STATS 600 HW2

October 3, 2023

Contents

```r
# Generating Design Matrix X with size n = 1000 and five covariates d = 5
n = 1000
d = 5
# In order to truly ensure that we have rank d, we can just use an identity matrix
# and append the rest of the matrix with 0 to form X. We have a large vertical matrix
# because the matrix is n x d. But randomly generating X will almost always produce
# something with full rank.
#X = rbind(diag(d), matrix(0, n-d, d))
# Instead of randomly generated beta we can just make it fixed.
# beta must be d x 1
#beta = seq(1, d)

set.seed(1)
beta = rnorm(d, 0, 1)
X = matrix(rnorm(n*d), nrow=n)

# This is for sigma-hat, not sigma-hat^2
for (k in 1:10) {
  epsilon = rnorm(d, mean = 0, sd = 1)
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_hat = sqrt(RSS / (n-d))
  #se_hat is the definition of se given in the problem
  se_hat = sigma_hat * sqrt(t(diag(d))%*%solve(t(X)%*%X)%*%diag(d))

  # Extracting the diagonal of the matrix se_hat and doing entry-wide division
  # from beta_hat and diagonal elements
  t = beta_hat / diag(se_hat) # Each entry of t is t_j
  t_square = t^2 # we store the t^2 values and not t

  # we store all the F scores in f
  f = list()
  H = X %*% solve(t(X)%*%X) %*% t(X)

  for (i in 1:d) {
    # drop the i-th column
    X_j = X[,-i]
    H_j = X_j %*% solve(t(X_j)%*%X_j) %*% t(X_j)
```

```
    # Use the formula for F_j and calculate the numerator and denominator
    numerator = norm((H - H_j)%*%Y , type = "2")^2
    denominator = norm((diag(n) - H)%*%Y, type = "2")^2
    f[i] = numerator / ((1/(n-d)) * denominator)
  }

  f = matrix(unlist(f), ncol = 1, byrow = TRUE)
  # we show a table where the first column is the value of F and second column it t^2
  table = cbind(f, t_square)

  print(table)
}
```

```
## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced
```

```
##            [,1]       [,2]
## [1,]  620.08066  620.08066
## [2,]   62.64931   62.64931
## [3,]  995.21973  995.21973
## [4,] 4043.31075 4043.31075
## [5,]  145.01333  145.01333
```

```
## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced
```

```
##           [,1]      [,2]
## [1,]  809.8885  809.8885
## [2,]  105.5298  105.5298
## [3,] 1533.5976 1533.5976
## [4,] 5695.6949 5695.6949
## [5,]  188.3195  188.3195
```

```
## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced
```

```
##           [,1]      [,2]
## [1,] 1326.7075 1326.7075
## [2,]  121.4778  121.4778
## [3,] 2304.1998 2304.1998
## [4,] 8428.6910 8428.6910
## [5,]  340.4851  340.4851
```

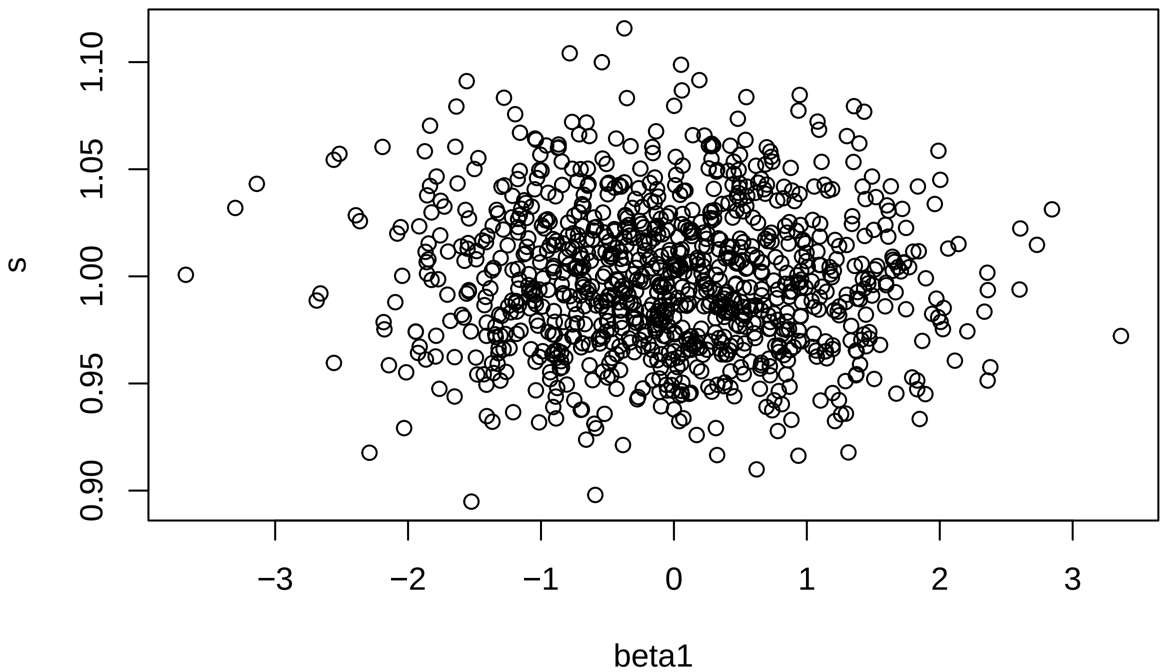
```
## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced
```

```
##            [,1]       [,2]
## [1,]  474.43904  474.43904
## [2,]   31.16872   31.16872
## [3,]  813.30313  813.30313
## [4,] 3273.45926 3273.45926
## [5,]  137.90180  137.90180
```

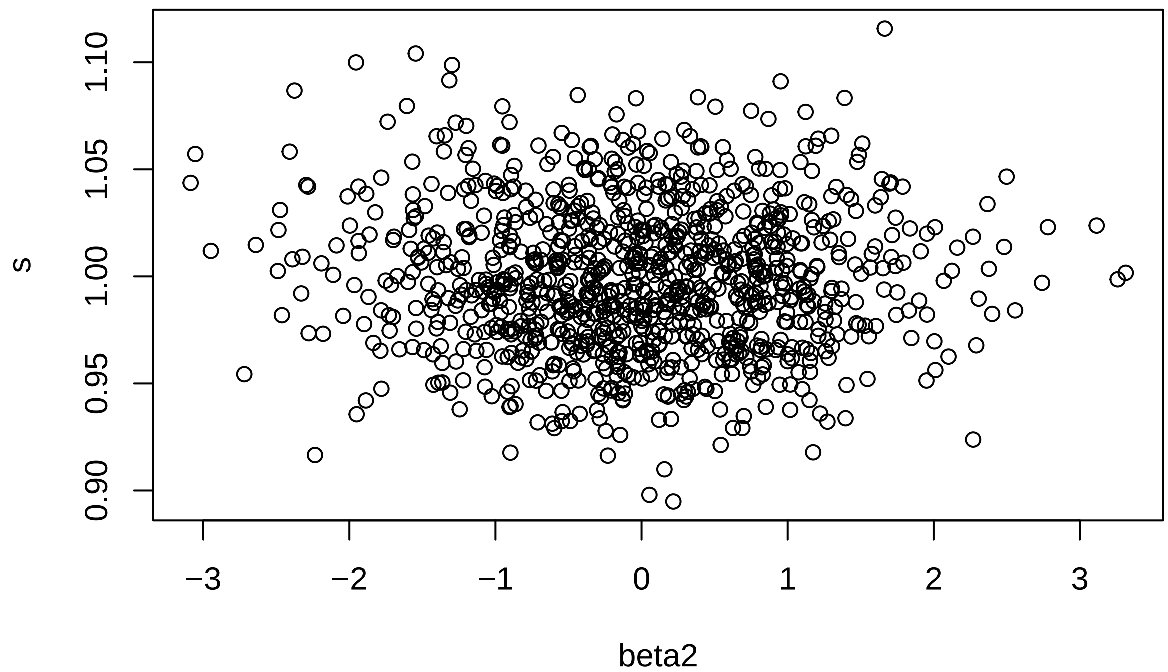
```
## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced
```

```
##             [,1]       [,2]
## [1,]  235.09722  235.09722
## [2,]   40.08654   40.08654
## [3,]  453.64251  453.64251
## [4,] 1598.35323 1598.35323
## [5,]   45.56749   45.56749

## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced

##            [,1]      [,2]
## [1,]  536.0007  536.0007
## [2,]   59.8007   59.8007
## [3,]  940.1787  940.1787
## [4,] 3571.5025 3571.5025
## [5,]  140.9260  140.9260

## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced

##             [,1]       [,2]
## [1,]  387.08218  387.08218
## [2,]   48.00703   48.00703
## [3,]  860.63521  860.63521
## [4,] 3274.52259 3274.52259
## [5,]  142.81523  142.81523

## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced

##             [,1]       [,2]
## [1,]  323.55402  323.55402
## [2,]   20.49727   20.49727
## [3,]  505.21040  505.21040
## [4,] 1592.92917 1592.92917
## [5,]   82.67968   82.67968

## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced

##             [,1]       [,2]
## [1,]  911.28289  911.28289
## [2,]   82.50828   82.50828
## [3,] 1645.13164 1645.13164
## [4,] 5902.56153 5902.56153
## [5,]  235.75105  235.75105

## Warning in sqrt(t(diag(d)) %*% solve(t(X) %*% X) %*% diag(d)): NaNs produced

##             [,1]       [,2]
## [1,]  343.26725  343.26725
## [2,]   27.22471   27.22471
## [3,]  580.68287  580.68287
## [4,] 1976.20406 1976.20406
## [5,]   97.89491   97.89491
```

```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 2
set.seed(1)

# we store the beta1-hat and beta2-hat to compare later
beta1 <- list()
beta2 <- list()
# s is going to store the sigma-hat
s <- list()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  epsilon = rLaplace(n, mu = 0, b = 1/sqrt(2)) # mean 0 and var 1
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_hat = sqrt(RSS / (n-p))
  sigma_hat

  beta1[i] <- beta_hat[1]
  beta2[i] <- beta_hat[2]
  s[i] <-sigma_hat
}

# plot out results for estimates of sigma and beta-hat
plot(beta1, s)
```

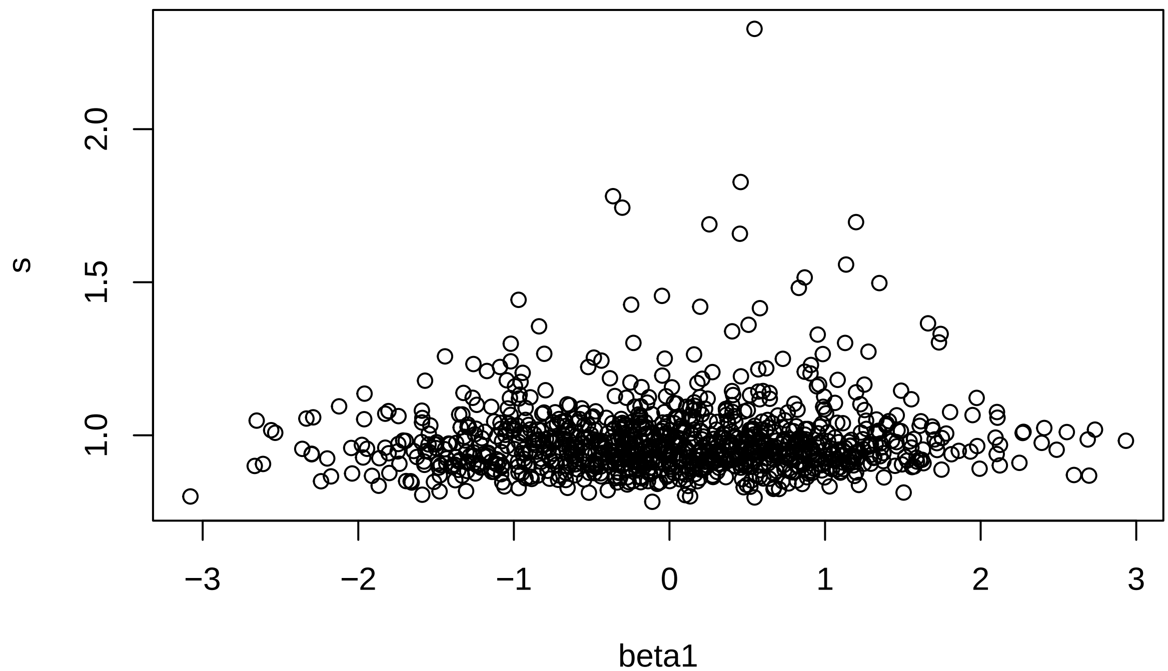



```r
plot(beta2, s)
```

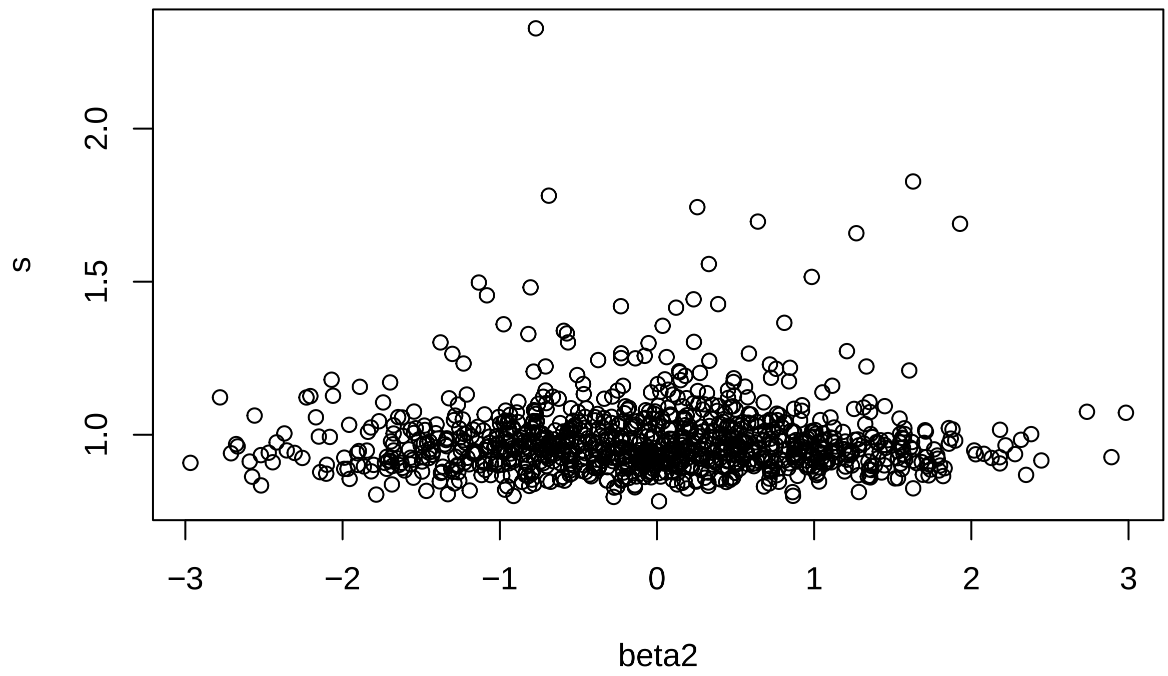



```
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 2
set.seed(1)

# we store the beta1-hat and beta2-hat to compare later
beta1 <- list()
beta2 <- list()
# s is going to store the sigma-hat
s <- list()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  epsilon = rt(n, df = 3) / sqrt(3) # mean zero and variance of 3
  # t_errs <- rt(n, df=100) / sqrt(100/(100-2)) # mean zero and var df/(df-2)
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y
  beta_hat

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_hat = sqrt(RSS / (n-p))
  sigma_hat
```

```
  beta1[i] <- beta_hat[1]
  beta2[i] <- beta_hat[2]
  s[i] <-sigma_hat
}

plot(beta1, s)
```

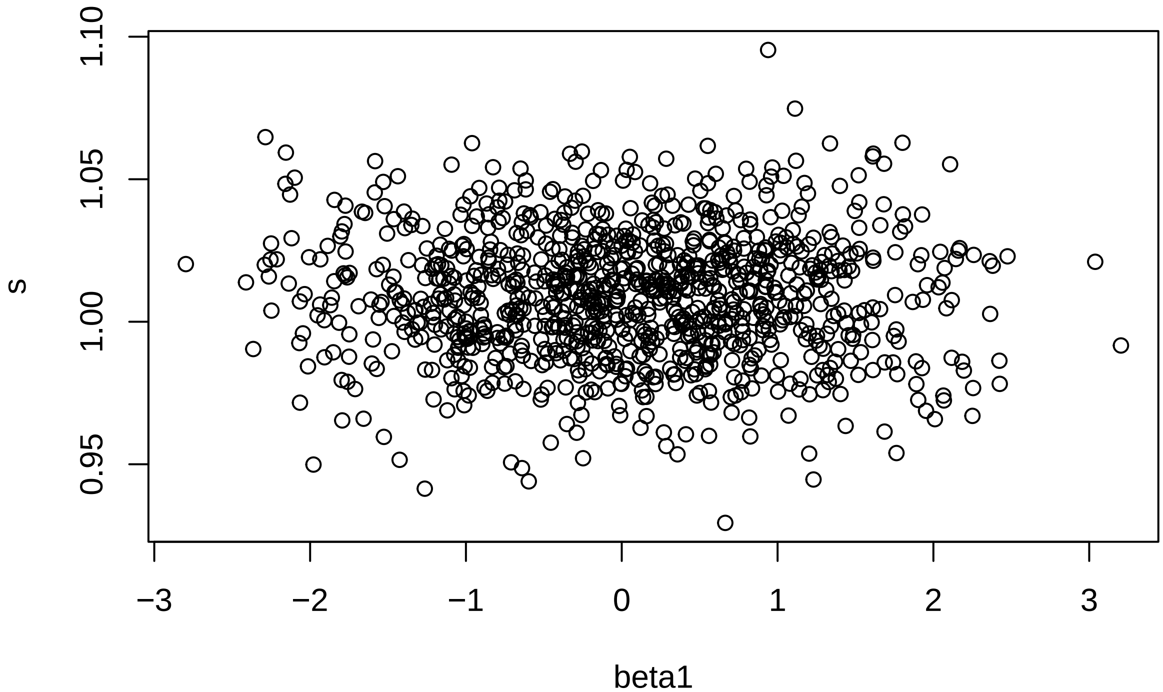



```
plot(beta2, s)
```

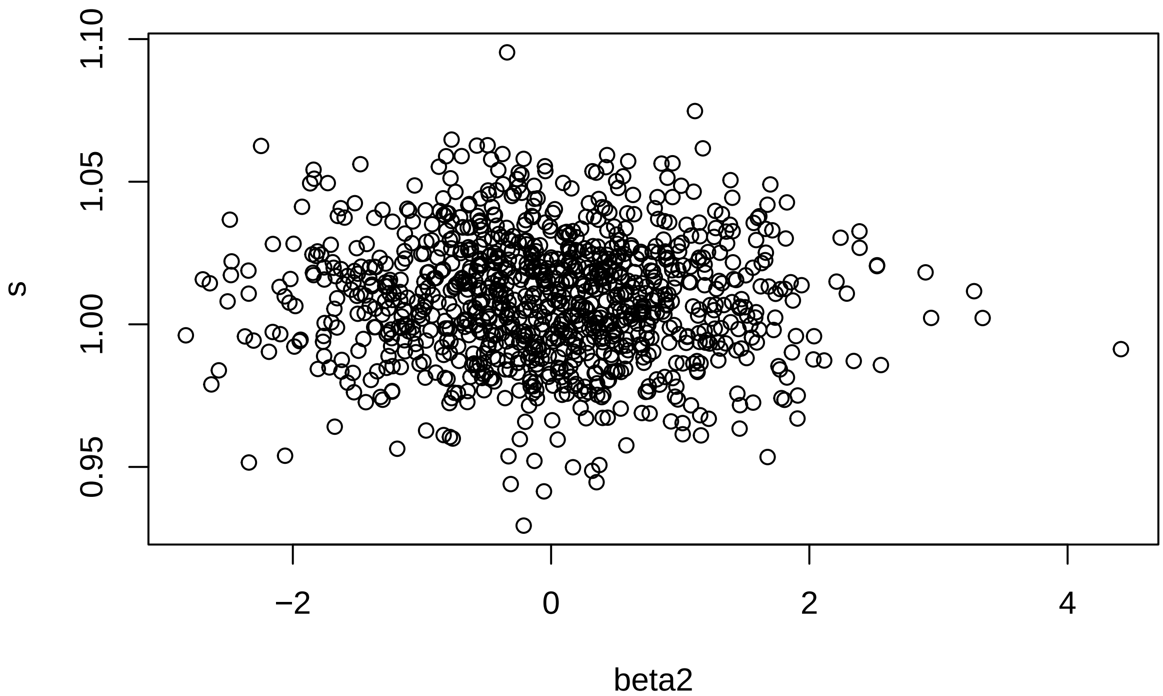



```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 2
set.seed(1)

# we store the beta1-hat and beta2-hat to compare later
beta1 <- list()
beta2 <- list()
# s is going to store the sigma-hat
s <- list()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  epsilon <- rt(n, df=100) / sqrt(1000/(1000-2)) # mean zero and var df/(df-2)
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y
  beta_hat

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_hat = sqrt(RSS / (n-p))
  sigma_hat

  beta1[i] <- beta_hat[1]
```

```r
  beta2[i] <- beta_hat[2]
  s[i] <-sigma_hat
}

plot(beta1, s)
```

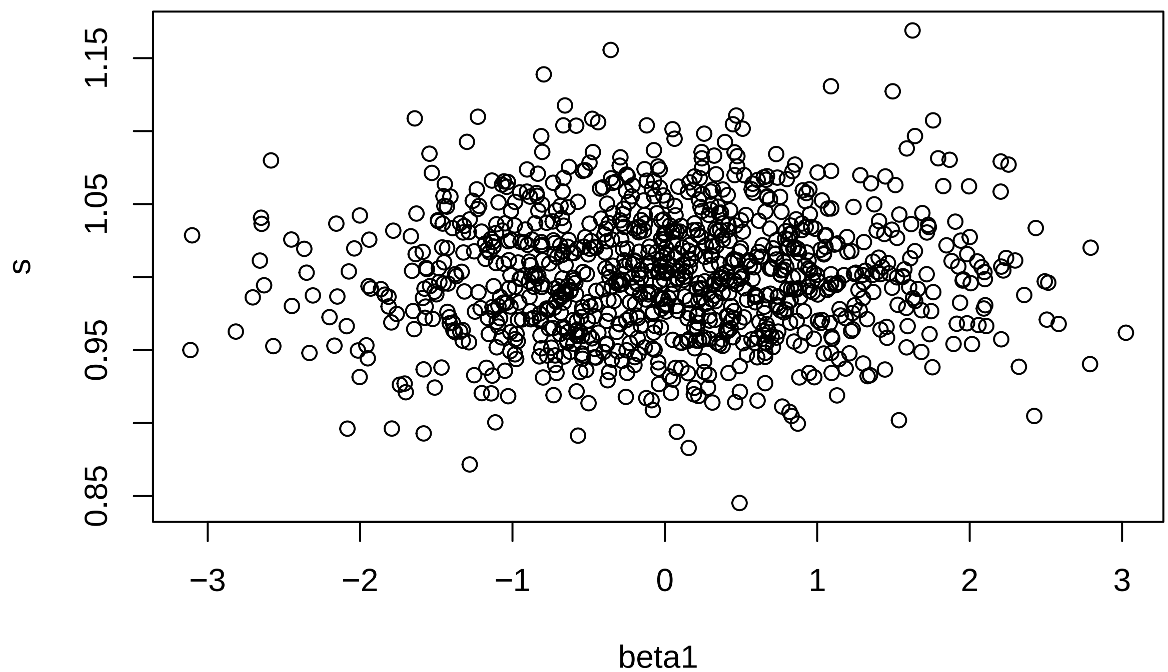



```r
plot(beta2, s)
```

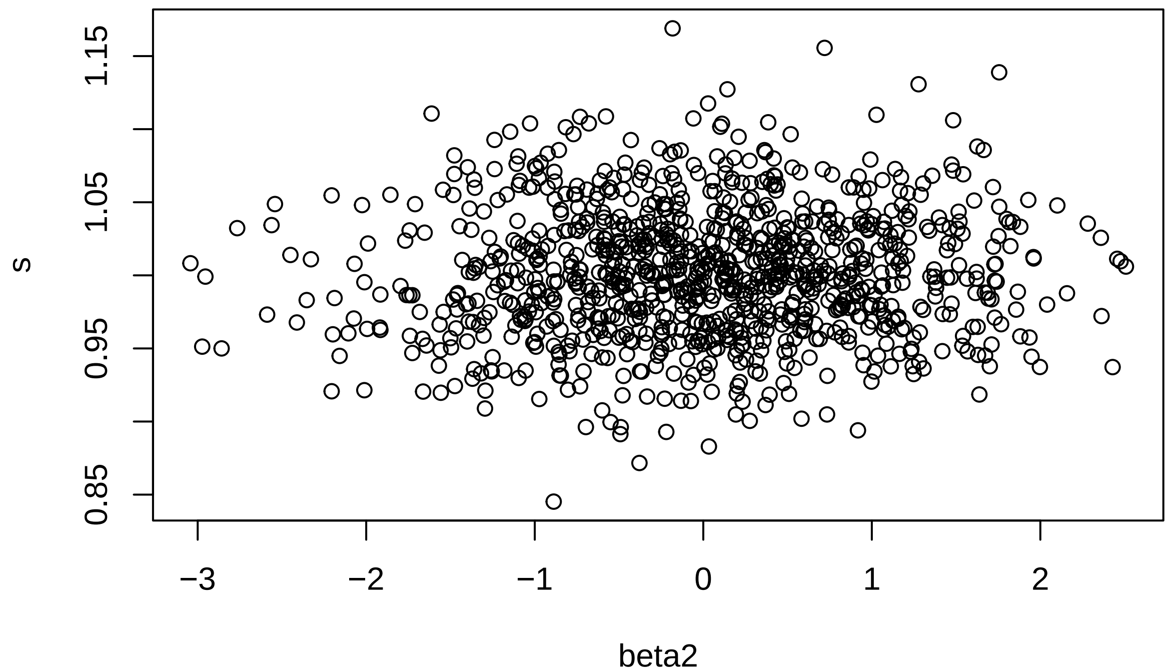



```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 2
set.seed(1)

# we store the beta1-hat and beta2-hat to compare later
beta1 <- list()
beta2 <- list()
# s is going to store the sigma-hat
s <- list()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  epsilon = rexp(n, rate = 1) - 1 # mean 0 variance 1
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y
  beta_hat

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_hat = sqrt(RSS / (n-p))
  sigma_hat

  beta1[i] <- beta_hat[1]
```

```
  beta2[i] <- beta_hat[2]
  s[i] <-sigma_hat
}

plot(beta1, s)
```




```
plot(beta2, s)
```




```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 3
set.seed(2)

sigma_sq1 = c()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  # sigma is the scale parameter
  epsilon = rLaplace(n, mu = 0, b = 1/sqrt(2)) # mean 0 and var 1
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_sq1[i] = RSS / (n-p)
}

# this is variance of sigma^2
var(sigma_sq1)
```

```
## [1] 0.004923938
```

```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 3

sigma_sq2 = c()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  # df = degree of freedom
  epsilon = rt(n, df = 3) / sqrt(3) # mean 0 and var 3
  # epsilon <- rt(n, df=100) / sqrt(100/(100-2)) # mean zero and var df/(df-2)
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_sq2[i] = RSS / (n-p)
}

# this is variance of sigma^2
var(sigma_sq2)
```

```
## [1] 0.06843827
```

```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 3

sigma_sq5 = c()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  # df = degree of freedom
  epsilon <- rt(n, df=100) / sqrt(1000/(1000-2)) # mean zero and var df/(df-2)
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_sq5[i] = RSS / (n-p)
}

# this is variance of sigma^2
var(sigma_sq5)
```

```
## [1] 0.002065806
```

```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 3

sigma_sq3 = c()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  # df = degree of freedom
  epsilon = rexp(n, rate = 1) - 1 # mean 0 var 1
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_sq3[i] = RSS / (n-p)
}

# this is variance of sigma^2
var(sigma_sq3)
```

```
## [1] 0.007612446
```

```r
# Generating Design Matrix X with size n = 1000 and two covariates
n = 1000
p = 3

sigma_sq4 = c()

for (i in 1:1000) {
  X = matrix(rnorm(n*p, 0, 1), n, p)
  beta = rnorm(p, 0, 1)

  # df = degree of freedom
  epsilon = rnorm(n, mean = 0, sd = 1)
  Y = X%*%beta + epsilon
  beta_hat = solve(t(X)%*%X) %*% t(X) %*% Y

  Yhat = X%*%beta_hat
  RSS = norm(Y - Yhat, type = "2")^2
  sigma_sq4[i] = RSS / (n-p)
}

var(sigma_sq4)
```

```
## [1] 0.001987236
```

```r
all_sigma_square = cbind(var(sigma_sq1), var(sigma_sq2), var(sigma_sq3), var(sigma_sq4), 2/(n-p), var(s
all_sigma_square
```

```
##              [,1]       [,2]        [,3]        [,4]        [,5]        [,6]
## [1,] 0.004923938 0.06843827 0.007612446 0.001987236 0.002006018 0.002065806
```

```r
# Generating Design Matrix X with size n = 500 and 10 covariates
library(combinat)
```

```
##
## Attaching package: 'combinat'

## The following object is masked from 'package:utils':
##
##     combn
```

```r
library(MASS)
n = 500
d = 10
set.seed(1)
comb = combn(c(1,2,3,4,5,6,7,8,9,10), 3)

for (i in 1:20) {
  X = matrix(rnorm(n*d, 0, 1), n, d)
  beta = matrix(c(5, 5, 0, 0, 0, 0, 0, 0, 0, 0), 10, 1)
  epsilon <- rnorm(n, 0, 1)
  RSS = c()
  RSSi = c()

  Y = X %*% beta + epsilon

  # Here we go through all possible models of size 3
  for (j in 1:ncol(comb)) {
    beta_hat = solve(t(X[,comb[,j]])%*%X[,comb[,j]]) %*% t(X[,comb[,j]]) %*% Y
    Yhat = X[,comb[,j]]%*%beta_hat
    H = X[,comb[,j]] %*% solve(t(X[,comb[,j]])%*%X[,comb[,j]]) %*% t(X[,comb[,j]])

    for (k in 1:n) {
      RSSi[k] = norm((Y[k,] - Yhat[k,]) / (1 - H[k,k]), type = "2")^2
    }

    RSS[j] = sum(RSSi)
  }

  best_model = which.min(RSS)
  print(comb[,best_model])
}
```

```
## [1] 1 2 7
## [1] 1 2 8
## [1] 1 2 4
## [1] 1 2 4
## [1] 1 2 8
## [1] 1 2 9
## [1]  1  2 10
```

```
## [1] 1 2 5
## [1] 1 2 5
## [1] 1 2 7
## [1] 1 2 7
## [1]  1  2 10
## [1] 1 2 7
## [1] 1 2 9
## [1] 1 2 3
## [1]  1  2 10
## [1] 1 2 3
## [1]  1  2 10
## [1] 1 2 9
## [1] 1 2 4
```