Kathleen Zhen

999210972

STA 141A HW 5

`The codes and results derived by using these codes constitute my own work. I have consulted the following resources regarding this assignment: Noa Shadom

NUMBER 1

1. I first set.seed(100) so I can reproduce the same results. To generate a random sample of size n = 100 following the univariate regression model, where X's are independent Chi-squared random variables with degrees of freedom 6, and errors are i.i.d with a normal distribution, I did the following:

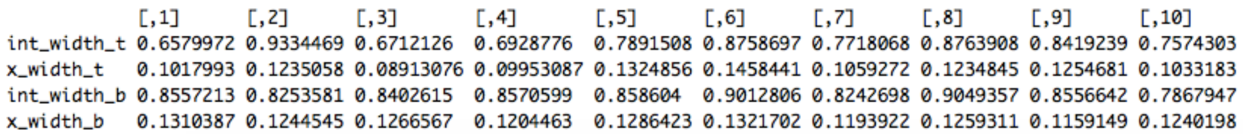
   ```
   norm <- rnorm(100, 0, 1) #generate 100 random data based on N(0, 1)
   x = rchisq(n = 100, df = 6) #generate 100 chi square random vars with df 6
   y = -5 + (2*x) + norm #solve the univariate regression model
   ```

2. Using lm(y ~ x) to fit my model, I got $B_0 = -5.055$ and $B_1 = 2.005$ and $\text{sigma}^2 = 1.049129$

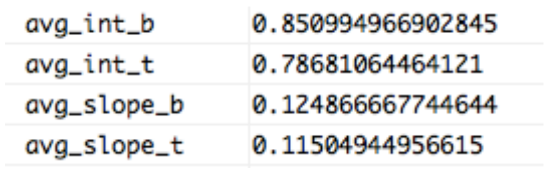
3. The 95% confidence interval for B_0 and B_1 after using a parametric bootstrap procedure with 400 bootstrap replicates is:

   B_0 : [-5.483866, -4.60724]
   B_1 : [1.941765, 2.073203]

4.



```
             [,1]      [,2]      [,3]       [,4]       [,5]      [,6]      [,7]      [,8]      [,9]      [,10]
int_width_t 0.6579972 0.9334469 0.6712126  0.6928776  0.7891508 0.8758697 0.7718068 0.8763908 0.8419239 0.7574303
x_width_t   0.1017993 0.1235058 0.08913076 0.09953087 0.1324856 0.1458441 0.1059272 0.1234845 0.1254681 0.1033183
int_width_b 0.8557213 0.8253581 0.8402615  0.8570599  0.858604  0.9012806 0.8242698 0.9049357 0.8556642 0.7867947
x_width_b   0.1310387 0.1244545 0.1266567  0.1204463  0.1286423 0.1321702 0.1193922 0.1259311 0.1159149 0.1240198
```



avg_int_b	0.850994966902845
avg_int_t	0.78681064464121
avg_slope_b	0.124866667744644
avg_slope_t	0.11504944956615

int_width_t is the width of the theoretical intercept confidence interval.
x_width_t is the width of the theoretical slope confidence interval.

int_width_b is the width of the bootstrap intercept confidence interval.
x_width_b is the width of the bootstrap slope confidence interval.

avg_int_b is the average width for the bootstrap intercept widths.
avg_int_t is the average width for the theoretical intercept widths.
avg_slope_b is the average width for the bootstrap slope widths.
avg_slope_t is the average width for the theoretical slope widths.

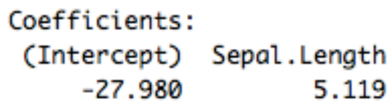
We can see that that bootstrap widths for both intercept and slope are much more narrow than their respective theoretical widths. This is because the bootstrap is resampling a lot of data creating more accurate predictions, therefore, the confidence intervals have a smaller range because we are more confident that it is accurate whereas the theoretical values are doing predictions from one data set and therefore the ranges are wider to ensure a 95% confidence.

NUMBER 2

1. I used the following code to split the data for setosa and versicolor while leaving the last 10 measurements as the testing and the rest as training.

```
data = iris[1:100,] #extracts setosa and versicolor data
test = rbind(data[41:50,], data[91:100,]) #set testing set
train = rbind(data[1:40,], data[51:90,]) #set training set
```

2. Aftering fitting a logistic regression model using glm() and setting the family = binomial, I got the following model parameters:



```
Coefficients:
 (Intercept)  Sepal.Length
     -27.980         5.119
```

The confusion matrix is:

True\Model	Setosa	Versicolor
Setosa	10	0
Versicolor	2	8

3. The decision boundary for the linear discriminant analysis is 2.222592.

The confusion matrix is:

True\Model	Setosa	Versicolor
Setosa	10	0
Versicolor	3	7

4. Using k-nearest neighbors classification method, with k = 3, the confusion matrix is:

True\Model	Setosa	Versicolor
Setosa	10	0

Versicolor	3	7

The confusion matrix for k = 5 is:

True\Model	Setosa	Versicolor
Setosa	10	0
Versicolor	2	8

5. After performing three different classification procedures, all of the methods knew that if the flower was setosa, it predicted setosa. A few of the true versicolor flowers was predicted as setosa. The logistic regression and k = 5 nearest neighbors both had an accuracy of 90% (misclassification rate of 10%) and linear discriminant analysis and k = 3 nearest neighbors had an accuracy of 85% (misclassification of 15%). The average accuracy is 87.5%.

APPENDIX

```
set.seed(100) #set seed to get reproducible results

#part 1
norm <- rnorm(100, 0, 1) #generate 100 random data based on N(0, 1)
x = rchisq(n = 100, df = 6) #generate 100 chi square random vars with df 6
y = -5 + (2*x) + norm #solve the univariate regression model

## PART 2
mod = lm(y ~ x) #fit the model
s2 = summary(mod)$sigma #sigma squared
theor = confint(mod) #theoretical confidence interval

##PART 3
library(readr)
library(broom)

resid= augment(mod)
data = augment(mod)
resample = function(data) {
  n = nrow(data)
  # Sample row numbers (i) rather than values
  idx = sample(n, n, replace = TRUE)

  # Use row numbers to get new residuals.
  res_samp = data$.resid[idx]

  # y2_i =  b_0 + b_1 * x_i    + e
  y_samp =  data$.fitted      + res_samp

  # Insert new response (y_i) into data frame, keeping old covariates (x_i)
  data$new = y_samp

  # Fit the same model with new data (y2_i, x_i).
  new_mod = lm(new ~ x, data)

  return (coef(new_mod))
}

# Bootstrap 400 times
boot = replicate(400, resample(resid))

#95% CI
ci_intercept = quantile(boot[1, ], c(0.025, 0.975))
```

```r
ci_slope     = quantile(boot[2, ], c(0.025, 0.975))

#PART 4
accuracy_test = function(seed) {
  set.seed(seed)

  #part 1
  norm <- rnorm(100, 0, 1) #generate 100 random data based on N(0, 1)
  x = rchisq(n = 100, df = 6) #generate 100 chi square random vars with df 6
  y = -5 + (2*x) + norm #solve the univariate regression model

  ## PART 2
  mod = lm(y ~ x) #least squares regression model
  theoretical = confint(mod) #theoretical CI
  int_width_t = abs(theoretical[1] - theoretical[3]) #width of CI
  x_width_t = abs(theoretical[2] - theoretical[4]) #width of CI

  #part 3
  data = augment(mod)

  #400 bootstraps
  boot = replicate(400, resample(resid))

  #95% CI
  ci_intercept = quantile(boot[1, ], c(0.025, 0.975))
  ci_slope     = quantile(boot[2, ], c(0.025, 0.975))

  int_width_b = abs(ci_intercept[[1]] - ci_intercept[[2]]) #widths
  x_width_b = abs(ci_slope[[1]] - ci_slope[[2]]) #widths

  #put all widths into data frame
  widths = data.frame(int_width_t, x_width_t, int_width_b, x_width_b)

  return (widths)
}

#run function 10 times to get accurate data
all_ci_widths = sapply(1:10, accuracy_test)

#calculate mean of all widths
avg_int_t = mean(as.numeric(as.vector(all_ci_widths[1,])))
avg_slope_t = mean(as.numeric(as.vector(all_ci_widths[2,])))
avg_int_b = mean(as.numeric(as.vector(all_ci_widths[3,])))
avg_slope_b = mean(as.numeric(as.vector(all_ci_widths[4,])))
```

NUMBER 2

```r
data = iris[1:100,] #extracts setosa and versicolor data
test = rbind(data[41:50,], data[91:100,]) #set testing set
train = rbind(data[1:40,], data[51:90,]) #set training set

##part 2

#logistic regression model
log_model = glm(Species ~ Sepal.Length, train, family = binomial)

# Predict for test data. Use type = "response" to get class probabilities.
log_pred = predict(log_model, test, type = "response")
# Convert predictions to 1 or 2, for category 1 or 2 respectively.
log_pred = (log_pred > 0.5) + 1
# Convert predictions to setosa or versicolor, same category order as original data.
log_pred = levels(train$Species)[log_pred]

# Make a confusion matrix by tabulating true classes against predicted classes.
test = droplevels(test) #drop all other levels in the data
log_con = table(true = test$Species, model = log_pred)

##part 3
library (MASS)

train = droplevels(train) #remove other variables thats not setosa/versicolor

#lda model
lda_model = lda(Species ~ Sepal.Length, train)

# Predict for test data. Use type = "response" to get class probabilities.
lda_pred = predict(lda_model, test, type = "response")

# Convert predictions to sestosa or versicolor, same category order as original data.
lda_pred = levels(train$Species)[lda_pred$class]

# Make a confusion matrix by tabulating true classes against predicted classes.
lda_con = table(true = test$Species, model = lda_pred)

##PART 4

library(class)
# Fit knn (k = 3) model.
knn_pred = knn(
  train = train["Sepal.Length"], # 1-col data frame
  test  = test["Sepal.Length"],  # 1-col data frame
  cl    = train$Species,        # vector
  k     = 3
)

# Confusion matrix.
knn_con3 = table(true = test$Species, model = knn_pred)

#Fit knn (k = 5) model
knn_pred = knn(

  train = train["Sepal.Length"], # 1-col data frame
  test  = test["Sepal.Length"],  # 1-col data frame
  cl    = train$Species,         # vector
  k     = 5
)

# Confusion matrix.
knn_con5 = table(true = test$Species, model = knn_pred)
```