R-intro - Session 3 exercise solutions

João Gonçalves

27 de Julho de 2018

Contents

List objects

Exercise 1

1a.

```
my_list <- list(a = 1:5,
                b = matrix(1:20, nrow = 5, ncol = 4))
```

1b.

```
# Extracting a from the list
my_list$a
```

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

```
# Extracting the first column of b
my_list$b[,1]
```

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

```
# Multiply the two vectors (element-by-element) and attribute
# this to a new list element called "mult"
my_list[["mult"]] <- my_list$a * my_list$b[,1]

print(my_list$mult)
```

```
## [1]  1  4  9 16 25
```

1c.

```
# Using the dollar $ operator
my_list$mult[3]
```

```
## [1] 9
```

```
# Equivalent to before but using the [[]] operator
my_list[["mult"]][3]
```

```
## [1] 9
```

```
# Concatenation can be used to extract more than one element, for example:
my_list[["mult"]][c(1,3,5)]
```

```
## [1]  1  9 25
```

Exercise 2

2.

```
set.seed(123) # Used to make random numbers always the same

# Now define the list that will be used for the exercise
nestList <- list(

  x = list(
    a1 = 1:10,
    a2 = rnorm(10)
  ),

  y = list(
        b1 = 1:10,
        b2 = rnorm(10)
  )
)
```

2a.

```
# Using the $ operator
nestList$x
```

```
## $a1
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $a2
##  [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774
##  [6]  1.71506499  0.46091621 -1.26506123 -0.68685285 -0.44566197
```

```
# The same as before but with the [[]] operator
nestList[["x"]]
```

```
## $a1
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $a2
##  [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774
##  [6]  1.71506499  0.46091621 -1.26506123 -0.68685285 -0.44566197
```

2b.

```r
# the third element of `a1`
nestList$x$a1[3]
```

```
## [1] 3
```

```r
# equivalent to:
nestList[["x"]][["a1"]][3]
```

```
## [1] 3
```

```r
# the second to fifth elements of `b2`
nestList$y$b2[2:5]
```

```
## [1]  0.3598138  0.4007715  0.1106827 -0.5558411
```

```r
# equivalent to:
nestList[["y"]][["b2"]][2:5]
```

```
## [1]  0.3598138  0.4007715  0.1106827 -0.5558411
```

2c.

```r
nestList$x$a2 * nestList$y$b1
```

```
##  [1]  -0.5604756  -0.4603550   4.6761249   0.2820336   0.6464387
##  [6]  10.2903899   3.2264134 -10.1204899  -6.1816757  -4.4566197
```

```r
# the same as:
nestList[["x"]][["a2"]] * nestList[["y"]][["b1"]]
```

```
##  [1]  -0.5604756  -0.4603550   4.6761249   0.2820336   0.6464387
##  [6]  10.2903899   3.2264134 -10.1204899  -6.1816757  -4.4566197
```

QUICK-EXERCISE 1

QE-1) a)

```r
# Extract columns to atemporary vectors
ozone <- airquality$Ozone
temp <- airquality$Temp

# Calculate correlation
aq_cor <- cor.test(x = ozone, y = temp,method="pearson")

# Equivalent to before but without specifically declaring
# the x and y in cor.test. This way we are defining function
# arguments by their positions
aq_cor <- cor.test(ozone, temp)

# Also equivalent but not using temp vectors, instead
# using directly indexation to pull data
aq_cor <- cor.test(airquality$Ozone, airquality$Temp)
```

QE-1) b)

```r
# Get object structure
str(aq_cor)
```

```
## List of 9
```

```
##  $ statistic  : Named num 10.4
##   ..- attr(*, "names")= chr "t"
##  $ parameter  : Named int 114
##   ..- attr(*, "names")= chr "df"
##  $ p.value    : num 2.93e-18
##  $ estimate   : Named num 0.698
##   ..- attr(*, "names")= chr "cor"
##  $ null.value : Named num 0
##   ..- attr(*, "names")= chr "correlation"
##  $ alternative: chr "two.sided"
##  $ method     : chr "Pearson's product-moment correlation"
##  $ data.name  : chr "airquality$Ozone and airquality$Temp"
##  $ conf.int   : atomic [1:2] 0.591 0.781
##   ..- attr(*, "conf.level")= num 0.95
##  - attr(*, "class")= chr "htest"
```

```
# Get correlation value
aq_cor$estimate
```

```
##       cor
## 0.6983603
```

```
# Get correlation p-value
aq_cor$p.value
```

```
## [1] 2.931897e-18
```

User-defined functions

Exercise 3

3a.

```
Celsius2Kelvin <- function(tempCelsius){

  tempKelvin <- tempCelsius + 273.15

  return(tempKelvin)
}

a <- 20.5
Celsius2Kelvin(a)
```

```
## [1] 293.65
```

3b.

```
Celsius2Fahrenheit <- function(tempCelsius){

  return(tempCelsius * 9/5 + 32)
}
```

```r
b <- 16.7
Celsius2Fahrenheit(b)
```

```
## [1] 62.06
```

3c.

```r
recode2NA <- function(x){
  # Modify x values to NA
  x[x < 10 | x > 100] <- NA
  # return the modified x
  return(x)
}

v <- c(1, 5, 10, 15, 25, 78, 90, 34, 55, 120, 100, 105, 103, 12, 101)
recode2NA(v)
```

```
##  [1]  NA  NA  10  15  25  78  90  34  55  NA 100  NA  NA  12  NA
```

3d.

```r
standardize.me <- function(x){
  # A temp variable with the standardized x
  x.std <- (x - mean(x)) / sd(x)
  return(x.std)
}

d <- rnorm(100, 100, 5)
standardize.me(d)
```

```
##   [1] -1.20092019 -0.23629559 -1.15345306 -0.81621339 -0.69833577
##   [6] -1.90337085  0.96205166  0.18520463 -1.28072953  1.43426593
##  [11]  0.49517789 -0.32380438  1.02713411  1.00784707  0.94365701
##  [16]  0.79276194  0.63984446 -0.05915536 -0.33616647 -0.42073748
##  [21] -0.77741241 -0.22487962 -1.42517589  2.47300079  1.38222037
##  [26] -1.26367153 -0.44617840 -0.51856141  0.89642063 -0.08351063
##  [31]  0.29864809 -0.02128432 -0.03754250  1.56455583 -0.24514455
##  [36]  1.73239442 -1.74680087  0.67468620  0.15169910  0.25622323
##  [41]  0.44202926 -0.55904664 -0.36709071 -1.14502066 -1.20542354
##  [46]  0.35563932  0.51986028  0.07128048  1.05794154  2.33807543
##  [51] -0.54622928 -2.60991448  1.15268573 -0.79386364 -0.76980940
##  [56]  1.17519709 -0.31211505 -1.37446320  0.21690711 -0.14653145
##  [61]  0.01766046  0.44843201 -0.40960150  0.74252023 -0.23914644
##  [66]  0.38770837  1.25608974  0.50507246 -0.35883228  1.31507693
##  [71]  1.13879871  0.63357818  0.28209130 -0.70158975  1.55553234
##  [76] -0.67020948  2.49385972  1.75071422 -0.25641493 -1.15392575
##  [81] -0.79523229  0.30269478 -0.26889090 -0.38336195 -1.06902104
##  [86] -0.03999112 -0.87979147 -1.88208702 -0.42045998  1.05422894
##  [91] -0.64193230  0.70119037 -1.82526708 -0.05194805  0.60067323
##  [96]  0.35294325  0.13106603 -0.71611837 -0.95334285 -1.15132408
```

3e.

```r
CoeffVar <- function(x){

  cv <- mean(x) / sd(x)
  return(cv)
```

```r
}

e <- rnorm(100)
CoeffVar(e)
```

```
## [1] -0.02338485
```

3f.

```r
mult2 <- function(x){

  x.sort <- sort(x, decreasing = TRUE)
  highest2 <- x.sort[1:2]

  return((highest2[1] * highest2[2]) / 2)
}

f <- rnorm(1000, 100, 10)
mult2(f)
```

```
## [1] 8459.032
```

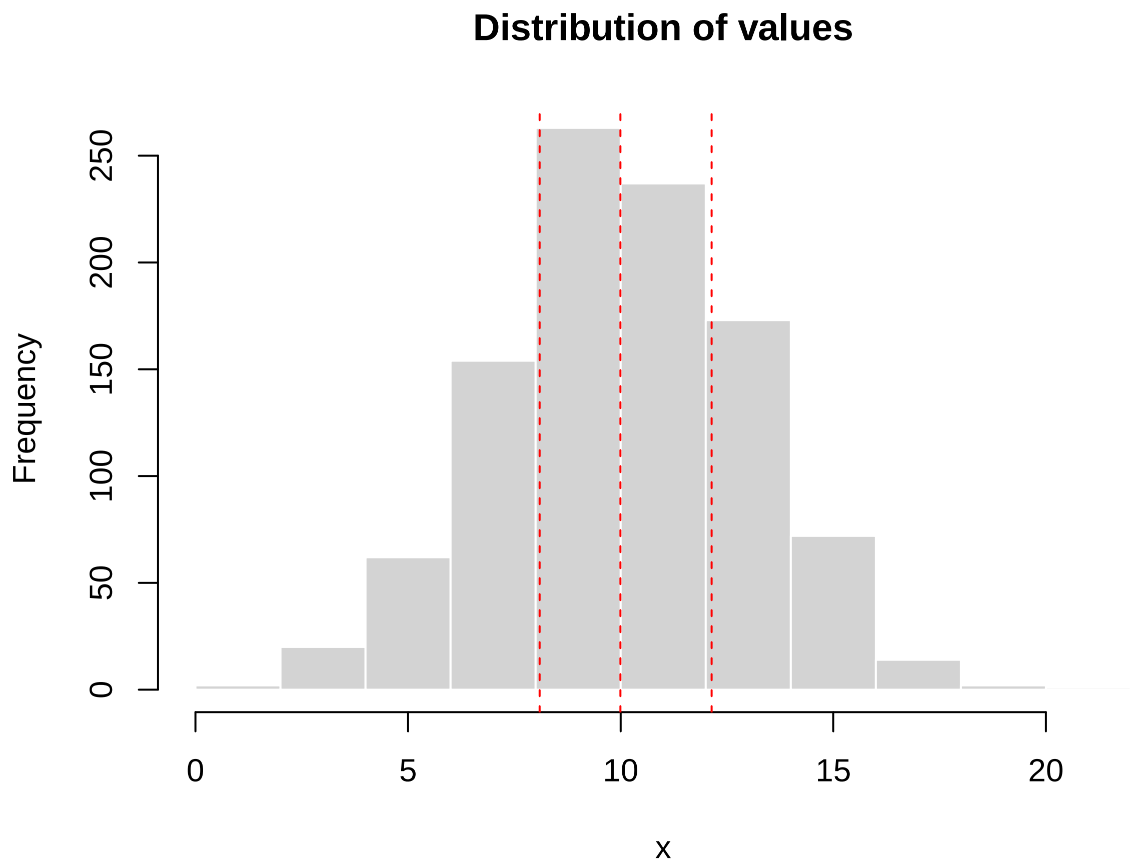
3g.

```r
myHist <- function(x, ...){

  # Calculate the 25%, 50% and the 50% quantiles
  qts <- quantile(x, probs = c(0.25, 0.50, 0.75))
  # Make the histogram with the input x vector
  hist(x, ...)
  # Add the vertical lines using abline function
  abline(v = qts, lwd=1, lty=2, col="red")
}


g <- rnorm(1000, 10, 3)

myHist(g, main="Distribution of values",
       col = "light grey",
       border = "white")
```




```r
# To save the result to a file use (for example) png()
# folowed by the function that makes the plot
# then close the file by using dev.off()
png(filename = "myHist.png", width = 1200, height = 900, res = 300)
myHist(g, main="Distribution of values",
       col = "light grey",
       border = "white")
dev.off()
```

```
## pdf
##   2
```

If conditionals

Exercise 4

4a.

```r
isPositive <- function(x){
  # The test condition of an if goes inside the ( ) what happens
```

```
  # if the condition is TRUE goes within the { }
  if(x > 0){
    TRUE
  }else{
    FALSE
  }
}

isPositive(10)
```

```
## [1] TRUE
```

```
isPositive(-10)
```

```
## [1] FALSE
```

4b.

```
checkCorrValue <- function(x, y){

  absCorrValue <- abs(cor.test(x, y)$estimate)

  if(absCorrValue > 0.6){
    return(TRUE)
  }else{
    return(FALSE)
  }
}

# Test with vectors Ozone and Temp from airquality dataset
checkCorrValue(x = airquality$Ozone, y = airquality$Temp)
```

```
## [1] TRUE
```

```
# Two additional tests
checkCorrValue(x = rnorm(10), y = rnorm(10))
```

```
## [1] FALSE
```

```
checkCorrValue(x = 1:10, y = 1:10)
```

```
## [1] TRUE
```

4c.

```
significanceSymbol <- function(x, y, ...){

  corTest <- cor.test(x,y,...)
  pvalue <- corTest$p.value

  if(pvalue >= 0.1){
    print("n.s.")
  }else if(pvalue < 0.001){
    print("***")
  }else if(pvalue < 0.01){
    print("**")
  }else if(pvalue < 0.05){
    print("*")
```

```r
  }else if(pvalue < 0.1){
    print("-")
  }
}

significanceSymbol(x = airquality$Ozone, y = airquality$Temp)
```

```
## [1] "***"
```

For loops

Exercise 5

5a.

```r
# Use na.rm = TRUE by default to avoid missing values
# in mean and sd calculation
#
# In addition let's define a number of decimal plates to
# round the output values (ndigits argument)
#
printMeanStd <- function(x, na.rm=TRUE, ndigits=2, ...){

  # Check if x is a matrix or a dataframe
  # If not stop the function execution
  #
  if(!is.matrix(x) & !is.data.frame(x)){
    stop("x must be a matrix or data frame")
  }

  # Calculate the mean and stdev for each column i
  # and print it to the console
  #
  for(i in 1:ncol(x)){ # let's iterate from 1 to ncol(x)

    # Paste is used to concatenate the text that is
    # outputed
    #
    print(paste("Column nr", i, colnames(x)[i], ":"))

    print(paste("Mean =",
                round(mean(x[,i], na.rm = na.rm), ndigits)))

    print(paste("Std-dev. =",
                round(sd(x[,i], na.rm = na.rm), ndigits)))
  }
}

printMeanStd(airquality)
```

```
## [1] "Column nr 1 Ozone :"
## [1] "Mean = 42.13"
## [1] "Std-dev. = 32.99"
## [1] "Column nr 2 Solar.R :"
## [1] "Mean = 185.93"
## [1] "Std-dev. = 90.06"
## [1] "Column nr 3 Wind :"
## [1] "Mean = 9.96"
## [1] "Std-dev. = 3.52"
## [1] "Column nr 4 Temp :"
## [1] "Mean = 77.88"
## [1] "Std-dev. = 9.47"
## [1] "Column nr 5 Month :"
## [1] "Mean = 6.99"
## [1] "Std-dev. = 1.42"
## [1] "Column nr 6 Day :"
## [1] "Mean = 15.8"
## [1] "Std-dev. = 8.86"
```

5b.

```r
calcCorrelation <- function(xvec, ymat, thresh){

  # Check if xvec is a numeric vector
  # If not stop the function execution
  if(!is.numeric(xvec)){
    stop("xvec must be a numeric vector")
  }

  # Check if ymat is a matrix or a dataframe
  # stop if not
  if(!is.matrix(ymat) & !is.data.frame(ymat)){
    stop("ymat must be a matrix or data frame")
  }

  for(i in 1:ncol(ymat)){
    # Calculate the correlation
    corTest <- cor.test(xvec, ymat[,i])
    corValue <- corTest$estimate

    if(abs(corValue) > thresh){
      print(colnames(ymat)[i]) # Print the i-th variable name
      print(corTest) # print the cor.test result
    }
  }
}

# Let's test the function
calcCorrelation(airquality$Ozone, airquality[,-1], thresh = 0.6)
```

```
## [1] "Wind"
##
##  Pearson's product-moment correlation
##
## data:  xvec and ymat[, i]
```

```
## t = -8.0401, df = 114, p-value = 9.272e-13
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.7063918 -0.4708713
## sample estimates:
##        cor
## -0.6015465
##
## [1] "Temp"
##
##  Pearson's product-moment correlation
##
## data:  xvec and ymat[, i]
## t = 10.418, df = 114, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5913340 0.7812111
## sample estimates:
##       cor
## 0.6983603
```