test1.R

User

2020-06-13

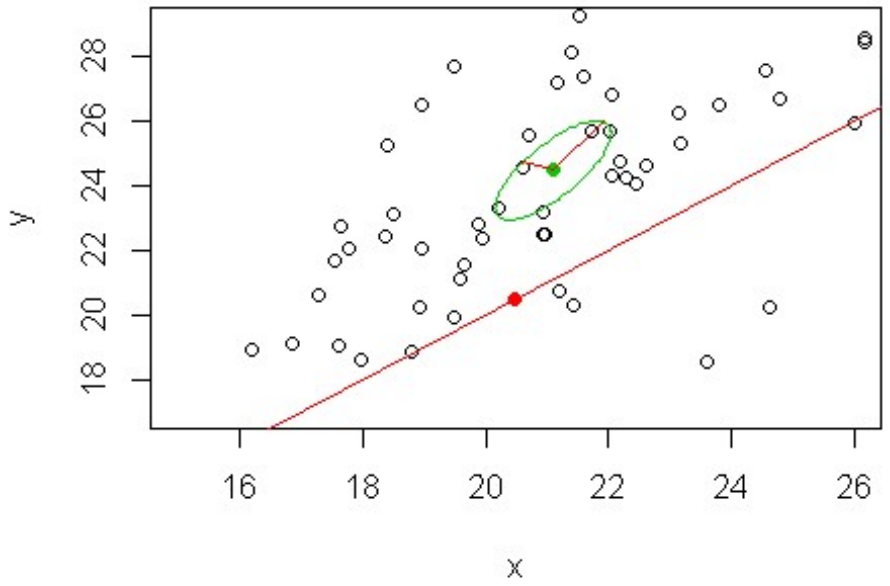
```
library(ellipse)

## Warning: package 'ellipse' was built under R version 3.6.3

##
## Attaching package: 'ellipse'

## The following object is masked from 'package:graphics':
##
##     pairs

library(Rfast)

## Warning: package 'Rfast' was built under R version 3.6.3

## Loading required package: Rcpp

## Loading required package: RcppZiggurat

## Warning: package 'RcppZiggurat' was built under R version 3.6.3

setwd("C:\\Users\\User\\Desktop\\School\\Math_537\\Test1")

data=read.csv("genderwage.csv")

###########################################
#1 a)

x=data$female
y=data$male
n=length(x)

variance=matrix(var(data),ncol=2)/n

muX=mean(x)
muY=mean(y)
mu=matrix(c(muX,muY),2,1)
muT=mean(cbind(x,y))

findMu=function(x,mu,variance)
{
  xy=c(x,x)
```

```
  result=t(mu-xy)%*%solve(variance)%*%(mu-xy)
  return(result)
}

result=optim(par=20,fn=findMu,mu=mu,variance=variance)

## Warning in optim(par = 20, fn = findMu, mu = mu, variance = variance): one
-dimensional optimization by Nelder-Mead is unreliable:
## use "Brent" or optimize() directly

mu0=matrix(c(result$par,result$par),2,1)

t2=t(mu-mu0)%*%solve(variance)%*%(mu-mu0)

pval=2*(1-pf((n-2)/((n-1)*2)*t2,2,n-2))

pval

##              [,1]
## [1,] 7.215637e-10
```

Using the optim function we find the theoretical average if both sexes had th
e same average wage. We do this by calculating the Mahalanobis distance from
a point (x,x) and then optim prescribed x=20.46973.From there we calculated t
he p-value using the above theoretical Mu and the data. Since the p-value is
so small we do not expect the average wage for women and men to be the same.

```
##############################################
#1 b)

eig=eigen(variance)

lam1=eig$values[1]
lam2=eig$values[2]

v1=eig$vectors[,1]
v2=eig$vectors[,2]

dist =(2*(n-1)/(n-2))*qf(.95,2,n-2)

a1=v1*sqrt(lam1)*sqrt(dist)
a2=v2*sqrt(lam2)*sqrt(dist)

plot(x,y,xlim=(c(15,26)),ylim=(c(17,29)))
points(muX,muY,pch=19,cex=1,col=3)
lines(ellipse(variance,centre=c(muX,muY)),col=3)
```

```
lines(c(muX,muX+a1[1]),c(muY,muY+a1[2]),lwd=.5,col=2)
lines(c(muX,muX+a2[1]),c(muY,muY+a2[2]),lwd=.5,col=2)

lines(c(0,28),c(0,28),lwd=.5,col=2)
points(mu0[1],mu0[2],pch=19,cex=1,col=2)
```



In the above graph the large red line cutting the plot is the line x=y and the red dot is where our theoretical Mu is at. The green ellipse is our confidence interval for the mu we found given the data. The red lines within the ellipse are our eigenvectors. The magnitude of these vectors shows how much information is captured in each direction.

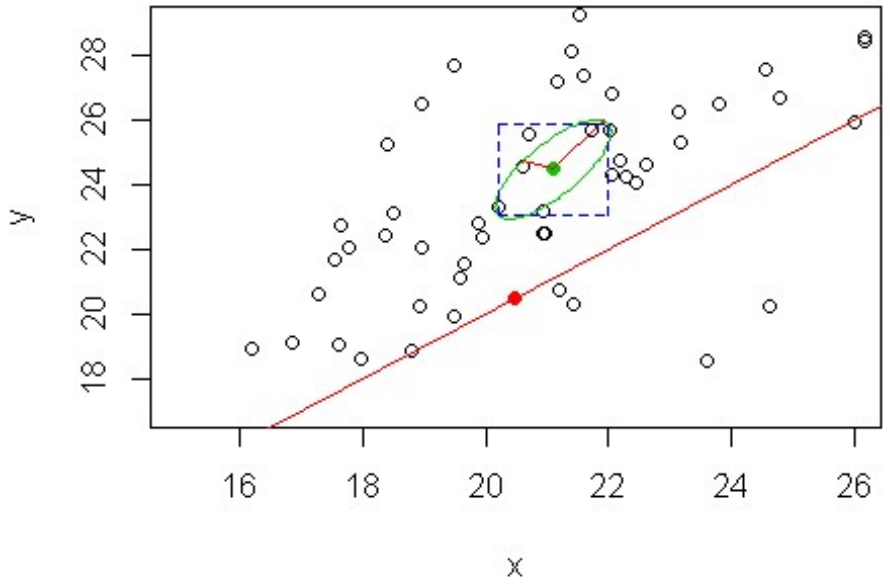
```
##############################################
#1 c)

lowerX=muX-qt(.975,n-1)*sd(x)/sqrt(n)
upperX=muX+qt(.975,n-1)*sd(x)/sqrt(n)

print(lowerX)

## [1] 20.335

print(muX)
```

```
## [1] 21.1021

print(upperX)

## [1] 21.8692

lowerY=muY-qt(.975,n-1)*sd(y)/sqrt(n)
upperY=muY+qt(.975,n-1)*sd(y)/sqrt(n)

print(lowerY)

## [1] 23.22128

print(muY)

## [1] 24.46516

print(upperY)

## [1] 25.70905
```

Separating the males and females into two different “bins” and then calculating their marginal confidence interval we get the above intervals (y is male, x is female). This does not actually capture a true 95% collectively and thus, we need to do better.

```
##############################################
#1 d)

xa = mean(x) - qt(.9875,length(x)-1)*sd(x)/sqrt(length(x))
xb = mean(x) + qt(.9875,length(x)-1)*sd(x)/sqrt(length(x))

ya = mean(y) - qt(.9875,length(y)-1)*sd(y)/sqrt(length(y))
yb = mean(y) + qt(.9875,length(y)-1)*sd(y)/sqrt(length(y))

plot(x,y,xlim=(c(15,26)),ylim=(c(17,29)))
points(muX,muY,pch=19,cex=1,col=3)
lines(ellipse(variance,centre=c(muX,muY)),col=3)

lines(c(muX,muX+a1[1]),c(muY,muY+a1[2]),lwd=.5,col=2)
lines(c(muX,muX+a2[1]),c(muY,muY+a2[2]),lwd=.5,col=2)

lines(c(0,28),c(0,28),lwd=.5,col=2)
points(mu0[1],mu0[2],pch=19,cex=1,col=2)

lines(c(xa,xb),c(ya,ya),lty=2,col=4)
lines(c(xa,xb),c(yb,yb),lty=2,col=4)
```

```
lines(c(xa,xa),c(ya,yb),lty=2,col=4)
lines(c(xb,xb),c(ya,yb),lty=2,col=4)
```



The blue box added here is the simultaneous interval. We obtain this by using alpha/(2*p) where p=2 since our degrees of freedom is 2. From there we calcul ate the intervals the same as part c. This means we can separate them for cal culation purposes because our “simultaneous” portion was found using a differ ent parameter for our bounds.

```
##############################################
#1 e)

muM=matrix(c(22.5,24.5),2,1)
muR=matrix(c(21.5,26),2,1)

sigM=matrix(c(12,8,8,12),2,2)
sigR=matrix(c(9,8,8,16),2,2)

likeRatio=prod(dmvnorm(as.matrix(data),muM,sigM))/prod(dmvnorm(as.matrix(data
),muR,sigR))

#Rachel is the winner, sorry M
```

The likeRatio=4.551284e-08 highly suggest that Rachel is the winner for the battle of class mates. Since both agreed that the data followed a bivariate normal distribution then we were able to use R's dmvnorm to calculate a ratio test with Mahalet's results being divided by Rachel's results. This created the above ratio value where Rachel's likelihood was much higher. This suggests that, while it may not be the right model, Rachel's model is much better then Mahalet's.

```
##########################################
#1 f)

eig=eigen(cov(as.matrix(data)))
vec=eig$vectors
l=eig$values

eMu0=vec[,1]%*%mu0
eX=as.matrix(data)%*%vec[,1]

t=(mean(eX)-eMu0)/(sd(eX)/sqrt(n))
2*(1-pt(t,n-1))

##              [,1]
## [1,] 8.056076e-07
```

Using the eigenvectors to calculate the P-value we see that we still get a substantially small p-value. This shows that our easier calculation using the eigenvectors and eigenvalues produces the same result as in part A. While we do lose some information (slightly higher p-value), it is not enough to change our answer from A. Since the eigenvalues are of magnitudes different, then that suggests that this would have been a good rout to take from the beginning to simplify the calculations while still provided the same relative results.