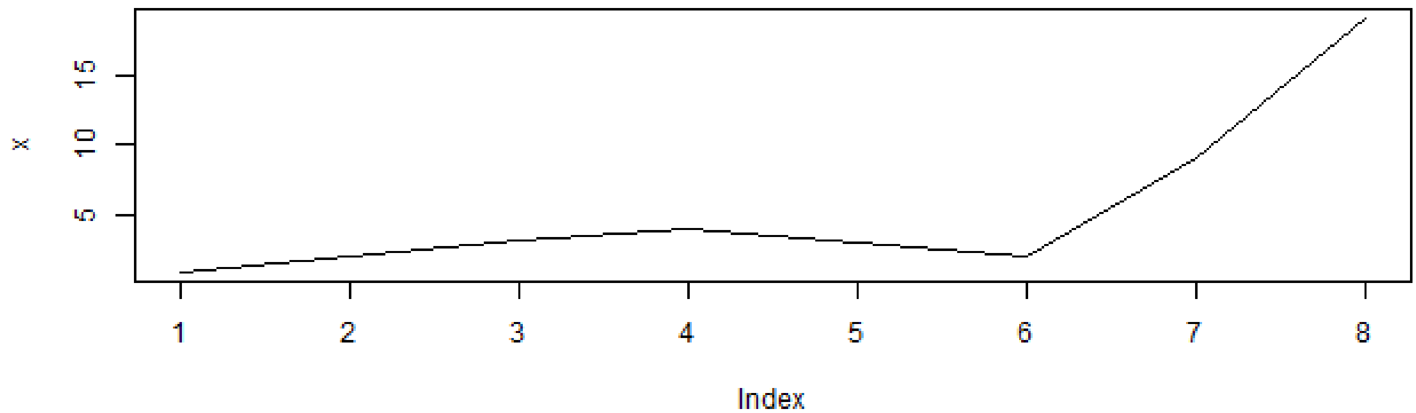
```r
# pass by reference
env_2 <- env
env_2$third <- 42

get("third", envir = env)
```

```
[1] 42
```

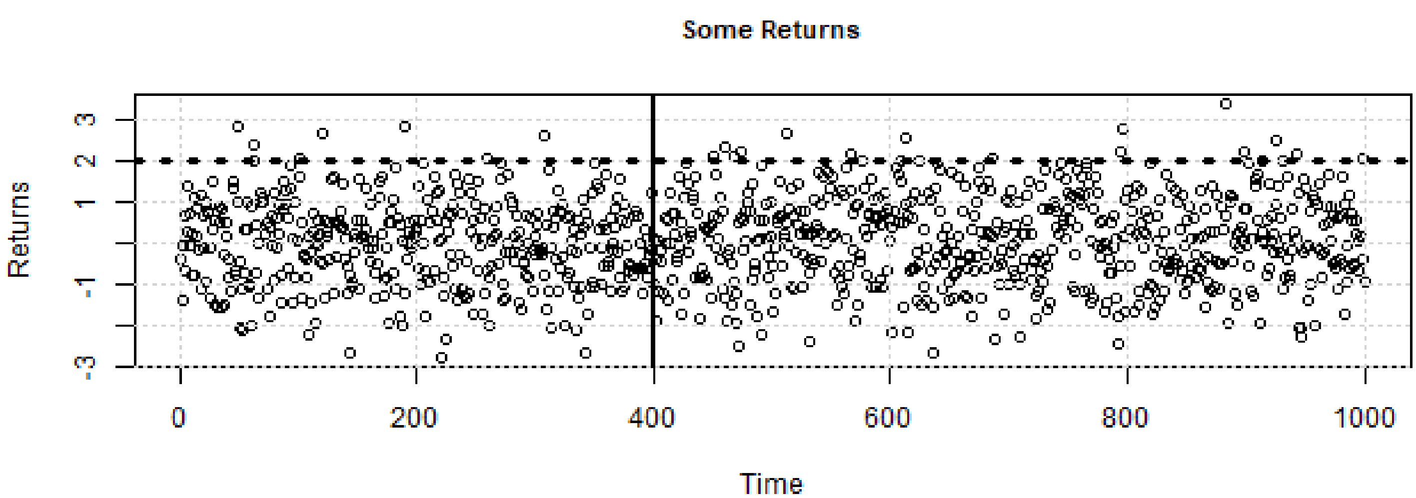
```r
x <- c(1, 2, 3.2, 4, 3, 2.1, 9, 19)
plot(x)
```




```r
plot(x, type = "l")
```

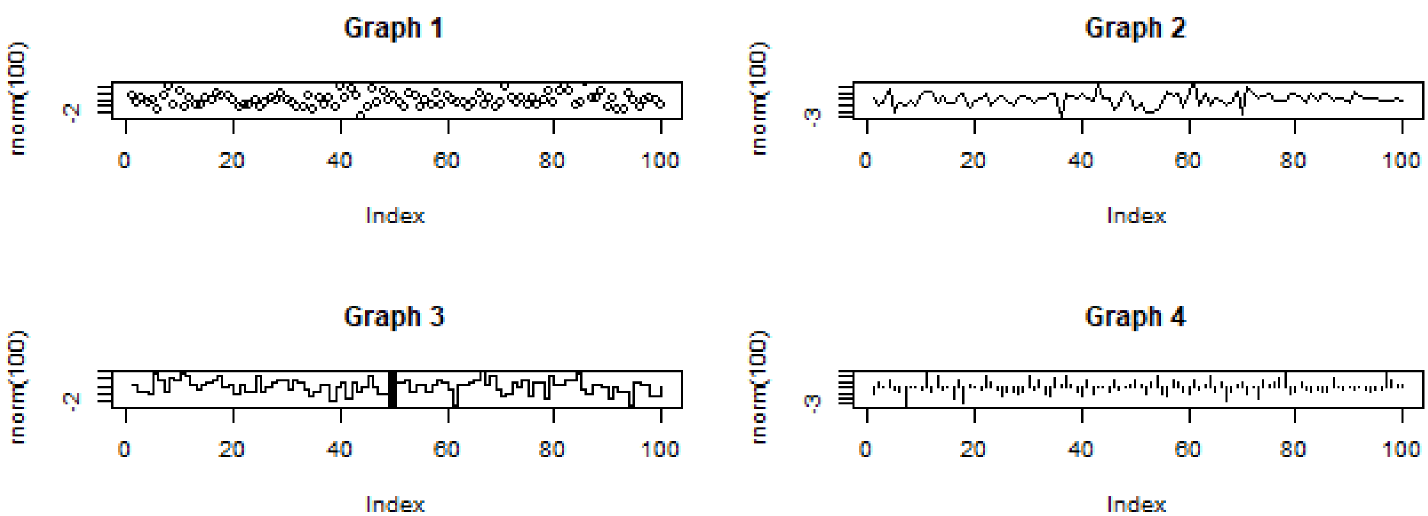



```r
# Setup the canvas
plot(rnorm(1000), main = "Some Returns", cex.main = 0.9,
     xlab = "Time", ylab = "Returns")
# Superimpose a grid
grid()
# Create a few vertical and horizontal lines
abline(v = 400, lwd = 2, lty = 1)
abline(h = 2, lwd = 3, lty = 3)
```



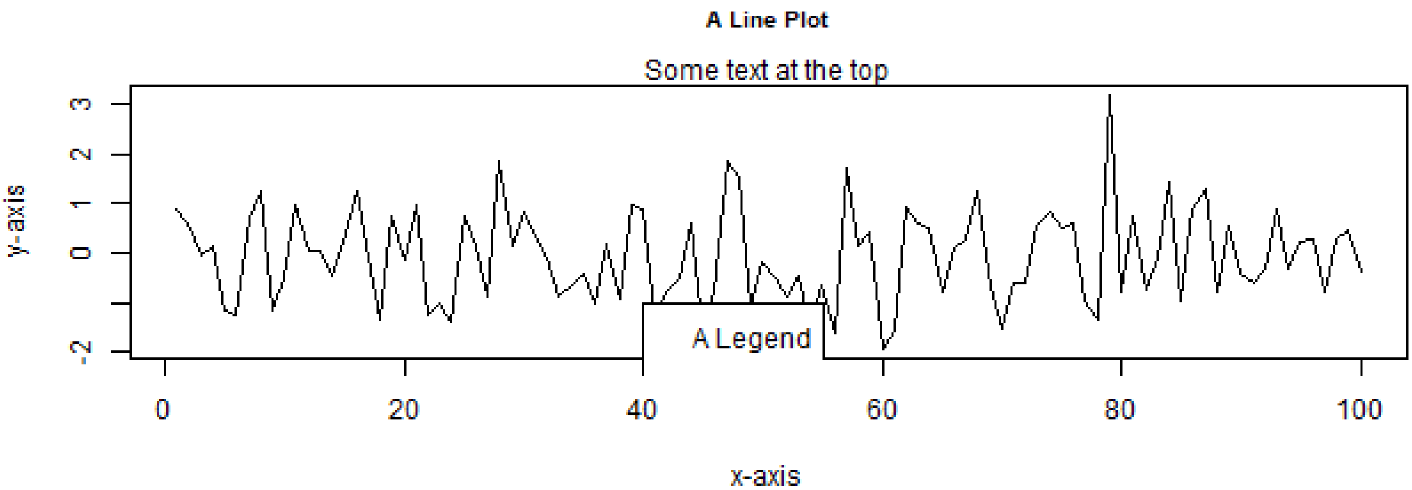

```r
# Create a 2-row, 2-column format
par(mfrow = c(2, 2))

# First plot (points)
plot(rnorm(100), main = "Graph 1")
```

```r
# Second plot (line)
plot(rnorm(100), main = "Graph 2", type = "l")

# Third plot (steps) with a vertical line
plot(rnorm(100), main = "Graph 3", type = "s")
abline(v = 50, lwd = 4)

plot(rnorm(100), type = "h", main = "Graph 4")
```




```r
# reset par
par(mfrow = c(1, 1))
```

```r
plot(rnorm(100), main = "A Line Plot",
     cex.main = 0.8,
     xlab = "x-axis",
     ylab = "y-axis",
     type = "l")

# Extra text
mtext("Some text at the top", side = 3)

# At x = 40, and y = -1 coordinates
legend(40, -1, "A Legend")
```




```r
formals(plot.default)
```

```
$x


$y
NULL

$type
[1] "p"

$xlim
NULL

$ylim
NULL

$log
[1] ""

$main
NULL

$sub
NULL

$xlab
NULL
```

```
$ylab
NULL

$ann
par("ann")

$axes
[1] TRUE

$frame.plot
axes

$panel.first
NULL

$panel.last
NULL

$asp
[1] NA

$xgap.axis
[1] NA

$ygap.axis
[1] NA

$...
```

Functional Programming

Functional:

```r
ans <- sum(1:100)

ans
```

```
[1] 5050
```

Imperative:

```r
answer <- 0
for(i in 1:100){
   answer = answer + i
}
```

```
answer
```

```
[1] 5050
```

Functions

```
# Create 100 standard normals
x <- rnorm(100, mean = 0, sd = 1)

# Find the length of the vector x.
length(x)
```

```
[1] 100
```

```
# Compute the mean of x.
mean(x)
```

```
[1] 0.1795094
```

```
# Compute the standard deviation of x.
sd(x)
```

```
[1] 0.923425
```

```
# Compute thee range (min, max) of a variable.
range(x)
```

```
[1] -2.513281  2.154217
```

```
# Find the sum of all the numbers.
sum(x)
```

```
[1] 17.95094
```

```
# Do a cumulative sum of the values in x.
cumsum(x)
```

```
  [1]  0.7371499  1.3945812  2.9126431  2.7620730  2.6266597  2.8057030
  [7]  4.0464533  5.0957957  6.5439274  7.1619043  7.3060174  7.0888445
 [13]  6.8434952  7.0310633  8.0129890  8.6932272  8.5324497  7.8926287
 [19]  7.9337941  7.5146411  6.9956750  7.3279772  6.2168107  4.6415356
 [25]  3.8463487  2.4959945  2.0432408  2.4908769  2.3413224  1.7426243
 [31]  2.9858976  3.0413348  2.0094078  3.9886215  3.5836472  3.4965739
 [37]  3.6679747  4.6032153  5.2183202  5.6656599  5.7645559  6.1426344
 [43]  7.0363985  6.4948516  7.3427078  7.7112560  8.4455083  9.2614302
 [49]  9.6959040  9.7273640 11.7321600 11.9107518 12.6009052 13.1168750
 [55] 13.2810812 13.1558443 13.8869984 15.1043203 14.9139295 13.8209335
 [61] 14.0982162 14.3164339 13.5725143 13.2138087 14.8675421 14.1626469
 [67] 16.3168637 16.9196236 16.5900724 16.4191131 16.2984766 18.1113318
```

```
 [73] 20.0879579 18.6321345 17.1605271 16.8674346 17.5909653 16.9363520
 [79] 18.5408090 18.8395865 19.5871888 20.3830588 21.3429234 20.6759120
 [85] 21.6561634 20.4557266 21.0152577 19.7601278 21.5775434 22.7862562
 [91] 22.0375917 19.5243107 18.5725099 18.0294363 17.2828486 18.9968337
 [97] 17.8319481 18.1247986 18.6883637 17.9509444
```

```r
# Display the first 3 elements of x.
head(x, 3)
```

```
[1] 0.7371499 0.6574312 1.5180619
```

```r
# Display the summary statistics on x.
summary(x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-2.5133 -0.4693  0.1788  0.1795  0.7398  2.1542 
```

```r
# Sort x from largest to smallest.
sort(x)
```

```
  [1] -2.51328097 -1.57527508 -1.47160742 -1.45582340 -1.35035415 -1.25512990
  [7] -1.20043675 -1.16488563 -1.11116653 -1.09299604 -1.03192696 -0.95180082
 [13] -0.79518693 -0.74866452 -0.74658770 -0.74391962 -0.73741927 -0.70489523
 [19] -0.66701134 -0.65461334 -0.63982107 -0.59869809 -0.54307364 -0.54154692
 [25] -0.51896609 -0.45275375 -0.41915305 -0.40497430 -0.35870562 -0.32955121
 [31] -0.29309249 -0.24534932 -0.21717286 -0.19039083 -0.17095927 -0.16077751
 [37] -0.15057010 -0.14955450 -0.13541328 -0.12523694 -0.12063653 -0.08707332
 [43]  0.03146005  0.04116546  0.05543717  0.09889603  0.14411306  0.16420623
 [49]  0.17140086  0.17859184  0.17904324  0.18756814  0.21821771  0.27728273
 [55]  0.29285049  0.29877750  0.33230227  0.36854818  0.37807848  0.43447380
 [61]  0.44733966  0.44763608  0.51596975  0.55953108  0.56356513  0.60275988
 [67]  0.61510493  0.61797689  0.65743125  0.68023824  0.69015343  0.72353072
 [73]  0.73115409  0.73425235  0.73714994  0.74760236  0.79586999  0.81592188
 [79]  0.84785620  0.89376408  0.93524057  0.95986455  0.98025135  0.98192569
 [85]  1.04934242  1.20871278  1.21732197  1.24075032  1.24327335  1.44813172
 [91]  1.51806191  1.60445700  1.65373347  1.71398512  1.81285515  1.81741564
 [97]  1.97662613  1.97921367  2.00479595  2.15421681
```

```r
# Compute the successive differences in x.
diff(x)
```

```
 [1] -0.07971869  0.86063066 -1.66863201  0.01515682  0.31445652  1.06170708
 [7] -0.19140790  0.39878930 -0.83015483 -0.47386384 -0.36128592 -0.02817646
[13]  0.43291746  0.79435756 -0.30168745 -0.84101574 -0.47904356  0.68098653
[19] -0.46031851 -0.09981304  0.85126836 -1.44346880 -0.46410854  0.78008814
[25] -0.55516722  0.89760040  0.90038983 -0.59719058 -0.44914358  1.84197143
[31] -1.18783618 -1.08736412  3.01114063 -2.38418798  0.31790098  0.25847418
[37]  0.76383971 -0.32013564 -0.16776528 -0.34844362  0.27918245  0.51568560
```

```
[43] -1.43531101  1.38940312 -0.47930802  0.36570417  0.08166954 -0.38144808
[49] -0.40301375  1.97333590 -1.82620411  0.51156159 -0.17418367 -0.35176353
[55] -0.28944317  0.85639103  0.48616788 -1.40771280 -0.90260521  1.37027877
[61] -0.05906502 -0.96213733  0.38521400  2.01243909 -2.35862870  2.85911204
[67] -1.55145693 -0.93231109  0.15859193  0.05032275  1.93349168  0.16377098
[73] -3.43244954 -0.01578402  1.17851493  1.01662322 -1.37814406  2.25907033
[79] -1.30567950  0.44882487  0.04826763  0.16399456 -1.62687588  1.64726268
[85] -2.18068810  1.75996783 -1.81466098  3.07254554 -0.60870286 -1.95737730
[91] -1.76461645  1.56148016  0.40872718 -0.20351406  2.46057282 -2.87887075
[97]  1.45773611  0.27071464 -1.30098440
```

```r
# Create an integer sequence from 1 to 10
1:10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```r
# A sequence from 1 to 10 in steps of 0.1
seq(1, 10, 0.1)
```

```
 [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4
[16]  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9
[31]  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4
[46]  5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9
[61]  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4
[76]  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9
[91] 10.0
```

If

```r
# Define a boolean variable
my_boolean <- 1 == 2

if( my_boolean) {
   print("not correct")
} else {
   print("XYZ")
}
```

```
[1] "XYZ"
```

```r
for(i in 1:5){
   cat(i, "\n")
}
```

```
1
2
3
4
5
```

```r
some_list <- list()

for(z in c("hello", "goodbye")) {
   some_list[[z]] <- z
}

some_list
```

```
$hello
[1] "hello"

$goodbye
[1] "goodbye"
```

```r
filter_and_sort_symbols <- function(symbols) {
   # Name: filter_symbols
   # Purpose: Convert to upper case if not
   # and remove any non valid symbols
   # Input: symbols = vector of stock tickers
   # Output: filtered_symbols = filtered symbols

   # Convert symbols to uppercase
   symbols <- toupper(symbols)

   # Validate the symbol names
   valid <- regexpr("^[A-Z]{2,4}$", symbols)

   # Return only the valid ones
   return(sort(symbols[valid == 1]))
}
```

```r
filter_and_sort_symbols(c("MOT", "cvx", "123", "Gog2", "XLe"))
```

```
[1] "CVX" "MOT" "XLE"
```

```r
extract_prices <- function(filtered_symbols, file_path) {
   # Name: extract_prices
   # Purpose: Read prices from specified file
   # Inputs: filtered_symbols = vector of symbols,
   #         file_path = location of price data
   # Output: prices = data.frame of prices per symbols

   # Read in the .csv prices
   all_prices <- read.csv(file = file_path, header = T,
                          stringsAsFactors = F)
```

```
    # Make the data row names
    rownames(all_prices) <- all_prices$Date

    # Remove the original Date column
    all_prices$Date <- NULL

    # Extract only the reelevant data columns
    valid_columns <- colnames(all_prices) %in% filtered_symbols

    return(all_prices[, valid_columns])
}
```

```
filter_prices <- function(prices) {
    # Name: filter_prices
    # Inputs: Identify the rows with missing values
    # Outputs: missing_rows = vector of indexes wehre
    #  data is missing in any of the columns

    # Returns a boolean vector of good or bad rows
    valid_rows <- complete.cases(prices)

    # Identify the index of the missing rows
    missing_rows <- which(valid_rows == F)

    return(missing_rows)
}
```

```
compute_pairwise_correlations <- function(prices) {
    # Name: compute_pairwise_correlations
    # Purpose: Calculates pairwise correlations of returns
    # and plots the pairwise relationships

    # Inputs: prices = data.frame of prices
    # Output: correlation_matrix = A corrleation matrix

    # Convert prices to returns
    returns <- apply(prices, 2, function(x) diff(log(x)))

    # Plot all the pairwise relationships
    pairs(returns, main = "Pairwise return scatterplot")
}
```

```
# Stock Symbols
symbols <- c("IBM", "XOM", "2SG", "TEva",
             "GOog", "CVX", "AAPL", "BA")
```

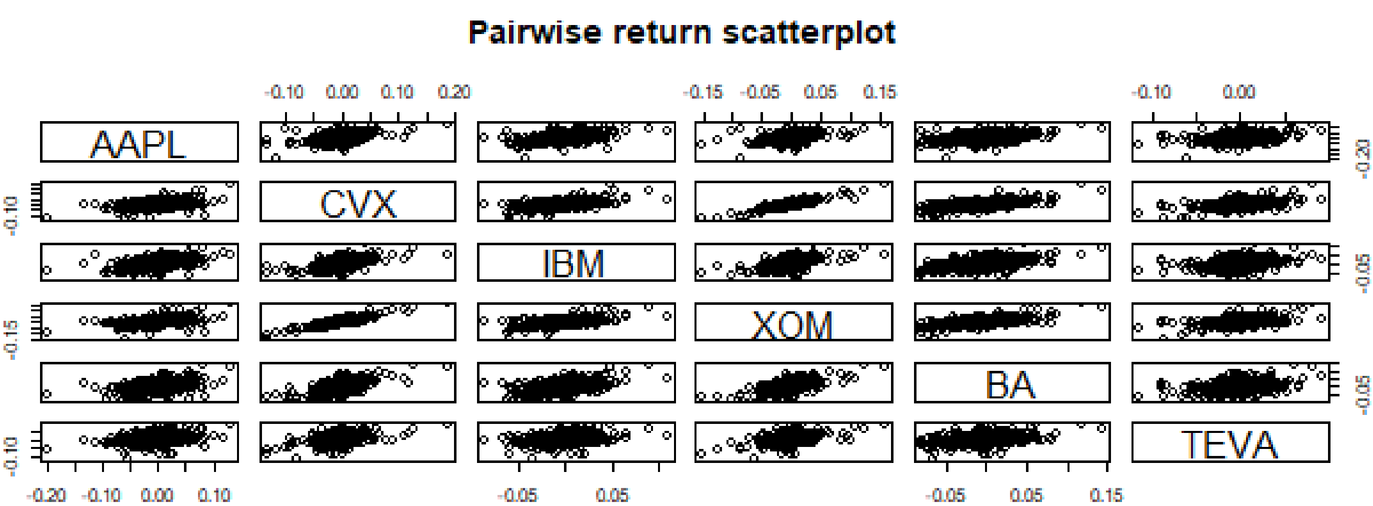
```r
# Location of the price database

price.file <- file.path(here::here(), "/Quantitative Trading/prices.csv")
prices <- data.table::fread(price.file)

# Filter and sort the symbols
filtered_symbols <- filter_and_sort_symbols(symbols)
filtered_symbols
```

```
[1] "AAPL" "BA"   "CVX"  "IBM"  "TEVA" "XOM"
```

```r
# Extract Prices
prices <- extract_prices(filtered_symbols, price.file)

# Filter Prices
missing_rows <- filter_prices(prices)
missing_rows
```

```
integer(0)
```

```r
# Compute Correlations
correlation_matrix <- compute_pairwise_correlations(prices)
```




```r
correlation_matrix
```

```
NULL
```