Homework 3

Jacob Adams

Table of contents

! Important

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:

```
library(readr)
library(tidyr)
library(dplyr)
```

```
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(car)
```

```
Warning: package 'car' was built under R version 4.3.2

Loading required package: carData

Warning: package 'carData' was built under R version 4.3.2

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 4.3.2

Loading required package: Matrix

Warning: package 'Matrix' was built under R version 4.3.2

Attaching package: 'Matrix'
```

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

    expand, pack, unpack

Loaded glmnet 4.1-8
```

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/winequality-w

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


df1 <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winec
df2 <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winec
```

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 <- df1 %>%
  mutate(type = "white")
df2 <- df2 %>%
  mutate(type = "red")
df <- bind_rows(df1,df2)
df <- df %>%
  rename(fixed_acidity = fixed.acidity, volatile_acidity = volatile.acidity, citric_acid = c
df <- df %>%
  mutate(fixed_acidity = NULL, free_sulfur_dioxide = NULL) %>%
  mutate(type = as.factor(type))
df<- na.omit(df)
```

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

```r
dim(df)
```

```
[1] 6497   11
```

```
[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
df_stats <- df %>%
  group_by(type) %>%
  summarise(mean = mean(quality), sd = sd(quality), n = length(quality))

diff_mean <- df_stats$mean %>%
  diff()
```

```
sp <- sqrt(sum(df_stats$sd^2 * (df_stats$n-1)) / (sum(df_stats$n - 2)) * (1/nrow(df1) + 1/nr
t1 <-  diff_mean / sp
t1
```

```
[1] 9.684158
```

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`.

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

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

```
typewhite
  9.68565
```

1.6 (5 points)

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

From these values we can conclude the t-statistic is very significant

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

```
                   t typewhite 
 9.684158 -9.685650  9.685650 
```

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?

Based off the p-values of each of the predictors, it seems almost all of them are significant predictors in their own regard. Thus, wine quality is based off a multitude of predictors that can all accurately suggest the quality of wine.

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

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)
```

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

2.3 (5 points)

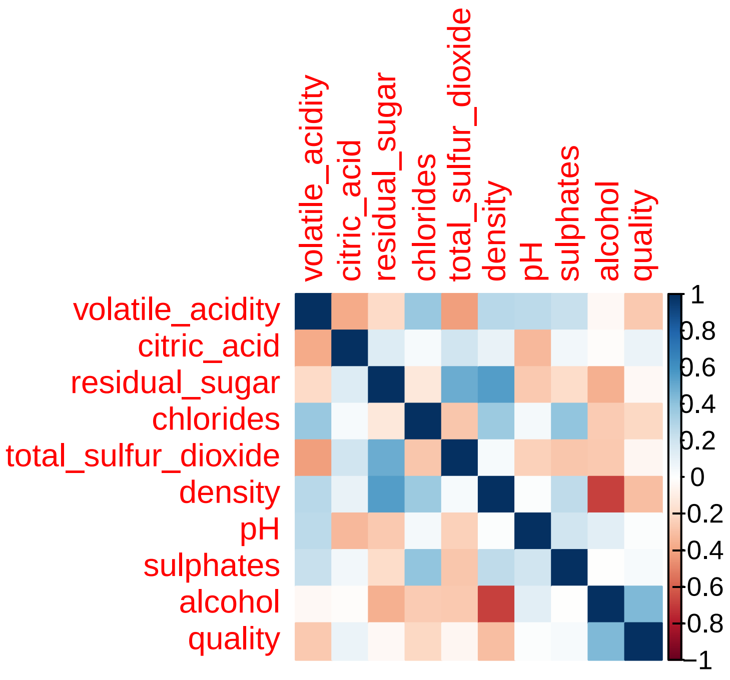
Visualize the correlation matrix of all numeric columns in `df` using `corrplot()`

```r
library(corrplot)
```

```
Warning: package 'corrplot' was built under R version 4.3.2
```

```
corrplot 0.92 loaded
```

```
num_df <- df[sapply(df, is.numeric)]
correlationmatrix <- cor(num_df)
corrplot(correlationmatrix,method = "color")
```



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?

From the VIF of the predictors for the model, we can conclude that most of the predictors have some correlation between each other. Thestrongest case of collinearity is density, but it is not above 10.

```
vif(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
```

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

```r
backwards_model <- step(model, scope = formula(model), direction = "backward")
```

```
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
```

```r
backward_formula <- formula(backwards_model)
```

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, df)
forward_model <- step(null_model, scope = formula(model), direction = "forward")
```

```
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
```

```
forward_model
```

```

Call:
lm(formula = quality ~ alcohol + volatile_acidity + sulphates + 
    residual_sugar + type + density + chlorides + pH, data = df)

Coefficients:
     (Intercept)           alcohol  volatile_acidity         sulphates  
        57.22518           0.28073          -1.62632           0.65234  
  residual_sugar         typewhite           density         chlorides  
         0.04425          -0.41760         -54.87625          -0.95067  
              pH
        0.17589
```

```r
forward_formula <- formula(forward_model)
```

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`.

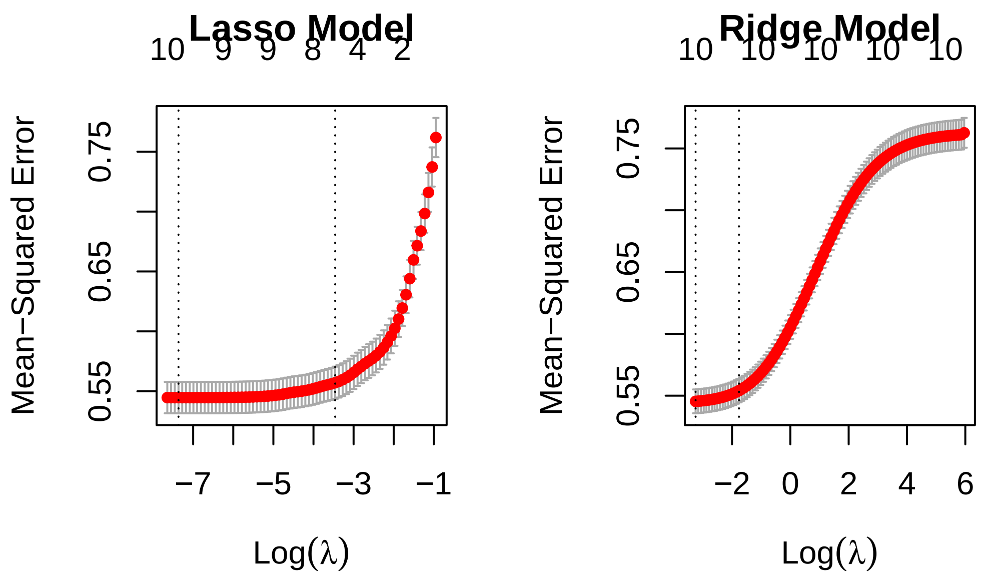
```r
library(glmnet)
y = c(df$quality)
make_model_matrix <- function(formula){
  X <- model.matrix(formula, df)[, -1]
  cnames <- colnames(X)
  for(i in 1:ncol(X)){
    if(!cnames[i] == "typewhite"){
      X[, i] <- scale(X[, i])
    } else {
      colnames(X)[i] <- "type"
    }
  }
  return(X)
}
model_matrix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

lasso_model <- cv.glmnet(model_matrix, y, alpha = 1)
ridge_model <- cv.glmnet(model_matrix, y, alpha = 0)
```

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

For ridge regression, we should be seeing the sum of squared error tending towards a flat line, and we do. We can see it actually tends towards .55. This means the value of lambda may be too high towards the right and middle end of the graph, or the model is suffering from over fitting. Towards the left tail of the graph, the line tends towards .55 which is what we want to see for a good model. For lasso regression we can see the minimum error occcurs *when log(lambda) = -7. The graph is relatively flat which is a good thing. This means our sum of residual error is expected and not random. Overall, both models level out at a stable mean-squared error. The only noteworthy concern is the standard error is rather high.*

```r
par(mfrow=c(1, 2))
plot(lasso_model, main = "Lasso Model")
plot(ridge_model, main = "Ridge Model")
```



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_model$lambda1se
```

```
NULL
```

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

```
11 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)           5.81837771
volatile_acidity     -0.19128674
citric_acid           .
residual_sugar        0.03943232
chlorides             .
total_sulfur_dioxide  .
density               .
pH                    .
sulphates             0.05379620
alcohol               0.36366674
type                  .
```

```r
lasso_formula <- make_formula(rownames(lasso_vars)[-1])
lasso_formula
```

```
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type
<environment: 0x00000223c26b4cf8>
```

3.5 (5 points)

Print the coefficient values for ridge regression at the `lambda.1se` value? What are the variables selected here? *The variables selected are volatile_acidity, citric_acid, residual_sugar, chlorides, total_sulfur_dioxide, density, pH, sulphates, alcohol, and type.*

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_model$lambda1se
```

```
NULL
```

```r
ridge_vars <- coef(ridge_model)
print(ridge_vars)
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)           5.88519724
volatile_acidity     -0.18384619
citric_acid           0.01816935
residual_sugar        0.10718760
chlorides            -0.04540275
total_sulfur_dioxide -0.04032981
density              -0.08770244
pH                    0.02540356
sulphates             0.08442069
alcohol               0.28041795
type                 -0.08863342
```

```r
ridge_formula <- make_formula(rownames(ridge_vars)[-1])
ridge_formula
```

```
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type
<environment: 0x00000223c8b8b998>
```

3.6 (10 points)

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

When we had stepwise regression, the slope of each predictory variable what very high. This is very common in multiple regression, and it is a classic case of overfitting. The LASSO and ridge regression models were able to minimize these slopes to fit the sum of squared error. Thus, allowing the model to be better suited for addition data instead of the df we were provided with.

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. *In the context of choosing all the different potential predictors without replacement and order doesn't matter, we can choose 10 different combinations out of 10 total potential covariates. Thus, the total possible permutations is 10 chose 10.*

```r
total <- sum(choose(10, 0:10))
total
```

```
[1] 1024
```

4.2 (20 points)

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

```r
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.

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

formulas <- formulas[!duplicated(formulas)]
```

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

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

```
[1] "quality ~ volatile_acidity + residual_sugar + density + pH"
[2] "quality ~ volatile_acidity + residual_sugar + total_sulfur_dioxide + pH + alcohol + typ
[3] "quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides + total_sulfur_di
[4] "quality ~ volatile_acidity + citric_acid + chlorides + total_sulfur_dioxide + density +
```

```r
# 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 + de
# [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`.

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

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.

```
get_adj_r_squareds <- function(formula, df){
  model2 <- lm(formula, data = df)
  return(summary(model2)$adj.r.squared)
}

#get list of all the adj.r.squared
adj.r.squared <- sapply(summaries, get_adj_r_squareds)
```

Store resulting formula as a variable called `rsq_formula`.

```
#get formula where max matches
rsq_formula <- formulas[which.max(adj.r.squared)]
rsq_formula
```

```
[[1]]
quality ~ volatile_acidity
<environment: 0x00000223c94eeb78>
```

4.5 (5 points)

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

```
get_AIC_values <- function(formula, df){
  model2 <- lm(formula, data = df)
  return(summary(model2))

}
AIC <- sapply(summaries, function(summary) summary$AIC)
```

Store resulting formula as a variable called `aic_formula`.

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

```
[[1]]
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type
<environment: 0x00000223ca0fde60>
```

4.6 (15 points)

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

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

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

```
[[1]]
quality ~ 1

[[2]]
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type

[[3]]
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type

[[4]]
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type + density + chlorides + pH
```

```
[[5]]
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type
<environment: 0x00000223c26b4cf8>

[[6]]
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type
<environment: 0x00000223c8b8b998>

[[7]]
quality ~ volatile_acidity
<environment: 0x00000223c94eeb78>

[[8]]
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type
<environment: 0x00000223ca0fde60>
```

- 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's were not the same, but they did share some covariates. For example, they both had volatile_acidity as its X1 predictory variable. Compared to the Lasso and Ridge they were not similar. They had far less predictory variables.

- Which of these is more reliable? Why? *The AIC model will be more reliable. The model with the highest R-squared just indicates its well-suited for the training data. This indicates nothing towards the test data, and it may be a symptom of over fitting.*
- If we had a dataset with 10,000 columns, which of these methods would you consider for your analyses? Why? *I would've probably used lasso or ridge regression in this context, because its automates ways to get rid of unecessary variables.*

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,
 \(x) broom::glance(lm(x, data = df)) %>%
    select(sigma, adj.r.squared, AIC, df, p.value)
) %>% bind_rows()

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

sigma	adj.r.squared	AIC	df	p.value
0.8732553	0.0000000	16679.64	NA	NA
0.7370527	0.2876152	14486.26	10	0
0.7370314	0.2876563	14483.89	8	0
0.7370314	0.2876563	14483.89	8	0
0.7370527	0.2876152	14486.26	10	0
0.7370527	0.2876152	14486.26	10	0
0.8419317	0.0704531	16205.99	1	0
0.7370314	0.2876563	14483.89	8	0

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 4.3.1 (2023-06-16 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 11 x64 (build 22621)

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

time zone: America/New_York
tzcode source: internal

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

other attached packages:
[1] corrplot_0.92 glmnet_4.1-8  Matrix_1.6-5  car_3.1-2     carData_3.0-5
[6] purrr_1.0.2   dplyr_1.1.2   tidyr_1.3.0   readr_2.1.4

loaded via a namespace (and not attached):
 [1] jsonlite_1.8.7    compiler_4.3.1    Rcpp_1.0.11       tidyselect_1.2.0
 [5] splines_4.3.1     yaml_2.3.7        fastmap_1.1.1     lattice_0.21-8
 [9] R6_2.5.1          generics_0.1.3    shape_1.4.6       knitr_1.43
[13] backports_1.4.1   iterators_1.0.14  tibble_3.2.1      pillar_1.9.0
[17] tzdb_0.4.0        rlang_1.1.1       utf8_1.2.3        broom_1.0.5
[21] xfun_0.40         cli_3.6.1         withr_2.5.0       magrittr_2.0.3
[25] digest_0.6.33     foreach_1.5.2     grid_4.3.1        rstudioapi_0.15.0
[29] hms_1.1.3         lifecycle_1.0.3   vctrs_0.6.3       evaluate_0.21
[33] glue_1.6.2        codetools_0.2-19  survival_3.5-5    abind_1.4-5
[37] fansi_1.0.4       rmarkdown_2.24    tools_4.3.1       pkgconfig_2.0.3
[41] htmltools_0.5.6
```