```
##############################################################################
######## One possible way to solve Homework 4 - Bioinformatics 2010-2011 #######
##############################################################################

#setting the dir where everything is stored
setwd("C:/Zkristel 1jan2010/Homework 4 Bioinf")

#exploring the data
pedigree.ped <- read.table(file="data.dat",header=TRUE,sep=" ")
pedigree.ped[1:10,1:10]
ncol(pedigree.ped)
phenotype.phe <- read.table(file="pheno.dat",header=TRUE,sep=" ")
phenotype.phe[1:10,]

#make subselection of parents only
pedigree.ped <- subset(pedigree.ped, FID==0 & MID==0)
pedigree.ped[1:10,1:10]
nrow(pedigree.ped)
phenotype.phe <- subset(phenotype.phe, phenotype.phe[,1] %in% pedigree.ped[,2])
phenotype.phe[1:10,]
nrow(phenotype.phe)

#pedigree and pheno data
SNPphe <- cbind(pedigree.ped[,7:(ncol(pedigree.ped))],phenotype.phe[,3])

#prepare the data for use with SNPassoc
library(genetics)
SNP012 <- SNPphe[,1:(ncol(SNPphe)-1)]
SNPgenotype <- as.data.frame(SNP012) #no missing data assumed in the following
SNPgenotype[SNP012==0]<-"A/A"
SNPgenotype[SNP012==1]<-"A/B"
SNPgenotype[SNP012==2]<-"B/B"
SNPdata <- makeGenotypes(data.frame(SNPgenotype))

library(SNPassoc)
SNPandPHE <- as.data.frame(cbind(SNPdata,SNPphe[,ncol(SNPphe)]))
SNPandPHE <- setupSNP(SNPandPHE, 1:(ncol(SNPphe)-1), sort = FALSE, info, sep =
"/") #manual p4
names(SNPandPHE)[1] <- "status"

#perform first quality checks on the data
summary(SNPandPHE)  #manual p5
class(summary(SNPandPHE))
plotMissing(SNPandPHE)  #manual p6
res<-tableHWE(SNPandPHE)   #manual p6
res

#perform a GWA analysis     # manual p9
All<-WGassociation(status~1,data=SNPandPHE,model="all")
summary(All)
print(All)
pvalAll<-pvalues(All)
WGstats(All)
plot(All)
labels(All)

#more elaborate ways to assess genome-wide significance   # manual p12
All.perm<-scanWGassociation(status ~1, data=SNPandPHE,model="log-add", nperm=2)
  #take 1000 permutations if possible
res.perm<- permTest(All.perm)
print(res.perm)
plot(res.perm)
```

```
res.perm.rtp<- permTest(All.perm,method="rtp",K=20)
print(res.perm.rtp)

#interaction analysis
int.cod <- interactionPval(status~1, data=SNPandPHE,model="codominant")
int.dom <- interactionPval(status~1, data=SNPandPHE,model="dominant")
int.rec <- interactionPval(status~1, data=SNPandPHE,model="recessive")
int.ove <- interactionPval(status~1, data=SNPandPHE,model="overdominant")
int.log <- interactionPval(status~1, data=SNPandPHE,model="log-additive")
print(int.log)
plot(int.log)

#basic random forests approach
library(randomForest)
as.data.frame(SNPandPHE)
set.seed(1969)
data.rf <- randomForest(status ~ ., data=SNPandPHE, mtry=3,importance=TRUE,
na.action=na.omit)
print(data.rf)
round(importance(data.rf), 2) # Show "importance" of variables: higher value
means more import

#flexible function to perform machine learning applications
library(MLInterfaces)
help(MLearn)
set.seed(1969)
kp = sample(1:nrow(SNPandPHE), size=75*nrow(SNPandPHE)/100)
data.rf = MLearn(status~., data=crabs, randomForestI, kp, ntree=600
impV = getVarImp(data.rf)
plot(impV, n=15)
```