Stat 528 HW1

Dongyang Wang

2023-01-15

Question 1

```
rm(list=ls())
set.seed(42)

beta = rnorm(50, 0, 1)
n_vector = c(50, 100, 200, 400)
```

```
library(mvtnorm)
n = 50
nx = 50

# data generation
x = rmvnorm(n, mean = rep(0, nx), sigma = diag(nx))
e = rnorm(n, 0, 1)
y = 3* tanh(beta*x) + e
```

Question 1.2

```
#required package
library(mvtnorm)
library(randomForest)
library(xtable)
#n
n_all=c(50,100,200,400)
#simulate beta=(beta_1,...,beta_50)^T (50*1)
beta=rnorm(n=50,mean=0,sd=1)
#(1)
#function for 100 times simulation for a particular n
one_n=function(i)
{
#determine n
n=n_all[i]
  #simulate data and run regression (repeat N times)
result=do.call(rbind,lapply(c(1:100),function(rept) {
    #simulate X, each column is a \vec{X}=(x1,...,x50)^T, have n X (n columns) (50*n)
X=t(rmvnorm(n+1,mean=rep(0,50),sigma=diag(50))) #simulate error term
    epsilon=rnorm(n+1,mean=0,sd=1)
    #generate y=3*tanh(beta^T*X)+epsilon
    fix_effect=as.vector(t(beta)%*%X)
    y=3*tanh(fix_effect)+epsilon
    #combine the dataset
```

```
simdata=as.data.frame(cbind(y,t(X))) #learn OLS with i=1,2,...,n/2
M=lm(y~.,data=simdata[1:(n/2),]) #compute residuals for i=n/2+1,...,n
R=abs(y[(n/2+1):n]-predict(M,newdata=simdata[(n/2+1):n,])) #90% (alpha=0.1) prediction interval for y_{
C=predict(M,newdata=simdata[n+1,])+c(-1,1)*quantile(R,0.9) #whether y_{n+1}\in C
    cover=y[n+1]>=C[1] & y[n+1]<=C[2]
    #output results
    data.frame(cover=cover,C_lower=C[1],C_upper=C[2])
  }
))
  #cat('Finish',i,'in',100,'\n')
  #calculate average coverage rate and average 90% CI
  result=apply(result,2,mean)
  return(result)
}

#run 100 trials and output results
results=NULL
for (i in 1:4){
  results=rbind(results,one_n(i))
}
xtable(results)
```

```
## % latex table generated in R 4.2.0 by xtable 1.8-4 package
## % Wed Feb 22 10:33:02 2023
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrr}
##   \hline
##  & cover & C\_lower & C\_upper \\
##   \hline
## 1 & 0.89 & -83.53 & 96.01 \\
##   2 & 0.87 & -317.22 & 319.57 \\
##   3 & 0.89 & -4.14 & 4.48 \\
##   4 & 0.91 & -3.18 & 3.60 \\
##    \hline
## \end{tabular}
## \end{table}
```

```
#(2)
#function for 100 times simulation for a particular n
one_n=function(i)
{
#determine n
n=n_all[i]
  #simulate data and run regression (repeat N times)
result=do.call(rbind,lapply(c(1:100),function(rept) {
    #simulate X, each column is a \vec{X}=(x1,...,x50)^T, have n X (n columns) (50*n)
X=t(rmvnorm(n+1,mean=rep(0,50),sigma=diag(50))) #simulate error term
    epsilon=rnorm(n+1,mean=0,sd=1)
    #generate y=3*tanh(beta^T*X)+epsilon
    fix_effect=as.vector(t(beta)%*%X)
    y=3*tanh(fix_effect)+epsilon
    #combine the dataset
simdata=as.data.frame(cbind(y,t(X))) #learn random forest with i=1,2,...,n/2
```

```
M=randomForest(y~.,type='regression',data=simdata[1:(n/2),],ntree=1000,mtry=5) #compute residuals for i
R=abs(y[(n/2+1):n]-predict(M,newdata=simdata[(n/2+1):n,])) #90% (alpha=0.1) prediction interval for y_{
C=predict(M,newdata=simdata[n+1,])+c(-1,1)*quantile(R,0.9) #whether y_{n+1}\in C
cover=y[n+1]>=C[1] & y[n+1]<=C[2] #output results
7
data.frame(cover=cover,C_lower=C[1],C_upper=C[2])
  }
  ))
  #cat('Finish',i,'in',100,'\n')
  #calculate average coverage rate and average 90% CI
  result=apply(result,2,mean)
  return(result)
}
#run 100 trials and output results
results=NULL
for (i in 1:4){
  results=rbind(results,one_n(i))
}
xtable(results)
```

```
## % latex table generated in R 4.2.0 by xtable 1.8-4 package
## % Wed Feb 22 10:34:10 2023
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrr}
##   \hline
##  & cover & C\_lower & C\_upper \\
##   \hline
## 1 & 0.89 & -4.21 & 4.14 \\
##   2 & 0.86 & -4.00 & 3.91 \\
##   3 & 0.91 & -3.88 & 3.97 \\
##   4 & 0.89 & -3.84 & 3.83 \\
##    \hline
## \end{tabular}
## \end{table}
```

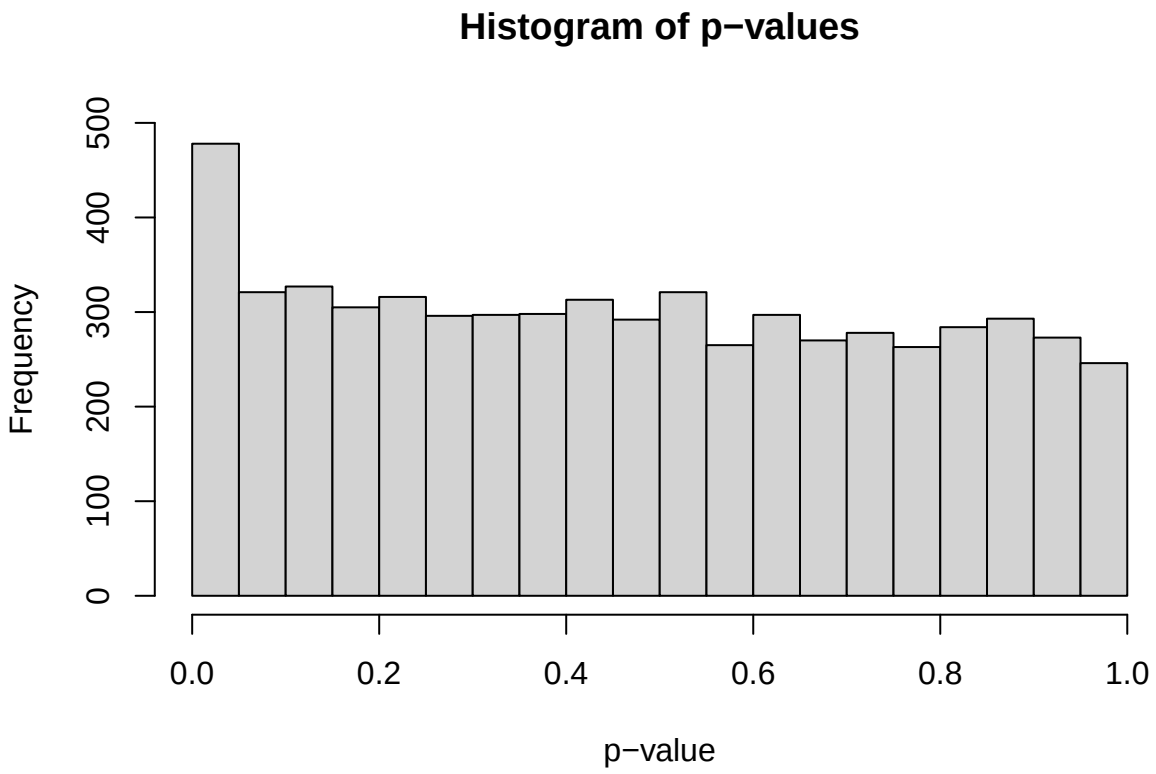
Question 2

Question 2.1

```
# Load data
url <- "https://web.stanford.edu/~hastie/CASI_files/DATA/prostz.txt"
df <- scan(url)
```

```
#calculate p-values
p_value=2*pnorm(abs(df),lower.tail=FALSE)

#plot histogram
hist(p_value,breaks=20,ylim=c(0,500),xlab='p-value',main = "Histogram of p-values",
)
```



Question 2.2

```
#Bonferroni
a=0.05
a_star=0.05/length(p_value)
cat("Significant genes under Bonferroni correction:",
    which(p_value<=a_star))
```

```
## Significant genes under Bonferroni correction: 332 610 1720
```

```
#Holm's procedure
p_value_sorted=sort(p_value,decreasing=F)
threshold=a/(length(p_value)-1:length(p_value)+1)
i0=min(which(p_value_sorted>threshold))
cat("Significant genes under Holm's procedure:",
    which(p_value <= max(p_value_sorted[1:(i0-1)])))
```

```
## Significant genes under Holm's procedure: 332 610 1720
```

```
#FDR control
a=0.1
I=(1:length(p_value))*a/length(p_value)
R=max(which(p_value_sorted<=I))
P_T=p_value_sorted[R]
cat("Significant genes under FDR control:",
    which(p_value<=P_T))
```

```
## Significant genes under FDR control: 2 11 332 364 377 579 610 637 694 698 702 721 735 739 805 905 914
```

```
#if (!require("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")
#BiocManager::install("qvalue")
library(qvalue)
```

```
## Warning: package 'qvalue' was built under R version 4.2.1
```

```
#Storey's q-values
cat("Significant genes under FDR control:",
    which(qvalue(p_value, fdr.level = 0.1)$significant == TRUE))
```

```
## Significant genes under FDR control: 2 11 292 298 332 364 377 452 579 610 637 694 698 702 721 735 739
```

Question 2.3

Question 3

```
p_vals = c(0.0011, 0.031, 0.017, 0.32, 0.11, 0.9, 0.07, 0.006, 0.004, 0.0009)
p_vals < 0.005
```

```
##  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
p_ordered = p_vals[order(p_vals)]
```

```
p_ordered
```

```
##  [1] 0.0009 0.0011 0.0040 0.0060 0.0170 0.0310 0.0700 0.1100 0.3200 0.9000
```

```
q = 0.05
j = seq(1, 10)
thres = q*j/10
p_ordered < thres
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
```

```
which(p_vals  %in% p_ordered[p_ordered < thres])
```

```
## [1]  1  3  8  9 10
```

```
p_ordered
```

```
##  [1] 0.0009 0.0011 0.0040 0.0060 0.0170 0.0310 0.0700 0.1100 0.3200 0.9000
```

```
q = 0.2
j = seq(1, 10)
thres = q*j/10
p_ordered[sum(p_ordered < thres)]
```

```
## [1] 0.11
```

```
which(p_vals <= p_ordered[sum(p_ordered < thres)])
```

```
## [1]  1  2  3  5  7  8  9 10
```

```
#which(p_vals  %in% p_ordered[p_ordered < thres])
```