Introduction to R (II) - Exercises 20-11-2019

Daniel Zanchetta and Lais Zanchetta

26/11/2019

# Lais Silva Almeida Zanchetta and Daniel Ferreira Zanchetta

# Exercise 1

x <- rnorm(n=1000,mean=5,sd=4)

## with sort

xsrt <- sort(x,decreasing=TRUE)  
xbig <- c(xsrt[1:5])  
xbig

## [1] 19.82430 15.41067 15.32074 15.06895 14.98362

## SORT - Min and Max

min(xsrt,na.rm = TRUE)

## [1] -11.27482

max(xsrt,na.rm = TRUE)

## [1] 19.8243

## with order

x2<-x[order(-x)[1:5]] #the order indexed will get the 5 first values in descending order

## ORDER - Min and Max

min(x2,na.rm = TRUE)

## [1] 14.98362

max(x2,na.rm = TRUE)

## [1] 19.8243

### Are the results repeatable between runs? How would you make the run repeatable?

R.: No, they are not repeatable. They can be repeatable if the function is normalized (N(0,1))

# Exercise 2

A <- sample(1:10000, 1000, replace=FALSE)  
B <- sample(1:10000, 1000, replace=FALSE)

## A Union B

U <- union(A,B)  
length(U)

## [1] 1885

## A intersection B

I <- intersect(A,B)  
length(I)

## [1] 115

## A diferents of B

D <- setdiff(A,B)  
length(D)

## [1] 885

# Exercise 3

bcnpisos <- read.table("C:/Users/Daniel/Documents/Certificados & Faculdade/UPC Master Big Data/Data Analytics/Aula 2 - 13-11/exer\_Descr/bcn\_pisos.txt", header=TRUE)

## A) As the parameters to determine the intervals were not informed, we assume for the exercise the interquartile range

Q1 <- summary(bcnpisos$Superf)[2]  
Q3 <- summary(bcnpisos$Superf)[4]  
IQR <- Q3 - Q1  
Superf <- cut(bcnpisos$Superf,c(0,Q1,IQR,Q3), labels= c("Peque","Grande","MuyGrande"))  
table(Superf)

## Superf  
## Peque Grande MuyGrande   
## 0 584 776

## B)

#install.packages("dummies")  
library("dummies")

## dummies-1.5.6 provided by Decision Patterns

DistDum <- dummy(bcnpisos$Dist)

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =  
## FALSE): non-list contrasts argument ignored

# Exercise 4

wines <- read.table("C:/Users/Daniel/Documents/Certificados & Faculdade/UPC Master Big Data/Data Analytics/Aula 3 - 20-11/wine.data",sep = ",") #import file

## A) Column names

names(wines)=c("label","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols","Flavanoids","Nonflavanoid phenols","Proanthocyanins","Color intensity","Hue","OD280/OD315 of diluted wines","Proline")

## B) Force the label (first column) to be a factor

wines$label <- as.factor(wines$label)

## C) For train control, use 10-fold Cross Validation protocol

#install.packages("caret")  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

tC <- trainControl(method="cv", number=10)

## D) Try the same model we used during the class (linear svm).Try different values for the cost parameters and print the resulting model. What is the best C?

#install.packages("e1071")  
library("e1071")  
pG <- expand.grid(C = c(0.01,0.05,0.1,0.25,0.5,0.75,1))  
modelSVM <-train(label~., data = wines, method = "svmLinear", trControl = tC, tuneGrid=pG)  
print(modelSVM)

## Support Vector Machines with Linear Kernel   
##   
## 178 samples  
## 13 predictor  
## 3 classes: '1', '2', '3'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 160, 162, 160, 160, 160, 160, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 0.01 0.9836257 0.9753060  
## 0.05 0.9891813 0.9836781  
## 0.10 0.9773757 0.9656637  
## 0.25 0.9655702 0.9480923  
## 0.50 0.9600146 0.9396018  
## 0.75 0.9489035 0.9228576  
## 1.00 0.9489035 0.9228576  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was C = 0.05.

### R.: The best C to train the model is C = 0.5.

## Block II

#linear model using lm() function  
irisTwoVariables <- iris[,c("Sepal.Length","Sepal.Width")]  
irislm <- lm(Sepal.Length~Sepal.Width, data = irisTwoVariables)  
irislm

##   
## Call:  
## lm(formula = Sepal.Length ~ Sepal.Width, data = irisTwoVariables)  
##   
## Coefficients:  
## (Intercept) Sepal.Width   
## 6.5262 -0.2234

summary(irislm)

##   
## Call:  
## lm(formula = Sepal.Length ~ Sepal.Width, data = irisTwoVariables)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.5561 -0.6333 -0.1120 0.5579 2.2226   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.5262 0.4789 13.63 <2e-16 \*\*\*  
## Sepal.Width -0.2234 0.1551 -1.44 0.152   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8251 on 148 degrees of freedom  
## Multiple R-squared: 0.01382, Adjusted R-squared: 0.007159   
## F-statistic: 2.074 on 1 and 148 DF, p-value: 0.1519

#Libraries  
library("caret")  
library("parallel")  
  
#K Fold Cross Validation  
  
#Decoupling the k folds number  
kNumber<-10  
method <- "lm"  
  
kfolds <- createFolds(iris$Sepal.Length, k=kNumber)  
tC <- trainControl(method="cv",number = kNumber,allowParallel = TRUE)  
  
modelKCV<-function(irisTwoVariables,kfolds,tC,method,seq){  
 #prepare a test data K-1 other folds  
 irisWithoutKFolds <- irisTwoVariables[-kfolds[[seq]],]  
 modelLM<-train(Sepal.Length~., data = irisWithoutKFolds, method = method,trControl = tC)  
 modelLM$results$Rsquared  
}  
  
#This method is not parallelized, and aims to store in a vector with kNumber positions the Rsquared for each train execution  
RsquaredNotParal <- sapply(1:kNumber,FUN=function(seq){  
modelKCV(irisTwoVariables,kfolds,tC,method,seq)  
})  
  
#Return the average of the accuracy obtained (accuracy = linear model Rsquared obtained)  
mean(RsquaredNotParal)

## [1] 0.07790036

#Preparing the parameters for the parallelized apply: detect cores and cluster export (to make available in the workspace)  
cl <- makeCluster(detectCores())  
clusterExport(cl=cl,varlist=list("irisTwoVariables","kfolds","tC","method","seq","modelKCV","train"))  
  
#Now, this method is parallelized, and aims to store in a vector with kNumber positions the Rsquared for each train execution  
RsquaredParal<-parSapply(cl,1:kNumber,FUN=function(seq){  
modelKCV(irisTwoVariables,kfolds,tC,method,seq)  
})  
  
mean(RsquaredParal)

## [1] 0.08058632