Intro to R for Decision Modelling

Introduction to R - Recap

SickKids and DARTH

Change eval to TRUE if you want to knit this document.

This worksheet recaps the programming capabilities of R that we will need to develop our decision models. There are three exercises and we have left all the *code chunks* empty for you to fill in. We will also provide solutions after the session.

# 0. Load data into R

Before we begin this session, we need to load our Framingham dataset into R.

data <- read.csv('framingham.csv', header = TRUE)

We will modify a few important variables so that we can understand and visualize the results better.

# 1. Data manipulation

Nested ifelse statements should be read from ‘outside-in’. That is, start with the most outer ifelse statement and work your way in. For the purpose of working through these exercises, you do not have to fully understand the code in the below code chunk.

library(dplyr)  
data1 <- data %>%   
 mutate(SEX = ifelse(!is.na(SEX),   
 ifelse(SEX == 1,   
 'male', 'female'),   
 NA)) %>%  
 mutate(PREVSTRK = ifelse(!is.na(PREVSTRK),   
 ifelse(PREVSTRK == 1,   
 'yes', 'no'),   
 NA)) %>%  
 mutate(PREVMI = ifelse(!is.na(PREVMI),   
 ifelse(PREVMI == 1,   
 'yes', 'no'),   
 NA)) %>%   
 mutate(DIABETES = ifelse(!is.na(DIABETES),   
 ifelse(DIABETES== 1,   
 'yes', 'no'),   
 NA)) %>%  
 mutate(CURSMOKE = ifelse(!is.na(CURSMOKE),   
 ifelse(CURSMOKE== 1,   
 'yes', 'no'),   
 NA)) %>%  
 mutate(BPMEDS = ifelse(!is.na(BPMEDS),   
 ifelse(BPMEDS== 1,   
 'yes', 'no'),   
 NA))

# 2. Data Manipulation

create a data frame that contains the BMI and whether the individual has had a stroke (PREVSTRK) for all individuals with a time of death over 8760 days:

# Your turn  
library(dplyr)  
BMI.STROKE <- data %>%   
 filter(TIMEDTH >= 8760) %>%  
 dplyr::select(BMI, PREVSTRK)  
  
  
## Or  
BMI.STROKE <- data[data$TIMEDTH >= 8760, c("BMI", "PREVSTRK")]

# 3. Matrix multiplication

Create a function that takes two square matrices as inputs and return the following:

1. Their sum
2. Their element-wise product (the Hadamard product)
3. Their product
4. The product of the first matrix with the transpose of the second matrix

Embed a check that returns an error message when any of the two matrices is not a square matrix.

The output of the function should be a list containing the above items.

Test the function on the following two sets of matrices:

m\_A\_set1 <- matrix(1:9, nrow=3, ncol=3)  
m\_B\_set1 <- matrix(6:14, nrow=3, ncol=3)  
  
m\_A\_set2 <- matrix(rnorm(6), nrow=2, ncol=3)  
m\_B\_set2 <- matrix(runif(6), nrow=3, ncol=2)

matrix\_ops <- function(matrixA, matrixB) {  
 if (dim(matrixA)[1] != dim(matrixA)[2] | dim(matrixB)[1] != dim(matrixB)[2]) {  
 print("Error: both matrices need to be square matrices")  
 } else{  
 list(sum = matrixA + matrixB,  
 had\_prod = matrixA \* matrixB,  
 prod = matrixA %\*% matrixB,  
 prod\_trans = matrixA %\*% t(matrixB))  
 }  
}

matrix\_ops(m\_A\_set1, m\_B\_set1)

## $sum  
## [,1] [,2] [,3]  
## [1,] 7 13 19  
## [2,] 9 15 21  
## [3,] 11 17 23  
##   
## $had\_prod  
## [,1] [,2] [,3]  
## [1,] 6 36 84  
## [2,] 14 50 104  
## [3,] 24 66 126  
##   
## $prod  
## [,1] [,2] [,3]  
## [1,] 90 126 162  
## [2,] 111 156 201  
## [3,] 132 186 240  
##   
## $prod\_trans  
## [,1] [,2] [,3]  
## [1,] 126 138 150  
## [2,] 153 168 183  
## [3,] 180 198 216

matrix\_ops(m\_A\_set2, m\_B\_set2)

## [1] "Error: both matrices need to be square matrices"

# 4. Loops

R makes it very simple to generate random numbers from different standard distributions, with a number of functions to achieve this. The general structure of these functions is a lower-case r followed by the name of the distribution:

1. rnorm: To simulate from a normal distribution.
2. rgamma: To simulate from a gamma distribution.
3. rexp: To simulate from an exponential distribution.
4. rbeta: To simulate from a beta distribution.

The first argument of these functions is the number of simulations you wish to generate, with the other arguments giving the parameters of the distribution. For example, to generate 10 values from a gamma distribution with shape parameter 2 and scale parameter 1, we would use:

# Simulate from a gamma distribution  
rgamma(10, shape = 2, scale = 1)

## [1] 2.1708104 2.3056691 3.8056051 0.1712853 6.6270838 4.1932435 2.6118609  
## [8] 0.6341914 1.3470847 0.5073715

Similarly, to simulate 5 values from a normal distribution with mean 2 and variance 5, we would use:

# Simulate from a normal distribution  
rnorm(5, mean = 0, sd = sqrt(5))

## [1] -3.439169 -1.872594 4.558355 -1.077709 1.043541

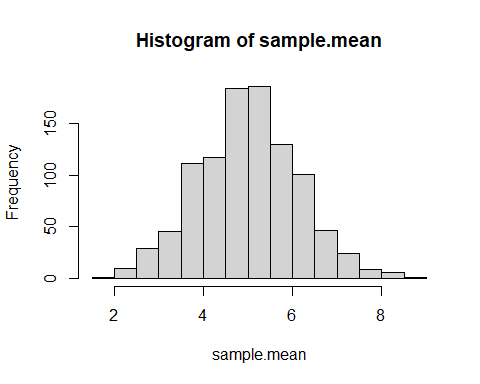
Note here that R uses the *standard deviation* rather than the variance in the definition of the normal distribution.

## Exercise

Determine the sampling distribution of the sample mean and variance of sample of size 20 from a normal distribution with mean 5 and variance 25. To achieve this,

1. Sample 20 values from the proposed normal distribution.
2. Calculate the sample mean and variance from these values.
3. Repeat these two steps 1000 times, storing the results in two vectors.
4. Plot a histogram of the results.

# Initialise the vectors  
sample.mean <- vector("numeric", 1000)  
sample.var <- vector("numeric", 1000)  
  
# Write a loop  
for(i in 1:1000){  
 # Simulate from a normal distribution  
 X <- rnorm(20, mean = 5, sd = sqrt(25))  
   
 # Calculate the statistics of interest  
 sample.mean[i] <- mean(X)  
 sample.var[i] <- var(X)  
}  
  
hist(sample.mean)



hist(sample.var)

