HW4

Nikhil Gopal

10/9/2020

1a

```
#1a

sample <- rnorm(10, mean =0 , sd =1)
mean(sample)
```

```
## [1] 0.05149495
```

As you can see here, the data is generated using a normal distribution, but the mean is further away from the true mean of 0 than in the below cases.

1b

```
sample400 = rnorm(400, mean =0 , sd =1)
mean(sample400)
```

```
## [1] -0.05960891
```

```
sample10000 = rnorm(10000, mean =0 , sd =1)
mean(sample10000)
```

```
## [1] 0.004998079
```

In these questions, the mean gets closer and closer to the true mean of 0 as the sample size increases. This is in accordance with the law of large numbers. If we take a large enough sample size, we will eventually get a sample average of the true mean, and larger samples should result in closer sample averages to the true mean.

1c

```
#create a matrix to hold generated samples, each row is a new sample
#10 columns because you're generating 10 datapoints (n = 10)
#average each row to get sample means
matrix_holding_samples <- matrix(rnorm(900, mean = 0, sd = 1), nrow = 900, ncol = 10, byrow = TRUE)

#calculate the sample means by summing the values of each row, then dividing by sample size
sample_means <- rowSums(matrix_holding_samples)/10

#sample variance of sample averages
var(sample_means)
```

```
## [1] 0.09436501
```

```
#variance of distribution
var(sample10000)
```

```
## [1] 0.9872661
```

```
#ratio of sample variance of sample averages : variance of my distribution
var(sample_means)/var(sample10000)
```

```
## [1] 0.09558214
```

The variance of sample average was about 0.095, and the variance of my distribution (n=10,000) was 1.03. The ratio of them is about 0.0921.

1d

```
#n = 400

#create a matrix to hold generated samples, each row is a new sample
#10 columns because you're generating 10 datapoints (n = 10)
#average each row to get sample means
matrix_holding_samples400 <- matrix(rnorm(900, mean = 0, sd = 1), nrow = 900, ncol = 400, byrow = TRUE)

#calculate the sample means by summing the values of each row, then dividing by sample size
sample_means400 <- rowSums(matrix_holding_samples)/400

#sample variance of sample averages
var(sample_means400)
```

```
## [1] 5.897813e-05
```

```
#variance of distribution
var(sample10000)
```

```
## [1] 0.9872661
```

```
#ratio of sample variance of sample averages : variance of my distribution
var(sample_means400)/var(sample10000)
```

```
## [1] 5.973884e-05
```

```
#n = 10,000





#create a matrix to hold generated samples, each row is a new sample
#10 columns because you're generating 10 datapoints (n = 10)
#average each row to get sample means
```

```r
matrix_holding_samples10k <- matrix(rnorm(900, mean = 0, sd = 1), nrow = 900, ncol = 10000, byrow = TRU

#calculate the sample means by summing the values of each row, then dividing by sample size
sample_means10k <- rowSums(matrix_holding_samples)/10000

#sample variance of sample averages
var(sample_means10k)
```

```
## [1] 9.436501e-08
```

```r
#variance of distribution
var(sample10000)
```

```
## [1] 0.9872661
```

```r
#ratio of sample variance of sample averages : variance of my distribution
var(sample_means10k)/var(sample10000)
```

```
## [1] 9.558214e-08
```

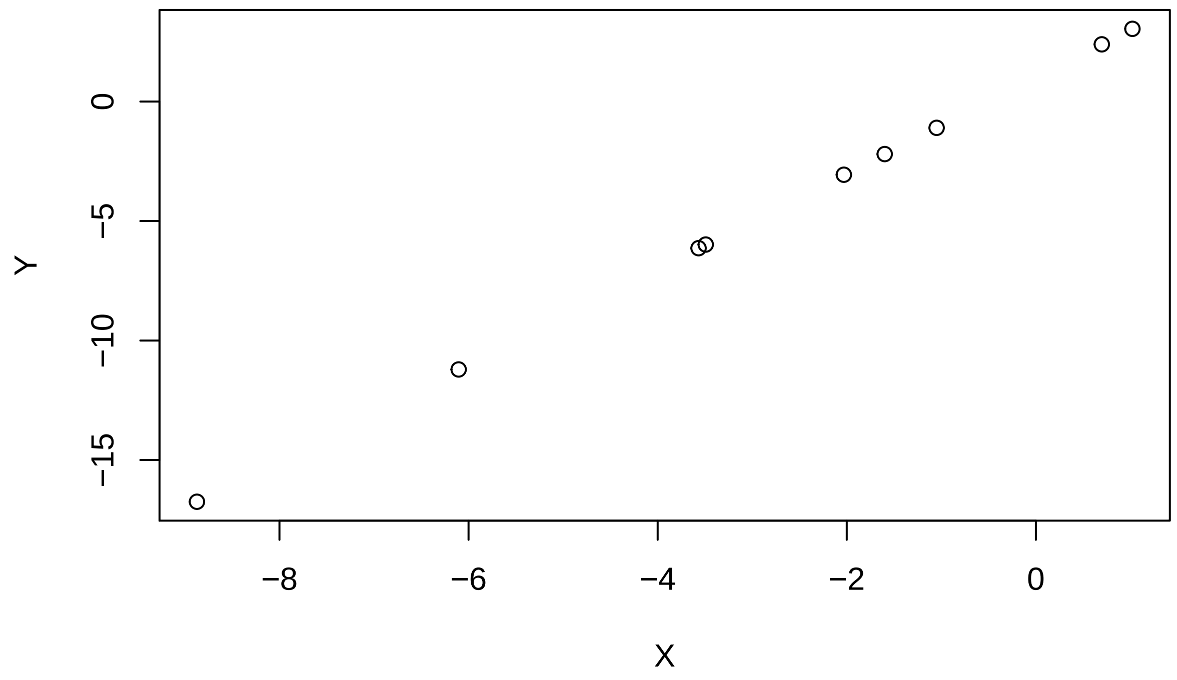
As we see the sample size increase, we observe the variance decreasing and the ratio of the variances also simultaneously decreasing.

2a

```r
#ancillary
X <- rnorm(9, mean = 0, sd = 3)

#outcome w/ conditional expectation
Y <- 1 + 2*X

plot(X,Y)
```



2b

```
#create a linear model for Beta hat
linear_model <- lm(Y ~ X)

summary(linear_model)
```

```
## Warning in summary.lm(linear_model): essentially perfect fit: summary may be
## unreliable
```

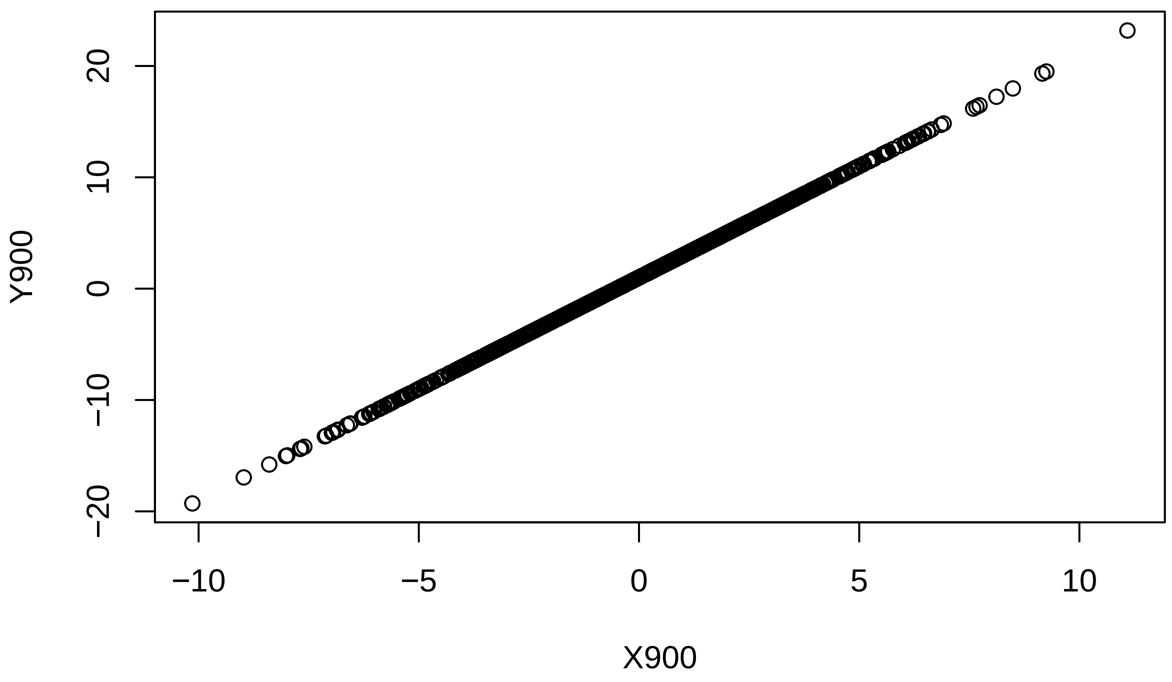
```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.082e-15 -2.614e-16 -1.132e-16  9.174e-17  1.094e-15
##
## Coefficients:
##              Estimate Std. Error   t value Pr(>|t|)
## (Intercept) 1.000e+00  3.052e-16 3.277e+15   <2e-16 ***
## X           2.000e+00  7.470e-17 2.678e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.714e-16 on 7 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 7.169e+32 on 1 and 7 DF,  p-value: < 2.2e-16
```

Our intercept is almost exactly 1 and our slope is 2, which is as expected.

2c

```
X900 <- rnorm(900, mean =0, sd = 3)
Y900 <- 1 + 2*X900

plot(X900, Y900)
```



```
linear_model_900 <- lm(Y900 ~ X900)

summary(linear_model_900)
```

```
## Warning in summary.lm(linear_model_900): essentially perfect fit: summary may be
## unreliable
```

```
##
## Call:
## lm(formula = Y900 ~ X900)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.644e-13 -2.200e-17  2.330e-16  4.030e-16  4.539e-15
##
## Coefficients:
##              Estimate Std. Error   t value Pr(>|t|)
## (Intercept) 1.000e+00  1.837e-16 5.444e+15   <2e-16 ***
## X900        2.000e+00  5.850e-17 3.419e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.51e-15 on 898 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.169e+33 on 1 and 898 DF,  p-value: < 2.2e-16
```

As above, the slope is nearly 2 and the intercept is nearly 1.

2d

```
matrix_X <- matrix(rnorm(10*400, mean =0, sd = 1), nrow =400, ncol = 9)
```

```
## Warning in matrix(rnorm(10 * 400, mean = 0, sd = 1), nrow = 400, ncol = 9): data
## length [4000] is not a sub-multiple or multiple of the number of columns [9]
```

```
epsilon = matrix(rnorm(10*400, mean =0, sd =0.5), nrow = 400, ncol = 9)
```

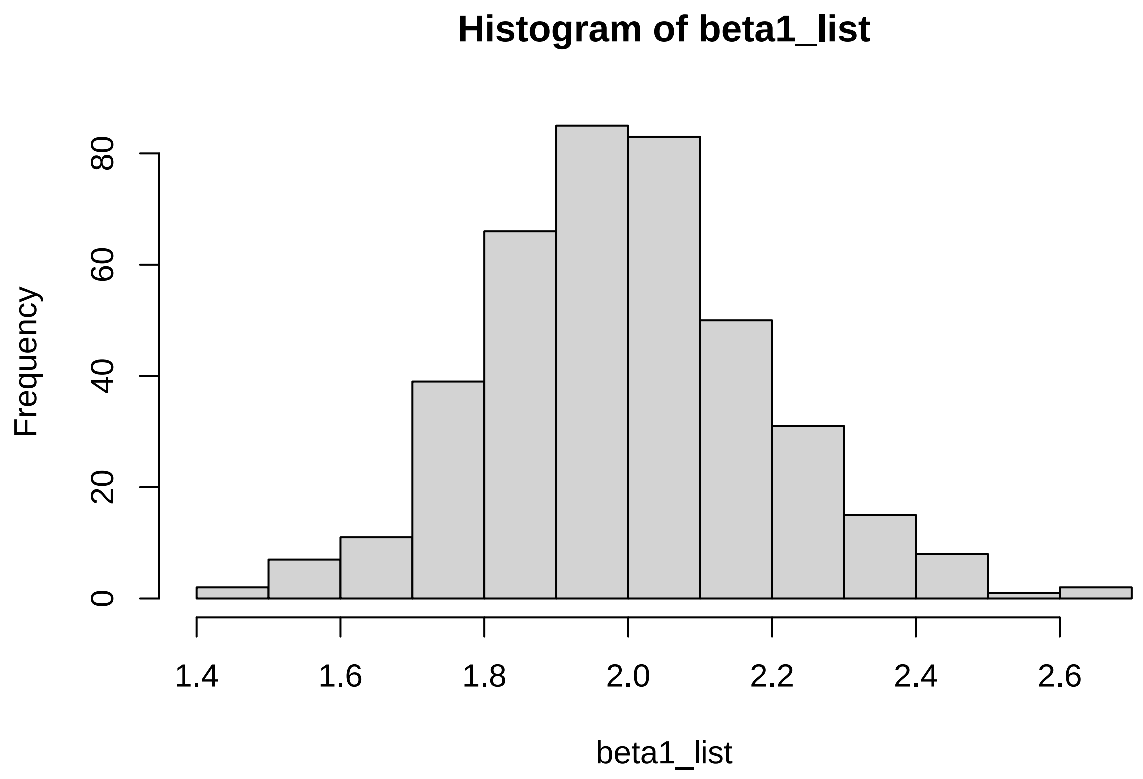
```
## Warning in matrix(rnorm(10 * 400, mean = 0, sd = 0.5), nrow = 400, ncol = 9):
## data length [4000] is not a sub-multiple or multiple of the number of columns
## [9]
```

```
matrix_Y <- 1 + 2*matrix_X + epsilon


#create am empty list to use later
beta1_list = NULL

#iterate thru matrix, then generate beta 1 coefficient for the given index and save to list to plot lat
for(i in 1:400){
  beta1_list <- c(beta1_list, lm(matrix_Y[i,] ~ matrix_X[i,])$coefficients[2])
}


hist(beta1_list)
```



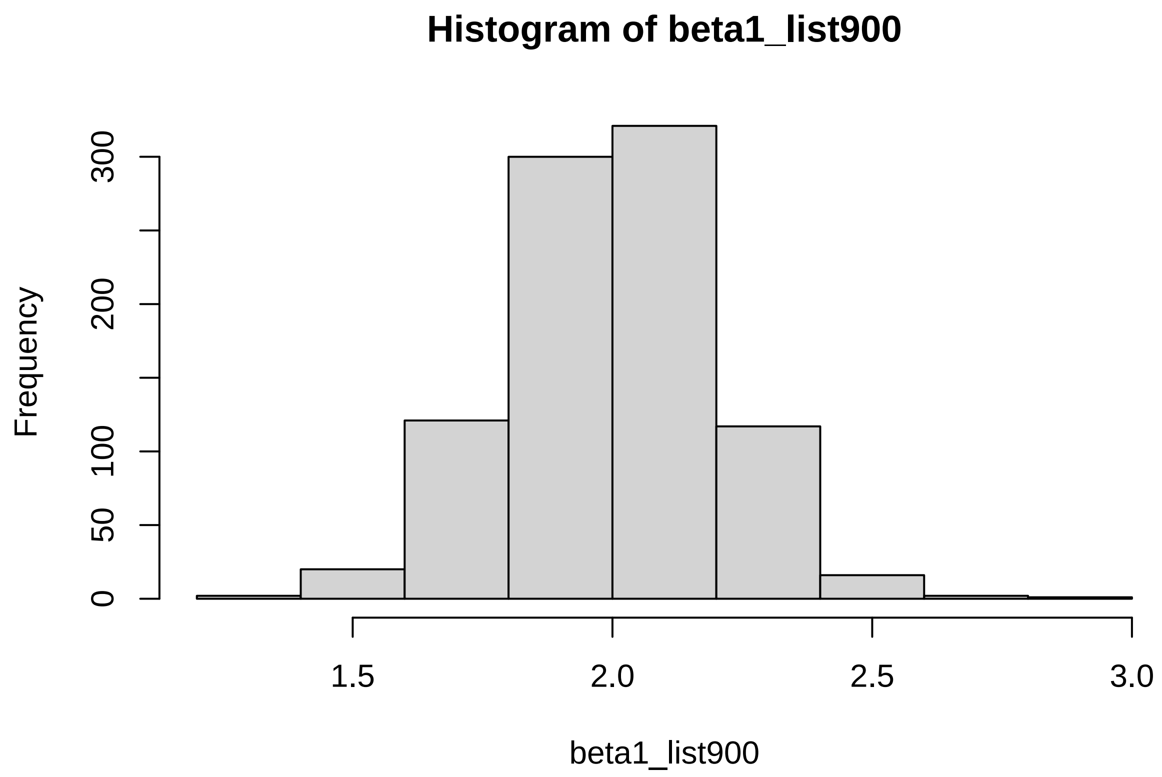
2e

```
matrix_X900 <- matrix(rnorm(10*900, mean =0, sd = 1), nrow =900, ncol = 9)
epsilon900 <- matrix(rnorm(10*900, mean =0, sd =0.5), nrow = 900, ncol = 9)
matrix_Y900 <- 1 + 2*matrix_X900 + epsilon900

#create am empty list to use later
beta1_list900 = NULL

#iterate thru matrix, then generate beta 1 coefficient for the given index and save to list to plot lat
for(i in 1:900){
  beta1_list900 <- c(beta1_list900, lm(matrix_Y900[i,] ~ matrix_X900[i,])$coefficients[2])
}


hist(beta1_list900)
```


Histogram of beta1_list900
Frequency
0
50
100
200
300
1.5
2.0
2.5
3.0
beta1_list900