Homework 3

Insert your name here

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)
```

```
Warning: package 'readr' was built under R version 4.0.5
```

```r
library(tidyr)
```

```
Warning: package 'tidyr' was built under R version 4.0.5
```

```r
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.0.5

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)
library(glmnet)
```

```
Warning: package 'glmnet' was built under R version 4.0.5

Loading required package: Matrix

Warning: package 'Matrix' was built under R version 4.0.5

Attaching package: 'Matrix'

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

    expand, pack, unpack

Loaded glmnet 4.1-4
```

```r
#library(car)
```

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`.

```r
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, header = TRUE, sep = ";")
df2 <- read.csv(url2, header = TRUE, 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.

```r
df1$type <- "red"
df2$type <- "white"
df <- rbind(df1, df2)

colnames(df) <- gsub(" ", "_", colnames(df))
```

```r
df <- df[, !(colnames(df) %in% c("fixed.acidity", "free.sulfur.dioxide"))]

df$type <- factor(df$type)

df <- na.omit(df)

dim(df)
```

```
[1] 6497   11
```

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
mean_white <- mean(df$quality[df$type == "white"])
mean_red <- mean(df$quality[df$type == "red"])
diff_mean <- mean_white - mean_red

n_white <- length(df$quality[df$type == "white"])
n_red <- length(df$quality[df$type == "red"])
var_white <- var(df$quality[df$type == "white"])
var_red <- var(df$quality[df$type == "red"])
sp_squared <- ((n_white - 1) * var_white + (n_red - 1) * var_red) / (n_white + n_red - 2)

se <- sqrt(sp_squared * (1 / n_white + 1 / n_red))
t1 <- diff_mean / se
```

```r
t1
```

```
[1] -9.68565
```

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
t_test <- t.test(quality ~ type, data = df, var.equal = TRUE)
t2 <- t_test$statistic

t2
```

```
      t
9.68565
```

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"]

t3
```

```
[1] -9.68565
```

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) # Insert your code here
```

```
                 t
-9.68565  9.68565 -9.68565
```

—

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

```
# A tibble: 11 x 5
   term                  estimate std.error statistic  p.value
   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)          57.1        9.30         6.15  8.38e-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
```

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?

```
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
```

```
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
```

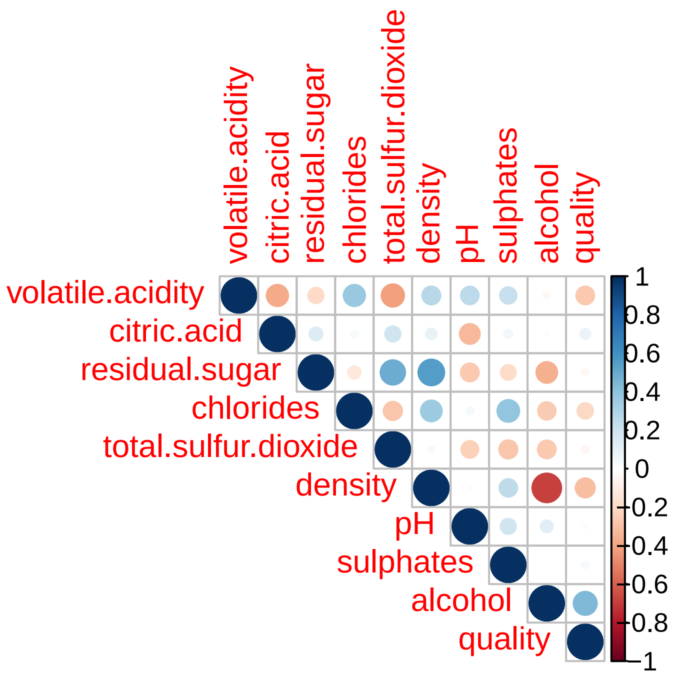
2.3 (5 points)

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

```
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
correlatoin_matrix <- df %>%
  keep(is.numeric) %>%
  cor()
corrplot(correlatoin_matrix, type = "upper", method = "circle")
```



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?

```
#vif_values <- vif(df) # Insert your code here
```

—

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 ~ ., 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

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

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
```

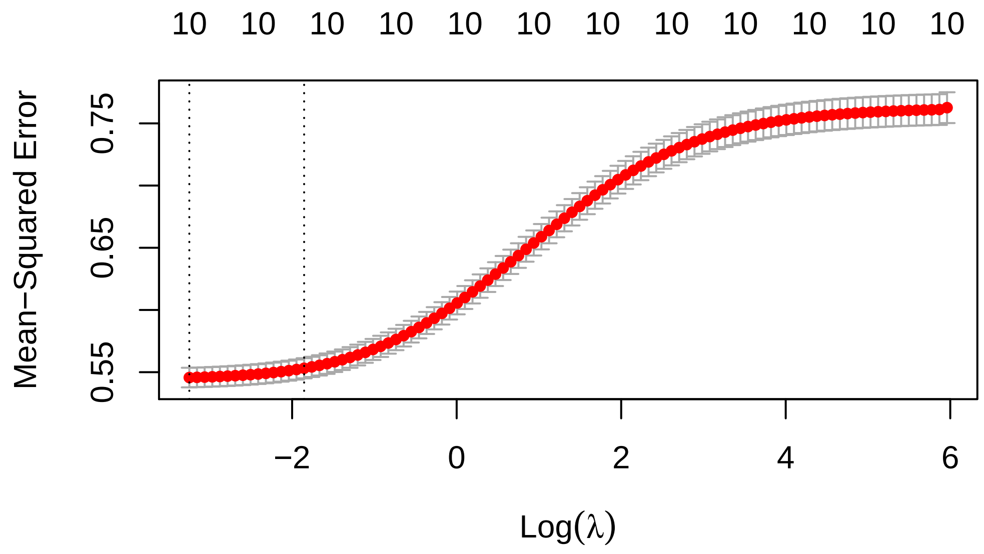
3.3 (10 points)

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`.

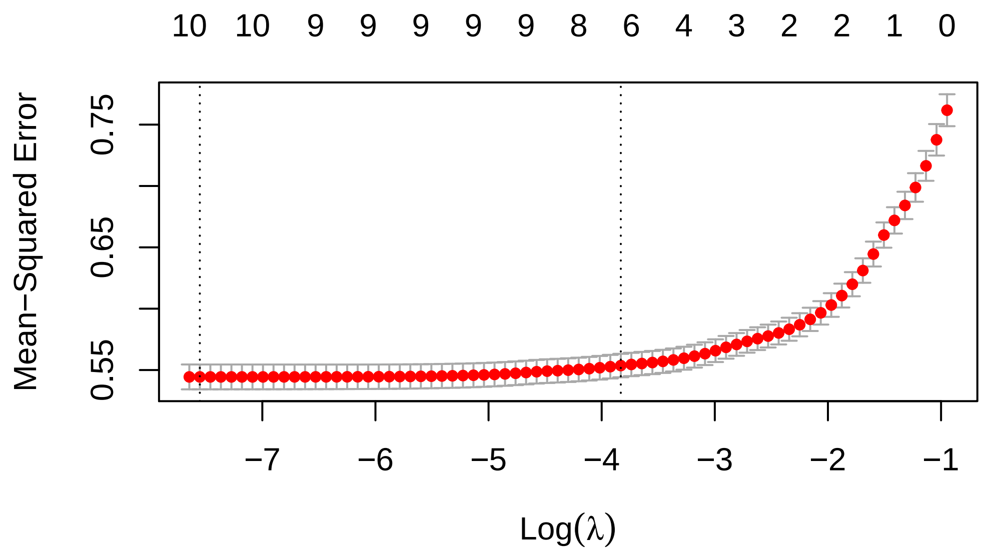
```r
y <- df$quality

make_model_matrix <- function(formula){
  X <- model.matrix(full_model, 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)
}

cv_ridge <- cv.glmnet(make_model_matrix(forward_formula), y, alpha = 0)
cv_lasso <- cv.glmnet(make_model_matrix(forward_formula), y, alpha = 1)
```

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

```r
plot(cv_ridge)
```



```r
plot(cv_lasso)
```



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_coefficient <- coef(cv_lasso, s = "lambda.1se")
lasso_coefficient
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                               s1
(Intercept)           5.816929919
volatile.acidity     -0.204848960
citric.acid           .
residual.sugar        0.059537104
chlorides            -0.001916886
total.sulfur.dioxide -0.008115301
density               .
```

```
pH                      0.003557628
sulphates               0.067073695
alcohol                 0.377201730
type                    0.005882625
```

```r
lasso_vars <- rownames(lasso_coefficient)[which((lasso_coefficient) > 0)]
lasso_vars
```

```
[1] "(Intercept)"    "residual.sugar" "pH"             "sulphates"
[5] "alcohol"        "type"
```

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

lasso_formula <- make_formula(lasso_vars)
lasso_formula
```

```
quality ~ (Intercept) + residual.sugar + pH + sulphates + alcohol +
    type
<environment: 0x0000000027c1c478>
```

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.

```r
ridge_coefficient <- coef(cv_ridge, s = "lambda.1se")
ridge_coefficient
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)           5.79431467
volatile.acidity     -0.18836482
citric.acid           0.01721478
residual.sugar        0.11167415
chlorides            -0.04481638
total.sulfur.dioxide -0.04027017
density              -0.08890876
pH                    0.02572467
sulphates             0.08571325
alcohol               0.28530948
type                  0.09777210
```

```
ridge_vars <- rownames(ridge_coefficient)[which((ridge_coefficient) > 0)]
ridge_vars
```

```
[1] "(Intercept)"    "citric.acid"    "residual.sugar" "pH"
[5] "sulphates"      "alcohol"        "type"
```

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

ridge_formula <- make_formula(ridge_vars)
ridge_formula
```

```
quality ~ (Intercept) + citric.acid + residual.sugar + pH + sulphates +
    alcohol + type
<environment: 0x0000000025de69c0>
```

3.6 (10 points)

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

```
"Stepwise uses 8 variables"
```

```
[1] "Stepwise uses 8 variables"
```

```
"Lasso uses 5 variables, not including the intercept"
```

```
[1] "Lasso uses 5 variables, not including the intercept"
```

```
"Ridge uses 6 variables, not including the intercept"
```

```
[1] "Ridge uses 6 variables, not including the intercept"
```

—

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.

```
"The number of different models we can create using any subset of n covariates is 2^n sinc
```

```
[1] "The number of different models we can create using any subset of n covariates is 2^n si
```

```
"In this case, we have n = 10 covariates, so the total number of different models we can c
```

```
[1] "In this case, we have n = 10 covariates, so the total number of different models we can
```

4.2 (20 points)

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

```
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.

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

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

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

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

```
[1] "quality ~ chlorides + density + pH + sulphates + alcohol + type"
[2] "quality ~ type"
[3] "quality ~ density + pH + alcohol"
[4] "quality ~ volatile.acidity + citric.acid + residual.sugar + total.sulfur.dioxide + sulph
```

```
# 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((-quality) ~ ., data=df)) # Insert your code here
summaries <- map(models,broom::glance) # Insert your code here
summaries_tibble <- bind_rows(summaries, .id="formula")
```

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.

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

max_adj_index <- which.max(R_squared)

r_form <- formulas[max_adj_index]
```

Store resulting formula as a variable called `rsq_formula`.

```
#rsq_formula <- ... #insert code here
```

4.5 (5 points)

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

```
... # Insert your code here
```

Store resulting formula as a variable called `aic_formula`.

```
aic_formula <- ... # Insert your code
```

4.6 (15 points)

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

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

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) ... # Insert your code here
) %>% 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

```
sessionInfo()
```

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19044)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

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

other attached packages:
[1] corrplot_0.92 glmnet_4.1-4  Matrix_1.4-1  purrr_0.3.4   dplyr_1.0.8
[6] tidyr_1.2.0   readr_2.1.2

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.8.3     pillar_1.8.1     compiler_4.0.3   iterators_1.0.14
 [5] tools_4.0.3      digest_0.6.29    jsonlite_1.8.0   evaluate_0.20
 [9] lifecycle_1.0.1  tibble_3.1.6     lattice_0.20-45  pkgconfig_2.0.3
[13] rlang_1.0.2      foreach_1.5.2    cli_3.2.0        rstudioapi_0.14
[17] yaml_2.3.5       xfun_0.20        stringr_1.4.0    knitr_1.30
[21] generics_0.1.0   vctrs_0.4.1      hms_1.1.2        grid_4.0.3
[25] tidyselect_1.1.2 glue_1.6.2       R6_2.5.1         fansi_0.4.2
[29] survival_3.2-7   rmarkdown_2.6    tzdb_0.3.0       magrittr_2.0.3
[33] backports_1.4.1  splines_4.0.3    codetools_0.2-16 ellipsis_0.3.2
[37] htmltools_0.5.1  shape_1.4.6      renv_0.16.0      utf8_1.1.4
[41] stringi_1.7.6    broom_1.0.3
```