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,]  0.216 -0.669  0.375  1.417 -1.733
[2,]  1.189  0.178  0.888  1.205  1.659
[3,] -0.311 -1.263 -1.239 -0.032  0.118
[4,]  0.822 -0.555  1.118 -0.452 -0.102
[5,]  0.552  2.861  1.079 -1.462 -0.086
```

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

```
              [,1]        [,2]         [,3]       [,4]         [,5]
 [1,] 3.232779e-03 0.046635132 0.4477492173 0.14070471 2.007288e+00
 [2,] 1.708153e-01 1.412756225 0.0315285373 0.78814538 1.451191e+00
 [3,] 7.657190e-03 0.096784843 1.5944968187 1.53520809 9.961697e-04
 [4,] 9.469063e-05 0.675479907 0.3080542146 1.25015738 2.040668e-01
 [5,] 6.856766e-01 0.304754818 8.1852834393 1.16459314 2.137836e+00
 [6,] 7.762066e-02 0.497849328 0.0002397166 2.24270730 2.140550e+00
 [7,] 4.350827e-01 0.085218418 2.6450951369 0.15927902 8.483330e-01
 [8,] 1.655075e-01 0.911908470 0.5313500358 1.46693507 3.611889e-03
 [9,] 1.072804e+00 0.370367095 3.9760650957 0.67143762 2.767373e+00
[10,] 4.705250e-01 1.620777433 0.0913223354 0.03118601 1.009274e+00
[11,] 6.924190e-01 1.193935195 0.0326241769 0.14501242 1.100095e+00
[12,] 2.113157e-02 6.421874039 3.5905712866 3.63973199 3.228633e-01
[13,] 3.442075e-01 0.193148010 0.0536682808 1.58003956 4.713509e-01
[14,] 3.382860e-02 0.933757389 0.6477013976 0.94639481 1.493993e-01
[15,] 5.925857e-01 0.690836114 2.9571957538 0.92716196 1.773176e+00
[16,] 3.800506e+00 1.185015193 0.4428915594 0.93919507 1.315394e-05
[17,] 1.299171e-01 3.282853718 0.2455025110 0.40512760 4.415003e-01
[18,] 5.069166e-01 0.987721692 1.2366327702 1.66073960 1.652975e+00
[19,] 1.256437e+00 0.031569993 0.1117259595 1.05560392 1.591750e+00
[20,] 1.027798e+00 0.005763322 2.6320825959 2.21498750 3.868036e-01
[21,] 6.211578e-02 4.416924949 3.1216087141 5.02891221 2.749635e+00
[22,] 7.261861e-02 0.069646432 1.4327976781 3.75122071 5.245422e-01
[23,] 5.687546e+00 0.247841477 2.9653876851 1.82587162 1.376173e+00
[24,] 1.342531e+00 0.400833900 0.0443055892 0.59779331 1.200194e-02
[25,] 9.932145e-01 1.289201653 0.9333497620 0.03452366 3.767737e+00
              [,6]        [,7]        [,8]         [,9]        [,10]
 [1,] 3.0047080348 0.025019333 0.002038921 3.3613920366 4.000978e-01
 [2,] 2.7534958280 2.328111409 0.260177288 0.7354307608 2.337777e-01
 [3,] 0.0138446756 2.208074177 0.141853410 0.1813696675 4.287824e-01
 [4,] 0.0104008012 2.665882557 0.018905875 2.0224278224 4.254108e+00
 [5,] 0.0074712387 0.014744475 1.120556165 0.1789628802 8.516287e-02
 [6,] 3.1240384203 0.154272600 0.687821831 0.3173953333 9.368815e-01
 [7,] 1.8548034249 1.868920810 3.445235624 0.0262629387 1.114716e-01
 [8,] 0.1685648353 0.932333732 0.786776324 4.6940996923 1.174216e+00
 [9,] 0.9326032674 0.740129367 0.720211782 2.0575338089 2.461809e+00
[10,] 0.8400681655 2.273433375 0.005290494 1.4866751348 1.811696e-01
```

```
[11,] 0.0053036647 0.944457954 0.625873056 0.4410199289 2.166682e-01
[12,] 0.2514587840 2.039618546 1.183433987 0.8519384139 1.156861e-03
[13,] 3.4162539291 0.366067554 0.983777926 2.6223392237 8.442594e-01
[14,] 0.0001860384 6.295541457 0.678469586 6.2294053821 1.783449e+00
[15,] 1.3457905816 0.591128639 1.615286451 0.8622545528 4.594545e-07
[16,] 0.0636477854 1.374142895 0.090118314 0.1768694150 3.895118e+00
[17,] 0.3743944383 0.830546057 1.935611209 0.7362875763 5.492570e-01
[18,] 0.2668338270 0.152799462 2.906302337 1.7670204922 2.287481e-01
[19,] 0.9034742117 0.272002771 0.037896651 0.3219673180 2.193571e-01
[20,] 2.2994434140 1.229828367 0.146675724 0.0157244858 1.572679e+00
[21,] 1.9731824520 0.182622330 0.255264414 0.1325810084 4.267371e-03
[22,] 0.0455230293 0.002208525 0.584361883 0.5495572985 9.975429e-02
[23,] 0.2206303836 2.258917321 1.924803700 0.6940591511 3.816880e-01
[24,] 0.1068864176 0.795823847 1.427407271 0.0154416598 3.392385e-02
[25,] 0.4843248855 0.032555383 1.093916548 0.0006590596 9.928846e-01
```

data.frame:

```
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
```

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

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

```
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
```

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

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

```
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"
```

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

second_element
```

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

```
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
```

```
class(part_of_list)
```

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

```
size_of_list <- length(my_list)

size_of_list
```

```
[1] 4
```

Env:

```
env <- new.env()

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

env
```

```
<environment: 0x000000001c9751c8>
```

```
ls(env)
```

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

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

```
[1] 5
```

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

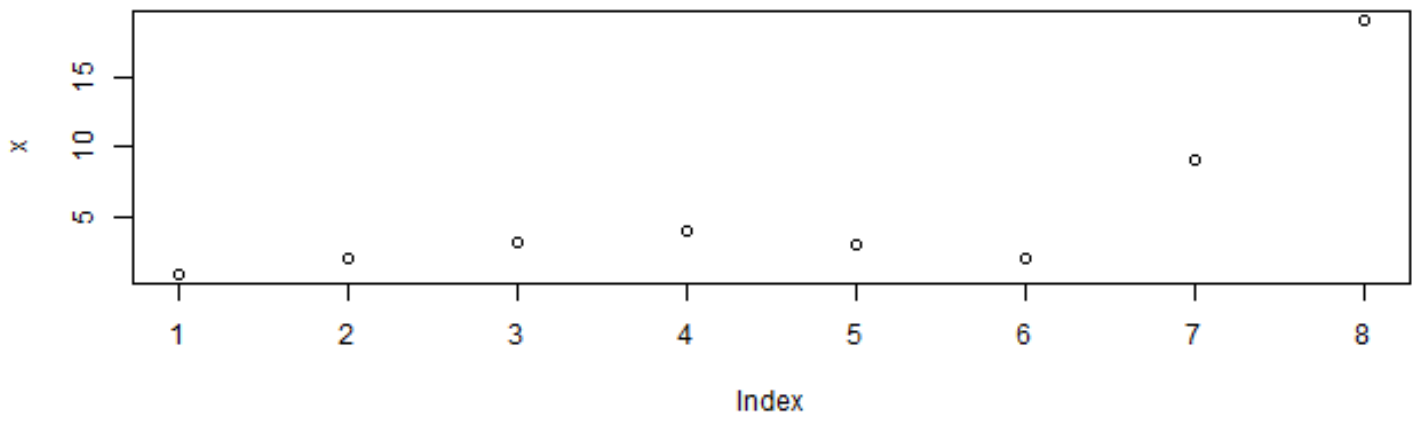
```
ls(env)
```

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

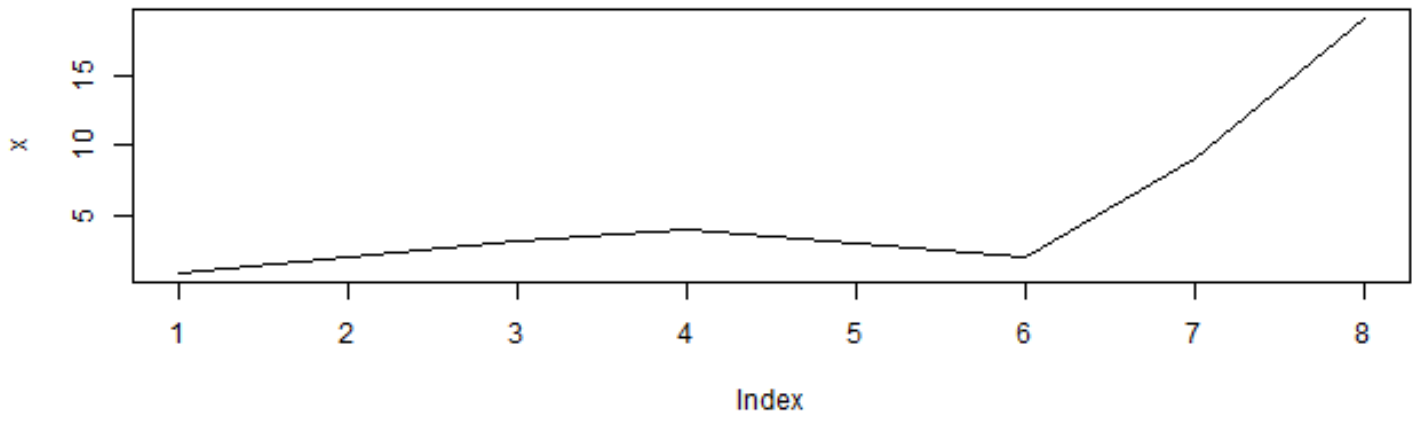
```
# pass by reference
env_2 <- env
env_2$third <- 42

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

```
[1] 42
```

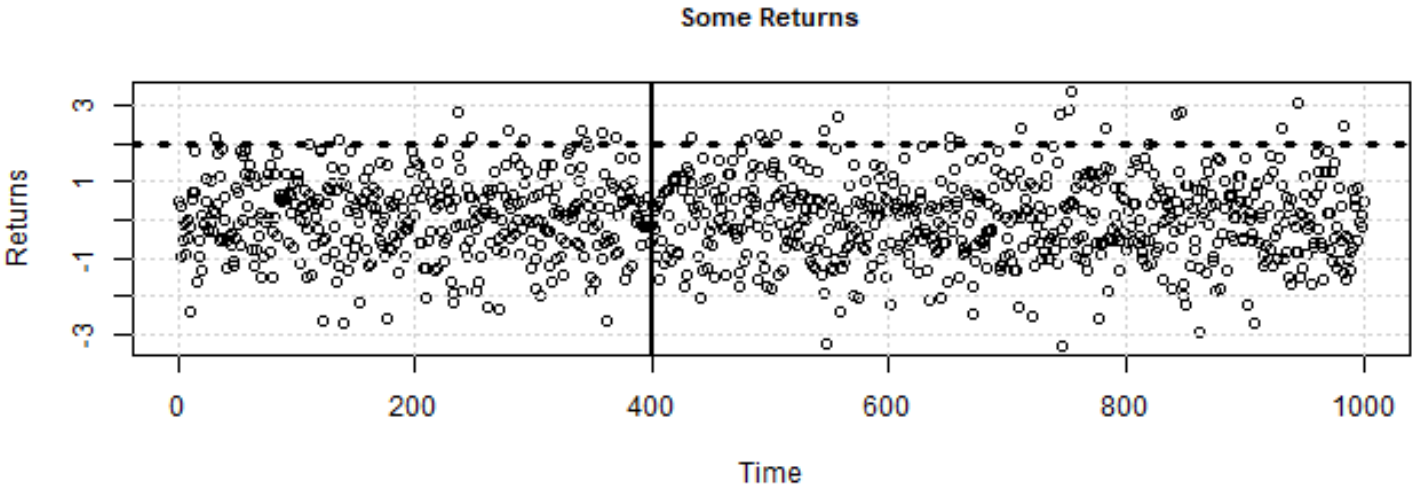
```
x <- c(1, 2, 3.2, 4, 3, 2.1, 9, 19)
plot(x)
```




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

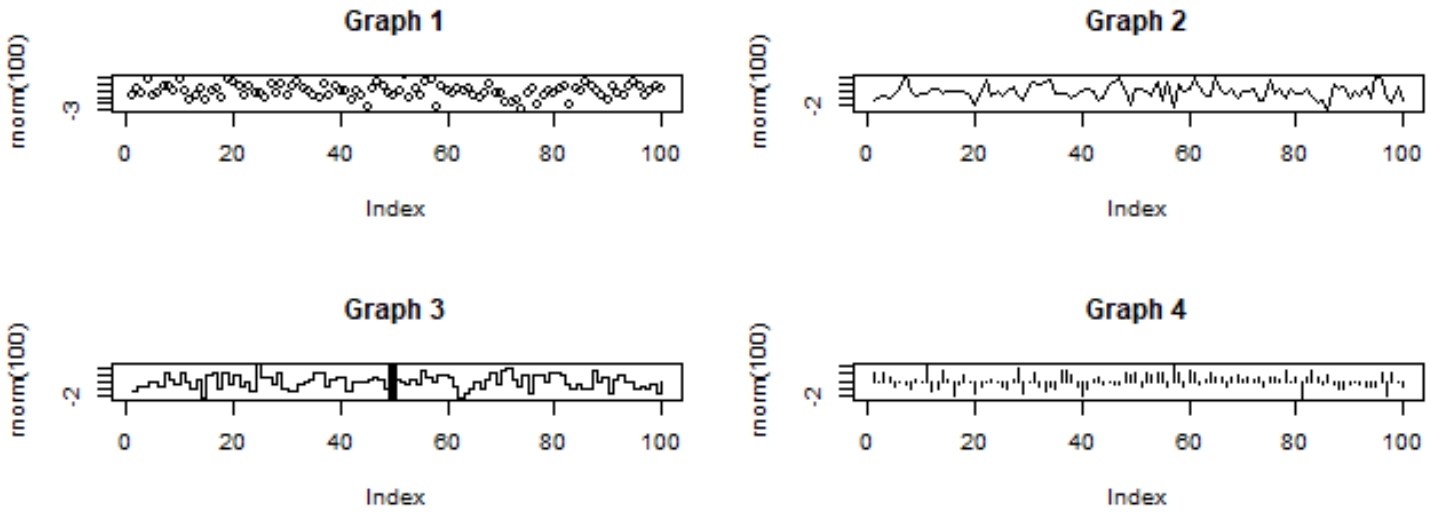



```
# Setup the canvas
plot(rnorm(1000), main = "Some Returns", cex.main = 0.9,
     xlab = "Time", ylab = "Returns")
# Superimpose a grid
grid()
# Create a few vertical and horizontal lines
abline(v = 400, lwd = 2, lty = 1)
abline(h = 2, lwd = 3, lty = 3)
```



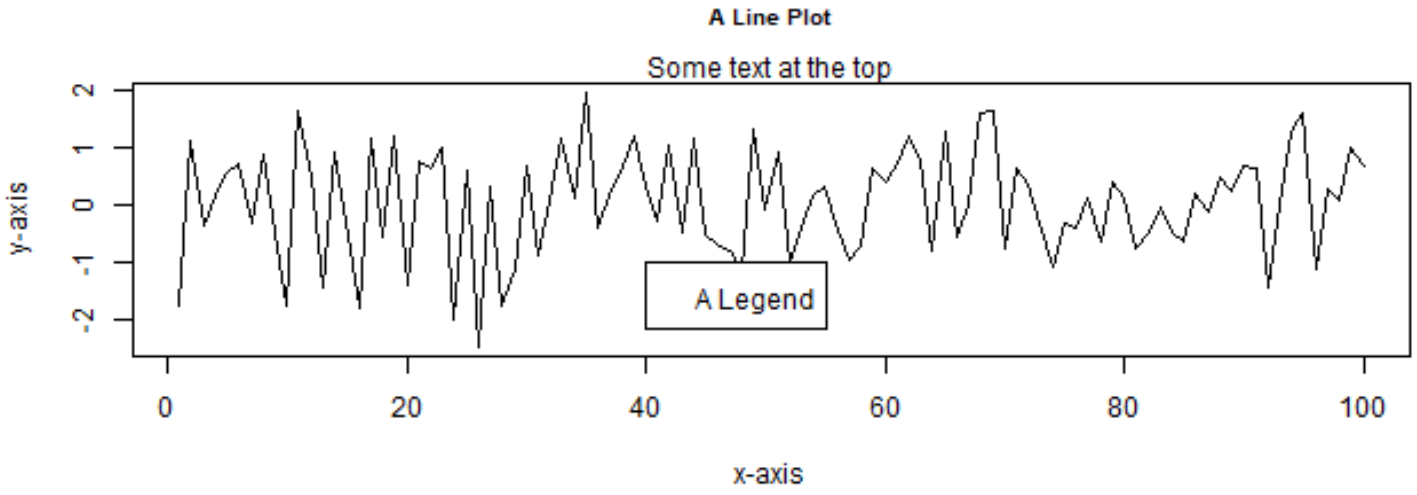

```
# Create a 2-row, 2-column format
par(mfrow = c(2, 2))

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

```
# 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")
```




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

```
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")
```




```
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:

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

ans
```

```
[1] 5050
```

Imperative:

```
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] -0.002689892
```

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

```
[1] 1.000342
```

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

```
[1] -2.008620  2.847385
```

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

```
[1] -0.2689892
```

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

```
  [1] -0.1789785 -0.1870025  0.3445454 -0.5057238  1.1220334  0.5635940
  [7] -0.1692210 -0.9345384 -1.5064345 -0.7694077  0.4528056  0.4496205
 [13] -1.0355282 -1.6792453 -1.4523583 -0.2355864  0.7280996  0.8644689
 [19]  3.7106157  4.2033580  3.7346506  3.8993876  2.3467463  3.2794730
 [25]  2.9961400  2.4845021  1.6841167  2.1844619  1.5265282  1.5368944
 [31]  1.6729845  2.5102739  4.9469283  5.7169783  4.5456579  3.1557434
 [37]  3.9267329  2.1921949  1.9928650  1.5294910  0.5514357  2.1341481
 [43]  2.0216269  2.1293842  3.1805669  4.6890297  4.4015567  4.4949871
 [49]  4.1507104  4.2401288  5.3222073  5.9588528  5.0830354  5.1412798
 [55]  6.1468911  4.4853023  3.4241605  3.7754068  5.3679139  4.3925276
 [61]  4.4209782  4.4416166  5.1124823  4.8919838  4.9946358  5.9517313
 [67]  6.5637181  5.5474803  4.4681401  5.4220728  6.2015978  6.1052675
```

```
 [73]  6.6545870  7.0583763  7.7004228  7.5758458  7.9677746  6.6747056
 [79]  5.7195921  4.1217412  4.2012903  4.4112354  3.3787331  3.0623561
 [85]  4.2798915  3.5185463  2.9304738  3.3408945  4.1141972  6.9615822
 [91]  5.1916444  6.7720691  5.5276225  3.5190023  3.0520950  2.9371830
 [97]  1.8568365  1.5737697  1.6518082 -0.2689892
```

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

```
[1] -0.178978464 -0.008024043  0.531547911
```

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

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.00862 -0.73995  0.01550 -0.00269  0.64925  2.84739
```

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

```
  [1] -2.008620168 -1.920797365 -1.769937831 -1.734537994 -1.661588875
  [6] -1.597850966 -1.552641363 -1.485148653 -1.389914461 -1.293069012
 [11] -1.244446598 -1.171320432 -1.080346518 -1.079340199 -1.061141788
 [16] -1.032502364 -1.016237844 -0.978055233 -0.975386300 -0.955113464
 [21] -0.875817349 -0.850269166 -0.800385413 -0.765317333 -0.761345210
 [26] -0.732815097 -0.657933721 -0.643717087 -0.588072453 -0.571896075
 [31] -0.558439369 -0.511637902 -0.468707378 -0.466907353 -0.463374009
 [36] -0.344276780 -0.316376970 -0.287472964 -0.283333012 -0.283066808
 [41] -0.220498490 -0.199329983 -0.178978464 -0.124577014 -0.114911933
 [46] -0.112521209 -0.096330330 -0.008024043 -0.003185173  0.010366217
 [51]  0.020638410  0.028450539  0.058244340  0.078038451  0.079549159
 [56]  0.089418483  0.093430415  0.102652010  0.107757281  0.136090096
 [61]  0.136369306  0.164737056  0.209945113  0.226886991  0.351246292
 [66]  0.391928788  0.403789208  0.410420708  0.492742290  0.500345183
 [71]  0.531547911  0.549319585  0.611986815  0.636645513  0.642046578
 [76]  0.670865744  0.737026776  0.770050011  0.770989518  0.773302731
 [81]  0.779524955  0.837289401  0.932726757  0.953932733  0.957095503
 [86]  0.963685951  1.005611371  1.051182702  1.082078415  1.216771917
 [91]  1.217535370  1.222213329  1.508462785  1.580424686  1.582712408
 [96]  1.592507154  1.627757179  2.436654412  2.846146781  2.847384986
```

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

```
 [1]  0.170954420  0.539571954 -1.381817078  2.478026345 -2.186196548
 [6] -0.174375729 -0.032502236  0.193421258  1.308922851  0.485186553
[11] -1.225398502 -1.481963480  0.841431566  0.870604078  0.989884925
[16] -0.253085965 -0.827316646  2.709777475 -2.353404491 -0.961449668
```

```
[21]  0.633444435 -1.717378419  2.485368120 -1.216059769 -0.228304890
[26] -0.288747511  1.300730596 -1.158278905  0.668299938  0.125723880
[31]  0.701199304  1.599365011 -1.666604401 -1.941370443 -0.218594029
[36]  2.160903979 -2.505527513  1.535208012 -0.264044027 -0.514681223
[41]  2.560767640 -1.695233617  0.220278490  0.943425421  0.457280083
[46] -1.795935749  0.380903379 -0.437707195  0.433695263  0.992659932
[51] -0.445432901 -1.512462862  0.934061689  0.947367031 -2.667200246
[56]  0.600447088  1.412388079  1.241260862 -2.567893454  1.003836839
[61] -0.007812129  0.650227334 -0.891364234  0.323150501  0.854443493
[66] -0.345108688 -1.628224659 -0.063102355  2.033272933 -0.174407778
[71] -0.875855285  0.645649915 -0.145530377  0.238257370 -0.766623592
[76]  0.516505802 -1.684997800  0.337955548 -0.642737502  1.677400125
[81]  0.130395954 -1.242447477  0.716125393  1.533912340 -1.978880580
[86]  0.173272756  0.998493161  0.362882023  2.074082255 -4.617322817
[91]  3.350362518 -2.824871284 -0.764173570  1.541712815  0.351995419
[96] -0.965434585  0.797279710  0.361105259 -1.998835815
```

```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
```

```
some_list <- list()

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

some_list
```

```
$hello
[1] "hello"

$goodbye
[1] "goodbye"
```

```
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]))
}
```

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

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

```
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")
}
```

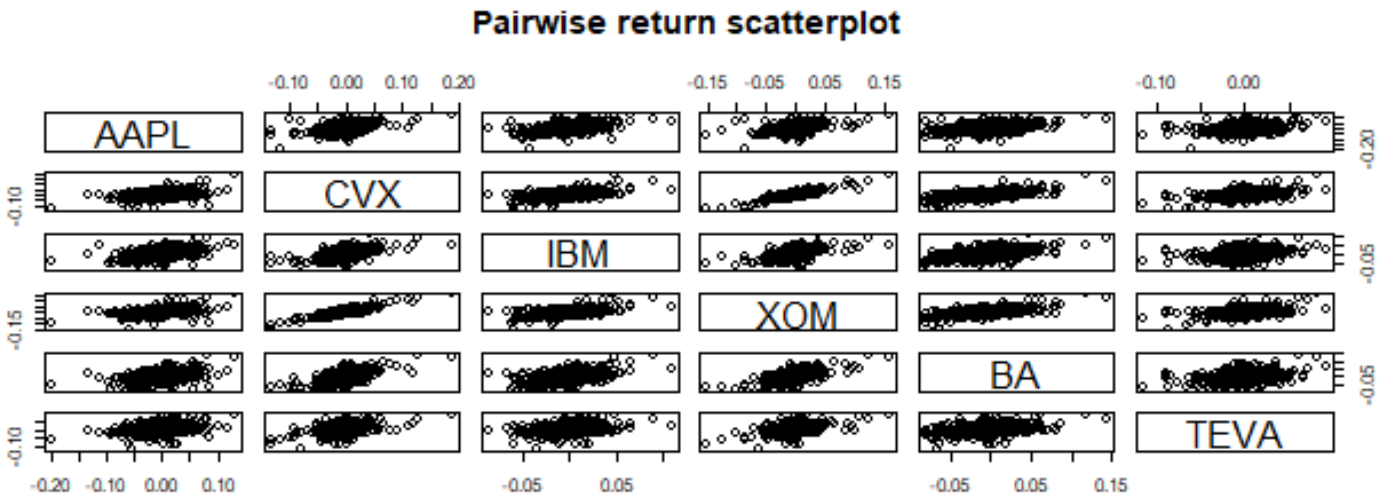
```
# Stock Symbols
symbols <- c("IBM", "XOM", "2SG", "TEva",
             "GOog", "CVX", "AAPL", "BA")

# 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"
```

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

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

```
integer(0)
```

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



```
correlation_matrix
```

```
NULL
```