

Homework 3

Insert your name here

Table of contents

Question 1	3
Question 2	7
Question 3	11
Print coefficient values	16
Identify variables with non-zero coefficients	16
Create formula object using make_formula() function	16
Print Ridge formula	16
Appendix	21

! 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

```
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

```
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

```
library(broom)
library(corrplot)
```

corrplot 0.92 loaded

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/winequality09.csv"
url2 <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality08.csv"

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

1.2 (5 points)

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

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

```
#1
df1$type <- "white"
df2$type <- "red"
df <- rbind(df1, df2)

#2
colnames(df) <- gsub("\\.", "_", colnames(df))

#3
df <- df %>%
  select(-fixed_acidity, -free_sulfur_dioxide)

#4
df$type <- as.factor(df$type)

#5
nrow(df)
```

```
[1] 6497
```

```
df <- na.omit(df)
dim(df) #none were removed
```

```
[1] 6497  11
```

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

```
[1] 6497  11
```

```
head(df)
```

	volatile_acidity	citric_acid	residual_sugar	chlorides	total_sulfur_dioxide	
1	0.27	0.36	20.7	0.045		170
2	0.30	0.34	1.6	0.049		132
3	0.28	0.40	6.9	0.050		97
4	0.23	0.32	8.5	0.058		186
5	0.23	0.32	8.5	0.058		186
6	0.28	0.40	6.9	0.050		97
	density	pH	sulphates	alcohol	quality	type
1	1.0010	3.00	0.45	8.8	6	white
2	0.9940	3.30	0.49	9.5	6	white
3	0.9951	3.26	0.44	10.1	6	white
4	0.9956	3.19	0.40	9.9	6	white
5	0.9956	3.19	0.40	9.9	6	white
6	0.9951	3.26	0.44	10.1	6	white

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`.

```
# 1
mean_red <- mean(df$quality[df$type == "red"])
mean_white <- mean(df$quality[df$type == "white"])
diff_mean <- mean_red - mean_white

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

# 3
```

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

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
```

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)

summary_coef <- summary(fit)$coefficients
t_stat <- summary_coef["typewhite", "t value"]
t3 <- t_stat
```

1.6 (5 points)

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

```
c(t1, t2, t3) # Insert your code here
```

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

Since t1 and t2 are positive while t3 is negative we see that there is a significant difference in mean quality between red and white wines.

Also, since the magnitude of each is the same we know that Welch's t-test and the two-sample t-test yield similar results.

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?

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

summary(lm_model)
```

Call:

```
lm(formula = quality ~ ., data = df)
```

Residuals:

Min	1Q	Median	3Q	Max
-3.3415	-0.4725	-0.0405	0.4573	3.1140

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	5.753e+01	9.331e+00	6.166	7.44e-10	***
volatile_acidity	-1.609e+00	8.057e-02	-19.965	< 2e-16	***
citric_acid	2.721e-02	7.833e-02	0.347	0.72827	
residual_sugar	4.509e-02	4.158e-03	10.844	< 2e-16	***
chlorides	-9.639e-01	3.328e-01	-2.897	0.00378	**

```
total_sulfur_dioxide -3.289e-04 2.623e-04 -1.254 0.20995
density -5.520e+01 9.320e+00 -5.922 3.34e-09 ***
pH 1.885e-01 6.613e-02 2.850 0.00438 **
sulphates 6.620e-01 7.584e-02 8.730 < 2e-16 ***
alcohol 2.767e-01 1.418e-02 19.514 < 2e-16 ***
typewhite -3.858e-01 5.493e-02 -7.023 2.39e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 0.7371 on 6486 degrees of freedom
Multiple R-squared: 0.2887, Adjusted R-squared: 0.2876
F-statistic: 263.3 on 10 and 6486 DF, p-value: < 2.2e-16

```
tidy(lm_model)
```

```
# A tibble: 11 x 5
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <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

I found that volatile acidity, residual sugar, chlorides, density, sulphates, alcohol, and the type of wine are significant predictors of wine quality. On the other hand, variables like citric acid and total sulfur dioxide do not seem to have a significant effect on quality.

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)

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

```
summary(model_sulfur)
```

Call:

```
lm(formula = quality ~ total_sulfur_dioxide, data = df)
```

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	5.8923848	0.0246717	238.831	< 2e-16 ***
total_sulfur_dioxide	-0.0006394	0.0001915	-3.338	0.000848 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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

The results are very comparable to the previous one. They show that, variables like citric acid and total sulfur dioxide do not seem to have a significant effect on quality.

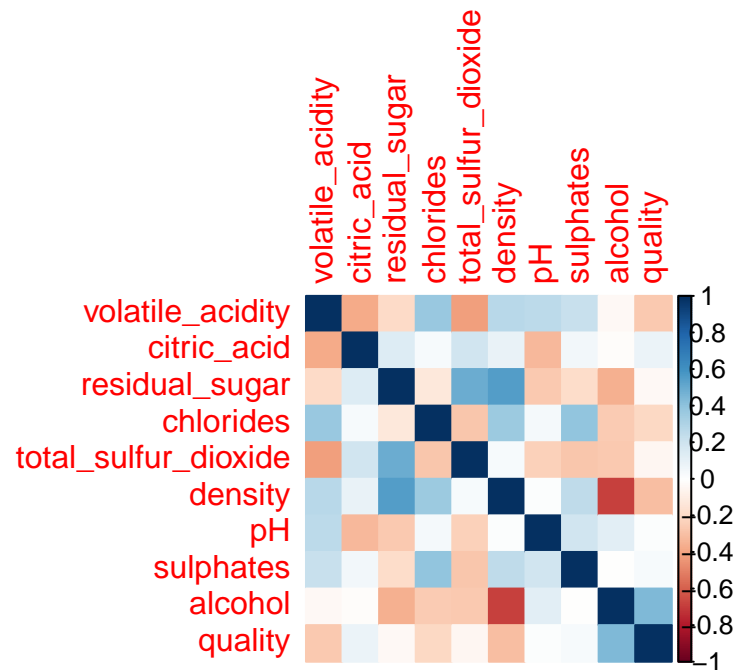
2.3 (5 points)

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

```
numeric_df <- df %>% select_if(is.numeric)

correlation_matrix <- cor(numeric_df)

corrplot(correlation_matrix, 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?


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

vif_scores <- vif(lm_model)
vif_scores
```

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 VIF scores indicate that there is potential multicollinearity in density and type, where VIF values exceeded the common threshold of 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
<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

```
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

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

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

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

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

```
forward_formula
```

```
quality ~ 1
```

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`.

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

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

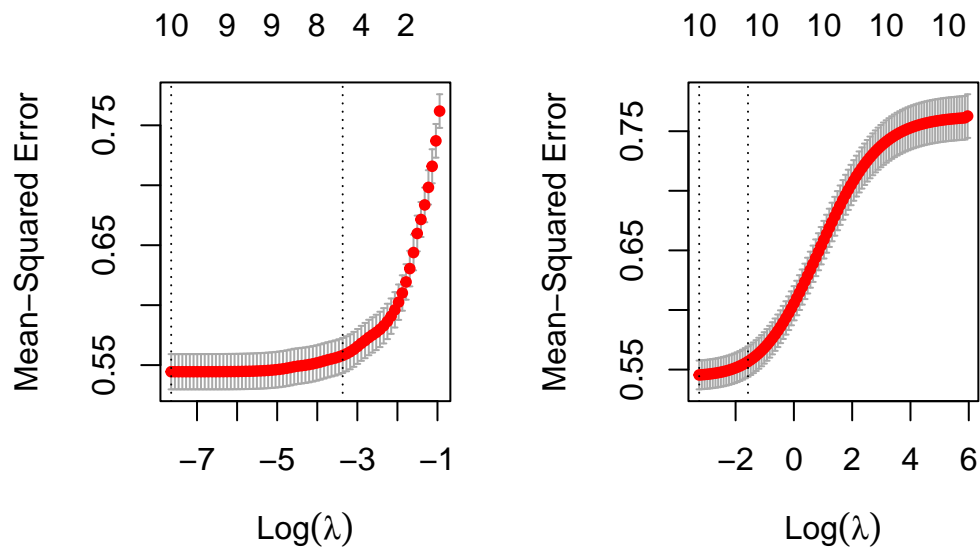
```
# 1
y <- df$quality

# 2
X <- as.matrix(model.matrix(full_model))

# 3
lasso_model <- cv.glmnet(X, y, alpha = 1)

ridge_model <- cv.glmnet(X, y, alpha = 0)

par(mfrow=c(1, 2))
plot(lasso_model)
plot(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.

```
# Extract coefficients at lambda.1se for LASSO model
lasso_coef <- coef(lasso_model, s = "lambda.1se")

# Print coefficient values
print(lasso_coef)

# Identify variables with non-zero coefficients
lasso_vars <- names(lasso_coef)[lasso_coef != 0]

# Remove the response variable "quality"
lasso_vars <- lasso_vars[!lasso_vars %in% "quality"]

# Create the LASSO formula
```

```
lasso_formula <- as.formula(paste("quality ~", paste(lasso_vars, collapse = " + ")))

# Print LASSO formula
print(lasso_formula)
```

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# Extract coefficients at `lambda.1se` for Ridge model `ridge_coef <- coef(ridge_model, s = "lambda.1se")`”

Print coefficient values

```
print(ridge_coef)
```

Identify variables with non-zero coefficients

```
ridge_vars <- names(ridge_coef)[ridge_coef != 0]
```

Create formula object using `make_formula()` function

```
ridge_formula <- make_formula(ridge_vars)
```

Print Ridge formula

```
print(ridge_formula)
```

3.6 (10 points)

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

Question 4
::: {.callout-tip}
70 points

Variable selection
:::

4.1 (5 points)

Excluding `quality` from `df` we have 10 possible predictors as the covariates. How many different combinations of predictors are possible?

::: {.cell}

:::

4.2 (20 points)

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

```
::: {.cell}

```{r .cell-code}
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)
 map(vars, \(var_set) make_formula(var_set))
 }
) %>% 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, ~lm(., data = df))
summaries <- map(models, glance)
```

```
summaries <- bind_rows(summaries, .id = "model")
head(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.

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

best_model_index <- which.max(adj_r_squared)
```

Store resulting formula as a variable called `rsq_formula`.

```
rsq_formula <- formulas[[best_model_index]]
```

---

4.5 (5 points)

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

```
aic_values <- summaries$AIC

best_model_index <- which.min(aic_values)
```

Store resulting formula as a variable called `aic_formula`.

```
aic_formula <- formulas[[best_model_index]]
print(aic_formula)
```

---

4.6 (15 points)

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

```

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

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

```

- Are `aic_formula` and `rsq_formula` the same? How do they differ from the formulas shortlisted in question 3?
- Which of these is more reliable? Why?
- If we had a dataset with 10,000 columns, which of these methods would you consider for your analyses? Why?

---

#### 4.7 (10 points)

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

```

summary_table <- map(
 final_formulas,
 \(formula) {
 model <- lm(formula, data = df)
 glance(model)
 }
) %>%
bind_rows()

summary_table %>% knitr::kable()

```

---

## Appendix

### Convenience function for creating a formula object

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

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

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

### Convenience function for glmnet

The **make\_model\_matrix** function below takes a **formula** as input and outputs a **rescaled** model matrix **X** in a format amenable for **glmnet()**

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

### Session Information

Print your R session information using the following command

```
sessionInfo()
```