Homework 3

Marc Hughes

Table of contents

Link to the Github repository

❗ Due: Thu, Mar 2, 2023 @ 11:59pm

Please read the instructions carefully before submitting your assignment.

1. This assignment requires you to only upload a `PDF` file on Canvas
2. Don't collapse any code cells before submitting.
3. Remember to make sure all your code output is rendered properly before uploading your submission.

Please add your name to the author information in the frontmatter before submitting your assignment

For this assignment, we will be using the Wine Quality dataset from the UCI Machine Learning Repository. The dataset consists of red and white *vinho verde* wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests

We will be using the following libraries:

```r
library(readr)
library(tidyr)
```

```
Warning: package 'tidyr' was built under R version 4.2.2
```

```r
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.2.2
```

```

Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(purrr)
```

```
Warning: package 'purrr' was built under R version 4.2.2
```

```r
library(car)
```

```
Warning: package 'car' was built under R version 4.2.2
```

```
Loading required package: carData
```

```
Warning: package 'carData' was built under R version 4.2.2
```

```

Attaching package: 'car'
```

```
The following object is masked from 'package:purrr':

    some
```

```
The following object is masked from 'package:dplyr':

    recode
```

```r
library(glmnet)
```

```
Warning: package 'glmnet' was built under R version 4.2.2
```

```
Loading required package: Matrix
```

```
Attaching package: 'Matrix'
```

```
The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

```
Loaded glmnet 4.1-6
```

```r
library(corrplot)
```

```
Warning: package 'corrplot' was built under R version 4.2.2
```

```
corrplot 0.92 loaded
```

```r
library(broom)
```

```
Warning: package 'broom' was built under R version 4.2.2
```

Question 1

50 points

Regression with categorical covariate and *t*-Test

1.1 (5 points)

Read the wine quality datasets from the specified URLs and store them in data frames `df1` and `df2`.

```
url1 <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequalit

url2 <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequalit


df1 <- read.csv(url1, sep = ';')
df2 <- read.csv(url2, sep = ';')
```

1.2 (5 points)

Perform the following tasks to prepare the data frame `df` for analysis:

1. Combine the two data frames into a single data frame `df`, adding a new column called `type` to indicate whether each row corresponds to white or red wine.
2. Rename the columns of `df` to replace spaces with underscores
3. Remove the columns `fixed_acidity` and `free_sulfur_dioxide`
4. Convert the `type` column to a factor
5. Remove rows (if any) with missing values.

```
# adding new column to both data frames that will distinguish the two types of wine when b
df1$type = "white"
df2$type = "red"

# binding both data frames
df <- rbind(df1, df2)

# replacing the periods with and underscore
colnames(df) <- gsub('\\.', '_', colnames(df))

# removing columns 'fixed_acidity' and 'free_sulfur_dioxide' from the data frame
df <- df %>%
  select(!fixed_acidity & !free_sulfur_dioxide)
```

```r
# changing the 'type' column to a factor
df$type <- factor(df$type)

# dropping any missing values
df <- df %>%
  drop_na()
```

Your output to R `dim(df)` should be

```
[1] 6497   11
```

1.3 (20 points)

Recall from STAT 200, the method to compute the t statistic for the the difference in means (with the equal variance assumption)

1. Using `df` compute the mean of `quality` for red and white wine separately, and then store the difference in means as a variable called `diff_mean`.
2. Compute the pooled sample variance and store the value as a variable called `sp_squared`.
3. Using `sp_squared` and `diff_mean`, compute the t Statistic, and store its value in a variable called `t1`.

```r
# creating temporary df to calculate mean
temp_df <-
  df %>%
  group_by(type) %>%
  summarise("quality_mean" = mean(quality))

# calculating mean
diff_mean <- abs(temp_df$quality_mean[temp_df$type == "white"] - temp_df$quality_mean[temp

# finding lengths
n1 <- length(df$quality[df$type == "white"])
n2 <- length(df$quality[df$type == "red"])

var1 <- var(df$quality[df$type == "white"])
var2 <- var(df$quality[df$type == "red"])

# manually calculating sp_squared
```

```r
sp_squared <- ((n1-1)*var1 + (n2-1)*var2) / (n1+n2-2)

# calculating standard deviation
sd1 <- sd(df$quality[df$type == "white"])
sd2 <- sd(df$quality[df$type == "red"])

# calculating the t-statistic
t1 <-  diff_mean / sqrt(sp_squared*(1/n1 + 1/n2))
```

1.4 (10 points)

Equivalently, R has a function called `t.test()` which enables you to perform a two-sample *t*-Test without having to compute the pooled variance and difference in means.

Perform a two-sample t-test to compare the quality of white and red wines using the `t.test()` function with the setting `var.equal=TRUE`. Store the t-statistic in `t2`.

```r
# using 't.test()' function to calculate the t-statistic
t_test <- t.test(df$quality[df$type == "white"], df$quality[df$type == "red"], var.equal =
t2 <- t_test$statistic
```

1.5 (5 points)

Fit a linear regression model to predict `quality` from `type` using the `lm()` function, and extract the *t*-statistic for the `type` coefficient from the model summary. Store this *t*-statistic in `t3`.

```r
fit <- lm(quality ~ type, data = df)
t3 <- summary(fit)$coefficients[2, "t value"]
```

1.6 (5 points)

Print a vector containing the values of `t1`, `t2`, and `t3`. What can you conclude from this? Why?

```r
c(t1, t2, t3)
```

```
              t
9.68565 9.68565 9.68565
```

You can conclude that all three of the methods are valid ways of extracting the t-statistic due to all t values being the exact same regardless of the method. In addition, this shows that the t-statistic is significant which allows us to reject the null hypothesis.

Question 2

25 points

Collinearity

2.1 (5 points)

Fit a linear regression model with all predictors against the response variable `quality`. Use the `broom::tidy()` function to print a summary of the fitted model. What can we conclude from the model summary?

```r
full_model <- lm(quality ~ ., data = df)
broom::tidy(summary(full_model))
```

```
# A tibble: 11 x 5
   term                  estimate std.error statistic  p.value
   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)          57.5       9.33         6.17  7.44e-10
 2 volatile_acidity     -1.61      0.0806     -20.0   4.07e-86
 3 citric_acid           0.0272    0.0783       0.347 7.28e- 1
 4 residual_sugar        0.0451    0.00416     10.8   3.64e-27
 5 chlorides            -0.964     0.333       -2.90  3.78e- 3
 6 total_sulfur_dioxide -0.000329  0.000262    -1.25  2.10e- 1
 7 density             -55.2       9.32        -5.92  3.34e- 9
 8 pH                          0.188     0.0661       2.85  4.38e- 3
 9 sulphates                   0.662     0.0758       8.73  3.21e-18
10 alcohol                     0.277     0.0142      19.5   1.87e-82
11 typewhite                  -0.386     0.0549      -7.02  2.39e-12
```

We can conclude that all of the p-values are significant which means we reject the null hypothesis and accept the alternative. In addition, the summary shows, evident by the "statistic" column, that only a select few of the t values are statistically significant.

2.2 (10 points)

Fit two **simple** linear regression models using `lm()`: one with only `citric_acid` as the predictor, and another with only `total_sulfur_dioxide` as the predictor. In both models, use `quality` as the response variable. How does your model summary compare to the summary from the previous question?

```r
model_citric <- lm(quality ~ citric_acid, data = df)
summary(model_citric)
```

```

Call:
lm(formula = quality ~ citric_acid, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.9938 -0.7831  0.1552  0.2426  3.1963 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.65461    0.02602 217.343   <2e-16 ***
citric_acid  0.51398    0.07429   6.918    5e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8701 on 6495 degrees of freedom
Multiple R-squared:  0.007316,  Adjusted R-squared:  0.007163 
F-statistic: 47.87 on 1 and 6495 DF,  p-value: 5.002e-12
```

```r
model_sulfur <- lm(quality ~ total_sulfur_dioxide, data = df)
summary(model_sulfur)
```

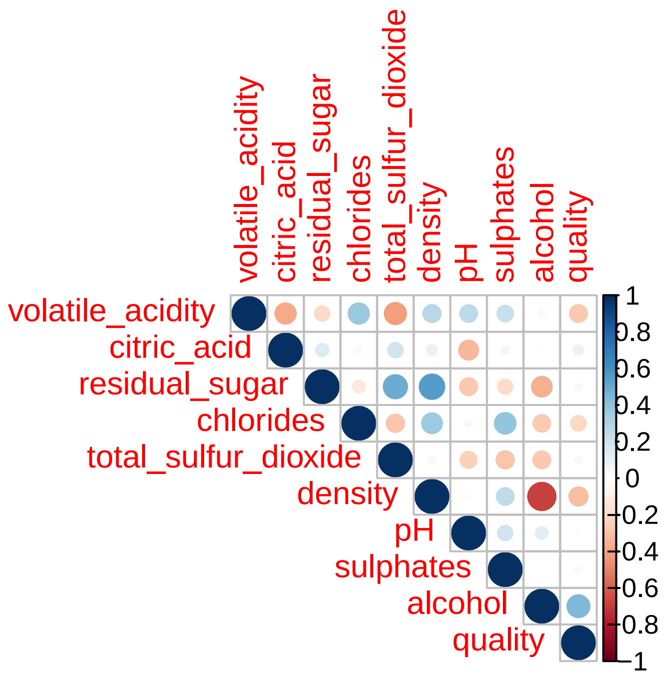
```

Call:
lm(formula = quality ~ total_sulfur_dioxide, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.8866 -0.7971  0.1658  0.2227  3.1965 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           5.8923848  0.0246717 238.831  < 2e-16 ***
total_sulfur_dioxide -0.0006394  0.0001915  -3.338 0.000848 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8726 on 6495 degrees of freedom
Multiple R-squared:  0.001713,	Adjusted R-squared:  0.001559 
F-statistic: 11.14 on 1 and 6495 DF,  p-value: 0.000848
```

The t value of the "citric_acid" predictor under the model_citric model shows a large increase to the t value shown in the model with all predictors. The increase was so much so that "citric_acid" actually has a significant t value in "model_critic". On the other hand, the t value of the "total_sulfur_dioxide" predictor is even smaller and remains insignificant.

2.3 (5 points)

Visualize the correlation matrix of all numeric columns in `df` using `corrplot()`

```r
# creating data frame with only numeric columns
df_numeric <- df %>%
  keep(is.numeric)

# creating correlation matrix
cor_mat <- cor(df_numeric)
# visualizing the correlation matrix
corrplot(cor_mat, type = 'upper')
```



2.4 (5 points)

Compute the variance inflation factor (VIF) for each predictor in the full model using `vif()` function. What can we conclude from this?

```r
vif(full_model)
```

```
volatile_acidity          citric_acid       residual_sugar
        2.103853             1.549248             4.680035
       chlorides total_sulfur_dioxide              density
        1.625065             2.628534             9.339357
              pH            sulphates              alcohol
        1.352005             1.522809             3.419849
            type
        6.694679
```

We can conclude that predictors like; 'volatile_acidity', 'residual_sugar', 'total_sulfur_dioxide', 'density', 'alcohol', and 'type' all of relatively high variance inflation factors. Having a high variance inflation means that the predictors listed are highly correlated with other variables in the model. This means that values like the t-statistic and p-value vastly different for variables with a high VIF depending on the inclusion of other highly correlated variables within the model.

Question 3

40 points

Variable selection

3.1 (5 points)

Run a backward stepwise regression using a `full_model` object as the starting model. Store the final formula in an object called `backward_formula` using the built-in `formula()` function in R

```
full_model <- lm(quality ~ ., data = df)

backward_formula <- step(full_model, direction = "backward", scope=formula(full_model))
```

```
Start:  AIC=-3953.43
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides + 
    total_sulfur_dioxide + density + pH + sulphates + alcohol + 
    type

                       Df Sum of Sq    RSS     AIC
- citric_acid           1     0.066 3523.6 -3955.3
- total_sulfur_dioxide  1     0.854 3524.4 -3953.9
<none>                              3523.5 -3953.4
- pH                    1     4.413 3527.9 -3947.3
- chlorides             1     4.559 3528.1 -3947.0
- density               1    19.054 3542.6 -3920.4
- type                  1    26.794 3550.3 -3906.2
- sulphates             1    41.399 3564.9 -3879.5
- residual_sugar        1    63.881 3587.4 -3838.7
- alcohol               1   206.860 3730.4 -3584.8
- volatile_acidity      1   216.549 3740.0 -3567.9
```

```
Step:  AIC=-3955.3
quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide +
    density + pH + sulphates + alcohol + type

                       Df Sum of Sq    RSS     AIC
- total_sulfur_dioxide  1     0.818 3524.4 -3955.8
<none>                              3523.6 -3955.3
- chlorides             1     4.495 3528.1 -3949.0
- pH                    1     4.536 3528.1 -3948.9
- density               1    20.794 3544.4 -3919.1
- type                  1    26.943 3550.5 -3907.8
- sulphates             1    41.491 3565.1 -3881.2
- residual_sugar        1    67.371 3590.9 -3834.3
- alcohol               1   235.151 3758.7 -3537.6
- volatile_acidity      1   252.565 3776.1 -3507.5

Step:  AIC=-3955.8
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type

                   Df Sum of Sq    RSS     AIC
<none>                          3524.4 -3955.8
- pH                1     4.295 3528.7 -3949.9
- chlorides         1     4.523 3528.9 -3949.5
- density           1    21.540 3545.9 -3918.2
- sulphates         1    40.711 3565.1 -3883.2
- type              1    43.664 3568.0 -3877.8
- residual_sugar    1    66.572 3591.0 -3836.2
- alcohol           1   244.545 3768.9 -3521.9
- volatile_acidity  1   256.695 3781.1 -3501.0
```

3.2 (5 points)

Run a forward stepwise regression using a `null_model` object as the starting model. Store the final formula in an object called `forward_formula` using the built-in `formula()` function in R

```r
null_model <- lm(quality ~ 1, data = df)
```

```r
forward_formula <- step(null_model, direction = "forward", scope = formula(full_model))
```

```
Start:  AIC=-1760.04
quality ~ 1

                       Df Sum of Sq    RSS     AIC
+ alcohol               1    977.95 3975.7 -3186.9
+ density               1    463.41 4490.3 -2396.2
+ volatile_acidity      1    349.71 4604.0 -2233.7
+ chlorides             1    199.47 4754.2 -2025.1
+ type                  1     70.53 4883.2 -1851.2
+ citric_acid           1     36.24 4917.4 -1805.7
+ total_sulfur_dioxide  1      8.48 4945.2 -1769.2
+ sulphates             1      7.34 4946.3 -1767.7
+ residual_sugar        1      6.77 4946.9 -1766.9
+ pH                    1      1.88 4951.8 -1760.5
<none>                              4953.7 -1760.0

Step:  AIC=-3186.88
quality ~ alcohol

                       Df Sum of Sq    RSS     AIC
+ volatile_acidity      1   307.508 3668.2 -3707.9
+ residual_sugar        1    85.662 3890.1 -3326.4
+ type                  1    54.335 3921.4 -3274.3
+ citric_acid           1    40.303 3935.4 -3251.1
+ chlorides             1    39.696 3936.0 -3250.1
+ total_sulfur_dioxide  1    31.346 3944.4 -3236.3
+ sulphates             1     7.859 3967.9 -3197.7
+ pH                    1     5.938 3969.8 -3194.6
<none>                              3975.7 -3186.9
+ density               1     0.005 3975.7 -3184.9

Step:  AIC=-3707.89
quality ~ alcohol + volatile_acidity

                       Df Sum of Sq    RSS     AIC
+ sulphates             1    48.259 3620.0 -3791.9
+ density               1    38.704 3629.5 -3774.8
+ residual_sugar        1    29.751 3638.5 -3758.8
+ type                  1    28.895 3639.3 -3757.3
+ total_sulfur_dioxide  1     5.619 3662.6 -3715.9
+ pH                    1     5.533 3662.7 -3715.7
<none>                              3668.2 -3707.9
+ chlorides             1     0.162 3668.1 -3706.2
+ citric_acid           1     0.099 3668.1 -3706.1

Step:  AIC=-3791.94
quality ~ alcohol + volatile_acidity + sulphates

                       Df Sum of Sq    RSS     AIC
+ residual_sugar        1    43.989 3576.0 -3869.4
+ density               1    18.661 3601.3 -3823.5
+ type                  1     6.012 3614.0 -3800.7
+ chlorides             1     4.988 3615.0 -3798.9
+ citric_acid           1     2.031 3617.9 -3793.6
+ pH                    1     1.903 3618.1 -3793.4
<none>                              3620.0 -3791.9
+ total_sulfur_dioxide  1     0.817 3619.2 -3791.4

Step:  AIC=-3869.37
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar

                       Df Sum of Sq    RSS     AIC
+ type                  1   20.7581 3555.2 -3905.2
+ total_sulfur_dioxide  1   13.3542 3562.6 -3891.7
+ pH                    1    6.6430 3569.3 -3879.5
+ citric_acid           1    4.3384 3571.6 -3875.3
+ chlorides             1    1.8907 3574.1 -3870.8
<none>                              3576.0 -3869.4
+ density               1    0.0071 3576.0 -3867.4

Step:  AIC=-3905.19
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar + 
    type

                       Df Sum of Sq    RSS     AIC
+ density               1   20.4623 3534.8 -3940.7
+ chlorides             1    6.6602 3548.6 -3915.4
+ citric_acid           1    5.2242 3550.0 -3912.7
+ pH                    1    3.9477 3551.3 -3910.4
+ total_sulfur_dioxide  1    1.2539 3554.0 -3905.5
<none>                              3555.2 -3905.2

Step:  AIC=-3940.7
```

```
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type + density

                       Df Sum of Sq    RSS     AIC
+ chlorides             1    6.0826 3528.7 -3949.9
+ pH                    1    5.8541 3528.9 -3949.5
<none>                              3534.8 -3940.7
+ citric_acid           1    0.8471 3533.9 -3940.3
+ total_sulfur_dioxide  1    0.5646 3534.2 -3939.7

Step:  AIC=-3949.89
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type + density + chlorides

                       Df Sum of Sq    RSS     AIC
+ pH                    1    4.2945 3524.4 -3955.8
<none>                              3528.7 -3949.9
+ total_sulfur_dioxide  1    0.5765 3528.1 -3948.9
+ citric_acid           1    0.2338 3528.4 -3948.3

Step:  AIC=-3955.8
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type + density + chlorides + pH

                       Df Sum of Sq    RSS     AIC
<none>                              3524.4 -3955.8
+ total_sulfur_dioxide  1   0.81762 3523.6 -3955.3
+ citric_acid           1   0.02919 3524.4 -3953.9
```

3.3 (10 points)

1. Create a `y` vector that contains the response variable (`quality`) from the `df` dataframe.
2. Create a design matrix `X` for the `full_model` object using the `make_model_matrix()` function provided in the Appendix.
3. Then, use the `cv.glmnet()` function to perform LASSO and Ridge regression with `X` and `y`.

```
# creating y vector
y <- df$quality
```

```r
# creating a design matrix
make_model_matrix <- function(formula){
  X <- model.matrix(formula, df)[, -1]
  cnames <- colnames(X)
  for(i in 1:ncol(X)){
    if(!cnames[i] == "typewhite"){
      X[, i] <- scale(X[, i])
    } else {
      colnames(X)[i] <- "type"
    }
  }
  return(X)
}
# performing lasso regression
lassoReg <- cv.glmnet(x = make_model_matrix(forward_formula), y = y, alpha = 1)
lassoReg
```

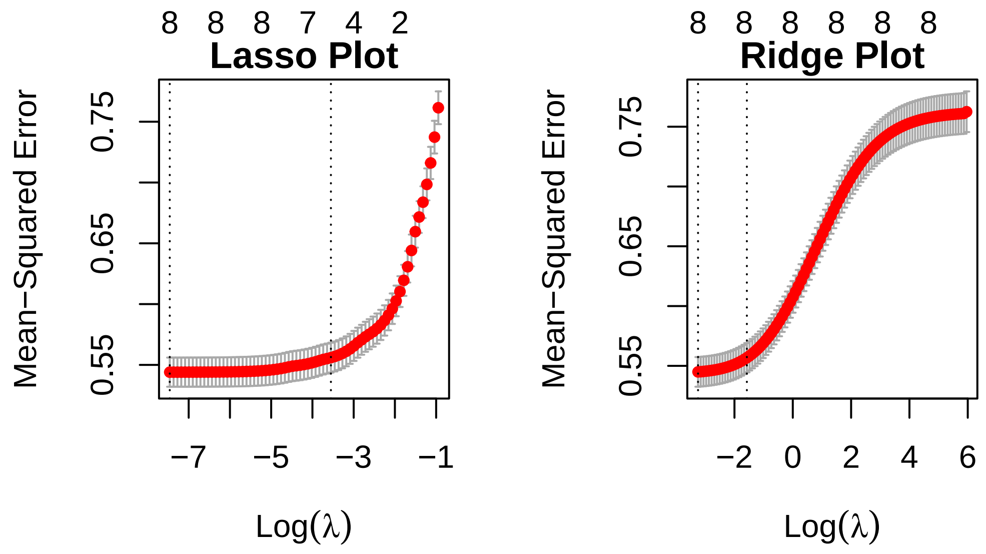
```
Call:  cv.glmnet(x = make_model_matrix(forward_formula), y = y, alpha = 1)

Measure: Mean-Squared Error

      Lambda Index Measure      SE Nonzero
min 0.000576    71   0.544 0.01199       8
1se 0.028674    29   0.556 0.01231       4
```

```r
# performing ridge regression
ridgeReg <- cv.glmnet(x = make_model_matrix(forward_formula), y = y, alpha = 0)
ridgeReg
```

```
Call:  cv.glmnet(x = make_model_matrix(forward_formula), y = y, alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure      SE Nonzero
min 0.0388   100  0.5449 0.01241       8
1se 0.2071    82  0.5568 0.01225       8
```

Create side-by-side plots of the ridge and LASSO regression results. Interpret your main findings.

```r
par(mfrow=c(1, 2))

plot(lassoReg)
title("Lasso Plot", line = .3)

plot(ridgeReg)
title("Ridge Plot", line = .3)
```



The LASSO plot suggests the lambda that is most optimal for the minimization of mean-squared error. According to the plot, this model has 4 variables. The ridge plot, on the other hand, uses a lambda of approximately -2 to minimize mean-squared error and includes all variables because ridge regression does not do variable selection.

3.4 (5 points)

Print the coefficient values for LASSO regression at the `lambda.1se` value? What are the variables selected by LASSO?

Store the variable names with non-zero coefficients in `lasso_vars`, and create a formula object called `lasso_formula` using the `make_formula()` function provided in the Appendix.

```r
lasso_coef <- coef(lassoReg, s = "lambda.1se")
lasso_coef
```

```
9 x 1 sparse Matrix of class "dgCMatrix"
                          s1
(Intercept)       5.81837771
alcohol           0.36804234
volatile_acidity -0.19397495
sulphates         0.05807566
residual_sugar    0.04408679
type              .
density           .
chlorides         .
pH                .
```

```r
lasso_vars <- rownames(lasso_coef)[which(abs(lasso_coef) > 0)][-1] # exclude the intercept
lasso_vars
```

```
[1] "alcohol"          "volatile_acidity" "sulphates"        "residual_sugar"
```

```r
make_formula <- function(x){
  as.formula(
    paste("quality ~ ", paste(x, collapse = " + "))
  )
}

lasso_formula <- make_formula(lasso_vars)
```

The variables selected are “volatile_acidity”, “residual_sugar”, “sulphates”, and “alcohol”.

3.5 (5 points)

Print the coefficient values for ridge regression at the `lambda.1se` value? What are the variables selected here?

Store the variable names with non-zero coefficients in `ridge_vars`, and create a formula object called `ridge_formula` using the `make_formula()` function provided in the Appendix.

```
ridge_coef <- coef(ridgeReg, s = "lambda.1se")
ridge_coef
```

```
9 x 1 sparse Matrix of class "dgCMatrix"
                          s1
(Intercept)       5.90326244
alcohol           0.27936003
volatile_acidity -0.17882869
sulphates         0.08259364
residual_sugar    0.08901663
type             -0.11259617
density          -0.08152371
chlorides        -0.04317661
pH                0.02001726
```

```
ridge_vars <- rownames(ridge_coef)[which(abs(ridge_coef) > 0)][-1] # exclude the intercept
ridge_vars
```

```
[1] "alcohol"          "volatile_acidity" "sulphates"        "residual_sugar"
[5] "type"             "density"          "chlorides"        "pH"
```

```
make_formula <- function(x){
  as.formula(
    paste("quality ~ ", paste(x, collapse = " + "))
  )
}

ridge_formula <- make_formula(ridge_vars)
```

All the variables are selected; “volatile_acidity”, “citric_acid”, “residual_sugar”, “chlorides”, “total_sulfur_dioxide”, “density”, “pH”, “sulphates”, “alcohol”, and “type”.

3.6 (10 points)

What is the difference between stepwise selection, LASSO and ridge based on you analyses above?

Stepwise selection uses 'AIC' to do variable selection and seems to retain more variables than LASSO regression. On the other hand, LASSO regression uses mean-squared error to do variable selection. Finally, ridge regression does not do variable selection whatsoever.

Question 4

70 points

Variable selection

4.1 (5 points)

Excluding `quality` from `df` we have 10 possible predictors as the covariates. How many different models can we create using any subset of these 10 coavriates as possible predictors? Justify your answer.

We can create 2^10 different models, so 1024 different models. This is because order doesn't matter and you can either include a predictor or exclude a predictor.

4.2 (20 points)

Store the names of the predictor variables (all columns except `quality`) in an object called `x_vars`.

```r
# storing predictor variables
x_vars <- colnames(df %>% select(-quality))
```

Use:

- the `combn()` function (built-in R function) and
- the `make_formula()` (provided in the Appendix)

to **generate all possible linear regression formulas** using the variables in `x_vars`. This is most optimally achieved using the `map()` function from the `purrr` package.

```
formulas <- map(
  1:length(x_vars),
  function(x){
    vars <- combn(x_vars, x, simplify = FALSE)
    map(vars, make_formula)
  }
) %>% unlist()
```

If your code is right the following command should return something along the lines of:

```
sample(formulas, 4) %>% as.character()
```

```
[1] "quality ~ volatile_acidity + residual_sugar + density + sulphates + type"
[2] "quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide + alcohol
[3] "quality ~ citric_acid + residual_sugar + density + alcohol"
[4] "quality ~ citric_acid + residual_sugar + chlorides + total_sulfur_dioxide + pH + sulphat
```

```
# Output:
# [1] "quality ~ volatile_acidity + residual_sugar + density + pH + alcohol"
# [2] "quality ~ citric_acid"
# [3] "quality ~ volatile_acidity + citric_acid + residual_sugar + total_sulfur_dioxide +
# [4] "quality ~ citric_acid + chlorides + total_sulfur_dioxide + pH + alcohol + type"
```

4.3 (10 points)

Use `map()` and `lm()` to fit a linear regression model to each formula in `formulas`, using `df` as the data source. Use `broom::glance()` to extract the model summary statistics, and bind them together into a single tibble of summaries using the `bind_rows()` function from `dplyr`.

```
models <- map(formulas, ~ lm(formula = .x, data = df))
summaries <- bind_rows(map(models, broom::glance))

summaries
```

```
# A tibble: 1,023 x 12
   r.squared adj.r.~1 sigma stati~2   p.value    df logLik    AIC    BIC devia~3
       <dbl>    <dbl> <dbl>   <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>   <dbl>
 1    0.0706   0.0705 0.842   493.  2.06e-105     1 -8100. 16206. 16226.   4604.
 2  0.00732  0.00716  0.870   47.9  5.00e- 12     1 -8314. 16634. 16654.   4917.
 3  0.00137  0.00121  0.873    8.89 2.87e-  3     1 -8333. 16673. 16693.   4947.
 4  0.0403   0.0401   0.856  273.   5.32e- 60     1 -8204. 16415. 16435.   4754.
 5  0.00171  0.00156  0.873   11.1  8.48e-  4     1 -8332. 16671. 16691.   4945.
 6  0.0935   0.0934   0.831  670.   9.66e-141     1 -8019. 16044. 16064.   4490.
 7  0.000380 0.000227 0.873    2.47 1.16e-  1     1 -8337. 16679. 16700.   4952.
 8  0.00148  0.00133  0.873    9.63 1.92e-  3     1 -8333. 16672. 16692.   4946.
 9  0.197    0.197    0.782 1598.   1.50e-312     1 -7623. 15253. 15273.   3976.
10  0.0142   0.0141   0.867   93.8  4.89e- 22     1 -8291. 16588. 16609.   4883.
# ... with 1,013 more rows, 2 more variables: df.residual <int>, nobs <int>,
#   and abbreviated variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

4.4 (5 points)

Extract the `adj.r.squared` values from `summaries` and use them to identify the formula with the ***highest*** adjusted R-squared value.

```
extraction1 <- summaries$adj.r.squared
```

Store resulting formula as a variable called `rsq_formula`.

```
rsq_formula <- formulas[which.max(extraction1)]
```

4.5 (5 points)

Extract the `AIC` values from `summaries` and use them to identify the formula with the ***lowest*** AIC value.

```
extraction2 <- summaries$AIC
```

Store resulting formula as a variable called `aic_formula`.

```
aic_formula <- formulas[which.min(extraction2)]
```

4.6 (15 points)

Combine all formulas shortlisted into a single vector called `final_formulas`.

```r
null_formula <- formula(null_model)
full_formula <- formula(full_model)

final_formulas <- c(
  null_formula,
  full_formula,
  backward_formula,
  forward_formula,
  lasso_formula,
  ridge_formula,
  rsq_formula,
  aic_formula
)
```

- Are `aic_formula` and `rsq_formula` the same? How do they differ from the formulas shortlisted in question 3?

The 'aic_formula' and 'rsq_formula' are not the same and they differ from the some of the other formulas in the way in which they were found and what they represent. For example, 'rsq_formula' was found by simply extracting the formula with the highest adjusted r-squared from the model summaries and matches no other formulas shortlisted. 'aic_formula', on the other hand, actually matches both the 'backward_formula' and the 'forward_formula'. This is because the 'aic_formula' was found by finding the formula with the lowest AIC which is exactly what backward and forward selection achieves.

- Which of these is more reliable? Why?

The 'aic_formula' is more reliable than 'rsq_formula' because it uses an already tried and tested way of finding an optimal model which is choosing a model that minimizes AIC.

- If we had a dataset with $10,000$ columns, which of these methods would you consider for your analyses? Why?

If we had a dataset with $10,000$ columns I would choose to use LASSO regression because they can handle high-dimensional datasets LASSO engages in feature selection so it can reduce the amount of parameters that would otherwise make the regression model much more computationally intensive.

4.7 (10 points)

Use map() and glance() to extract the sigma, adj.r.squared, AIC, df, and p.value statistics for each model obtained from final_formulas. Bind them together into a single data frame summary_table. Summarize your main findings.

```r
summary_table <- map(
  final_formulas,
  function(x) {
    ~ glance(x) %>%
      select(sigma, adj.r.squared, AIC, df, p.value) %>%
      mutate(formula = as.character(formula(x)))
  }
)
# could not get code below to work
# %>% bind_rows()

# summary_table %>% knitr::kable()
```

Appendix

Convenience function for creating a formula object

The following function which takes as input a vector of column names `x` and outputs a `formula` object with `quality` as the response variable and the columns of `x` as the covariates.

```
make_formula <- function(x){
  as.formula(
    paste("quality ~ ", paste(x, collapse = " + "))
  )
}

# For example the following code will
# result in a formula object
# "quality ~ a + b + c"
make_formula(c("a", "b", "c"))
```

Convenience function for `glmnet`

The `make_model_matrix` function below takes a `formula` as input and outputs a **rescaled** model matrix `X` in a format amenable for `glmnet()`

```
make_model_matrix <- function(formula){
  X <- model.matrix(formula, df)[, -1]
  cnames <- colnames(X)
  for(i in 1:ncol(X)){
    if(!cnames[i] == "typewhite"){
      X[, i] <- scale(X[, i])
    } else {
      colnames(X)[i] <- "type"
    }
  }
  return(X)
}
```

Session Information

Print your R session information using the following command

```r
sessionInfo()
```

```
R version 4.2.1 (2022-06-23 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22000)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

other attached packages:
 [1] broom_1.0.3   corrplot_0.92 glmnet_4.1-6  Matrix_1.4-1  car_3.1-1
 [6] carData_3.0-5 purrr_1.0.1   dplyr_1.1.0   tidyr_1.3.0   readr_2.1.4

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.10       pillar_1.8.1      compiler_4.2.1    iterators_1.0.14
 [5] tools_4.2.1       digest_0.6.31     jsonlite_1.8.4    evaluate_0.20
 [9] lifecycle_1.0.3   tibble_3.1.8      lattice_0.20-45   pkgconfig_2.0.3
[13] rlang_1.0.6       foreach_1.5.2     cli_3.6.0         yaml_2.3.7
[17] xfun_0.37         fastmap_1.1.1     withr_2.5.0       knitr_1.42
[21] generics_0.1.3    vctrs_0.5.2       hms_1.1.2         grid_4.2.1
[25] tidyselect_1.2.0 glue_1.6.2        R6_2.5.1          fansi_1.0.4
[29] survival_3.3-1    rmarkdown_2.20    tzdb_0.3.0        magrittr_2.0.3
[33] backports_1.4.1   splines_4.2.1     codetools_0.2-18 ellipsis_0.3.2
[37] htmltools_0.5.4   abind_1.4-5       shape_1.4.6       renv_0.16.0-53
[41] utf8_1.2.3
```