GETTING STARTED WITH THE ASSIGNMENT IN R

We start with reading the credit scoring data from the lectures in R.
The data can be found here.
Suppose you saved the data to a file "credit.txt" in the directory "dm"
on the C drive. To read it into R type (">" denotes the R prompt):

```
> credit.dat <- read.csv("C:/dm/credit.txt")
```

You have now assigned this data set to a variable called "credit.dat"
(" <- " is the assignment symbol in R).

To display its value, just type its name at the command line:

```
> credit.dat
   age married house income gender class
1   22       0     0     28      1     0
2   46       0     1     32      0     0
3   24       1     1     24      1     0
4   25       0     0     27      1     0
5   29       1     1     32      0     0
6   45       1     1     30      0     1
7   63       1     1     58      1     1
8   36       1     0     52      1     1
9   23       0     1     40      0     1
10  50       1     1     28      0     1
```

(the first column are row numbers, and the first row are column names)

"credit.dat" is now an object of type "data.frame". This is similar to (but subtly
different from) a matrix. In any case, you can index a data frame like a matrix.

Select the first row of credit.dat:

```
> credit.dat[1,]
  age married house income gender class
1  22       0     0     28      1     0
```

Select the fourth column of credit.dat:

```
> credit.dat[,4]
 [1] 28 32 24 27 32 30 58 52 40 28
```

Select the element in row 5, column 1:

```
> credit.dat[5,1]
[1] 29
```

Give the distinct values of income, sorted from low to high:

```
> sort(unique(credit.dat[,4]))
[1] 24 27 28 30 32 40 52 58
```

Add all the entries of the sixth column:

```
> sum(credit.dat[,6])
[1] 5
```

Add the entries of each column of credit.dat:

```
> apply(credit.dat,2,sum)
    age married   house  income  gender   class
    363       6       7     351       5       5
```

Add the entries of each row:

```
> apply(credit.dat,1,sum)
  1   2   3   4   5   6   7   8   9  10
 51  79  51  53  63  78 125  91  65  81
```

Select all rows where the first column is bigger than 27:

```
> credit.dat[credit.dat[,1] > 27,]
   age married house income gender class
2   46       0     1     32      0     0
5   29       1     1     32      0     0
6   45       1     1     30      0     1
7   63       1     1     58      1     1
8   36       1     0     52      1     1
```

```
10  50       1     1     28      0     1
```

Construct a vector "x" with the numbers 2,5,10 in that order:

```
> x <- c(2,5,10)
> x
[1] 2  5  10
```

Construct a vector consisting of the numbers 1 through 10:

```
> c(1:10)
 [1]  1  2  3  4  5  6  7  8  9 10
```

Select the *row numbers* of the rows where the first column of credit.dat is bigger than 27:

```
> c(1:10)[credit.dat[,1] > 27]
[1]  2  5  6  7  8 10
```

Draw a random sample of size 5 from the numbers 1 through 10 (without replacement):

```
> index <- sample(10,5)

> index
[1] 5 1 7 4 6
```

Select the corresponding rows:

```
> train <- credit.dat[index,]
> train
  age married house income gender class
5  29       1     1     32      0     0
1  22       0     0     28      1     0
7  63       1     1     58      1     1
4  25       0     0     27      1     0
6  45       1     1     30      0     1
```

Select all rows with row number not in "index":

```
> test <- credit.dat[-index,]
> test

   age married house income gender class
2   46       0     1     32      0     0
3   24       1     1     24      1     0
8   36       1     0     52      1     1
9   23       0     1     40      0     1
10  50       1     1     28      0     1
```

Consult the help page of the function "sample"

```
> help(sample)
```

At the end of a session (and also during a session), save your workspace to a file (choose "Save Workspace" from the file menu). Otherwise all results (the functions you created, etc.) will be lost after you quit R.

Practice exercise 1

Assume we have a classification problem with only 2 classes that are labeled 0 and 1 respectively. Write a function that computes the impurity of a vector (of arbitrary length) of class labels. Use the gini-index as impurity measure. Do not use a loop structure in your function, this is not necessary.

Example:

```
> y <- c(1,0,1,1,1,0,0,1,1,0,1)
> y
 [1] 1 0 1 1 1 0 0 1 1 0 1

> impurity(y)
[1] 0.2314050
```

If you are not working in Rstudio, to create the function, use:

```
> fix(impurity, editor="Notepad")
```

This will open a Notepad window. Type in the function definition, save the file and exit the editor.

Practice exercise 2

Write a function "bestsplit(x,y)" that computes the best split value on a numeric attribute x.
Here x is a vector of numeric values, and y is the vector of class labels (assume there
are only two classes, coded as 0 and 1). x and y must be of the same length: y[i] is the class
label of the i-th observation, and x[i] is the corresponding value of attribute x.
Only consider splits of type "x <= c" where "c" is the average of two consecutive values of x in the sorted order.
So one child contains all elements with "x <= c" and the other child contains all elements with "x > c".
The best split is the split that achieves the highest impurity reduction.

Example (best split on income):

```
> bestsplit(credit.dat[,4],credit.dat[,6])
[1] 36
```

Hint: Clever use of "subscripting" (selecting elements of vectors and matrices) is important in R. For example,
y[x > 29] produces a vector with all elements of y whose corresponding x-element (that is the element of x with
the same index) is bigger than 29. More formally: y[x > 29] = {y[i]: x[i] > 29}. The result is a vector, not a set, i.e.
duplicate values may occur. Just try it!

Hint: Example of how to determine candidate split points

```
> income.sorted <- sort(unique(credit.dat[,4]))
> income.sorted
 [1] 24 27 28 30 32 40 52 58
> income.splitpoints <- (income.sorted[1:7]+income.sorted[2:8])/2
> income.splitpoints
 [1] 25.5 27.5 29.0 31.0 36.0 46.0 55.0
```

Note: use the "brute force" approach, i.e. don't implement the "segment borders" algorithm.