

**Solutions:**

First we setup the golub dataset.

> data(golub, package="multtest")

> H4j<-grep("H4/j",golub.gnames[,2])

> APS<-grep("APS Prostate specific antigen", golub.gnames[,2])

> gol.fac<- factor(golub.cl, levels=0:1, labels=c("ALL","AML"))

R script for a)

> print(t.test(golub[H4j, gol.fac=="ALL"], mu= -1, alternative="greater"))

One Sample t-test

data: golub[H4j, gol.fac == "ALL"]

t = 3.2743, df = 26, p-value = 0.001497

alternative hypothesis: true mean is greater than -1

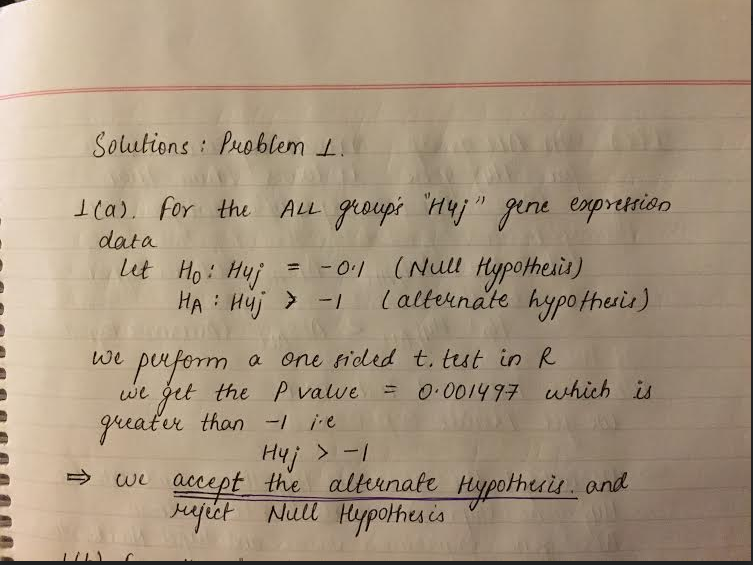
95 percent confidence interval:

-0.844439 Inf

sample estimates:

mean of x

-0.6753033



1(b)

R script for b)

> print(t.test(golub[H4j, gol.fac=="ALL"], golub[H4j, gol.fac=="AML"]))

Welch Two Sample t-test

data: golub[H4j, gol.fac == "ALL"] and golub[H4j, gol.fac == "AML"]

t = -1.4988, df = 29.978, p-value = 0.1444

alternative hypothesis: true difference in means is not equal to 0

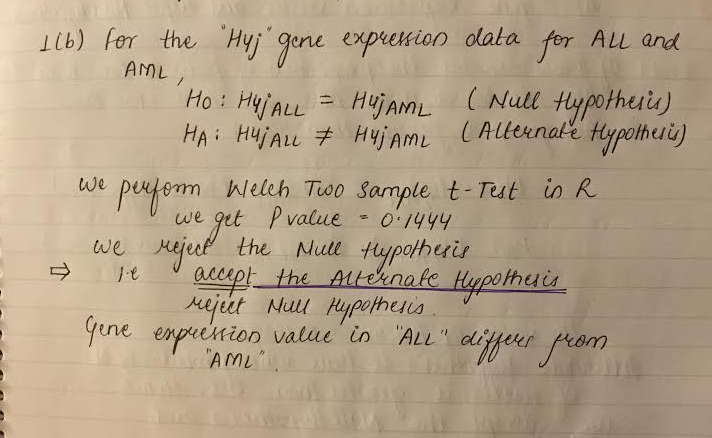
95 percent confidence interval:

-0.48627436 0.07463315

sample estimates:

mean of x mean of y

-0.6753033 -0.4694827



1c)

R script for 1c)

> print(t.test(golub[H4j, gol.fac=="ALL"], golub[APS, gol.fac=="ALL"], alternative="less", paired=T))

Paired t-test

data: golub[H4j, gol.fac == "ALL"] and golub[APS, gol.fac == "ALL"]

t = -1.8366, df = 26, p-value = 0.03886

alternative hypothesis: true difference in means is less than 0

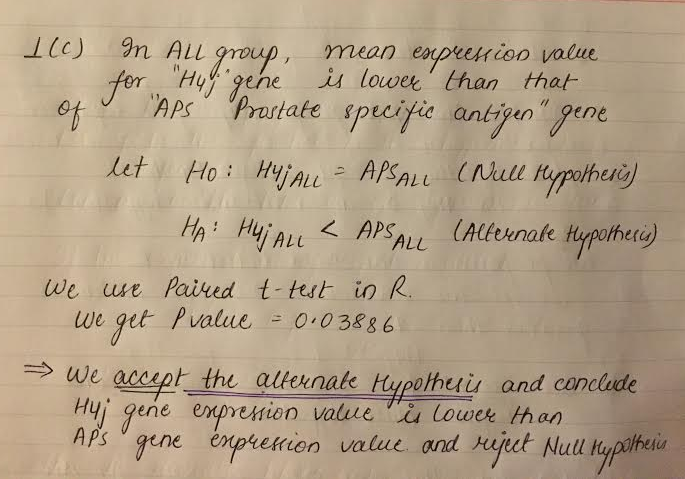
95 percent confidence interval:

-Inf -0.02175309

sample estimates:

mean of the differences

-0.3050307



1d)

R script for 1 d)

> Plow<-sum(golub[H4j, gol.fac=="ALL"] < golub[APS, gol.fac=="ALL"])

> H4jlength<-length(golub[H4j, gol.fac=="ALL"])

> print(binom.test(x=Plow, n=H4jlength, p=0.5, alternative="greater"))

Exact binomial test

data: Plow and H4jlength

number of successes = 17, number of trials = 27,

p-value = 0.1239

alternative hypothesis: true probability of success is greater than 0.5

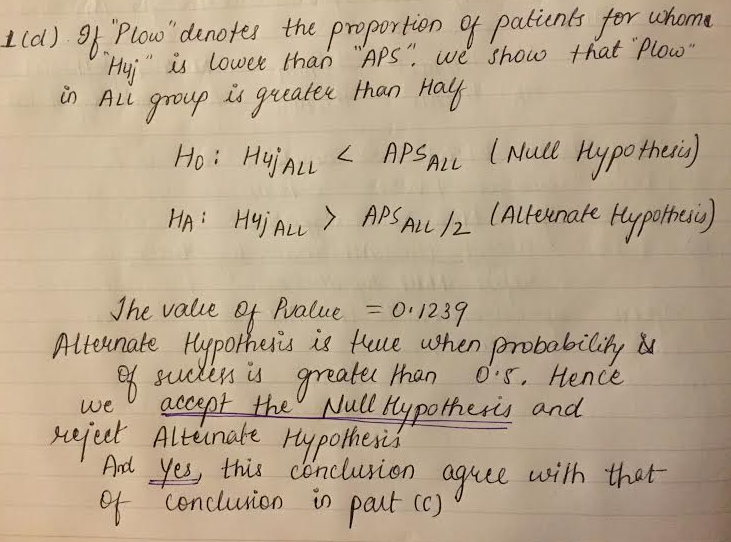
95 percent confidence interval:

0.4533598 1.0000000

sample estimates:

probability of success

0.6296296



1e)

R script for 1e

> PH4j<-sum(golub[H4j, gol.fac=="ALL"] > -0.5)

> PH4jlen<-length(golub[H4j, gol.fac=="ALL"])

> print(binom.test(x=PH4j, n=PH4jlen, p=0.5, alternative="less"))

Exact binomial test

data: PH4j and PH4jlen

number of successes = 8, number of trials = 27,

p-value = 0.02612

alternative hypothesis: true probability of success is less than 0.5

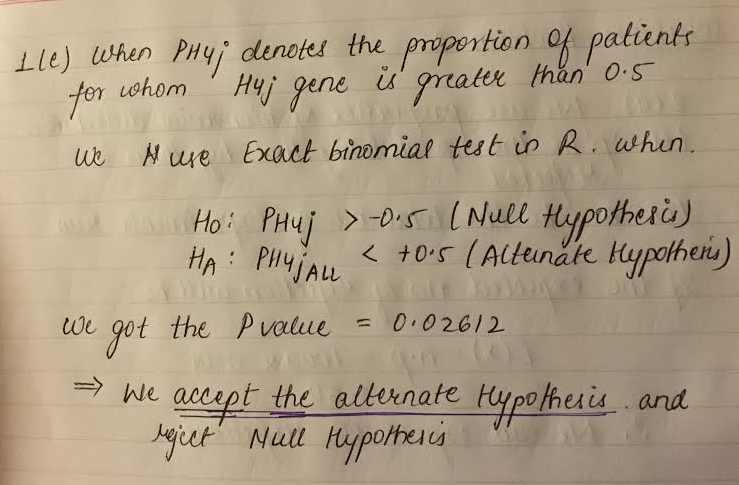
95 percent confidence interval:

0.0000000 0.4713915

sample estimates:

probability of success

0.2962963



1f)

R script for 1f

> PH4j.ALL<-sum(golub[H4j, gol.fac=="ALL"] > -0.5)

> PH4j.AML<-sum(golub[H4j, gol.fac=="AML"] > -0.5)

> ALL.length<- length(golub[H4j, gol.fac=="ALL"])

> AML.length<- length(golub[H4j, gol.fac=="AML"])

> print(prop.test(x= c(PH4j.ALL,PH4j.AML), n=c(ALL.length, AML.length), alternative="two.sided"))

2-sample test for equality of proportions with

continuity correction

data: c(PH4j.ALL, PH4j.AML) out of c(ALL.length, AML.length)

X-squared = 0.3086, df = 1, p-value = 0.5785

alternative hypothesis: two.sided

95 percent confidence interval:

-0.5631762 0.2466779

sample estimates:

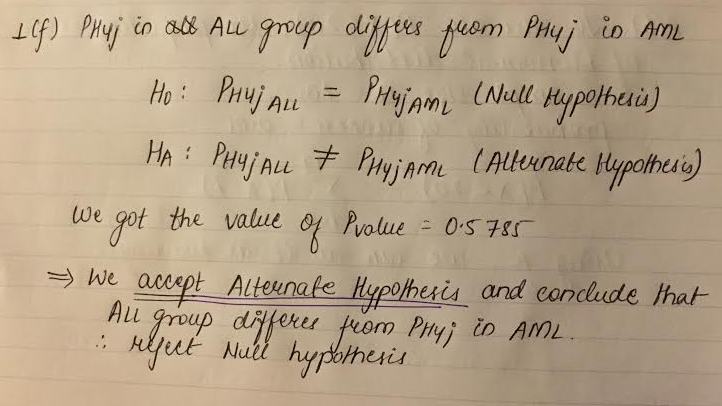
prop 1 prop 2

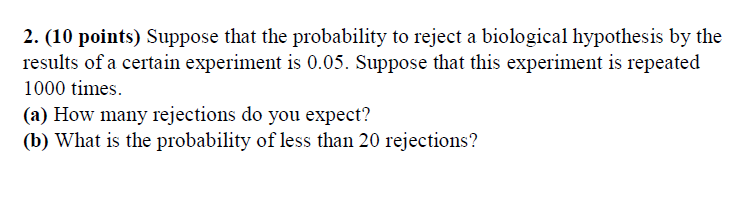
0.2962963 0.4545455

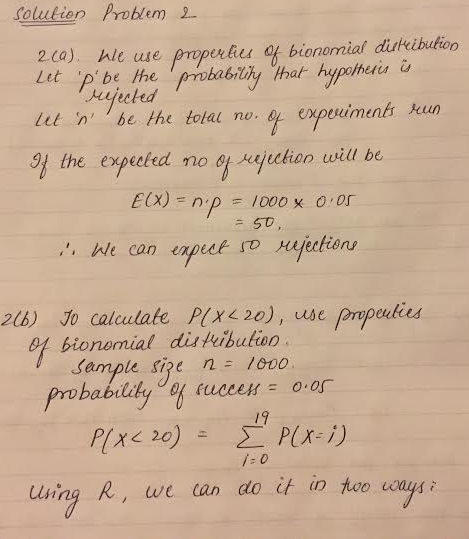
Warning message:

In prop.test(x = c(PH4j.ALL, PH4j.AML), n = c(ALL.length, AML.length), :

Chi-squared approximation may be incorrect







> # using dbinom to find the probability of fewer than 20 rejects.

> sum(dbinom(x=0:19, size= 1000, prob =0.05))

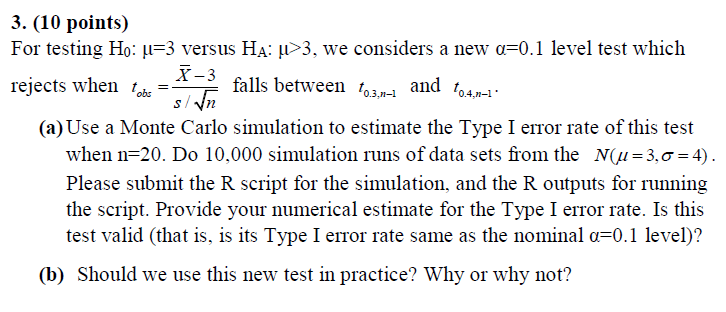
[1] 2.879692e-07

> # using pbinom to find the probability of fewer than 20 rejects.

> pbinom(q=19, size=1000, prob=0.05)

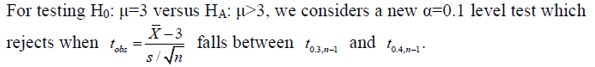
[1] 2.879692e-07

The probability of less than 20 rejection is 2.879692e-07 i.e approximately = 0



Solution:

3a)

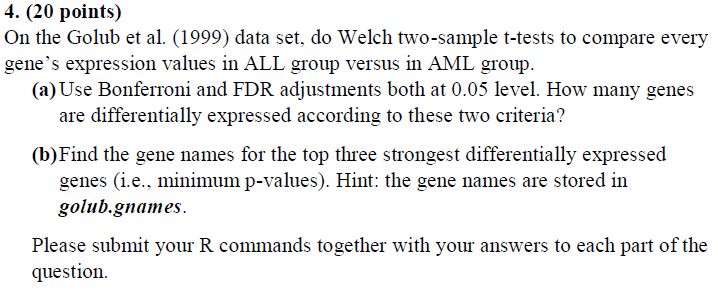


|  |
| --- |
| > #creating the data set  > x.simul<- matrix(rnorm(10000\*20, mean=3, sd=4), ncol=20)  > # t-test  > tstat<- function(x) (mean(x)-3)/(sd(x)/sqrt(length(x)))  > tstat.simul<-apply(x.simul,1,tstat)  > #calculating the rejection rate  > power.simul<- mean(tstat.simul > qt(0.3, df=19) & tstat.simul < qt(0.4, df=19) )  > # type I error rate with its 95% CI  > print(power.simul+ c(-1,0,1)\* qnorm(0.975)\*sqrt(power.simul\*(1-power.simul)/10000))  [1] 0.09100468 0.09680000 0.10259532 |
|  |
| |  | | --- | |  | |

The rejection rate is 0.09680 with 95% CI of (0.0910, 0.1025)

3b)

We should not use this as , for type I error, we reject the null hypothesis 10% . And also because to prove the significance of alternate hypothesis HA the = 0.10 is not sufficient enough.



Solution:

4a) The R code for problem 4a) is :

> # problem 4

>

> rm(list=ls())

>

> # 4(a) Use Bonferroni and FDR adjustments both at 0.05 level. How many genes are differentially expressed according to these two criteria?

>

> # load the golub data set and create factor

> data(golub, package="multtest")

> gol.fac<- factor(golub.cl, levels=0:1, labels=c("ALL","AML"))

>

> # to get the no. of genes and apply welch two-sample test

>

> length<- length(golub.gnames[,2])

> pvalues <- NULL

> for (i in 1:length){

+ pvalue <- t.test(golub[i,gol.fac=="ALL"], golub[i,gol.fac=="AML"])$p.value

+ pvalues <- c(pvalues, pvalue)

+ }

>

> # performing Bonferroni and FDR adjustment

> p.bon<-p.adjust(p=pvalues, method="bonferroni")

> p.fdr<-p.adjust(p=pvalues, method="fdr")

>

The output showing the number of genes expressed is

> print("total number of genes differentially expressed at 0.05 level no adjustment")

[1] "total number of genes differentially expressed at 0.05 level no adjustment"

> sum(pvalues < 0.05 )

[1] 1078

> print("total number of genes differentially expressed at 0.05 level with bonferroni ")

[1] "total number of genes differentially expressed at 0.05 level with bonferroni "

> sum(p.bon<0.05)

[1] 103

> print("total number of genes differentially expressed at 0.05 level with FDR")

[1] "total number of genes differentially expressed at 0.05 level with FDR"

> sum(p.fdr<0.05)

[1] 695

When there were no adjustments the number of genes differentially expressed = 1078

With bonferroni , no of genes differentially expressed = 103

With FDR , no of genes differentially expressed = 695

4b) the R code for 4b) is :

> # 4(b) Find the gene names for the top three strongest differentially expressed genes (i.e., minimum p-values). Hint: the gene names are stored in golub.gnames

>

> # load the golub data set and create factor

> data(golub, package="multtest")

> gol.fac<- factor(golub.cl, levels=0:1, labels=c("ALL","AML"))

>

> # to get the no. of genes and apply welch two-sample test

>

> length<- length(golub.gnames[,2])

> pvalues <- NULL

> for (i in 1:length){

+ pvalue <- t.test(golub[i,gol.fac=="ALL"], golub[i,gol.fac=="AML"])$p.value

+ pvalues <- c(pvalues, pvalue)

+ }

>

> # performing Bonferroni and FDR adjustment

> p.bon<-p.adjust(p=pvalues, method="bonferroni")

> p.fdr<-p.adjust(p=pvalues, method="fdr")

>

|  |
| --- |
| > #printing results  >  > print("top three strongest differentially espressed genes for FDR")  [1] "top three strongest differentially espressed genes for FDR"  > p.fdr<-p.adjust(p=pvalues,method="fdr")  > orderAML<-order(p.fdr, decreasing=FALSE)  > golub.gnames[orderAML[1:3],2]  [1] "Zyxin"  [2] "FAH Fumarylacetoacetate"  [3] "APLP2 Amyloid beta (A4) precursor-like protein 2"  >  > print("top three strongest differentially espressed genes for Bonferroni")  [1] "top three strongest differentially espressed genes for Bonferroni"  > p.bon<-p.adjust(p=pvalues,method="bon")  > orderAL<-order(p.bon, decreasing=FALSE)  > golub.gnames[orderAML[1:3],2]  [1] "Zyxin"  [2] "FAH Fumarylacetoacetate"  [3] "APLP2 Amyloid beta (A4) precursor-like protein 2" |
|  |
| |  | | --- | |  | |

The output showing the top three strongest differentially expressed gene are:

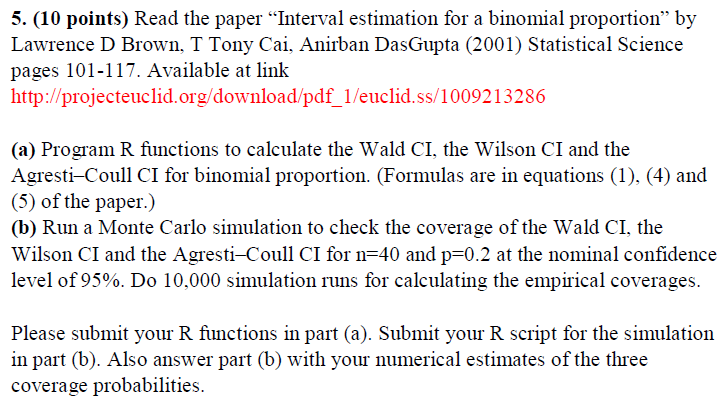
|  |
| --- |
|  |
|  |
|  |

The three strongest differentially expressed genes after FDR and Bonferroni adjustment are

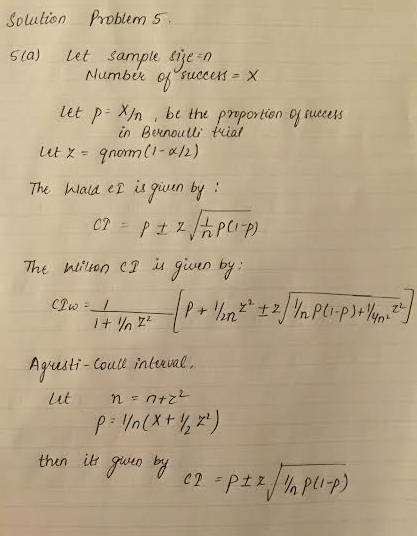
"Zyxin"

"FAH Fumarylacetoacetate"

"APLP2 Amyloid beta (A4) precursor-like protein 2"



Solution:



The code for problem 5a) is :

|  |
| --- |
| > # 5 (a) Program R functions to calculate the Wald CI, the Wilson CI and the Agresti–Coull CI for binomial proportion.  >  > wald.CI<- function(X,n,alpha=0.05){  + p<- X/n  + z<- qnorm(1-alpha/2)  + c(c(p,(p + c(-1,1) \* z \*sqrt((p\*(1-p))/n))))  + }  >  >  > wilson.CI<- function(X,n,alpha=0.05){  + p<- X/n  + z<- qnorm(1-alpha/2)  + c(p,((1/(1+z^2/n))\* (p+(z^2/(2\*n))+ c(-1,1)\*z\*sqrt((p\*(1-p))/n+z^2/(4\*n^2)))))  + }  >  >  > AgC.CI <- function(X,n,alpha=0.05){  + z<- qnorm(1-alpha/2)  + N<- n + z^2  + p<- (X + z^2/2)/N  + c(p,(p + c(-1,1)\*z\*sqrt((p\*(1-p)))))  + } |
|  |
| |  | | --- | |  | |

Solution for problem 5 b) : the code for the program is

> n.40<- rbinom(n=1, size=40, p=0.2)

> n.40.wald<-wald.CI(n.40,40)

> n.40.wilson<-wilson.CI(n.40,40)

> n.40.AgC<-AgC.CI(n.40,40)

> # run 10000 simulations to calculate the emprical changes.

>

> simul<- rbinom(n=10000, size=40, p=0.2)

> simul.wald<- NULL

> simul.wilson<- NULL

> simul.AgC<- NULL

> for(i in simul){

+ simul.wald<- rbind(simul.wald, wald.CI(i,40))

+ simul.wilson<- rbind(simul.wilson, wilson.CI(i,40))

+ simul.AgC<- rbind(simul.AgC, AgC.CI(i,40))

+ }

>

The point estimates and CI’s for single run when n= 40 p=0.2

> print(" the proportion of success for n=40 & p=0.2 ")

[1] " the proportion of success for n=40 & p=0.2 "

> print(n.40)

[1] 8

> print("The 95% CI for n=40 & p=0.2 ")

[1] "The 95% CI for n=40 & p=0.2 "

> print(" for Wald CI")

[1] " for Wald CI"

> print(rbind(n.40.wald))

[,1] [,2] [,3]

n.40.wald 0.2 0.07604099 0.323959

> print(" for wilson CI")

[1] " for wilson CI"

> print(rbind(n.40.wilson))

[,1] [,2] [,3]

n.40.wilson 0.2 0.1049999 0.3475731

> print(" for AgC CI")

[1] " for AgC CI"

> print(rbind(n.40.AgC))

[,1] [,2] [,3]

n.40.AgC 0.2262865 -0.5938148 1.046388

the proportion of success =8

for Wald cI p=0.0764 and 95% cI is (0.2,0.32)

for Wilson CI p=0.104 and 95% CI is (0.2,0.34)

for AgC CI p= -0.59 and 95% CI is (0.226,1.046)

The coverage of CI intervals after 10000 simulations

|  |
| --- |
| > # calculating the coverage of CI intervals and printing the results  >  > print("The estimated coverage after 10000 simulations of n=40, p=0.2")  [1] "The estimated coverage after 10000 simulations of n=40, p=0.2"  > print("for wald CI coverage")  [1] "for wald CI coverage"  > wald.coverage<- mean(0.2 > simul.wald[,2] & 0.2 < simul.wald[,3])  > print(wald.coverage)  [1] 0.9037  > print("for wilson CI coverage")  [1] "for wilson CI coverage"  > wilson.coverage<- mean(0.2 > simul.wilson[,2] & 0.2 < simul.wilson[,3])  > print(wilson.coverage)  [1] 0.9266  > print("for AgC CI coverage")  [1] "for AgC CI coverage"  > AgC.coverage<- mean(0.2 > simul.AgC[,2] & 0.2 < simul.AgC[,3])  > print(AgC.coverage)  [1] 1 |
|  |
| |  | | --- | |  | |

The estimated coverage after 10000 simulations of n=40 , p=0.2

For wald coverage: 90.35%

For Wilson CI coverage: 92.66%

For AgC CI coverage: 100%