Exercise1A

Hanzhong Shen

2016/10/21

Exercise 1A

Exercise 1.1: Packages and R Studio

1. Navigate to http://cran.r-project.org/ and look for a package called rpart. Have a brief look into its documentation to learn about its functionality

    - **Description:**Recursive partitioning for classification, regression and survival trees. An implementation of most of the functionality of the 1984 book by Breiman, Friedman, Olshen and Stone.

2. Use R Studio to download and install the package rpart

```
install.packages("rpart")
```

3. Load the package using the library command

```
library(rpart)
```

4. Use the R help to query the functioning of the method rpart

```
help("rpart")
```

Exercise 1.2: Variables and basic operations

1. Create two variables a and b and assign values of 3 and 4.5.

```
a <- 3
b <- 4.5
```

2. Query the type of variable a.

```
a
```

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

3. Check whether variable b is a text variable (type character)

```
is.character(a)
```

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

4. Calculate $a^2 + \frac{1}{b}$, $\sqrt{a * b}$, and $\log_2(a)$

```
print(paste("a^2+1/b=", a^2 + 1/b))
print(paste("sqrt(a*b)=", sqrt(a * b)))
print(paste("log(a,2)=", log(a, 2)))
```

```
## [1] "a^2+1/b= 9.22222222222222"
## [1] "sqrt(a*b)= 3.67423461417477"
## [1] "log(a,2)= 1.58496250072116"
```

Exercise 1.3: Variables and basic operations

Create three additional variables as follows:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} B = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} y = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

```
A <- matrix(c(1:8, 10), ncol = 3, byrow = T)
B <- matrix(c(1:8, 10), ncol = 3, byrow = F)
y <- matrix(c(1, 2, 3))
```

1. a * A

```
a * A
```

```
##      [,1] [,2] [,3]
## [1,]    3    6    9
## [2,]   12   15   18
## [3,]   21   24   30
```

2. A * B

```
A %*% B
```

```
##      [,1] [,2] [,3]
## [1,]   14   32   53
## [2,]   32   77  128
## [3,]   53  128  213
```

3. The inverse of matrix A and store the result in a variable invA

```
invA <- solve(A)
```

4. Multiply A and invA and verify that the result is the identity matrix

```
A %*% invA == diag(1, nrow = nrow(A), ncol = ncol(invA))
```

```
##       [,1]  [,2] [,3]
## [1,] FALSE FALSE TRUE
## [2,] FALSE FALSE TRUE
## [3,]  TRUE  TRUE TRUE
```

5. The transpose of matrix B

```
t(B)
```

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

6. Fill the first row of matrix B with ones

```
B[1, ] <- rep(1, times = ncol(B))
```

Exercise 1.4: Variables and basic operations

1. Check the values of variables A, B, and y from exercise 1.3

```
A
```

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

```
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    2    5    8
## [3,]    3    6   10
```

```
y
```

```
##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
```

2. Access the second element in the third row of A and the first element in the second row of B, and compute their product

```r
A[3, 2] * B[2, 1]
```

```
## [1] 16
```

3. Multiply the first row of A and the third column of B

```r
A[1, ] * B[, 3]
```

```
## [1]  1 16 30
```

4. Calculate the least square estimator $\beta = (A^T * A)^{-1} \cdot y$

```r
solve(t(A) %*% A) %*% y
```

```
##            [,1]
## [1,] -0.6666667
## [2,] -0.6666667
## [3,]  1.0000000
```

Exercise 1.5: Using inbuilt functions

The density of the normal distribution with expected value μ and variance σ is given as

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

1. The function dnorm implements the normal distribution in R. Find a way how to compute values of the standard normal distribution using this function.

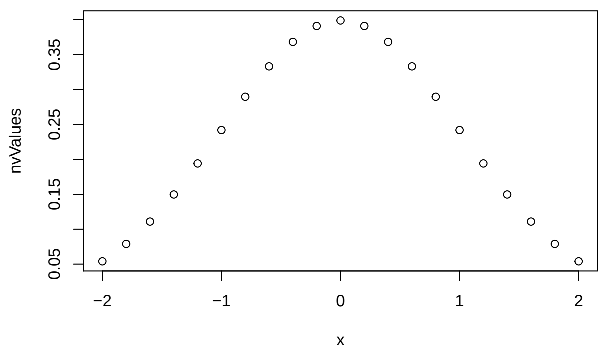
   - UsageL: dnorm(x, mean = 0, sd = 1, log = FALSE)
   - Arguments: Logical, if TRUE, probabilities p are given as log(p).

2. Calculate the value of the standard normal distribution (with $\mu = 0$ and $\sigma = 1$) at x = -2, -1.8, . . . , +2 and store the results in a variable nvValues

```r
x <- seq(-2, 2, by = 0.2)
nvValues <- dnorm(x, 0, 1)
```

3. Create a simple graph of the resulting values using the plot function

```r
plot(x, nvValues)
```



Exercise 1.6: Working with data.frames

1. The UCI Machine Learning repository provides a large number of data sets relating to a variety of domains. Download the data set German Credit from the UCI Repository (use the version german.data as opposed to german.numeric)

   ```
   fileUrl <- "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data"
   ```

2. Load the data into a variable ger using the function read.csv()
3. Note that you will have to set some function parameters to read the data properly
4. Use a text editor for a first examination of the data. Then check the help of read.csv and determine which parameters you have to set.

   ```
   ger <- read.csv(url(fileUrl), header = F, sep = " ")
   ```

5. Query the type of variable ger

   ```
   typeof(ger)
   ```

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

Exercise 1.7: Working with data.frames

The data set does not include column headers. However, a data dictionary is available in the file german.doc on the UCI website.

1. Read the data dictionary. Then identify an appropriate R function to set the column headers (i.e., variable names) in the data set

   ```
   columnName <- c("Status of existing checking account", "Duration in month",
       "Credit history", "Purpose", "Credit amount", "Savings account/bonds",
       "Present employment since", "Installment rate% of disposable income",
       "Personal status and sex", "Other debtors / guarantors",
       "Present residence since", "Property", "Age in year", "Other installment plans",
       "Housing", "Number of existing credits at this bank", "Job",
       "Number of people being liable to provide maintenance for",
       "Telephone", "foreign worker", "classification")
   names(ger) <- columnName
   ```

2. Set all variable names in the data set and save the resulting data set to a CSV file named German-Credit.csv

   ```
   write.csv(ger, file = "GermanCredit.csv")
   ```

3. Use appropriate R functions to answer the following questions

   - What is the most frequent credit purpose?
   - What are the minimal, maximal and average credit amounts?
   - What is the fraction of good and bad credit risks in the data set?

```r
ger$classification <- as.factor(ger$classification)
summary(select(ger, 4:5, classification))
```

```
##     Purpose    Credit amount   classification
##  A43    :280   Min.   :  250   1:700
##  A40    :234   1st Qu.: 1366   2:300
##  A42    :181   Median : 2320
##  A41    :103   Mean   : 3271
##  A49    : 97   3rd Qu.: 3972
##  A46    : 50   Max.   :18424
##  (Other): 55
```