title: “Homework 3” author: “Arjun Laxman” toc: true title-block-banner: true title-block-style: default format: pdf # format: pdf

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
options(repos = c(CRAN = "https://cloud.r-project.org"))

install.packages("car")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

```r
install.packages("corrplot")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

```r
install.packages("tidyverse")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

```r
install.packages("dplyr")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

```r
install.packages("tidyverse")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

```r
install.packages("glmnet")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

```r
install.packages("curl")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

```r
library(MASS)
install.packages("glmnet")
```

```

The downloaded binary packages are in
	/var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

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

```
corrplot 0.92 loaded
```

```r
library(dplyr)
```

```

Attaching package: 'dplyr'

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

    select

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
```

```r
install.packages("curl")
```

```

The downloaded binary packages are in
    /var/folders/kr/zyr76x5136x61sbwmb2_w_qr0000gn/T//RtmpW3LrSI/downloaded_packages
```

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

# Store the formula in a variable
my_formula <- make_formula(c("a", "b", "c"))

# Print the formula
print(my_formula)
```

```
quality ~ a + b + c
```

```
<environment: 0x11db30710>
```

```
# generate a model matrix 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)
}
```

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

df1 <- read_delim(url1, delim=";")
```

```
Rows: 4898 Columns: 12
── Column specification ────────────────────────────────────────────────────────
Delimiter: ";"
dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df2 <- read_delim(url2, delim=";")
```

```
Rows: 1599 Columns: 12
── Column specification ────────────────────────────────────────────────────────
Delimiter: ";"
dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
```

```
ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

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
library(dplyr)
library(stringr)  # Ensure this library is loaded

df <- bind_rows(
  df1 %>% mutate(type = "white"),
  df2 %>% mutate(type = "red")
) %>%
  rename_all(~str_replace_all(., " ", "_")) %>%
  select(-c(fixed_acidity, free_sulfur_dioxide)) %>%
  mutate(type = as.factor(type)) %>%
  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
df_summary <- df %>%
  group_by(type) %>%
  summarize(
    mean = mean(quality),
    sd = sd(quality),
    n = length(quality)
  )
```

```r
diff_mean <- df_summary$mean %>% diff()
sp <- sum(
  df_summary$sd^2 * (df_summary$n - 1)
) / sum(df_summary$n - 2)

t1 <- diff_mean / (sqrt(sp) * sqrt(1/nrow(df1) + 1/nrow(df2)))
```

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(
  df %>% filter(type == "white") %>% select(quality),
  df %>% filter(type == "red") %>% select(quality),
  var.equal=TRUE
)
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, df)
t3 <- coef(summary(fit))[, "t value"][2]
```

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 typewhite
9.684158  9.685650  9.685650
```

The similarity in the values of t2 and t3, and the very close value of t1, suggest that regardless of the slight methodological differences, the statistical evidence pointing towards a certain hypothesis (such as the significance of a predictor in a regression model) is robust.

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
library(broom)
# Fit the model
full_model <- lm(quality ~ ., data = df)
tidy_summary <- tidy(full_model)
# Print the summary using broom::tidy()
print(tidy_summary)
```

```
# A tibble: 11 × 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 with citric_acid as the predictor
model_citric <- lm(quality ~ citric_acid, 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 with total_sulfur_dioxide as the predictor
model_sulfur <- lm(quality ~ total_sulfur_dioxide, 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
```

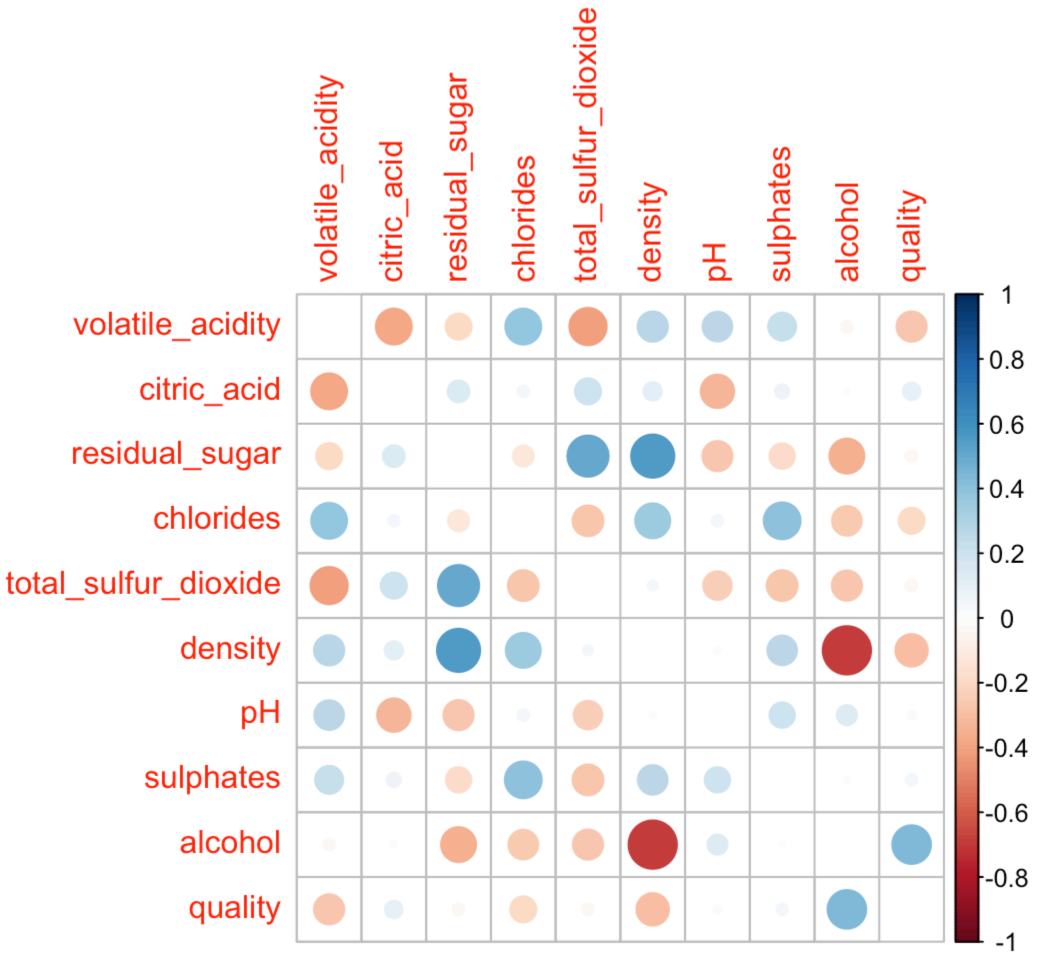
```r
model_sulfur <- ... # Insert your code here
```

2.3 (5 points)

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

```r
library(dplyr)       #
library(corrplot)

df %>%
  select(where(is.numeric)) %>%  # Select only numeric columns
  cor() %>%                      # Compute the correlation matrix
  round(digits=2) %>%
  corrplot(diag=F)               # Generate the correlation plot
```



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?

```
library(car)
vif(full_model) %>% knitr::kable()
```

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

The VIF for residual_sugar, density and type are notably large. This shows the multi-collinearity in full_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

```r
library(MASS)
null_model <- lm(quality ~ 1, data = df)
full_model <- lm(quality ~ ., data = df)
backward_model <- stepAIC(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
<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)
```

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
forward_model <- stepAIC(null_model, scope = list(lower = null_model, upper = 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
```

```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.

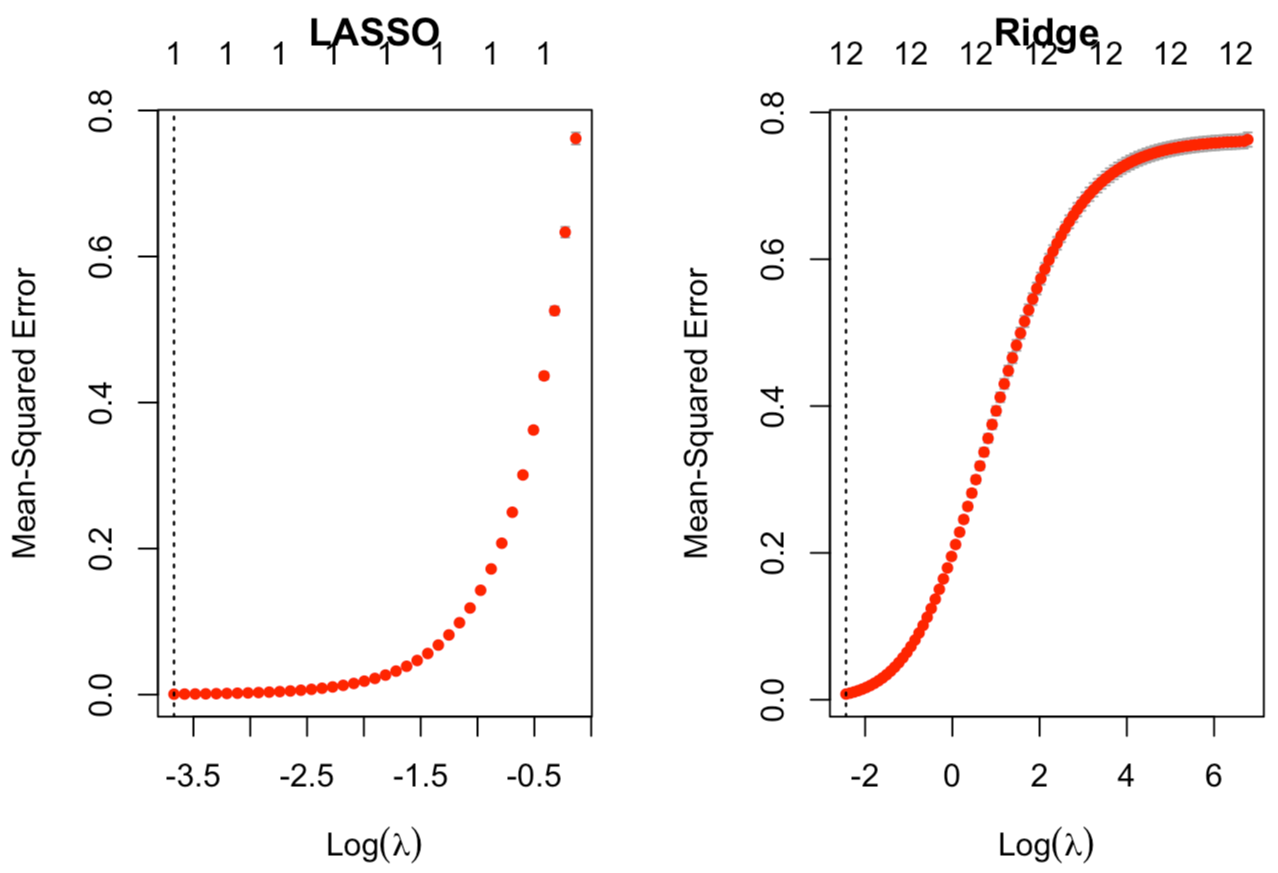
3. Then, use the `cv.glmnet()` function to perform LASSO and Ridge regression with `X` and `y`.

```r
library(glmnet)
y <- df$quality
X <- model.matrix(~ . -1, data = df) # Assuming 'df' is your dataset with 'quality' as on

lasso_model <- cv.glmnet(X, y, alpha = 1) # LASSO
ridge_model <- cv.glmnet(X, y, alpha = 0) # Ridge
```

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

```r
par(mfrow = c(1, 2)) # Setting up the plot area for two side-by-side plots
plot(lasso_model, main = "LASSO")
plot(ridge_model, main = "Ridge")
```



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(lasso_model, s = "lambda.1se")
lasso_vars <- rownames(lasso_coef)[lasso_coef[,1] != 0]
lasso_formula <- make_formula(lasso_vars[-1]) # Excluding the intercept
```

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_coef <- coef(ridge_model, s = "lambda.1se")
ridge_vars <- rownames(ridge_coef)[ridge_coef[,1] != 0]
ridge_formula <- make_formula(ridge_vars[-1]) # Excluding the intercept
```

3.6 (10 points)

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

LASSO tends to select a smaller number of variables compared to Ridge regression

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 = 1023 different models. This is because each predictor can either be included or not included in a model,

From those 2^10 combinations. We have to subtract 1 because we exclude the empty model the one with no predictors.

4.2 (20 points)

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

```r
# Install dplyr if it's not already installed
if (!require(dplyr)) install.packages("dplyr")
```

```
# Load dplyr, which will also load magrittr allowing you to use the pipe operator '%>%'
library(dplyr)

x_vars <- colnames(df %>% select(-quality))
if (!require(purrr)) install.packages("purrr")
library(purrr)
```

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, make_formula)
  }
) %>% unlist()
```

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

```
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, \(f) lm(f, data = df))
summaries <- map(models, broom::glance) %>% bind_rows()
```

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_rsq <- summaries %>% filter(adj.r.squared == max(adj.r.squared))
```

Store resulting formula as a variable called `rsq_formula`.

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

4.5 (5 points)

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

```
best_aic <- summaries %>% filter(AIC == min(AIC))
```

Store resulting formula as a variable called `aic_formula`.

4.6 (15 points)

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

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

```
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.
```

~~~
::: {.cell}

```{.r .cell-code}
print(colnames(df))

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

```
print(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 ~ quality
<environment: 0x12d1f55d8>

[[6]]
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    quality + typered + typewhite
<environment: 0x12d4e6db8>

[[7]]
quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide +
    density + pH + sulphates + alcohol + type
<environment: 0x11f7720a8>

[[8]]
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type
<environment: 0x11f7a44a8>
```

```
library(broom)  # Ensure broom is loaded for using glance()

# Modify the map function to include error handling
summary_table <- map(
  final_formulas,
  \(x) {
    tryCatch({
      model <- lm(x, data = df)
      glance(model) %>%
        select(sigma, adj.r.squared, AIC, df.residual, p.value)  # Select the required st
    }, error = function(e) {
      message("Error in formula: ", deparse(x), "\nError Message: ", e$message)
      return(data.frame(sigma = NA, adj.r.squared = NA, AIC = NA, df.residual = NA, p.val
    })
  }
) %>%
bind_rows()  # Combine all model summaries into one data frame
```

```
Warning in model.matrix.default(mt, mf, contrasts): the response appeared on
the right-hand side and was dropped

Warning in model.matrix.default(mt, mf, contrasts): problem with term 1 in
model.matrix: no columns are assigned

Error in formula: quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
total_sulfur_dioxide + density + pH + sulphates + alcohol +     quality + typered +
typewhite
Error Message: object 'typered' not found
```

```
# Adding names to the summary table for clarity
summary_table <- cbind(Formula = c("Null Formula", "Full Formula", "Backward Formula", "F
                                   "Ridge Formula", "Rsq Formula", "Aic Formula"), summar

# Use knitr to create a better display table
summary_table %>% knitr::kable()
```

Formula	sigma	adj.r.squared	AIC	df.residual	p.value
Null Formula	0.8732553	0.0000000	16679.64	6496	NA
Full Formula	0.7370527	0.2876152	14486.26	6486	0
Backward Formula	0.7370314	0.2876563	14483.89	6488	0
Forward Formula	0.7370314	0.2876563	14483.89	6488	0
Lasso Formula	0.8732553	0.0000000	16679.64	6496	NA
Ridge Formula	NA	NA	NA	NA	NA

Formula	sigma	adj.r.squared	AIC	df.residual	p.value
Rsq Formula	0.7370027	0.2877118	14484.38	6487	0
Aic Formula	0.7370314	0.2876563	14483.89	6488	0

:::

~~~
:::{.hidden unless-format="pdf"}
\pagebreak
:::

<br><br><br><br>
<br><br><br><br>
---


# 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.

::: {.cell}

```{.r .cell-code}
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()`

::: {.cell}

```{.r .cell-code}
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 4.3.3 (2024-02-29)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.4.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;
LAPACK version 3.11.0

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

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

```
other attached packages:
 [1] broom_1.0.5     stringr_1.5.1   glmnet_4.1-8    Matrix_1.6-5
 [5] car_3.1-2       carData_3.0-5   purrr_1.0.2     dplyr_1.1.4
 [9] corrplot_0.92   tidyr_1.3.1     readr_2.1.5     MASS_7.3-60.0.1

loaded via a namespace (and not attached):
 [1] utf8_1.2.4        generics_0.1.3    shape_1.4.6.1     stringi_1.8.3
 [5] lattice_0.22-5    hms_1.1.3         digest_0.6.35     magrittr_2.0.3
 [9] evaluate_0.23     grid_4.3.3        iterators_1.0.14  fastmap_1.1.1
[13] foreach_1.5.2     jsonlite_1.8.8    backports_1.4.1   survival_3.5-8
[17] fansi_1.0.6       codetools_0.2-19  abind_1.4-5       cli_3.6.2
[21] rlang_1.1.3       crayon_1.5.2      bit64_4.0.5       splines_4.3.3
[25] withr_3.0.0       tools_4.3.3       parallel_4.3.3    tzdb_0.4.0
[29] curl_5.2.1        vctrs_0.6.5       R6_2.5.1          lifecycle_1.0.4
[33] bit_4.0.5         vroom_1.6.5       pkgconfig_2.0.3   pillar_1.9.0
[37] glue_1.7.0        Rcpp_1.0.12       xfun_0.43         tibble_3.2.1
[41] tidyselect_1.2.1  rstudioapi_0.16.0 knitr_1.45        htmltools_0.5.8.1
[45] rmarkdown_2.26    compiler_4.3.3
```