```r
#1a
load("~/STAT3301/arsenic.rdata")
arsenic
```

```
##    arsenic.toenail arsenic.water age gender
## 1            0.119       0.00087  44 Female
## 2            0.118       0.00021  45 Female
## 3            0.099       0.00000  44   Male
## 4            0.118       0.00115  66 Female
## 5            0.277       0.00000  37   Male
## 6            0.358       0.00000  45 Female
## 7            0.080       0.00013  47   Male
## 8            0.158       0.00069  38 Female
## 9            0.310       0.00039  41 Female
## 10           0.105       0.00000  49 Female
## 11           0.073       0.00000  72 Female
## 12           0.832       0.04600  45 Female
## 13           0.517       0.01940  53   Male
## 14           2.252       0.13700  86 Female
## 15           0.851       0.02140   8 Female
## 16           0.269       0.01750  32 Female
## 17           0.433       0.07640  44   Male
## 18           0.141       0.00000  63 Female
## 19           0.275       0.01650  42   Male
## 20           0.135       0.00012  62   Male
## 21           0.175       0.00410  36   Male
```

```r
X = cbind(1, arsenic$arsenic.water, 1*(arsenic$gender=="Female"),arsenic$age, 1*(arsenic$gender=="Femal
X
```

```
##       [,1]    [,2] [,3] [,4]    [,5]
##  [1,]    1 0.00087    1   44 0.00087
##  [2,]    1 0.00021    1   45 0.00021
##  [3,]    1 0.00000    0   44 0.00000
##  [4,]    1 0.00115    1   66 0.00115
##  [5,]    1 0.00000    0   37 0.00000
##  [6,]    1 0.00000    1   45 0.00000
##  [7,]    1 0.00013    0   47 0.00000
##  [8,]    1 0.00069    1   38 0.00069
##  [9,]    1 0.00039    1   41 0.00039
## [10,]    1 0.00000    1   49 0.00000
## [11,]    1 0.00000    1   72 0.00000
## [12,]    1 0.04600    1   45 0.04600
## [13,]    1 0.01940    0   53 0.00000
## [14,]    1 0.13700    1   86 0.13700
## [15,]    1 0.02140    1    8 0.02140
## [16,]    1 0.01750    1   32 0.01750
## [17,]    1 0.07640    0   44 0.00000
## [18,]    1 0.00000    1   63 0.00000
## [19,]    1 0.01650    0   42 0.00000
## [20,]    1 0.00012    0   62 0.00000
## [21,]    1 0.00410    0   36 0.00000
```

```r
y = log(arsenic$arsenic.toenail)
y
```

```
##  [1] -2.1286318 -2.1370707 -2.3126354 -2.1370707 -1.2837378 -1.0272223
##  [7] -2.5257286 -1.8451602 -1.1711830 -2.2537949 -2.6172958 -0.1839228
## [13] -0.6597124  0.8118187 -0.1613432 -1.3130439 -0.8370176 -1.9589954
## [19] -1.2909842 -2.0024805 -1.7429693
```

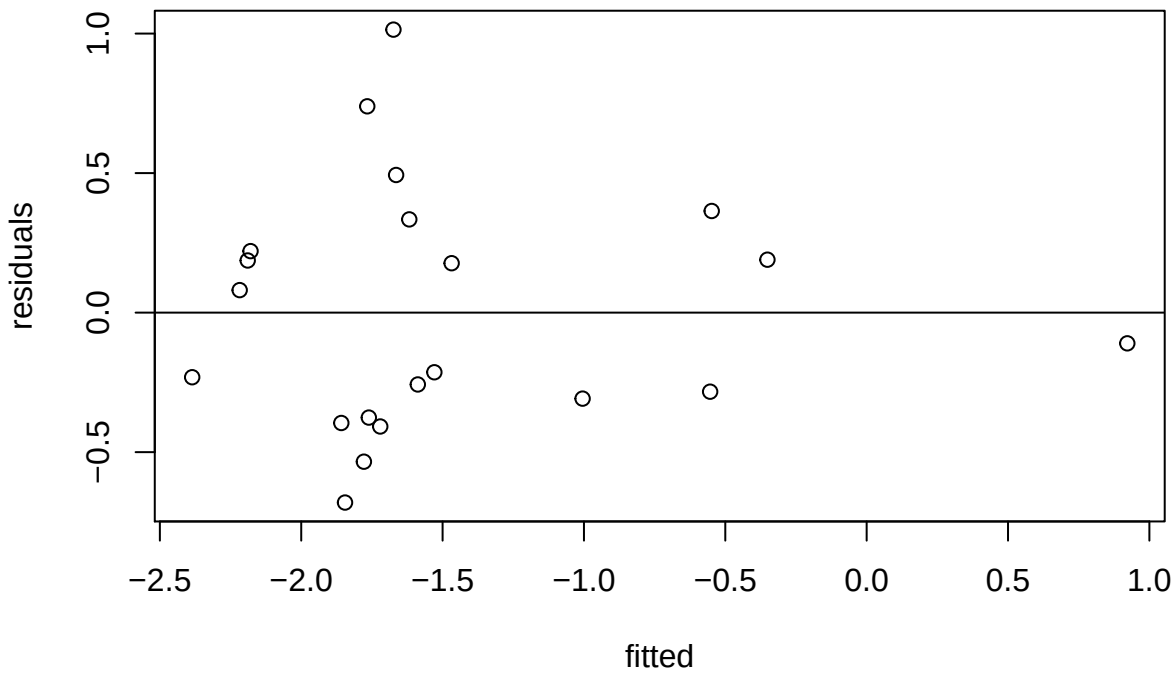
```
beta.hat = qr.solve(crossprod(X), crossprod(X,y))
#regression coefficients
beta.hat
```

```
##             [,1]
## [1,] -0.76923387
## [2,] 16.03106323
## [3,]  0.03483843
## [4,] -0.02293539
## [5,] 10.45649872
```

```
fitted = X%*%beta.hat
residuals = y- fitted
sE = sqrt(sum(residuals^2)/(length(y)-length(beta.hat)))
#estimate of error standard deviation
sE
```

```
## [1] 0.4864371
```

```
plot(fitted, residuals)
abline(h=0)
```




```
#1b
# yi is the predicted variable representing arsenic.toenail
# yi is the realization of the random variable Yi = -1.1636 + 23.934*xi1 + .188*xi2 + .4959*x13 where x
```

```
#1c
XtX = crossprod(X)
XtXinv = qr.solve(XtX)
est.std.err2 = sE*sqrt(XtXinv[4,4])
t.val = beta.hat[4]/est.std.err2
```

```r
moe = qt(0.995, length(y) - length(beta.hat))*est.std.err2
c(beta.hat[4]-moe, beta.hat[4]+moe)
```

```
## [1] -0.044551773 -0.001319013
```

```r
#we reject null hypothesis  and conclude that the interval indicates that age is significant because th
```

```r
#1d
#estimate of error standard deviation

t= beta.hat[5]/(sE*sqrt(XtXinv[5,5]))
p.val = 2*pt(-abs(t), length(y)-length(beta.hat))
p.val
```

```
## [1] 0.214602
```

```r
#the power is ~.0119
#so we reject the null hypthosis with statistical evidence that the interaction of gender and arsenic.w
```

```r
#1e
#X = cbind(1, arsenic$arsenic.water, arsenic$gender, arsenic$age)
#y = log(arsenic$arsenic.toenail)
beta.hat=qr.coef(qr(X), y=y)
residuals=y-X%*%beta.hat
sE=sqrt(sum(residuals^2)/(length(y)-length(beta.hat)))
XtXinv=qr.solve(crossprod(X))
xnew=c(1,0,0,30,0)
est.mean=sum(beta.hat*xnew)
moe=qt(0.995,length(y)-length(beta.hat)) * sE * sqrt(1+t(xnew)%*%XtXinv%*%xnew)[1]
c(exp(est.mean-moe), exp(est.mean+moe))# the 99% prediction interval for yet to be observed value of ar
```

```
## [1] 0.04824673 1.12393498
```

```r
#1f
set.seed(3301); n=length(y); p=length(beta.hat)
alpha=.05
reps=1e4
beta=beta.hat
sigma = sE
xnew=c(1,0, 0, 30,0)
XtX = crossprod(X)
XtXinv = qr.solve(XtX)
sqrtquadform=sqrt(1+t(xnew)%*%XtXinv%*%xnew)[1]
captured.list=numeric(reps)
for(r in 1:reps)
{
## generate the realization of the diastolic blood pressures:
y.s = X %*% beta + sigma*sqrt(12)*(runif(n=n,0,1)-.5)
## compute the realization of betahat
beta.s=qr.coef(qr(x=X), y =y.s)
sE.s=sqrt(sum((y.s-X%*%beta.s)^2 )/(n-p))
## compute the 95% prediction interval for Y_new
predict.int= (X%*%beta.s)[1]+c(-1,1)*qt(1-alpha/2, n-p)*sE.s*sqrtquadform
exp.predict = exp(predict.int)
## generate the realization of Y_new
ynew= xnew%*%beta.s+sigma*sqrt(12)*(runif(n=1,0,1)-.5)
```

```
exp.ynew = exp(ynew)
## check if ynew was captured
captured.list[r]=1*(exp.predict[1]< exp.ynew) & (exp.ynew < exp.predict[2])
}
mean(captured.list)
```

```
## [1] 0.9301
```

```
prop.test(x=sum(captured.list), n = reps, conf.level=.99, correct= FALSE)$conf.int[1:2]
```

```
## [1] 0.9232430 0.9363866
```

```
#The 99%confidence interval of the coverage probability of the 95% prediction inverval doesn't captures
```

```
#there are n subjects and two numerical explanatory variables
#2a
set.seed(3301)
gen.design.matrix=function(n,mu,sigma.X,row){
  A.list = rnorm(n, mean=0, sd= sqrt(row)*sigma.X)
  Z2.list = rnorm(n, mean=0, sd= sqrt(1-row)*sigma.X)
 Z3.list = rnorm(n, mean=0, sd= sqrt(1-row)*sigma.X)
 error.list = rnorm(n, mean=0, sd=sigma.X)
 X1.list = c(rep(1,n))
 X2.list = mu +A.list + Z2.list
 X3.list = mu + A.list + Z3.list
 X= cbind(X1.list, X2.list, X3.list)
 return(X)
}
```

```
#2b
set.seed(3301)
gen.pvals.f.test=function(X, beta, sigma, which.zero, reps)
{
## get information from the arguments
n=nrow(X)
p=ncol(X)
d=length(which.zero)
## compute quantities that stay the same in each
## replication
Xbeta=X%*%beta
X.XtXinv.Xt=X%*%qr.solve(crossprod(X))%*%t(X)
X0=X[,-which.zero, drop=FALSE]
X0.X0tX0inv.X0t=X0%*%qr.solve(crossprod(X0))%*%t(X0)
## allocate the memory for the observed p-values
pval.list=numeric(reps)
for(r in 1:reps)
{
## generate a realization of Y=X beta + epsilon
y=Xbeta + rnorm(n=n, mean=0, sd=sigma)
Xbeta.hat = X.XtXinv.Xt%*%y
## compute the residual sum of squares
## for the full model fit
rssf=sum((y-Xbeta.hat)^2)
## compute the fitted responses for the null
## model
X0beta.hat.0=X0.X0tX0inv.X0t%*%y
```

```
## compute the residual sum of squares
## for the null model fit
rss0=sum((y-X0beta.hat.0)^2)
## compute the realization of F
f=((rss0 - rssf)/d )/(rssf/(n-p))
## compute the observed p-value
pval.list[r]=1-pf(f, d, n-p)
}
return(pval.list)
}

power.fun = function(n, mu, sigma.X, row, beta, sigma, p, reps =1e4, alpha){
  X = gen.design.matrix(n, mu, sigma.X, row)
  pval.list =gen.pvals.f.test(X, beta, sigma, which.zero=p, reps)
  return(mean(pval.list<alpha))
}

n.list = seq(from=10, to =150, by =2)
n=length(n.list)
power.list = numeric(n)
for(i in 1:n){
  power.list[i] = power.fun(n=n.list[i], mu=68,sigma.X =2, row=.5, beta=c(1,0,-1), sigma=4, p=3, alpha=
  if(isTRUE((.89<power.list[i])&(power.list[i]< .91))){
    n.value= n.list[i]
  }
}
n.value
```

```
## [1] 62
```

```
n.list = seq(from=1030, to =1070, by =2)
n=length(n.list)
power.list = numeric(n)
for(i in 1:n){
  power.list[i] = power.fun(n=n.list[i], mu=68,sigma.X =2, row=.98, beta=c(1,0,-1), sigma=4, p=3, alpha=
  if(isTRUE((.89<power.list[i])&(power.list[i]< .91))){
    n.value= n.list[i]
  }
}
n.value
```

```
## [1] 1068
```