wcgs2=read.table("wcgs2.dat",header=T)

wcgsmatrix=wcgs2[,1:5]

WCGS=as.matrix(wcgsmatrix)

a=nrow(WCGS)

c=0

for(i in 1:a)

{b=WCGS[i,5]

if(b>0)

c=c+1

}

EVER=matrix(0,c,4)

NEVER=matrix(0,a-c,4)

k=1

p=1

for(i in 1:a)

{b=WCGS[i,5]

if(b>0)

{for(j in 1:4)

{EVER[k,j]=WCGS[i,j]}

k=k+1

}

else

{for(j in 1:4)

{NEVER[p,j]=WCGS[i,j]}

p=p+1}

}

muhatE = apply(EVER,2,mean)

sigmahatE = var(EVER)

muhatN = apply(NEVER,2,mean)

sigmahatN = var(NEVER)

dev.copy2pdf(Device2(ACTIVE))

, out.type=“pdf”)

dev.copy2pdf(pdf=”plot.pdf”)

dev.copy2pdf(out.type = "pdf")

dev.copy2pdf(,file=”pdf”)

pdf("plotname.pdf")

plot(......)

.....  ## until you're done with this plot

dev.off()

#EVER

d2 = NULL

for (i in 1:c) {

d2 = c(d2,(EVER[i,] - muhatE)%\*%solve(sigmahatE)%\*% (EVER[i,]-muhatE))

}

plot(qchisq((1:c)/c,4),sort(d2),

xlab="Chi-squared percentiles for ever smoke, df = 4",ylab="d2")

# Plot 45 degree line

abline(0,1)

#NEVER

e=a-c

d3 = NULL

for (i in 1:e) {

d3 = c(d3,(NEVER[i,] - muhatN)%\*%solve(sigmahatN)%\*% (NEVER[i,]-muhatN))

}

Plot2=plot(qchisq((1:e)/e,4),sort(d3),

xlab="Chi-squared percentiles for never smoke, df = 4",ylab="d3")

# Plot 45 degree line

abline(0,1)

mu0 =(c(55,70,165,120))

WCGS2=WCGS[,1:4]

muhat= apply(WCGS2,2,mean)

n = dim(WCGS2)[1]

sigmahat = var(WCGS2)

# One-sample T statistic for testing mu = mu0

# dimension of vector

p = 4

# Here, I pooled all constants into one statistic

T2 = (n\*(n-p)/(p\*(n-1)))\*(muhat - mu0)%\*% solve(sigmahat) %\*%

(muhat-mu0)

# p-value

pval = 1 - pf(T2,df1=p,df2=n-p)

mu0=(c(52,68,175,120))

n= dim(NEVER)[1]

Z = n\*(muhatN - mu0) %\*% solve(sigmahatN) %\*%

(muhatN-mu0)

# p-value

pval = 1 - pchisq(Z,df=4)

# R function for implementing confidence intervals

# using simultaneous univariate intervals ideas

#

# est is the estimates of the parameter

#

# Note that the length of est must be p

confreg <- function(X,alpha=0.05){

n = dim(X)[1]

p = dim(X)[2]

s=cov(X)

simucr=matrix(0,p,2)

dg2=n-p

cr=qf((1-alpha),p,n-p)

cr1=sqrt(p\*(n-1)\*cr/(n-p))

se=sqrt(diag(s))/sqrt(n)

est = colMeans(X)

simucr[,1]=est-cr1\*se

simucr[,2]=est+cr1\*se

print("C.R. based on T^2")

print(simucr)

indvcr=matrix(0,p,2)

q=1-(alpha/2)

cr=qt(q,(n-1))

indvcr[,1]=est-cr\*se

indvcr[,2]=est+cr\*se

print("CR based on individual t")

print(indvcr)

bonfcr=matrix(0,p,2)

q=1-(alpha/(2\*p))

cr=qt(q,(n-1))

bonfcr[,1]=est-cr\*se

bonfcr[,2]=est+cr\*se

print("CR based on Bonferroni")

print(bonfcr)

asymcr=matrix(0,p,2)

cr=sqrt(qchisq((1-alpha),p))

asymcr[,1]=est-cr\*se

asymcr[,2]=est+cr\*se

print("Asymp. simu. CR")

print(asymcr)

return(list(simucr=simucr,indvcr=indvcr,bonfcr=bonfcr,asymcr=asymcr))

}