Homework 3

Md Muhtasim Billah, Jugal Marfatia, Namrata Ray

10/24/2019

Question 1

Pooled t-test

```
#define the vectors
rationA=c(10,8,12,11,9,6,6,9,5,6)
rationB=c(13,15,14,13,17,9,10,8,10,8)
#calculate the params
n1=length(rationA)
n2=length(rationB)
xbar1=mean(rationA)
xbar2=mean(rationB)
s1=sd(rationA)
s2=sd(rationB)
#observed statistic
obs=xbar1-xbar2
obs
```

```
## [1] -3.5
```

```
#the pooled estimate of standard deviation
s.p=sqrt(((n1-1)*s1^2+(n2-1)*s2^2)/(n1+n2-2))
s.p
```

```
## [1] 2.785877
```

```
#the t statistic
t.stat=(xbar1-xbar2)/(s.p*sqrt(1/n1+1/n2))
t.stat
```

```
## [1] -2.809255
```

```
#degrees of freedom
df=n1+n2-2
df
```

```
## [1] 18
```

```
#the p-value
p=pt(t.stat,df)
p
```

```
## [1] 0.005802106
```

```r
#using the built-in package
t.test(rationA,rationB,alternative="less",var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  rationA and rationB
## t = -2.8093, df = 18, p-value = 0.005802
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -1.339561
## sample estimates:
## mean of x mean of y
##       8.2      11.7
```

Welch's t test

```r
#define the vectors
rationA=c(10,8,12,11,9,6,6,9,5,6)
rationB=c(13,15,14,13,17,9,10,8,10,8)
#calculate the params
n1=length(rationA)
n2=length(rationB)
xbar1=mean(rationA)
xbar2=mean(rationB)
var1=var(rationA)
var2=var(rationB)
#the t statistic
t.prime=(xbar1-xbar2)/sqrt((var1/n1)+(var2/n2))
t.prime
```

```
## [1] -2.809255
```

```r
#degrees of freedom
c=(var1/n1)/((var1/n1)+(var2/n2))
df=((n1-1)*(n2-1))/((n2-1)*(c^2)+(1-c^2)*(n1-1))
df
```

```
## [1] 9
```

```r
#the p-value
p=pt(t.prime,df)
p
```

```
## [1] 0.01020094
```

```r
#using the built in package
t.test(rationA,rationB,alternative="less")
```

```
##
##  Welch Two Sample t-test
##
## data:  rationA and rationB
## t = -2.8093, df = 16.85, p-value = 0.006072
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -1.331543
## sample estimates:
## mean of x mean of y
##       8.2      11.7
```

Wilcoxon rank sum

```
a= c(10,6,8,6,12,9,11,5,9,6)
b= c(13,9,15,10,14,8,13,10,17,8)
wilcox.test(b,a, correct=FALSE)
```

```
## Warning in wilcox.test.default(b, a, correct = FALSE): cannot compute exact
## p-value with ties
```

```
##
##  Wilcoxon rank sum test
##
## data:  b and a
## W = 80, p-value = 0.02246
## alternative hypothesis: true location shift is not equal to 0
```

Bootstrap

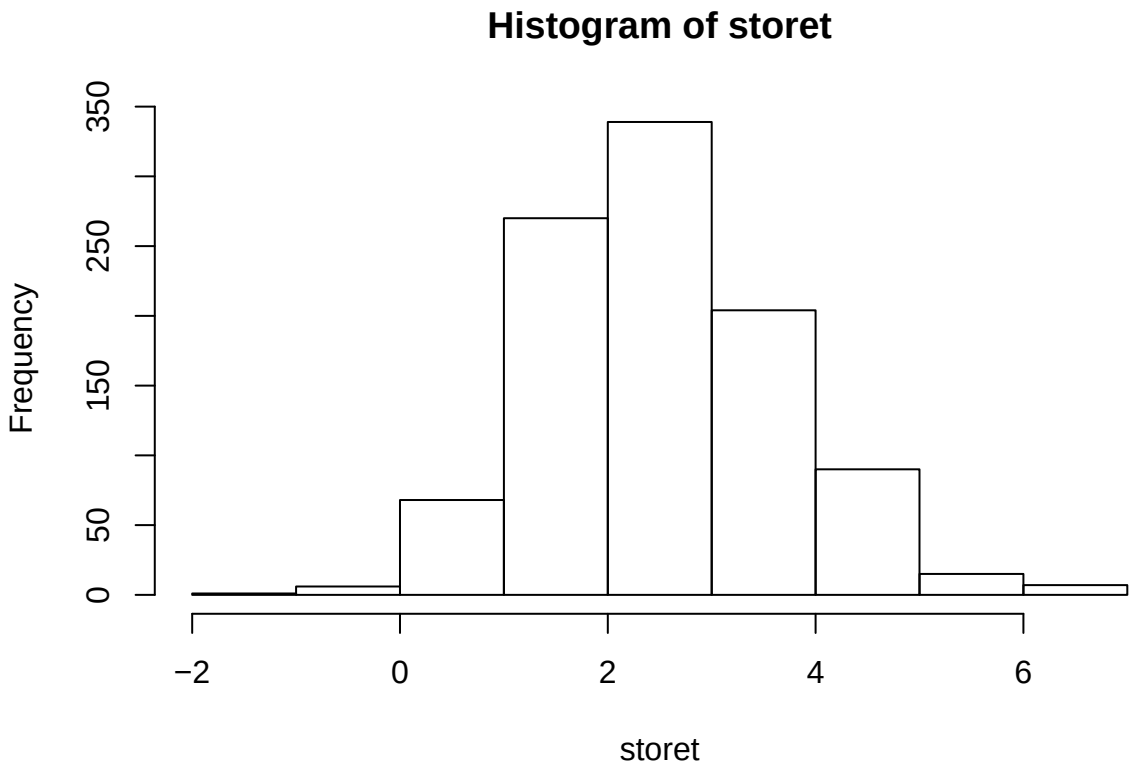
```
x1= c(10,6,8,6,12,9,11,5,9,6)
x2= c(13,9,15,10,14,8,13,10,17,8)

a1=mean(x1)
a2=mean(x2)
n1=length(x1)
n2=length(x2)

#observed t statistic
t0=abs( a1-a2)/sqrt((var(x1)/n1)+(var(x2)/n2))

# initialize a vector
storet=0
x=cbind(x1,x2)
# a loop for the 1000 bootstrap samples
for (i in 1:1000)
{
  x1b=sample(x,size=n1,replace=T)
```

```
  x2b=sample(x,size=n2,replace=T)
  tb=(mean(x1b)-mean(x2b)-a1+a2)/sqrt((var(x1b)/n1)+(var(x2b)/n1))
  storet[i]=tb
}
hist(storet)
```




```
tq=quantile(storet,prob=seq(0,1,.025))
tq[2]
```

```
##      2.5%
## 0.5971989
```

```
tq[40]
```

```
##   97.5%
## 4.92696
```

```
t0
```

```
## [1] 2.809255
```

```
p.value=(sum(storet>t0))/1000
p.value
```

```
## [1] 0.369
```

Question 2

Bonferroni method

```
g= c(0.1352, 0.4054, 0.9605, 0.3055, 0.9413, 0.0001, 0.2873, 0.0063, 0.0015, 0.4540, 0.2662, 0.9216, 0.0

reject = c(rep(0.05/100, 100))

print(g[g<reject])
```

```
## [1] 1e-04 3e-04 4e-04 2e-04 3e-04
```

```
print("Number of genes that Reject the null: ")
```

```
## [1] "Number of genes that Reject the null: "
```

```
print(length(g[g<reject]))
```

```
## [1] 5
```

FDR method

```
g_sort= sort(g, decreasing = FALSE)

reject = c(rep(0.05/100, 100) * seq(1, 100,1))

print(g_sort[g_sort<reject])
```

```
##  [1] 0.0001 0.0002 0.0003 0.0003 0.0004 0.0005 0.0005 0.0007 0.0007 0.0009
## [11] 0.0010 0.0015 0.0045 0.0046 0.0049 0.0059 0.0063 0.0085 0.0090
```

```
print("Number of genes that Reject the null: ")
```

```
## [1] "Number of genes that Reject the null: "
```

```
print(length(g_sort[g_sort<reject]))
```

```
## [1] 19
```

Bonferroni-Holm method

```r
g_sort= sort(g, decreasing = FALSE)

reject = c(rep(0.05, 100) / (100 - seq(1, 100, 1) + 1) )

print(g_sort[g_sort<reject])
```

```
## [1] 1e-04 2e-04 3e-04 3e-04 4e-04 5e-04 5e-04
```

```r
print("Number of genes that Reject the null: ")
```

```
## [1] "Number of genes that Reject the null: "
```

```r
print(length(g_sort[g_sort<reject]))
```

```
## [1] 7
```

Step up method

```r
g_sort= sort(g, decreasing = TRUE)

reject = c(rep(0.05, 100) / (100 - seq(1, 100, 1) + 1) )

print(g_sort[g_sort<reject])
```

```
##  [1] 0.0015 0.0010 0.0009 0.0007 0.0007 0.0005 0.0005 0.0004 0.0003 0.0003
## [11] 0.0002 0.0001
```

```r
print("Number of genes that Reject the null: ")
```

```
## [1] "Number of genes that Reject the null: "
```

```r
print(length(g_sort[g_sort<reject]))
```

```
## [1] 12
```