#zmena úloziste

getwd()

#MyDirectory <- "C:/Users/Client/Desktop/Ostatn?/Skola/University of Turku/R"

#TurkuDirectory<-'\\\\utuhome.utu.fi\\tokypt\\R'

setwd(MyDirectory)

#Exercise I.1

#a) Create a suitably named object that is a vector consisting of 10 numbers

#b) Perform all the basic arithmetic operations with your vector

#c) Extract the fourth and seventh elements from your vector and store these in an object. What class does this second object have?

#d) Calculate the mean value of your vector. Make a logical vector stating for each element whether it smaller than the vector's mean value or not. How many values are below the mean?

#e) Construct a matrix out of your vector

#f) Extract the values in the second row and the values in the second column of your matrix

#g) Extract the value of the first row and second column of your matrix

#h) Turn your matrix of values into a matrix of logicals stating for all values in your matrix whether they are larger than the mean value or not. How many values are below the mean?

#prirazuju objekt>nasobim>delim>odmocnuju>umocnuju

vektor10<-c(10:19)

vektor10\*2

vektor10/2

sqrt(vektor10)

vektor10^2

#vytvarim objekt z ctvrte a sedme hodnoty vektoru

element47<-vektor10[c(4,7)]

class(element47)

#class: integer (cele cislo)

mean(vektor10)

#aritmeticky prumer je 15

vektor10<mean(vektor10)

#5 values are below, 5 are above

matrix((vektor10),2)

matrix10<-matrix((vektor10),2)

#vytvoril jsem matrix s 10 cisly

matrix10[nrow=2,ncol=2]

matrix10[nrow=1,ncol=2]

#exktraktuju hodnoty z matrixu

matrix10<mean(matrix10)

#5 values are below

#Exercise I.2

#a) Get help for the function sum() which we encountered in chapter 1. Review the sections of the help page and run the examples.

#b) Use sum() to calculate the sum of the vector c(1,2,NA,4)

#c) Use sum() to calculate the sum of myMatrix's first row and of its first column

#d) Use the help page to find a related function that allows you to sum rows and columns of a matrix and use it for myMatrix

?sum()

sum(1:5)

sum(1,2,3,4,5)

sum(1:2,3:5)

sum(1:5,NA)

sum(1:5,NA,na.rm=TRUE)

#priklady z helpu

sum(c(1,2,NA,4))

sum(c(1,2,NA,4),na.rm=TRUE)

#pocitam soucet vektoru

myMatrix<-matrix(c(1,2,3,4),2)

sum(myMatrix[nrow=1,ncol=1])

#vytvarim matrix, extraktuju z nej specificke cislo

sum(myMatrix[1,])

sum(myMatrix[,2])

sum(myMatrix[1,],myMatrix[,2])

summary(myMatrix[nrow=1,ncol=1])

colSums(myMatrix)

rowSums(myMatrix)

#dalsi podobna funkce

#Exercise I.3

#a) Create and read in the MyData file (sections 3.2 and 3.3)

#b) Calculate the mean mass and mean length over all individuals

#c) Use logicals and indexing to calculate the mean mass of blue and of red individuals separately.

#TIP. approach this problem in steps. First work out how to use logicals to get a vector where the blue individuals are TRUE, then how to use this logical vector and indexing to obtain a vector of mass of blue individuals and lastly to calculate the mean of this vector.

myData<-read.csv2("myData.csv")

mean(myData$mass)

mean(myData$length)

#prumery

#c

myData$colour == "blue"

modra<-myData$colour == "blue"

#vytvoril jsem vektor modr?

myData$mass[c(modra)]

mean(myData$mass[c(modra)])

myData$colour == "red"

cervena<-myData$colour == "red"

myData$mass[c(cervena)]

mean(myData$mass[c(cervena)])

#to samy jina barva

#Exercise I.4

#a) Copy-paste the following information to a Notepad file and save it in your working directory. Import the data into R without changing any of the text.

#b) Read the data into R in such a way that habitat is coded as a factor. Investigate the differences in how the variables are interpreted using some of the functions introduced in this part to explore your data.

ctyrka<-read.csv("ex\_I\_4\_copied.txt", stringsAsFactors = TRUE)

str(ctyrka)

summary(ctyrka)

hist(ctyrka$count)

plot(ctyrka$count)

boxplot(ctyrka$count~ctyrka$Habitat)

head(ctyrka)

tail(ctyrka)

#Exercise I.5

#a) Make two vectors, with one of these containing one or more NA, and plot a histogram of each. What happens to the NA values?

#b) Plot one of your vectors against the other

#c) Use the function plot() to plot a "histogram" of one of your vectors (study the help page). What does it do?

vec1<-c(1,2,NA,4,NA,6,7)

vec2<-c(1,2,3,4,5,6,7)

hist(vec1)

hist(vec2)

#NA values get ignored

plot(vec1~vec2)

plot(vec1, vec2)

#NA values from vec1 remove values from vec2

plot(vec1, type ="h")

hist(vec1, breaks = c(0.5,1.5,2.5,3.5,4.5,6.5,7.5))

#EXERCISES II

#EXERCISE II.1

#a) Calculate the mean, SD and sample size of length for each colour in myData

#b) Use aggregate() to calculate the means of both mass and length in myData for each colour in one command line

#c) Code the construction of the following data.frame

#and use tapply() to calculate the mean, SD and sample size for each experimental group.

myData<-read.csv2("myData.csv")

tapply(myData$length,myData$colour,mean)

tapply(myData[,"length"],myData$colour,sd)

tapply(myData[,"length"],myData$colour,length)

aggregate(length~colour,myData,mean)

aggregate(mass~colour,myData,mean)

aggregate(cbind(length,mass)~colour,myData,mean)

?aggregate

#koukni na formulu v helpu

trial<-1:10

experiment<-c(rep("control",5),rep("treatment",5))

measure<-c(1,3,4,3,3,4,5,8,8,9)

pokusy<-data.frame(trial,experiment,measure)

#bez objektu: data.frame(trial=1:10,experiment=c(rep..

#vytvarim data.frame

tapply(pokusy$measure,pokusy$experiment,mean)

tapply(pokusy$measure,pokusy$experiment,sd)

tapply(pokusy$measure,pokusy$experiment,length)

#Exercise II.2

#Write a script which

#(a) imports the datafile "ExperimentalData.txt" which you can find in Moodle

#(b) Performs some checks the data import was successful

#(c) Recodes the 4 groups to:

# group Feeding.treatment Parasite.treatment

#1 not fed removed

#2 not fed retained

#3 fed removed

#4 fed retained

#(d) Extracts all data where the parasite was removed

#(e) Extracts data on all individuals which were "not fed" and parasite "retained" as well as "fed" and "removed" (one data.frame)

#(e) Calculates the mean for "fed" and "not fed" treatment groups

#(f) Calculates the variance for all each feeding and parasite treatment using aggregate()

?read.table

ExperimentalData<-read.csv("ExperimentalData.txt")

ExperimentalData

head(ExperimentalData)

tail(ExperimentalData)

str(ExperimentalData)

feeding.treatment<-c("fed","not fed")

parasite.treatment<-c("removed","retained")

ExperimentalDataRECODED<-data.frame(ExperimentalData,feeding.treatment,parasite.treatment)

#uploaduju nov? r?dky do dat

feeding.treatment<-ifelse(ExperimentalDataRECODED$group == 1 | (ExperimentalDataRECODED$group == 2),"not fed", "fed")

parasite.treatment<-ifelse(ExperimentalDataRECODED$group == 1|(ExperimentalDataRECODED$group == 3),"removed","retained")

#BETTER: ExperimentalDataRECODED$feeding.treatment<-ifelse(ExperimentalData$group==(1|2),"not fed","fed")

#prekodovani

ExperimentalDataRECODED<-data.frame(ExperimentalData,feeding.treatment,parasite.treatment)

#znovu naloadeni preformatovanejch dat

subset(ExperimentalData,parasite.treatment == "removed")

#(d) Extracts all data where the parasite was removed

#asi i takhle

#subset(ExperimentalDataRECODED,((Feeding.treatment=="not fed)&(Parasite.treatment=="retained))|(Feeding.treatment=="fed")&(Parasite.treatment=="removed")))

retnotfed<-data.frame(subset(ExperimentalDataRECODED,parasite.treatment == "retained"&feeding.treatment =="not fed"))

remfed<-data.frame(subset(ExperimentalDataRECODED,parasite.treatment == "removed"&feeding.treatment =="fed"))

parascomb<-rbind(retnotfed,remfed)

#Extracts data on all individuals which were "not fed" and parasite "retained" as well as "fed" and "removed" (one data.frame)

tapply(ExperimentalDataRECODED$size,ExperimentalDataRECODED$feeding.treatment,mean)

#Calculates the mean for "fed" and "not fed" treatment groups

#NEBO: mean(subset(ExperimentalData, Feeding.treatment=="not fed", size, drop=TRUE))

variance<-aggregate(size~parasite.treatment+feeding.treatment,ExperimentalDataRECODED,var)

#???Calculates the variance for all each feeding and parasite treatment using aggregate()

#Exercise II.3

#Use the data in "Fruit\_tasting\_data\_2018.txt" from Moodle

#Create four additional variables (i.e. "columns" in the data.frame holding the data) : score1.apples and score2.apples which hold score1 and score2 respectively only for apples, and score1.strawberies and score2.strawberries which contain score1 and score2 respectively but only for strawberries.

#a) Calculate the mean for each of these four variables

#b) Use subset() to create two datasets. One for apples and one for strawberries, but without any NA values

#c) Make an object of class numeric containing score1 values for apples

#d) Make an additional variable containing the name of each fruit tasted in the dataset, but as factors

ftdata<-read.table("Fruit\_tasting\_data\_2018.txt")

#filling the vectors with N values

score1.apples<-ifelse(ftdata$Fruit == "apple",ftdata$Score1, NA)

score2.apples<-ifelse(ftdata$Fruit == "apple",ftdata$Score2, NA)

score1.strawberries<-ifelse(ftdata$Fruit == "strawberry",ftdata$Score1, NA)

score2.strawberries<-ifelse(ftdata$Fruit == "strawberry",ftdata$Score2, NA)

#sorting the vectors out

newftdata<-data.frame(ftdata,score1.apples,score2.apples,score1.strawberries,score2.strawberries)

#dataframe with new variables sorted out.

tapply(newftdata$score1.apples,newftdata$Fruit,mean, na.rm=T)

tapply(newftdata$score2.apples,newftdata$Fruit,mean, na.rm=T)

tapply(newftdata$score1.strawberries,newftdata$Fruit,mean, na.rm=T)

tapply(newftdata$score2.strawberries,newftdata$Fruit,mean, na.rm=T)

#slo to i nejak takhle: aggregate(cbind(Score1,Score2)~Fruit,data = ftdata, mean)

#Calculate the mean for each of these four variables

jabka<-subset(newftdata, Fruit=="apple", Score1|Score2)

jahody<-subset(newftdata, Fruit=="strawberry", Score1|Score2)

jabkanumericky<-as.numeric(newftdata$score1.apples)

class(jabkanumericky)

#Make an object of class numeric containing score1 values for apples

ffruit<-as.factor(newftdata$Fruit)

ffruitdata<-data.frame(newftdata,ffruit)

class(ffruit)

summary(ffruitdata)

#Make an additional variable containing the name of each fruit tasted in the dataset, but as factors

#Exercise II.4

#Write a script which

#a) defines an object that is a matrix with 10 columns containing the numbers 1 to 100

#b) defines another object that is a matrix with 10 columns containing the numbers 1 to 100 such that the numbers increment by 1 for each row across the column (i.e. row 1 is 1, 2, 3, etc)

#c) names the rows and columns of these two objects

#d) Extracts the third row of each object

#e) Calculates the mean of each column in each object

desetsto<-matrix(1:100,10)

desetstorad<-matrix(1:100,10,byrow = TRUE)

rady<-c("rada1","rada2","rada3","rada4","rada5","rada6","rada7","rada8","rada9","rada10")

sloupy<-c("sloup1","sloup2","sloup3","sloup4","sloup5","sloup6","sloup7","sloup8","sloup9","sloup10")

rownames(desetsto)<-rady

colnames(desetsto)<-sloupy

colnames(desetstorad)<-sloupy

rownames(desetstorad)<-rady

#prejmenovavam matrix

desetsto["rada3",]

desetstorad[3,]

#extraktuju 3radu

colMeans(desetsto)

colMeans(desetstorad)

#calculating means of columns

#Exercise II.5

#a) Create the following data.frame

#Plot Species SoilType CationValue

#P1 Acer Wet 10

#P2.2 Acer Wet 12

#P2.3 Quercus Dry 4

#P3 Quercus Dry 3

#P10 Acer Dry 1

#P11 Quercus Wet 7

#P12 Quercus Unknown 2

#Use subset() to

#b) Make an object containing only information for Quercus

#c) Make an object containing only information for Quercus in dry soil types

#d) Make an object containing only information for Quercus in dry and wet soil types

#e) Make an object containing only information for Quercus in dry and Acer in dry and wet soil types

#f) Make an object containing only the soil type and cation values for Acer

#g) Make a vector of cation values for dry soil type

?data.frame

CationValue<-c(10,12,4,3,1,7,2)

SoilType<-c("Wet","Wet","Dry","Dry","Dry","Wet","Unknown")

Species<-c("Acer","Acer","Quercus","Quercus","Acer","Quercus","Quercus")

Plot<-c("P1","P2.2","P2.3","P3","P10","P11","P12")

stromy<-data.frame(Plot,Species, SoilType,CationValue)

#vytvoril jsem dataframe stromy

#TAKY: Quercus<-subset(stromy,Species=="Quercus")

Quercus<-subset(stromy,Species=="Quercus",c(Plot,SoilType,CationValue))

QuercusDRY<-subset(stromy,Species=="Quercus"&SoilType=="Dry",c(Plot,SoilType,CationValue))

QuercusDRYWET<-subset(stromy,Species=="Quercus"&SoilType=="Dry"|SoilType=="Wet",c(Plot,SoilType,CationValue))

#taky muzu subset(stromy, Species=="Quercus"&SoilType!="Unknown",c(Plot, SoilType,CationValue))

QuercusDRYacerDRYWET<-subset(stromy,c(Species=="Quercus"&SoilType=="Dry")|c(Species=="Acer"&SoilType=="Dry")|c(Species=="Acer"&SoilType=="Wet"),c(Plot,Species,SoilType,CationValue))

AcerSOILCAT<-subset(stromy,Species=="Acer",c(SoilType,CationValue))

#vytvoril jsem subsety na zaklade instrukci

stromy[SoilType=="Dry","CationValue"]

subset(stromy,SoilType=="Dry",CationValue,drop=T)

#a vector of cation values for dry soil types

#EXERCISES III

#Exercise III.1

#Use the data in "Fruit\_tasting\_data\_2018.txt" from Moodle

#a) Pimp your plot, step by step (one line of code for each addition; remember that you can copy-paste your line of code and then add the next step to it)

#. Plot "score 2" as a function of "score 1"

#. Rename the axes

#. Change the symbol plotted

#. Increase point size of the symbol plotted

#. Increase font size for the axes

#. Change the y axis so that the minimum value is 0 and the maximum is 10

#. Change the color of the points so that they are different for apples and for strawberries

#. Add a text on top of the figure

#. Add a legend

#b. You obtain the extra scores 2.2 and 4 for the apple Gyllenkroks Astrakan. Add this to the data object in R. Redraw the plot

#c. Code the exporting of the final plot as a PDF. Also export the plot as a PDF using the R Studio "Export" option. Are these two options producing the same outcome?

ftdata<-read.table("Fruit\_tasting\_data\_2018.txt")

plot(ftdata$Score1~ftdata$Score2)

plot(ftdata$Score1~ftdata$Score2,xlab="Score2",ylab="Score1")

plot(ftdata$Score1~ftdata$Score2,xlab="Score2",ylab="Score1",pch=11)

plot(ftdata$Score1~ftdata$Score2,xlab="Score2",ylab="Score1",pch=11,cex=0.7)

plot(ftdata$Score1~ftdata$Score2,xlab=list("Score2",cex=1.5),ylab=list("Score1",cex=1.5),pch=11,cex=0.7)

plot(ftdata$Score1~ftdata$Score2,xlab=list("Score2",cex=1.5),ylab=list("Score1",cex=1.5),pch=11,cex=0.7,ylim=c(0,10))

#vsechny kroky az do barev

ftdata$Fruit<-as.factor(ftdata$Fruit)

barvy1<-c("green2","red3")

plot(ftdata$Score1~ftdata$Score2,xlab=list("Score2",cex=1.5),ylab=list("Score1",cex=1.5),pch=11,cex=0.7,ylim=c(0,10),col=barvy1[ftdata$Fruit])

#prebarvil jsem body

plot(ftdata$Score1~ftdata$Score2,xlab=list("Score2",cex=1.5),ylab=list("Score1",cex=1.5),pch=11,cex=0.7,ylim=c(0,10),col=barvy1[ftdata$Fruit],main="X-Y plot of two scores given to apples and strawberries")

legend(x = "topright",pch=11,cex=1, col = c("red3","green2"), legend=unique(ftdata$Fruit))

#plot is finished

ftdatagyl<-rbind(ftdata,data.frame(Variety="Gyllenkroks Astrakan", Fruit="apple",Score1=2.2,Score2=4))

plot(ftdatagyl$Score1~ftdatagyl$Score2,xlab=list("Score2",cex=1.5),ylab=list("Score1",cex=1.5),pch=11,cex=0.7,ylim=c(0,10),col=barvy1[ftdatagyl$Fruit],main="X-Y plot of two scores given to apples and strawberries")

legend(x = "topright",pch=11,cex=1, col = c("red3","green2"), legend=unique(ftdata$Fruit))

#redrawing the plot

pdf(file="ExportCODEFruit.pdf",paper="a4")

plot(ftdatagyl$Score1~ftdatagyl$Score2,xlab=list("Score2",cex=1.5),ylab=list("Score1",cex=1.5),pch=11,cex=0.7,ylim=c(0,10),col=barvy1[ftdatagyl$Fruit],main="X-Y plot of two scores given to apples and strawberries")

legend(x = "topright",pch=11,cex=1, col = c("red3","green2"), legend=unique(ftdata$Fruit))

dev.off()

#exportovano skrz R

#Exercise III.2

#Use the data in "Fruit\_tasting\_data\_2018.txt" from Moodle

#(a) Calculate the mean and 95% quantile interval for score1 and score2 for apples and strawberries. The 95% quantile interval consists of 2 values. A lower and upper value. The 95% quantiles means that 95% of the data lies between these two values. The function quantile() is useful here (think carefully about which values to use as probability prob argument for this function).

#(b) Create a barplot of the mean score1 for apples and strawberries with a line indicating the 95% quantiles (i.e. the line goes from the lower value to the upper value, TIP: read carefully section 8.1 for this exercise). Make the plot look as good as you can.

#(c) Use par() to plot two panels in one graph and make a 2-panel plot showing the average score1 and score2 of apples and strawberries with their 95% quantile interval respectively. That is, one panel with a barplot of score1 and score2 of apples and the second panel showing barplot of score1 and score2 of strawberries. Something like this:

ftdata<-read.table("Fruit\_tasting\_data\_2018.txt")

#loaded up

means.per.fruit<-aggregate(cbind(Score1,Score2)~Fruit,data = ftdata, mean)

meanscore1<-means.per.fruit[,2]

meanapple<-(means.per.fruit[1,])[,-1]

meanstraw<-(means.per.fruit[2,])[,-1]

#means only numbers

subapp<-subset(ftdata,Fruit=="apple")

substraw<-subset(ftdata,Fruit=="strawberry")

kvantilapp1<-quantile(subapp$Score1,probs=c(0.025,0.975))

kvantilstraw1<-quantile(substraw$Score1,probs=c(0.025,0.975))

kvantilapp2<-quantile(subapp$Score2,probs=c(0.025,0.975))

kvantilstraw2<-quantile(substraw$Score2,probs=c(0.025,0.975))

#vytvoril jsem kvantil

kvantilapp1

windows(10,10)

barscore1<-barplot(meanscore1,col=c("green4","red4"),density=c(30,40),xlab=list("Fruit",cex=1.5),ylab=list("Scores",cex=1.5),ylim=c(0,10))

text(barscore1,0.5,means.per.fruit$Fruit)

segments(x0=0.7,y0=kvantilapp1[1],x1=0.7,y1=kvantilapp1[2],lwd=2)

barscore1

segments(x0=1.9,y0=kvantilstraw1[1],x1=1.9,y1=kvantilstraw1[2],lwd=2)

#mean Score 1s with their quantile segments

par(mfrow = c(1,2))

barapp<-barplot(as.numeric(meanapple),col=c("green4","lightgoldenrod2"),density=c(30,35),xlab=list("Apples",cex=1.5),ylab=list("Score",cex=1.5),ylim=c(0,7),names.arg = c('First','Second'),main = '(a)')

segments(x0=0.7,y0=kvantilapp1[1],x1=0.7,y1=kvantilapp1[2],lwd=2)

segments(x0=1.9,y0=kvantilapp2[1],x1=1.9,y1=kvantilapp2[2],lwd=2)

#apple barplot

barstraw<-barplot(as.numeric(meanstraw),col=c("red2","red4"),density=c(30,35),xlab=list("Strawberries",cex=1.5),ylab=list("Score",cex=1.5),ylim=c(0,7), names.arg = c('First','Second'),main = '(b)')

#segments(x0=0.7,y0=kvantilstraw1[1],x1=0.7,y1=kvantilstraw1[2],lwd=2)

#segments(x0=1.9,y0=kvantilstraw2[1],x1=1.9,y1=kvantilstraw2[2],lwd=2)

segments(x0=barstraw,y0=c(kvantilstraw1[1],kvantilstraw2[1]),x1=barstraw,y1=c(kvantilstraw1[2],kvantilstraw2[2]))

#strawberries barplot

#Exercise III.3

#You learn from Wikipedia (https://en.wikipedia.org/wiki/Standard\_error) that the standard error is defined as the standard deviation divided by the square root of the sample size.

#NE: calc.SE<-function(x) {sd(x)/sqrt(length(x))}

calc.SE<-function(x, na.rm=FALSE){sum(sd(x,na.rm=na.rm))/sqrt(length(na.omit(x)))}

calc.SE(1:10)

calc.SE(c(1:10,NA))

calc.SE(c(1:10,NA),na.rm = TRUE)

#function to calc Standard Error

df.year<-data.frame(Year=c(rep(1,5),rep(2,5)), Response=c(4.256898, 7.103097, 5.345175, 3.259264, 4.327387, 9.995211, 8.621862, 10.499859,10.032349, 8.821877))

aggregate(Response~Year,df.year,sd)

aggregate(Response~Year,df.year,mean)

aggregate(Response~Year,df.year,calc.SE)

tapply(df.year$Response,df.year$Year,sd)

tapply(df.year$Response,df.year$Year,mean)

tapply(df.year$Response,df.year$Year,calc.SE)

#b)Calculate the sample size, mean and SE of Response in both years in using both aggregate and tapply.

df.year<-rbind(df.year, data.frame(Year=c(rep(1,2),rep(2,2)), Response=c(4.33, NA, 10.1, NA)))

aggregate(Response~Year,df.year,sd)

extrojka1<-aggregate(Response~Year,df.year,length)

extrojka2<-aggregate(Response~Year,df.year,mean)

extrojka3<-aggregate(Response~Year,df.year,calc.SE)

tapply(df.year$Response,df.year$Year,function(x) sum(complete.cases(x)))

tapply(df.year$Response,df.year$Year,sd,na.rm = TRUE)

tapply(df.year$Response,df.year$Year,mean,na.rm = TRUE)

tapply(df.year$Response,df.year$Year,calc.SE,na.rm = TRUE)

extrojka.df<-data.frame(Year=unique(df.year$Year),n=extrojka1[,2],mean=round(extrojka2[,2],2),SE=round(extrojka3[,2],2))

write.table(extrojka.df, "extrojka.doc", quote=F, row.names = F, sep=";")

#Exercise III.4

ExperimentalData<-read.csv("ExperimentalData.txt")

feeding.treatment<-c("fed","not fed")

parasite.treatment<-c("removed","retained")

ExperimentalDataRECODED<-data.frame(ExperimentalData,feeding.treatment,parasite.treatment)

ExperimentalDataRECODED$feeding.treatment<-ifelse(ExperimentalData$group==1|(ExperimentalDataRECODED$group == 2),"not fed","fed")

ExperimentalDataRECODED$parasite.treatment<-ifelse(ExperimentalData$group==1|(ExperimentalDataRECODED$group == 3),"removed","retained")

#pripravuju experimentaldata

means.exp<-aggregate(size~group,data = ExperimentalDataRECODED, mean)

subremfed<-subset(ExperimentalDataRECODED,parasite.treatment == "removed"&feeding.treatment =="fed",na.rm=T)

subremnfed<-subset(ExperimentalDataRECODED,parasite.treatment == "removed"&feeding.treatment =="not fed",na.rm=T)

subretfed<-subset(ExperimentalDataRECODED,parasite.treatment == "retained"&feeding.treatment =="fed",na.rm=T)

subretnfed<-subset(ExperimentalDataRECODED,parasite.treatment == "retained"&feeding.treatment =="not fed",na.rm=T)

#pripravuju objekty

kvansubremfed<-quantile(subremfed$size,probs=c(0.025,0.975),na.rm = T)

kvansubremnfed<-quantile(subremnfed$size,probs=c(0.025,0.975),na.rm=T)

kvansubretfed<-quantile(subretfed$size,probs=c(0.025,0.975),na.rm = TRUE)

kvansubretnfed<-quantile(subretnfed$size,probs=c(0.025,0.975),na.rm=T)

#pripravuju kvantily

#TAKY: QuantLow<-tapply(ExperimentalDataRECODED,list(ExperimentalDataRECODED$Feeding.treatment,ExprerimentalData$Parasite.treatment),quantile,p=0.975)

#TAKY: QuantHigh......p=0.025)

barexp<-barplot(as.numeric(means.exp[,2]),xlab=list("Parasitic treatment",cex=1.5),ylab=list("Size",cex=1.5),ylim=c(0,20),col=c("grey65","grey60","grey55","grey50"),names.arg=c('removed','retained','removed','retained'))

text(barexp,0.5,c("not fed","not fed","fed","fed"))

#baruju

arrows(x0=0.7,y0=kvansubremnfed[1],x1=0.7,y1=kvansubremnfed[2],lwd=2,code=3,angle=90)

arrows(x0=1.9,y0=kvansubretnfed[1],x1=1.9,y1=kvansubretnfed[2],lwd=2,code=3,angle=90)

arrows(x0=3.1,y0=kvansubremfed[1],x1=3.1,y1=kvansubremfed[2],lwd=2,code=3,angle=90)

arrows(x0=4.3,y0=kvansubretfed[1],x1=4.3,y1=kvansubretfed[2],lwd=2,code=3,angle=90)

#sipky

#Exercise III.5

#(a) Use one line of code with aggregate()to compute how many different blocks there were per experiment group. The correct answer is not 5 per group.

#(b) Code the construction of a single data.frame that holds the descriptive statistics sample size, mean and standard deviation for "Measure" for each of the four possible combinations of "Experiment" and "block". The code should add each descriptive statistic one at a time.

#(c) Code the construction of a single data.frame with the descriptive statistics sample size, mean and standard deviation for "Measure2" for each of the four possible combinations of "Experiment" and "block". The code should add each descriptive statistic one at a time.

#(d) Code the construction of a single data.frame with the descriptive statistics sample size, mean and standard deviation for "Measure" and "Measure2". The code can add each descriptive statistic one at a time, but has to use a single call to aggregate() to calculate each statistic for each of the four possible combinations of "Experiment" and "block". Alternatively, you can code all in one line.

#(e) Are there differences between what you get in a,b,c,d? If so, why is this?

df.withNA<-data.frame(Trial=1:10, Experiment=c(rep("control",5),rep("treatment",5)),Measure=c(1,3,4,NA,3,4,5,NA,8,9))

df.withNA<-data.frame(df.withNA,Measure2=c(2,3,2,3,4,5,4,5,6,6),block=rep(c("B1","B2"),5))

aggregate(df.withNA,block~Experiment,function (df.withNA) length(unique(df.withNA)))

#adding dataframe, how many blocks per experiment

descriptiveStats.df.withNA<-aggregate(Measure~Experiment+block,df.withNA,mean)

names(descriptiveStats.df.withNA)[3]<-'mean'

descriptiveStats.df.withNA<-data.frame(descriptiveStats.df.withNA, n=aggregate(Measure~Experiment+block,df.withNA,length)$Measure)

descriptiveStats.df.withNA<-data.frame(descriptiveStats.df.withNA, sd=aggregate(Measure~Experiment+block,df.withNA,sd)$Measure)

#b

descriptiveStats2.df.withNA<-aggregate(Measure2~Experiment+block,df.withNA,mean)

names(descriptiveStats2.df.withNA)[3]<-'mean'

descriptiveStats2.df.withNA<-data.frame(descriptiveStats2.df.withNA, n=aggregate(Measure2~Experiment+block,df.withNA,length)$Measure2)

descriptiveStats2.df.withNA<-data.frame(descriptiveStats2.df.withNA, sd=aggregate(Measure2~Experiment+block,df.withNA,sd)$Measure2)

#c

#descriptiveStats\_comp<-aggregate(cbind(Measure,Measure2)~Experiment+block,df.withNA,mean)

#names(descriptiveStats\_comp)[3]<-"mean1"

#names(descriptiveStats\_comp)[4]<-"mean2"

#descriptiveStats\_comp<-data.frame(descriptiveStats\_comp, n1 = aggregate(cbind(Measure,Measure2)~Experiment+block,df.withNA,length)$Measure)

#descriptiveStats\_comp<-data.frame(descriptiveStats\_comp, n2 = aggregate(cbind(Measure,Measure2)~Experiment+block,df.withNA,length)$Measure)

#descriptiveStats\_comp<-data.frame(descriptiveStats\_comp, sd1 = aggregate(cbind(Measure,Measure2)~Experiment+block,df.withNA,sd)$Measure1)

#descriptiveStats\_comp<-data.frame(descriptiveStats\_comp, sd2 = aggregate(cbind(Measure,Measure2)~Experiment+block,df.withNA,sd)$Measure2)

#multiple lines

descriptiveStats\_comp<-aggregate(cbind(Measure,Measure2)~Experiment+block, df.withNA, function(x) c(n=length(x),mean=mean(x),sd=sd(x)))

#d

#Yes. The single call in d is obviously more simple but creates data you have to additionally comb through

#b omits cant calculate certain combinations due to missing values, instead it throws out missing value as a result

#d considers even 0 as a missing value

#e

#Exercises IV

#Exercise IV.1

census<-read.csv2("CensusData.txt",dec=".")

meadows<-read.csv2("MeadowsData.txt",dec=".")

visits<-read.csv2("VisitsData.txt",dec=".")

#vypoc?tat nejdr?v mean pro kazdou n?vstevu

meadows.visits<-merge(meadows,visits)

#merging for plot

meadows.temp<-aggregate(meadows.visits$Temperature~meadows.visits$Meadow.ID,meadows.visits,mean)

meadows.slope<-data.frame(unique(meadows.visits$Slope),unique(meadows.visits$Meadow.ID))

#calculating means of temp and seperating it with slope

#NEBO: aggregate(cbind(Temperature,Slope)~meadows.visits$Meadow.ID,meadows.visits,mean)

temp.slope<-merge(meadows.slope,meadows.temp,by.x="unique.meadows.visits.Meadow.ID.",by.y="meadows.visits$Meadow.ID")

names(temp.slope)<-c("Meadow.ID","Slope","MeanTemperature")

#merging slope and meantemp to make plot

plot(MeanTemperature~Slope,temp.slope,col="red",main="Temperature on slopes in visited meadows",ylab=list("Temperature",cex=1.5),xlab=list("Slope",cex=1.5),pch=20,cex=1.5,tck=0.02)

#v kompendiu asi prohozen?

#Exercise IV.2

census<-read.csv2("CensusData.txt",dec=".")

meadows<-read.csv2("MeadowsData.txt",dec=".")

visits<-read.csv2("VisitsData.txt",dec=".")

meadows.visits<-merge(meadows,visits)

mvc.all<-merge(meadows.visits,census)

n.unique <- function(x){length(unique(x))}

#making function to count how many unique values there are in a column

occ.sq.per.data.n.visit<-aggregate(Square~Meadow.ID+Visit.ID,mvc.all,n.unique)

hist(occ.sq.per.data.n.visit$Square,col="white",xlab="squares per meadow visited",main="")

#a

meantemp.mead<-aggregate(meadows.visits$Temperature~meadows.visits$Meadow.ID,meadows.visits,mean)

#calculating mean temperature of meadows

colnames(meantemp.mead)=c("Meadow.ID","MeanTemperature")

div<-function(x){

m<-x/30

return(m)}

propSq<-c(sapply(occ.sq.per.data.n.visit$Square, div))

mead.prop<-data.frame(visits,propSq)

mean.mead.prop<-aggregate(mead.prop$propSq~mead.prop$Meadow.ID,mead.prop,mean)

colnames(mean.mead.prop)=c("Meadow.ID","Proportion")

mean.mead.prop<-merge(mean.mead.prop,meantemp.mead,by="Meadow.ID")

plot(mean.mead.prop$Proportion~mean.mead.prop$MeanTemperature, xlab=list("Mean Temperature",cex=1.5),ylab="proportion non-empty squares", pch=16,xgap.axis=8)

#b

#Exercise IV.3

census<-read.csv2("CensusData.txt",dec=".")

meadows<-read.csv2("MeadowsData.txt",dec=".")

visitsNA<-read.csv2("VisitsDataWithNA.txt",dec=".")

meadows.visitsNA<-merge(meadows,visitsNA)

meadows.tempLEN<-aggregate(meadows.visitsNA$Temperature~meadows.visitsNA$Openness,meadows.visitsNA,length)

meadows.temp<-aggregate(meadows.visitsNA$Temperature~meadows.visitsNA$Openness,meadows.visitsNA,mean)

#a) Calculate the average temperature per open and closed meadow and the sample size used to calculate this average

meadows.visitsNA$TemperatureFILL<-meadows.visitsNA$Temperature

meadows.visitsNA$TemperatureFILL[is.na(meadows.visitsNA$TemperatureFILL)] <- mean(meadows.visitsNA$TemperatureFILL, na.rm = TRUE)

FILLmeadows.tempLEN<-aggregate(meadows.visitsNA$TemperatureFILL~meadows.visitsNA$Openness,meadows.visitsNA,length)

FILLmeadows.temp<-aggregate(meadows.visitsNA$TemperatureFILL~meadows.visitsNA$Openness,meadows.visitsNA,mean)

#making new column, filling NA values with mean, counting closed/open mean and sample size again

#NEBO: ifelse(is.na(meadows.visits$Temperature),(meanvalue of temperatures),meadows.visits$Temperature)

observer.visit<-aggregate(meadows.visitsNA$Visit.ID~meadows.visitsNA$Observer,meadows.visitsNA,length)

sum.na<-function (x) {sum(is.na(x))}

observer.miss<-aggregate(meadows.visitsNA$Temperature~meadows.visitsNA$Observer,meadows.visitsNA,sum.na,na.action = na.pass)

#b

observers.eff<-merge(observer.visit,observer.miss)

names(observers.eff)<-c("Who","Nvisits","Nmissed")

observers.eff$Propmissed<-observers.eff[,"Nmissed"]/observers.eff[,"Nvisits"]

#c

ID.meadows.tempLEN<-aggregate(meadows.visitsNA$Temperature~meadows.visitsNA$Meadow.ID,meadows.visitsNA,length)

NA.meadows<-aggregate(meadows.visitsNA$Temperature~meadows.visitsNA$Meadow.ID,meadows.visitsNA,sum.na,na.action = na.pass)

colnames(NA.meadows)[2]<-"TempMiss"

LEN.meadows.temp<-merge(NA.meadows,ID.meadows.tempLEN)

names(LEN.meadows.temp)<-c("Meadow.ID","MissedTemparature","MeasuredTemperature")

#how many times were temps measured

LEN.meadows.temp$Propmissed<-LEN.meadows.temp[,2]/3

#making proportion of missed temps

hist(LEN.meadows.temp$Propmissed,xlab="proportion missing",axes=T,main=" ",col="white",breaks=c(-0.1,0.1,0.2,0.3,0.5,0.7,0.8,0.9))

#Exercise IV.4

meadowpaper<-read.table(file="pieceofpaper.txt", header = TRUE,dec = ".")

meadows<-read.csv2("MeadowsData.txt",dec=".")

newMeadowData<-merge(meadows,meadowpaper,by.x="Meadow.ID",by.y = "Mdw")

newMeadowData<-data.frame(newMeadowData,ifelse(newMeadowData$Closed.=="no"|(newMeadowData$Closed.=="No"),"Open","Closed"))

colnames(newMeadowData)[8] <-"Opennes20"

#NEBO open.mead<-aggregate(Meadow.ID~Openness+Opennes20,newMeadowData,length)

meadow.entrance<-data.frame(newMeadowData$Meadow.ID,newMeadowData$Openness,newMeadowData$Opennes20)

names(meadow.entrance)<-c("Meadow.ID","Opennes","Opennes20")

#musim rozdelit na skupiny

cc<-subset(meadow.entrance,Opennes == "Closed"&Opennes20 =="Closed")

co<-subset(meadow.entrance,Opennes == "Closed"&Opennes20 =="Open")

oc<-subset(meadow.entrance,Opennes == "Open"&Opennes20 =="Closed")

oo<-subset(meadow.entrance,Opennes == "Open"&Opennes20 =="Open")

#subseting combinations

sum(table(cc))

sum(table(co))

sum(table(oc))

sum(table(oo))

#counting how many of each combination is there

open.sums<-c(sum(table(cc)),sum(table(co)),sum(table(oc)),sum(table(oo)))

open.states<-c("Closed-Closed","Closed-Open","Open-Closed","Open-Open")

change.opennes<-data.frame(open.sums,open.states)

barplot(change.opennes[,1],names.arg=change.opennes[,2],xlab="Change in opennes (old-new)",ylab="Number",col=c("grey60","grey60","grey15","grey15"))

#Exercise IV.5

alldata.ind<-read.csv2("AllDataIndividuals.txt",dec=".")

motherdata<-read.csv2("MotherData.txt",dec=".")

mother.mass<-merge(motherdata,alldata.ind,by.x="mother",by.y="individual")

mother.mass<-data.frame(mother.mass$mother,mother.mass$brood.x,mother.mass$body.mass)

names(mother.mass)<-c("mother","OGbrood","mothermass")

#seperating mother mass

mean.alldata.ind<-aggregate(alldata.ind$body.mass~alldata.ind$brood,alldata.ind,mean)

names(mean.alldata.ind)<-c("brood","broodmeanmass")

#mean of broods

broodmean.mothers<-merge(mother.mass,mean.alldata.ind,by.x="OGbrood",by.y="brood")

#merging mean of brood with mothers and their mean

plot(broodmeanmass~mothermass,broodmean.mothers)

#Exercises V

#Exercise V.1

dynamics<-read.table("dynamics.txt",dec=".")

#a

dynamic.mean<-sapply(dynamics,mean)

#NEBO: dynamic.mean<-colMeans(dynamics)

#b

plot(dynamic.mean, ylim=c(0,150),xlab="Time",ylab="Expected population size",cex=0.5)

#c

dynam.quantile<-list()

dynam.quantile<-lapply (dynamics,function(t) {

dynam.quantile[[length(dynam.quantile)+1]] <- quantile(t,c(0.025,0.975))})

dynam.quantile<-as.data.frame(dynam.quantile)

#d

plot(dynamic.mean, ylim=c(0,150),xlab="Time",ylab="Expected population size",cex=0.5)

polygon(c(1:100,100:1),c(dynam.quantile[1,], rev(dynam.quantile[2,])), col = "light gray", border = NA)

lines(dynamic.mean, type = "l", lwd = 2, col = "red")

#e

#Exercise V.2

PopData<-read.csv2("PopData.csv")

xtabs(N~Municipality,PopData)

hist(xtabs(N~Municipality,PopData))

#a

plot(N[order(year)]~sort(year),subset(PopData, Municipality == "Virrat"), type="b",main="Virrat")

#b

layout(matrix(c(1,2,3,4,5,6), 3, 2,byrow=T))

six.mun<-c("Asikkala", "Paltamo", "Padasjoki", "Jyvaskyla", "Loppi", "Kannonkoski")

for (y in six.mun){plot(N[order(year)]~sort(year),subset(PopData, Municipality == y), type="b",main=y)}

#c

#Exercise V.3

OutputData<-dir("C:\\Users\\Client\\Desktop\\Ostatn?\\Skola\\University of Turku\\R\\OutputData")

#reading all names of the files

site.data <- data.frame(matrix(ncol = 3, nrow=0))

colnames(site.data) <- c("site", "meanValue", "n")

#creating empty data.frame with proper header

#site.data<-rbind(site.data,data.frame(site=OutputData[1],meanValue=sapply(read.table(OutputData[1]),mean),n=sapply(read.table(OutputData[1]),length)))

setwd("C:\\Users\\Client\\Desktop\\Ostatn?\\Skola\\University of Turku\\R\\OutputData")

for(o in 1:length(OutputData)){

site.data<-rbind(site.data,data.frame(site=OutputData[o]

,meanValue=sapply(read.table(OutputData[o]),mean)

,n=sapply(read.table(OutputData[o]),length)))}

setwd("C:/Users/Client/Desktop/Ostatn?/Skola/University of Turku/R")

#loop to count each row

layout(matrix(c(1,2),1,2))

hist(site.data$meanValue,xlab="mean value",main="",col="white")

hist(site.data$n,xlab="sample size", main="",col="white")

#Exercise V.4

DataBlueTits<-read.csv2("DataBlueTits.csv",dec=".")

class(DataBlueTits$Ring)

class(DataBlueTits$Year)

class(DataBlueTits$Tarsus)

head(DataBlueTits)

tail(DataBlueTits)

#a

df.bt<-as.data.frame.matrix(xtabs(~Ring+Year,DataBlueTits))

df.bt[df.bt>0]<-1

#making it into dataframe of 0 or 1 values

#NEBO: df.bt<-as.data.frame.matrix(ifelse(table(subset(DataBlueTits,select=c(Ring,Year))),1,0))

binary<-data.frame(rowSums(df.bt))

colnames(binary)<-"Sum"

#making total by year

freq.year<-data.frame(n.years=c(1:7),frequency=rep(NA,7))

for (y in c(1:7)){

freq.year[y,2]<-length(binary$Sum[binary$Sum == y])}

#put the length of binary to 1:7

hist(binary$Sum, breaks=c(-0.5:8.5),main="(a)",xlab="Number of times observed",ylab="Frequency",xlim=c(0,8),ylim=c(0,1200))

#EXERCISE V.5:

DataBlueTits<-read.csv2("DataBlueTits.csv",dec=".",stringsAsFactors = TRUE)

#stringasfactor!!! otherwise no NA values

df.try<-as.data.frame.matrix(xtabs(Tarsus~Ring+Year, DataBlueTits))

df.try[df.try>0]<-1

more.binary<-data.frame(rowSums(df.try))

colnames(more.binary)<-"Sum"

freq.year.time<-data.frame(times=c(0:7), frequency=rep(NA,8))

for (y in c(0:7)){

freq.year.time[y+1,2]<-length(more.binary$Sum[more.binary$Sum == y])}

layout(matrix(c(1,2),1,2))

hist(binary$Sum, breaks=c(-0.5:8.5),main="(a)",xlab="Number of times observed",ylab="Frequency",xlim=c(0,8),ylim=c(0,1200),col="gray40")

hist(more.binary$Sum, breaks=c(-0.5:8.5),main="(b)",xlab="Number of times measured",ylab="Frequency",xlim=c(0,8),ylim=c(0,1200))

#EXERCISES VI

#EXERCISE VI.1

Offspring<-read.table("OffspringData.csv",sep=",",header=T)

Adults<-read.table("AdultData.csv",sep=",",header=T)

# reading in the data and check it

meanwing.offspring<-aggregate(wing.offspring~BroodID,Offspring,mean)

# compute the mean wing.length for offspring in each brood

wings.club<-merge(Offspring,Adults)

wings.club<-merge(wings.club,meanwing.offspring,by.x="BroodID",by.y="BroodID")

colnames(wings.club)[6]<-"mean.wing.offspring"

# merge the two datasets of offspring and parents

scale(wings.club$mean.wing.parent)

wings.club$scaled.parent<-scale(wings.club$mean.wing.parent)[,1]

wings.club$scaled.offspring<-scale(wings.club$mean.wing.offspring)[,1]

# transform the data on mean wing length of offspring and parents to Z-scores

plot(wings.club$scaled.parent,wings.club$scaled.offspring)

# graphically explore your data

lm.wings.scaled<-lm(scaled.parent~scaled.offspring,wings.club)

summary(lm.wings.scaled)$coefficients

str(lm.wings.scaled)

# implement the linear model and check the analysis is done properly

abline(lm.wings.scaled$coefficients)

legend("topleft", legend = paste("h2 =",format(lm.wings.scaled$coefficients[2],digits=2),format(lm.wings.scaled$coefficients[1],digits=2)),bty="n")

# plot the data that went into the analysis as well as the results of the analysis.

#Use the function abline() where the arguments for abline() are the value of the intercept (argument a) and the slope (argument b).

#Both these values you can extract from the summary output of the linear model by coding (use str() to see where they are)

#EXERCISE VI.2

owls.in.ghillie<-read.table("Koskenpato\_etal\_data\_citizen\_science.csv",sep=",",header=T)

str(owls.in.ghillie)

owls.in.snow <- subset(owls.in.ghillie, Snow=="TRUE")

#just owls in snow

owl.table<-xtabs(~Found+Colour,owls.in.snow)

owl.table

#making a table

chisq.test(owl.table)

owl.chisq.result<-chisq.test(owl.table)

str(owl.chisq.result)

owl.chisq.result$statistic

owl.chisq.result$parameter

owl.chisq.result$p.value

#using the table for the chisq test and extracting values

owl.plot <- barplot(owl.table,beside=T,ylab="Number found or not found",xlab="Morph",ylim=c(0,10000),col=c("red4","red4","Grey40","Grey40"))

text(owl.plot,1,rownames(owl.table),col=c("White","White","Black","Black"))

mtext(paste("X-sq =",format(owl.chisq.result$statistic,digits=2),"df =",owl.chisq.result$parameter,

"p =",format(owl.chisq.result$p.value,digits=2)))

#plotting the table, putting the extracted chisq test values in writing

owl.chisq.result$observed

owl.chisq.result$expected

owl.chisq.result$residuals

#just some checks

#In snowy environment there is statistical probability

#of grey owl being less likely to be spotted and

#of brown owl being more likely to be spotted

#therefore based on these result grey morph of the owl has

#better chance of masking itself in snowy environment

#I deduce this on the fact that p-value is very close

#to zero/under 5%

#null hypothesis that there is a equal chance to spot both morhps

#is close to O%, therefore the isnt equal chance to spot both

#Exercise VI.3

growth.data<-read.table("Growthdata.csv",sep=",",header=T)

str(growth.data)

summary(growth.data)

#import and check

boxplot(growth~treat, growth.data)

stripchart(growth~treat, growth.data,method="jitter")

#graphical check

ttest.growth<-t.test(growth~treat,growth.data)

#ttest

boxplot(growth~treat, growth.data,ylab="Growth",xlab="Treatment",col=c("lightblue","green3"))

mtext(paste("t =",format(ttest.growth$statistic,digits=2)

,"df =",format(ttest.growth$parameter,digits=2)

,"p =",format(ttest.growth$p.value,digits=2)))

stripchart(growth~treat, growth.data,method="jitter",ylab="Treatment",xlab="Growth",col=c("lightblue","green3"))

mtext(paste("t =",format(ttest.growth$statistic,digits=2)

,"df =",format(ttest.growth$parameter,digits=2)

,"p =",format(ttest.growth$p.value,digits=2)))

#nice graphical plotting

barplot(growth.data$growth,axes=F)

#art

#probability of the difference between control/treated is

#above 5% therefore I reject alternative hypothesis that

#the groups are not equal

#Did the growth differ between these groups?

#the growth did not differ because the groups are equal

#EXERCISE VI.4

data("ChickWeight")

str(ChickWeight)

finalweight<-subset(ChickWeight,Time==21)

#importing and extracting relevant data

stripchart(weight~Diet,finalweight,method="jitter",col=c("yellow4","blue","red","green3"))

chick.lm<-lm(weight~Diet,finalweight)

summary(chick.lm)

plot(chick.lm)

anova.chick<-anova(chick.lm)

plot(weight~Diet,finalweight)

mtext(paste("f =",format(anova.chick[1,4],digits=2)

,"df1 =",format(anova.chick[1,1],digits=2)

,"df2 =",format(anova.chick[2,1],digits=2)

,"p =",format(anova.chick[1,5],digits=2)))

#chci oboje df, fvalue a p value

#ftest v linearnim modelu

# v ttest t df p

#EXERCISE VI.5

tree.data<-read.table("TreeData.csv",sep=",",header=T)

str(tree.data)

stripchart(Volume~habitat,data=tree.data,method="jitter",col=c("yellow2","grey40"))

boxplot(Volume~habitat,data=tree.data,method="jitter",col=c("yellow2","grey40"))

t.test(Volume~habitat,tree.data)

lm.tree<-lm(Volume~habitat+Girth,tree.data)

summary(lm.tree)

plot(lm.tree)

lm.tree.anova<-anova(lm.tree)

str(lm.tree.anova)

summary(lm.tree.anova)

plot(Volume~Girth,tree.data,col=c("grey40","yellow3")[as.factor(tree.data$habitat)],pch=18,ylim=c(0,80),xlim=c(5,25))

tree.data$predictions<-predict(lm.tree)

lines(predictions~Girth,data = subset(tree.data,habitat=="shady"),col="grey40")

lines(predictions~Girth,data = subset(tree.data,habitat=="sunny"),col="yellow3")

abline(v=mean(tree.data$Girth),lty="dashed")

mtext(paste("f =",format(lm.tree.anova[1,4],digits=2)

,"df1 =",format(lm.tree.anova[1,1],digits=2)

,"df2 =",format(lm.tree.anova[3,1],digits=2)

,"p =",format(lm.tree.anova[1,5],digits=2)))

#EXERCISES VII

#Exercise VII.1

library(readxl)

library(ggplot2)

garden<-read\_excel("garden.xlsx")

garden$start\_yearF<-as.factor(garden$start\_year)

ggplot(garden, aes(y=quality,x=weed\_area,col=start\_yearF))+geom\_point()

ggsave("scatterex1.png", width = 5, height = 5)

#Add first the basic components and save the plot

ggplot(garden, aes(y=quality,x=weed\_area,col=start\_yearF))+geom\_point(size=2)

#Make a scatter plot but change the size of the points larger

ggplot(garden, aes(y=quality,x=weed\_area,col=start\_yearF))+geom\_point(size=2)+

theme\_minimal()

#Modify the background theme to be sure that it is white

ggplot(garden, aes(y=quality,x=weed\_area, colour=start\_yearF))+geom\_point(size=2)+

theme\_minimal()+scale\_colour\_brewer(palette="Spectral")+guides(colour=guide\_legend(title="Starting year"))

#Change the coloration of the points to your taste and add a legend title

ggplot(garden, aes(y=quality,x=weed\_area, colour=start\_yearF))+geom\_point(size=2)+

theme\_minimal()+scale\_colour\_brewer(palette="Spectral")+

guides(colour=guide\_legend(title="Starting year"))+

labs(colour="starting year",x="area of weeds",y="average quality of the produce",title = "Relation between the quality of the produce and area of weeds")

#Change the x & y -axis labels and add a title for your plot

ggplot(garden, aes(y=quality,x=weed\_area, colour=start\_yearF))+geom\_point(size=2)+

theme\_minimal()+scale\_colour\_brewer(palette="Spectral")+

guides(colour=guide\_legend(title="Starting year"))+

labs(colour="starting year",x="area of weeds",y="average quality of the produce",title = "Relation between the quality of the produce and area of weeds")+

theme(plot.title=element\_text(size=20),

axis.title.y=element\_text(size=15),

axis.title.x=element\_text(size=15))

#Change the label sizes to be larger

ggplot(garden, aes(y=quality,x=weed\_area, colour=start\_yearF))+geom\_point(size=2)+

theme\_minimal()+scale\_colour\_brewer(palette="Spectral")+

guides(colour=guide\_legend(title="Starting year"))+

labs(colour="starting year",x="area of weeds",y="average quality of the produce",title = "Relation between the quality of the produce and area of weeds")+

theme(plot.title=element\_text(size=20),

axis.title.y=element\_text(size=15),

axis.title.x=element\_text(size=15))+

stat\_smooth(method="lm",formula=y~x,se=FALSE)

#Add regression lines for the plot, but don`t include the standard error (se) in the lines this time

ggsave("scatterex1fin.png", width = 5, height = 5)

#Export the plot as a png file

#the closer to zero the area is, more quality it has

#weeds that grew in 2021 grew mostly in 0 areas

#most unsuccesful year of quality was the 2018

#Describe the plot by writing a short paragraph under your Figure

#Exercise VII.2

library(readxl)

garden<-read\_excel("garden.xlsx")

garden$locationF<-as.factor(garden$location)

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot()

ggsave("boxplotex2.png", width = 5, height = 5)

#• First add the basic components and save the plot

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot(outlier.shape=NA)

#• Make a boxplot but remove the outliers from the plot

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot(outlier.shape=NA)+ylim(0,50)

#• Change that the axis for economic value ranges from 0 to 50

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot(outlier.shape=NA)+ylim(0,50)+theme\_bw()

#• Modify the background theme so that it is white

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot(outlier.shape=NA)+ylim(0,50)+theme\_bw()+

scale\_fill\_manual(values=c("brown","orange","yellow"),name = "Location")

#• Change the coloration of the bars to your taste and add a legend title

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot(outlier.shape=NA)+ylim(0,50)+theme\_bw()+

scale\_fill\_manual(values=c("brown","orange","yellow"),name = "Location")+

coord\_flip()

#• Flip the plot so that the location is on y-axis and economic value is on x-axis

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot(outlier.shape=NA)+ylim(0,50)+theme\_bw()+

scale\_fill\_manual(values=c("brown","orange","yellow"),name = "Location")+

coord\_flip()+labs(y="economic value per box (€)",x="Location", title="Mean economic value and gardening location")

#• Change the x & y -axis labels and add a title for your plot

ggplot(garden, aes(y=econ\_box,x=locationF,fill=locationF))+

geom\_boxplot(outlier.shape=NA)+ylim(0,50)+theme\_bw()+

scale\_fill\_manual(values=c("brown","orange","yellow"),name = "Location")+

coord\_flip()+labs(x="Location",y="economic value per box (€)", title="Mean economic value and gardening location")+

theme(plot.title=element\_text(size=20),

axis.title.y=element\_text(size=15),

axis.title.x=element\_text(size=15),

legend.title =element\_text(size=15))

#• Change the label and title sizes larger. Here you can also modify the legend title and legend text in a similar way as the other texts: make them also larger if you can.

#Exercise VII.3

data(ChickWeight)

install.packages('cowplot')

library(cowplot)

plot\_list <- vector("list", length = 4)

for (i in 1:4) {

g <- ggplot(data=subset(ChickWeight,Diet==i), aes(y=weight,x=Time))+geom\_smooth()+

ylim(0,300)+theme\_bw()

plot\_list[[i]] <- g + facet\_grid(. ~Diet)}

cowplot::plot\_grid(plotlist = plot\_list, ncol = 4)

#NEBO PROSTE: ggplot(data=ChickWeight, aes(y=weight,x=Time))+geom\_smooth()+facet\_grid(. ~ Diet)

#Exercise VII.4

library(ggplot2)

library(readxl)

DataT<-read\_excel("DataT.xlsx")

str(DataT)

DataT.mean<-aggregate(measure~treatment,DataT,mean)

DataT.n<-aggregate(measure~treatment,DataT,length)[,2]

DataT.SD<-aggregate(measure~treatment,DataT,sd)[,2]

DataT.comp<-data.frame(DataT.mean,DataT.n,DataT.SD)

DataT.comp$SE<-DataT.comp$DataT.SD/sqrt(DataT.comp$measure)

DataT.comp$CI<-DataT.comp$SE\*1.96

ggplot(DataT.comp,aes(y=measure,x=treatment))+geom\_bar(stat='identity',fill="grey70")+

ylim(0,8)+theme\_classic()+geom\_errorbar(aes(ymin=measure-CI/2, ymax=measure+CI/2), width=.4)

#Exercise VII.5

library(readxl)

library(ggplot2)

fruit.taste<-read\_excel("Fruit\_tasting\_data\_2018.xlsx")

str(fruit.taste)

linear.fruit<-lm(Score2~Score1+Fruit+Score1\*Fruit,fruit.taste)

summary(linear.fruit)

anova(linear.fruit)

ggplot(fruit.taste, aes(y=Score2,x=Score1,col=Fruit))+

geom\_point(size=4)+stat\_smooth(method="lm",formula=y~x,se=TRUE)+

theme\_classic()