Homework 3

Kiran Charangat {style='background-color: yellow;'}

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:

```r
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)
```

Loading required package: carData

Attaching package: 'car'

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

    some

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

    recode

```r
library(glmnet)
```

Loading required package: Matrix

Attaching package: 'Matrix'

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

    expand, pack, unpack

Loaded glmnet 4.1-8

```
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"))
```

```
quality ~ a + b + c
<environment: 0x000002c7c4f83800>
```

```
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)
}
```

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_delim(url1, delim = ";", show_col_types = FALSE)
df2 <- read_delim(url2, delim = ";", show_col_types = FALSE)


colnames(df1)
```

```
 [1] "fixed acidity"       "volatile acidity"    "citric acid"
 [4] "residual sugar"      "chlorides"           "free sulfur dioxide"
 [7] "total sulfur dioxide" "density"            "pH"
[10] "sulphates"           "alcohol"             "quality"
```

```
head(df1)
```

```
# A tibble: 6 x 12
  `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides
          <dbl>              <dbl>         <dbl>            <dbl>     <dbl>
1             7               0.27          0.36             20.7     0.045
2           6.3               0.3           0.34              1.6     0.049
3           8.1               0.28          0.4               6.9     0.05
4           7.2               0.23          0.32              8.5     0.058
5           7.2               0.23          0.32              8.5     0.058
6           8.1               0.28          0.4               6.9     0.05
# i 7 more variables: `free sulfur dioxide` <dbl>,
#   `total sulfur dioxide` <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
#   alcohol <dbl>, quality <dbl>
```

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

4. Convert the **type** column to a factor
5. Remove rows (if any) with missing values.

```r
df <- bind_rows(df1 %>% mutate(type = "white"),
                df2 %>% mutate(type = "red")) %>%
    mutate(type = as.factor(type)) %>%
    drop_na() %>%
    rename_with(~ gsub(" ", "_", .x)) %>%
    select(-fixed_acidity, -free_sulfur_dioxide)
```

```r
colnames(df)
```

```
 [1] "volatile_acidity"    "citric_acid"          "residual_sugar"
 [4] "chlorides"           "total_sulfur_dioxide" "density"
 [7] "pH"                  "sulphates"            "alcohol"
[10] "quality"             "type"
```

```r
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**.

5

```
diff_mean <-df %>%
            group_by(type) %>%
            summarise(mean_quality = mean(quality)) %>%
            summarise(diff_mean = diff(mean_quality)) %>%
            pull(diff_mean)
diff_mean
```

[1] 0.2418868

```
sp_squared <- df %>%
            group_by(type) %>%
            summarise(n = n(),
                      var_quality = var(quality)) %>%
            summarise(sp_squared = ((n[1] - 1) * var_quality[1] + (n[2] - 1) *
            pull(sp_squared)

sp_squared
```

[1] 0.7518329

```
t1 <-   df %>%
        summarise(t1 = diff_mean / sqrt(sp_squared * (1/sum(df$type == "white") + 1
        pull(t1)

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)
t_test
```

```
    Two Sample t-test

data:  quality by type
t = -9.6856, df = 6495, p-value < 2.2e-16
alternative hypothesis: true difference in means between group red and group white is not eq
95 percent confidence interval:
 -0.2908436 -0.1929301
sample estimates:
  mean in group red mean in group white
          5.636023            5.877909
```

```r
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)
fit
```

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

Coefficients:
(Intercept)     typewhite
     5.6360        0.2419
```

```r
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
t_statistics <- c(t1, t2, t3)
t_statistics
```

```
            t
9.68565 -9.68565  9.68565
```

```r
# These t-statitistic values indicate a significant difference in wine quality
# between red and white wine. The negative sign in the second value (t2) suggests
# a reversal in direction of the effect. These results, however, show that the
# wine type significantly impacts quality, but it is important to understand the
# direction of the effect.
```

—

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?

```
broom::tidy(fit)
```

```
# A tibble: 2 x 5
  term         estimate std.error statistic  p.value
  <chr>           <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)     5.64    0.0217    260.    0
2 typewhite       0.242   0.0250      9.69 4.89e-22
```

```
# The coefficient for 'typewhite' indicates the estimated change in quality score when com
```

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)
```

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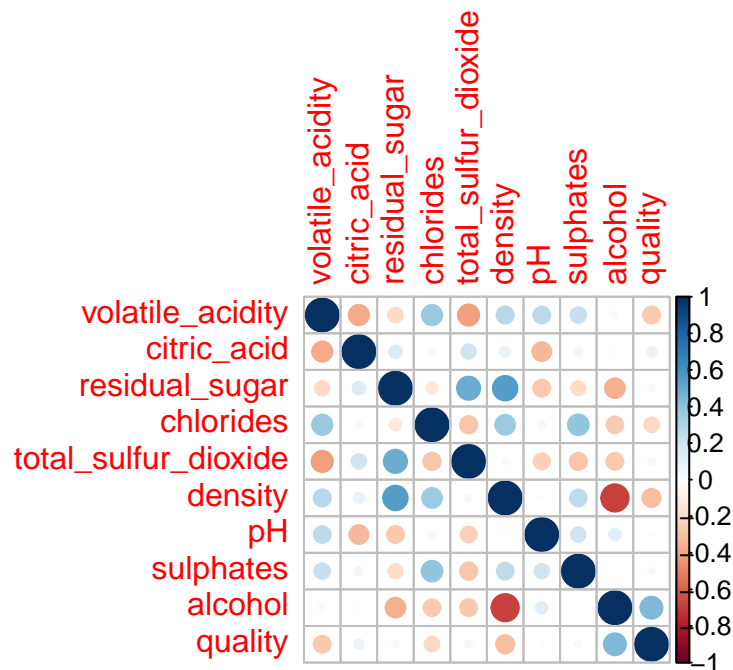

2.3 (5 points)

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

```
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
df %>%
  select_if(is.numeric) %>%
  cor() %>%
  corrplot(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?

```r
vif(lm(quality ~ ., data = df))
```

```
  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
```

```
# The predictor density has a VIF of over 9, and type has a VIF of close to 7, which sugge
```

10

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_model <- step(full_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
```

11

```
<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(backward_model)
  backward_formula
```

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

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)
  forward_model <- 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
```

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

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


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`.

```
y <- df$quality
X <- make_model_matrix(formula(full_model))
lasso_model <- cv.glmnet(X, y, alpha = 1)
ridge_model <- cv.glmnet(X, y, alpha = 0)
```

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

```
par(mfrow=c(1, 2))
plot(lasso_model, main="LASSO Regression")
plot(ridge_model, main="Ridge Regression")
```
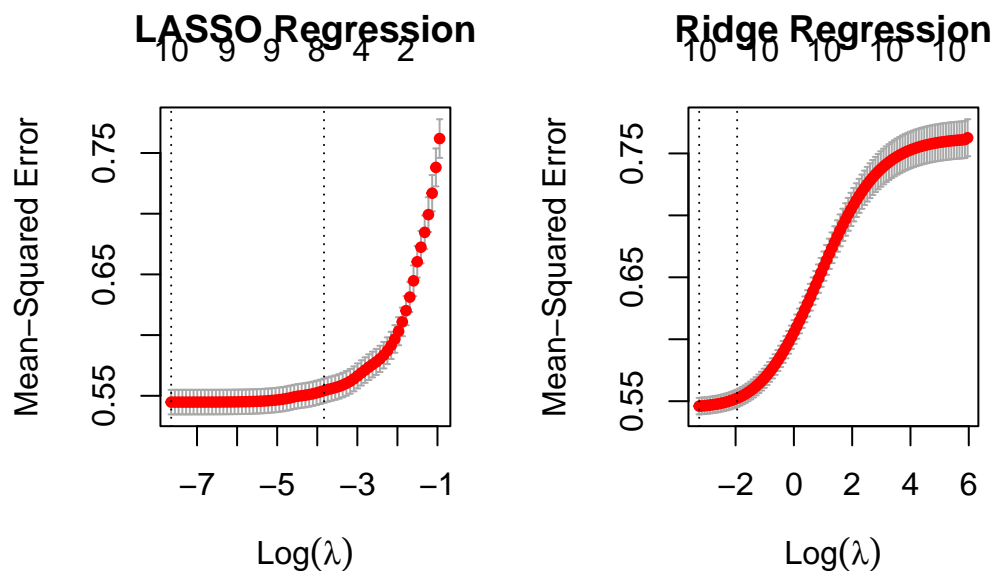


```
# From these models, it be concluded that both models benefit from regularization, as indi
```

3.4 (5 points)

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

```
lasso_coef_at_1se <- coef(lasso_model, s = "lambda.1se")
lasso_coef_at_1se
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)          5.822812544
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
```

```
# The variables selected by LASSO are volatile_acidity with a coefficient of approximately
```

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.

```
lasso_coef_vector <- as.numeric(lasso_coef_at_1se)
names(lasso_coef_vector) <- rownames(lasso_coef_at_1se)
lasso_vars <- names(lasso_coef_vector)[-1][lasso_coef_vector[-1] != 0]
lasso_formula <- make_formula(lasso_vars)
lasso_formula
```

```
quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide +
    pH + sulphates + alcohol + type
<environment: 0x000002c7ccb2bd20>
```

3.5 (5 points)

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

```
ridge_coef_at_1se <- coef(ridge_model, s = "lambda.1se")
ridge_coef_at_1se
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                             s1
(Intercept)          5.89912780
volatile_acidity    -0.19275639
citric_acid          0.01625926
residual_sugar       0.11607371
chlorides           -0.04421587
total_sulfur_dioxide -0.04011160
density             -0.09019465
pH                   0.02601554
sulphates            0.08689460
alcohol              0.28992182
type                -0.10711174
```

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_vars <- rownames(ridge_coef_at_1se)[-1]
ridge_formula <- make_formula(ridge_vars)
print(ridge_formula)
```

```
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type
<environment: 0x000002c7ce6e8e28>
```

3.6 (10 points)

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

```
# The stepwise selection would have given me a subset of predictors based on a criterion l
# The LASSO regression results showed which variables were most influential by keeping the
# The Ridge regression results would have included all predictors but with their influence
```

—

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 - 1 which is 1023 different models. Each predictor can either be incl
```

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))

```
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),
  \(x){
    vars <- combn(x_vars, x, simplify = FALSE)
    map(vars, \(combo) make_formula(combo))
  }
) %>% unlist(recursive = FALSE)
```

19

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

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

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

```
sample(formulas, 4) %>% as.character()
# 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(.x, data = df))
summaries <- map(models, ~broom::glance(.x))
all_summaries <- bind_rows(summaries)
all_summaries
```

```
# A tibble: 1,023 x 12
   r.squared adj.r.squared sigma statistic   p.value    df logLik    AIC    BIC
       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
 1   0.0706        0.0705  0.842     493.   2.06e-105     1 -8100. 16206. 16226.
 2   0.00732       0.00716 0.870      47.9  5.00e- 12     1 -8314. 16634. 16654.
 3   0.00137       0.00121 0.873       8.89 2.87e-  3     1 -8333. 16673. 16693.
 4   0.0403        0.0401  0.856     273.   5.32e- 60     1 -8204. 16415. 16435.
 5   0.00171       0.00156 0.873      11.1  8.48e-  4     1 -8332. 16671. 16691.
 6   0.0935        0.0934  0.831     670.   9.66e-141     1 -8019. 16044. 16064.
 7   0.000380      0.000227 0.873      2.47 1.16e-  1     1 -8337. 16679. 16700.
 8   0.00148       0.00133 0.873       9.63 1.92e-  3     1 -8333. 16672. 16692.
```

```
 9  0.197          0.197    0.782    1598.    1.50e-312      1 -7623. 15253. 15273.
10  0.0142         0.0141   0.867      93.8  4.89e- 22      1 -8291. 16588. 16609.
# i 1,013 more rows
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

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.

```
best_model_index <- which.max(all_summaries$adj.r.squared)
rsq_formula <- formulas[[best_model_index]]
rsq_formula
```

```
quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide +
    density + pH + sulphates + alcohol + type
<environment: 0x000002c7cfbf2810>
```

Store resulting formula as a variable called `rsq_formula`.

4.5 (5 points)

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

```
lowest_aic_index <- which.min(all_summaries$AIC)
aic_formula <- formulas[[lowest_aic_index]]
aic_formula
```

```
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type
<environment: 0x000002c7cfbb89c8>
```

Store resulting formula as a variable called `aic_formula`.

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 almost similar as the rsq_formula includes total_sulfur_dioxide and aic_formula does not. As for how they differ from the formulas shortlisted in question 3, The aic_formula and rsq_formula represent models selected based on optimizing specific statistical criteria. The formulas from question 3 however, like the ridge_formula and lasso_formula apply regularization techniques and the forward and backward formulas use step wise procedures.

- Which of these is more reliable? Why?

  – For prediction purposes and generalization, aic_formula and lasso_formula might be good options. For explanatory purposes, rsq_formula might be good like when explaining the variance in response is the primary goal.

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

  – I would choose the LASSO and Ridge regression for this analysis due to its effectiveness in handling multi-collinearity and its ability to prevent over fitting through regularization.

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) {
    model <- lm(x, data = df)
    broom::glance(model) %>%
      select(sigma, adj.r.squared, AIC, df = df.residual, p.value = p.value)
  }
) %>% bind_rows()

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

sigma	adj.r.squared	AIC	df	p.value
0.8732553	0.0000000	16679.64	6496	NA
0.7370527	0.2876152	14486.26	6486	0
0.7370314	0.2876563	14483.89	6488	0
0.7370314	0.2876563	14483.89	6488	0
0.7391172	0.2836187	14520.61	6488	0
0.7370527	0.2876152	14486.26	6486	0
0.7370027	0.2877118	14484.38	6487	0
0.7370314	0.2876563	14483.89	6488	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.

```r
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"))
```

```
quality ~ a + b + c
<environment: 0x000002c7889442c0>
```

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()`

```r
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 4.3.2 (2023-10-31 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_1.6-1.1 car_3.1-2      carData_3.0-5
> [6] purrr_1.0.2    dplyr_1.1.4    tidyr_1.3.0    readr_2.1.5
>
> loaded via a namespace (and not attached):
>  [1] utf8_1.2.4        generics_0.1.3    shape_1.4.6       lattice_0.21-9
>  [5] hms_1.1.3         digest_0.6.33     magrittr_2.0.3    evaluate_0.23
>  [9] grid_4.3.2        iterators_1.0.14  fastmap_1.1.1     foreach_1.5.2
> [13] jsonlite_1.8.7    backports_1.4.1   survival_3.5-7    fansi_1.0.5
> [17] codetools_0.2-19  abind_1.4-5       cli_3.6.1         rlang_1.1.2
> [21] crayon_1.5.2      bit64_4.0.5       splines_4.3.2     withr_2.5.2
> [25] yaml_2.3.7        tools_4.3.2       parallel_4.3.2    tzdb_0.4.0
> [29] broom_1.0.5       curl_5.1.0        vctrs_0.6.4       R6_2.5.1
> [33] lifecycle_1.0.4   bit_4.0.5         vroom_1.6.4       pkgconfig_2.0.3
> [37] pillar_1.9.0      glue_1.6.2        Rcpp_1.0.11       xfun_0.41
> [41] tibble_3.2.1      tidyselect_1.2.0  rstudioapi_0.15.0 knitr_1.45
> ```

```
[45] htmltools_0.5.7    rmarkdown_2.25    compiler_4.3.2
```