DSPG Practical Exam

Putri Khalilah binti Kamaluddin

2023-08-16

1. Exploring the diamonds dataset

Load the package

```
pacman::p_load(
  tidyverse, tidymodels, here,
  vip, janitor, visdat, naniar,
  corrplot.ggplot2,dplyr,yardstick
)
```

Load and read the data

```
data <-
  read_rds("C:\\Users\\PUTRI KHALILAH\\Desktop\\Trimester 2 2023\\STATS 7022\\WEEK 13\\DSPG_Practical_E
data
```

```
## # A tibble: 1,000 x 7
##        Y     X1      X2 X3                    X4 C1    C2
##    <dbl>  <dbl>   <dbl> <chr>              <dbl> <chr> <chr>
##  1  3.98 -0.150 -2.60   -1.74935165817275  3.19 c     W
##  2  2.86  1.11  -0.0636 0.685874962343099  2.06 b     W
##  3  1.89 -0.344 -1.07   2.86342100595413   2.50 b     X
##  4  3.07 NA     -0.164  0.592887995497871  3.06 c     Y
##  5  1.91 -1.53   0.988  2.76778554575478   3.48 a     Z
##  6  3.06 -0.777  0.921  1.17375123517006   3.46 a     Y
##  7  2.36 -1.56   0.697  2.75561166563404   3.47 a     W
##  8  3.41  1.10   1.36   1.24742213386541   2.64 d     Y
##  9  1.88  0.266  2.61   2.38658284906148   3.31 b     Y
## 10  2.80 -0.100 -0.902  2.21486518034953   3.34 c     X
## # i 990 more rows
```

Investigating the dataset.

The dataset has **1000** rows with 7 columns (3 characters type and 4 numerics type). After skimming the dataset, column **C2** has **15 missing values** and **X1** has **12 missing values**. X3 column should be in numeric instead of character type hence need to convert it.

```
skimr::skim(data)
```

Table 1: Data summary

Name	data
Number of rows	1000
Number of columns	7
Column type frequency:	
character	3
numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
X3	0	1.00	15	20	0	1000	0
C1	0	1.00	1	3	0	5	0
C2	15	0.98	1	1	0	6	0

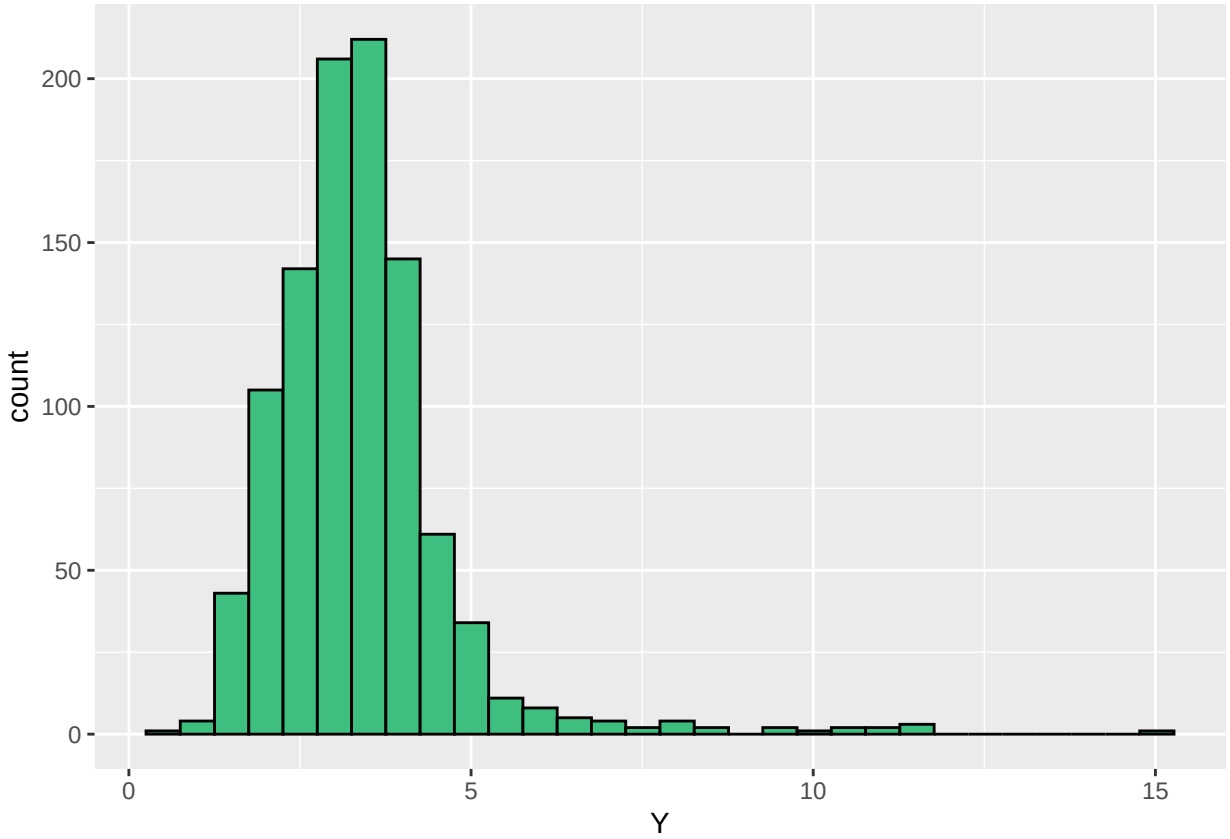
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Y	0	1.00	3.38	1.34	0.56	2.63	3.25	3.86	15.08	
X1	12	0.99	-0.06	0.95	-2.74	-0.71	-0.10	0.60	2.72	
X2	0	1.00	-0.58	4.49	-63.83	-1.23	-0.33	0.66	4.93	
X4	0	1.00	2.80	0.65	0.08	2.54	2.99	3.28	3.50	

Visualizations

Y

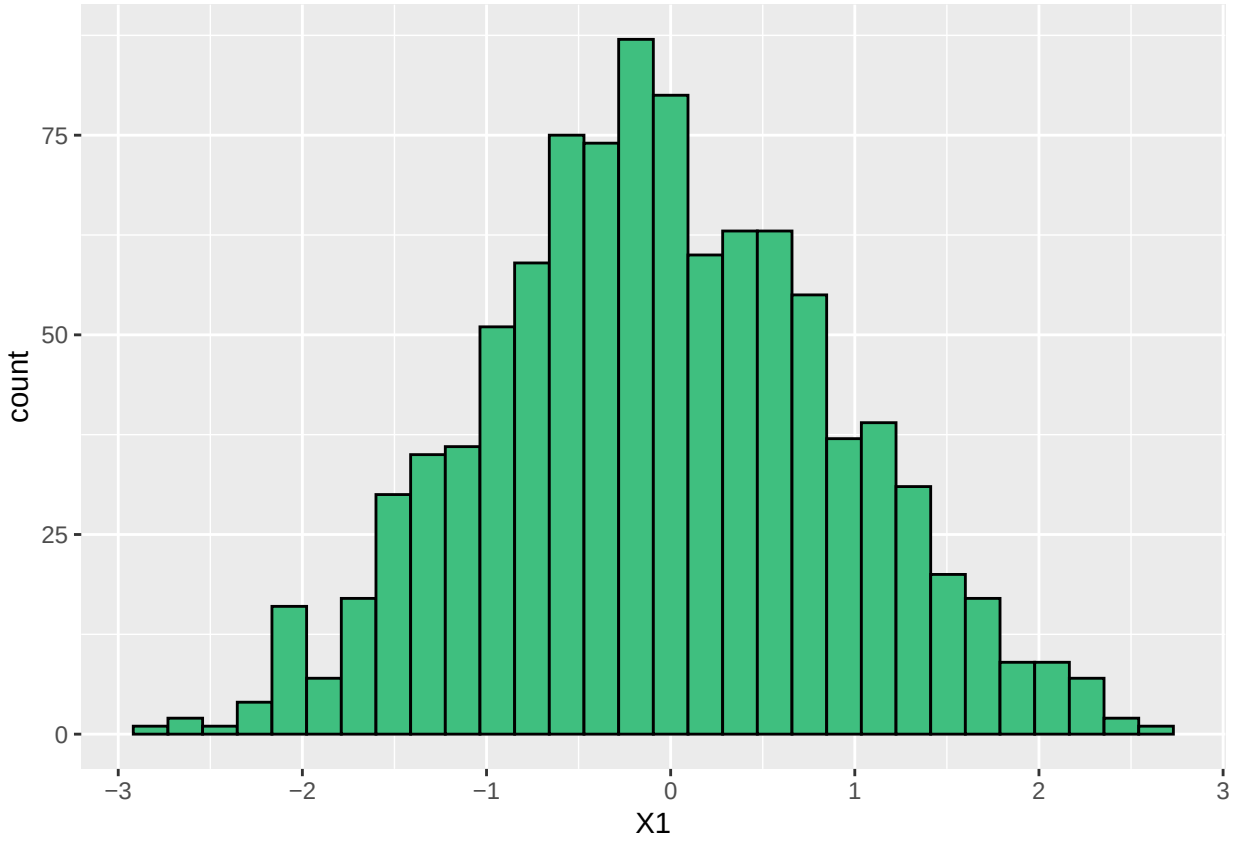
```
data %>% ggplot(aes(Y)) + geom_histogram(col="black",fill="#3fbf7f")
```



Y appears to has right-skewed distribution and unimodal. Its also appears that Y has **outlier above 12.5**.

X1

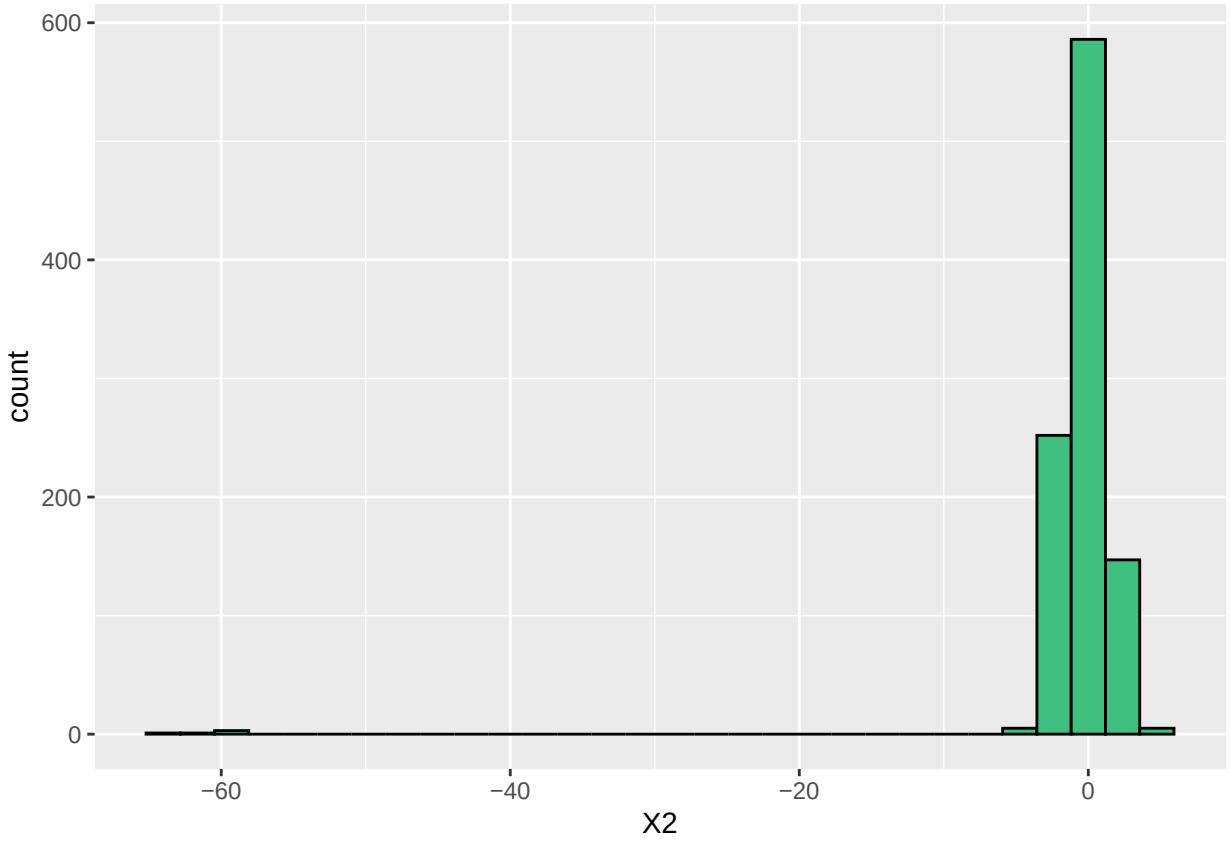
```
data %>% ggplot(aes(X1)) + geom_histogram(col="black",fill="#3fbf7f")
```



X1 appears roughly symmetric distribution and unimodal and there is no outlier shown in the histogram.

X2

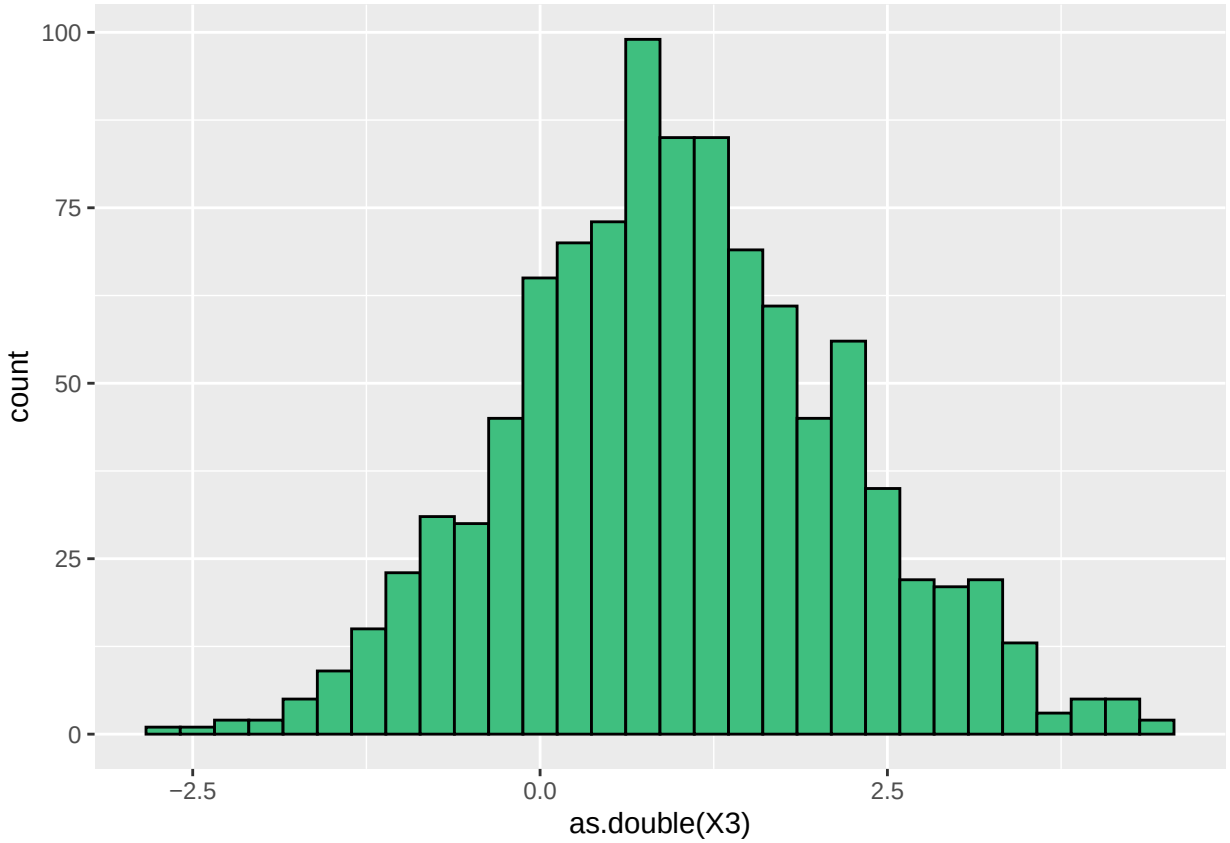
```
data %>% ggplot(aes(X2)) + geom_histogram(col="black",fill="#3fbf7f")
```



X2 appears to has roughly symmetric distribution and unimodal. Its also appears that X2 has **outlier below 0 at -60**.

X3

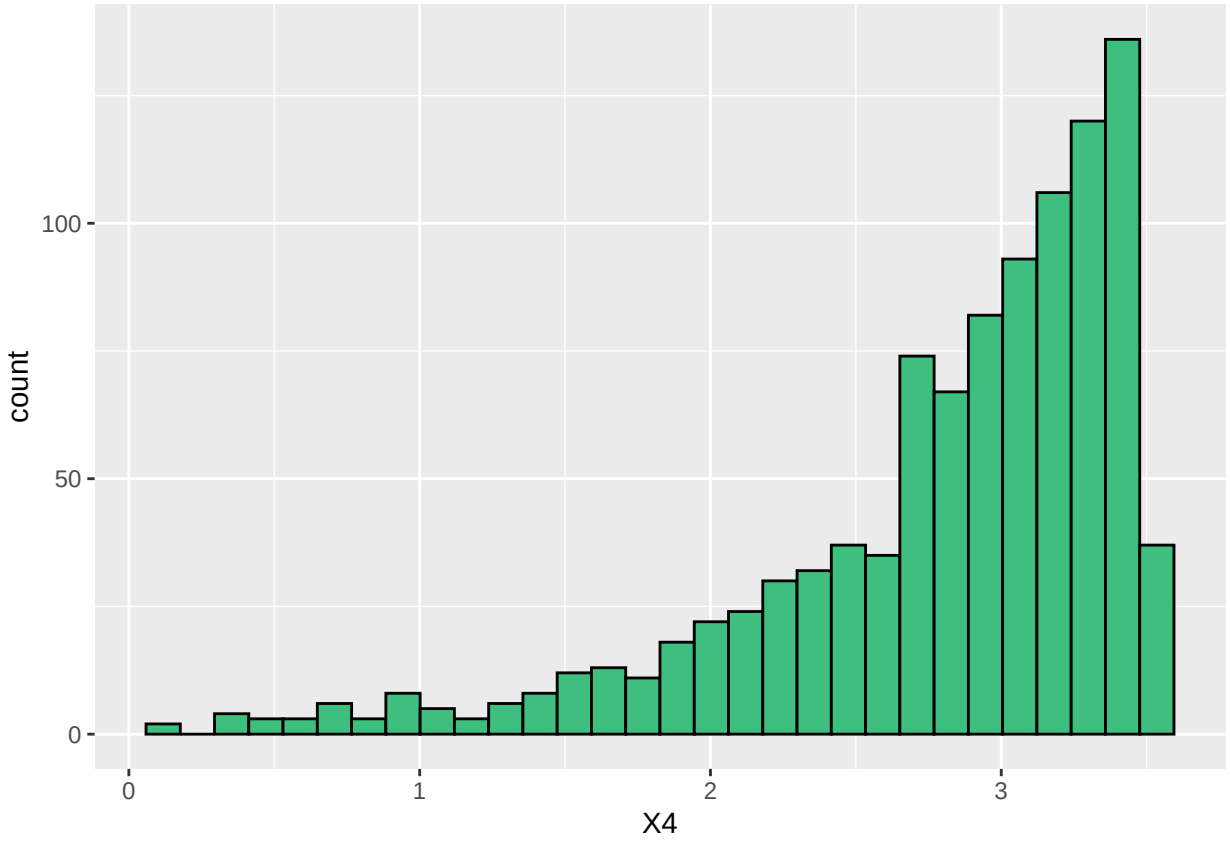
```
data %>% ggplot(aes(as.double(X3))) + geom_histogram(col="black",fill="#3fbf7f")
```



After masking the X3, it is X3 appears roughly symmetric distribution and unimodal and there is no outlier shown in the histogram.

X4

```
data %>% ggplot(aes(X4)) + geom_histogram(col="black",fill="#3fbf7f")
```



X4 appears to has left-skewed distribution and unimodal and there is no obvious outlier shown in the histogram.

Checking for outliers

```
data %>%
filter(Y > 12.5)
```

```
## # A tibble: 1 x 7
##       Y    X1    X2 X3                    X4 C1    C2
##   <dbl> <dbl> <dbl> <chr>              <dbl> <chr> <chr>
## 1  15.1  2.24 0.540 -1.65066617928017 0.0817 d     W
```

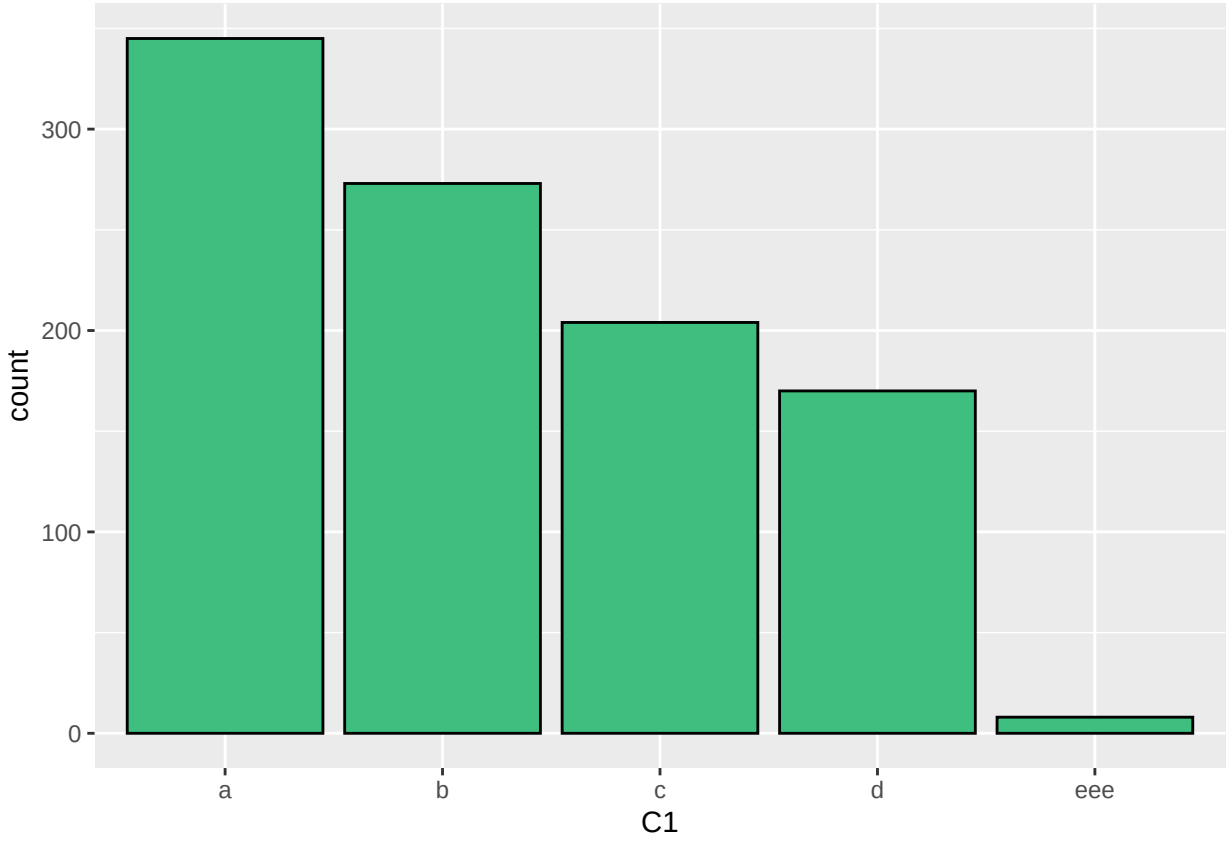
```
data %>%
filter(X2 < -20)
```

```
## # A tibble: 5 x 7
##       Y     X1    X2 X3                    X4 C1    C2
##   <dbl>  <dbl> <dbl> <chr>              <dbl> <chr> <chr>
## 1  2.99 -0.659 -62.0 -0.884191839300943  3.44 b     Z
## 2  1.71  0.577 -59.1 2.15097201440586    3.46 b     K
## 3  4.44  1.42  -58.3 -0.728374208890093  3.41 c     J
## 4  3.02  2.24  -60.3 0.297056055218382   2.95 c     X
## 5  4.90 -0.230 -63.8 -0.525592911445502  1.97 a     Z
```

It is observed that Y column has 1 outliers while X2 column has 5 outliers. The values were to far from the distribution and unlikely to be a correct value hence we can remove this from distribution during the data cleaning process.

C1

```
data %>% ggplot(aes(C1)) + geom_bar(col="black",fill="#3fbf7f")
```



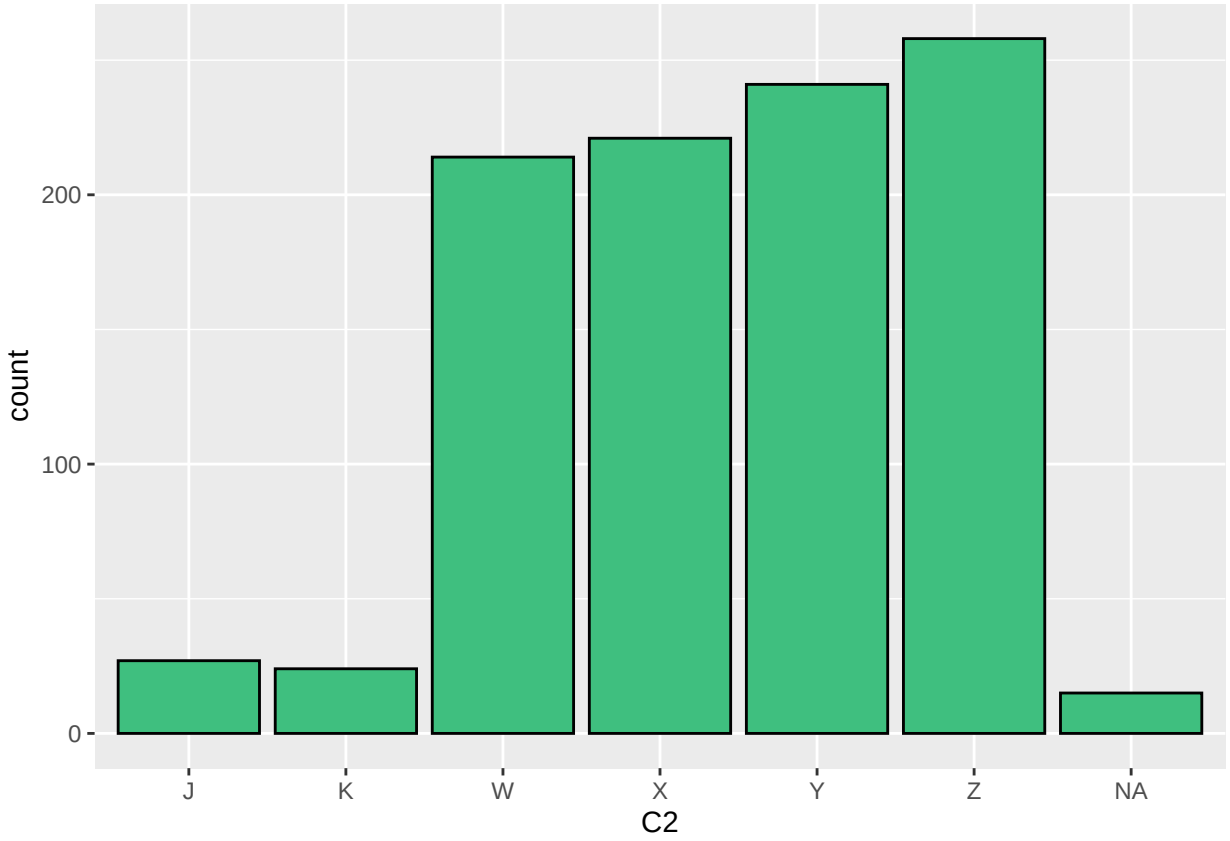

```
data %>%
filter(C1 == 'eee')
```

```
## # A tibble: 8 x 7
##       Y      X1      X2 X3                    X4 C1    C2
##   <dbl>   <dbl>   <dbl> <chr>              <dbl> <chr> <chr>
## 1 11.4  -0.819  -1.07   2.27806149384245    2.91 eee   Y
## 2 11.3  -0.109  -2.27   0.612955652656825   3.42 eee   Y
## 3 10.8   0.686  -0.891  -0.410664686874584  3.47 eee   W
## 4 10.1  -0.0814 -0.0716 0.767714070109195   3.43 eee   K
## 5 11.0   0.306   0.0897 -0.485118011472259  2.24 eee   Y
## 6  9.71 -0.925   0.534  1.95820912599341    2.97 eee   Y
## 7 10.6   0.653  -0.223  0.100147828501702   2.70 eee   Y
## 8 10.6  -0.205   0.342  -0.312610652790629  3.12 eee   Y
```

The levels of C1 seems monopolised by **a** at about **35%** of the distribution followed by b, c , d and e. **e** distribution has very small portion with **less than 1%** of the distribution.

C2

```
data %>% ggplot(aes(C2)) + geom_bar(col="black",fill="#3fbf7f")
```




```
data %>% filter(is.na(C2))
```

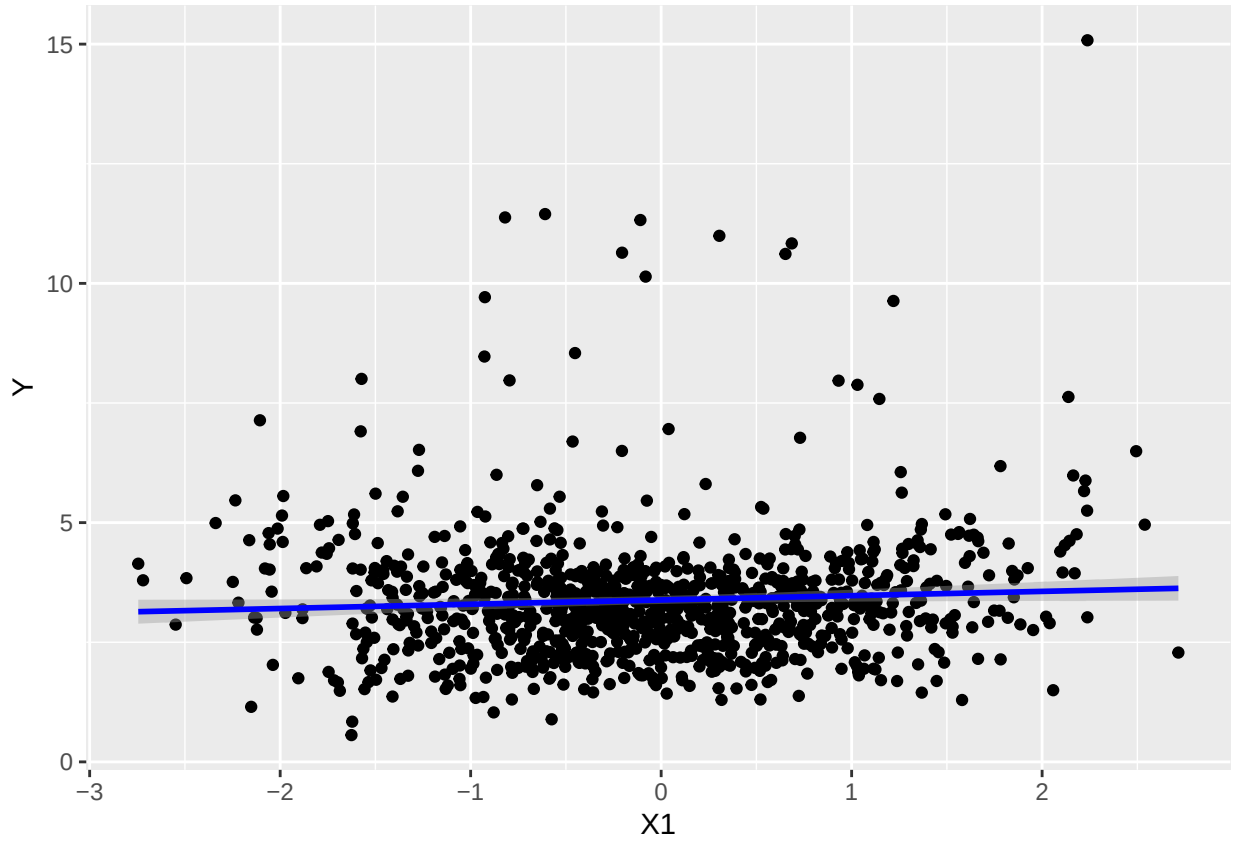
```
## # A tibble: 15 x 7
##        Y      X1     X2 X3                    X4 C1    C2
##    <dbl>   <dbl>  <dbl> <chr>              <dbl> <chr> <chr>
##  1  3.18  0.922  -1.33  0.391834343312658  2.85  a     <NA>
##  2  3.64  0.513   1.48  1.28487787236172   3.45  a     <NA>
##  3  2.01 -0.167  -0.449 2.03321465445502   2.94  b     <NA>
##  4  3.86  0.582   0.740 0.484097520046531  2.85  a     <NA>
##  5  4.00 -0.751  -1.18  -0.277734590728148 2.40  d     <NA>
##  6  2.57  0.0345  1.28  1.83623875224704   3.22  d     <NA>
##  7  6.96  0.0391 -0.984 0.436023987033152  0.591 a     <NA>
##  8  2.95 -0.903   0.677 0.473601220190164  2.96  d     <NA>
##  9  3.99  0.0816 -1.27  0.709799806973732  1.61  c     <NA>
## 10  7.63  2.14    4.33  -0.359828729454089 2.73  a     <NA>
## 11  3.66  0.0916  0.269 0.0104523611645457 3.26  a     <NA>
```

```
## 12  3.12  1.21   -1.44  -0.232899523125984 2.19  c     <NA>
## 13  4.24  0.117  -0.284 -0.575341431858281 2.65  a     <NA>
## 14  2.07  0.605  -1.32  -0.856797808173106 3.49  b     <NA>
## 15  1.84  0.768   0.452 1.99449876302465   3.44  b     <NA>
```

Its appear that **J** and **K** groups do not have many observations and may be worth consolidating these using step_other() functions. Its also appear there is 15 NA observations and judging by these 15 distributions, is still worth to keep it because these 15 data points are still within the rest of the data located. We can performs step impute by replacing in with highest frequency of the datapoint value.

X1 and Y

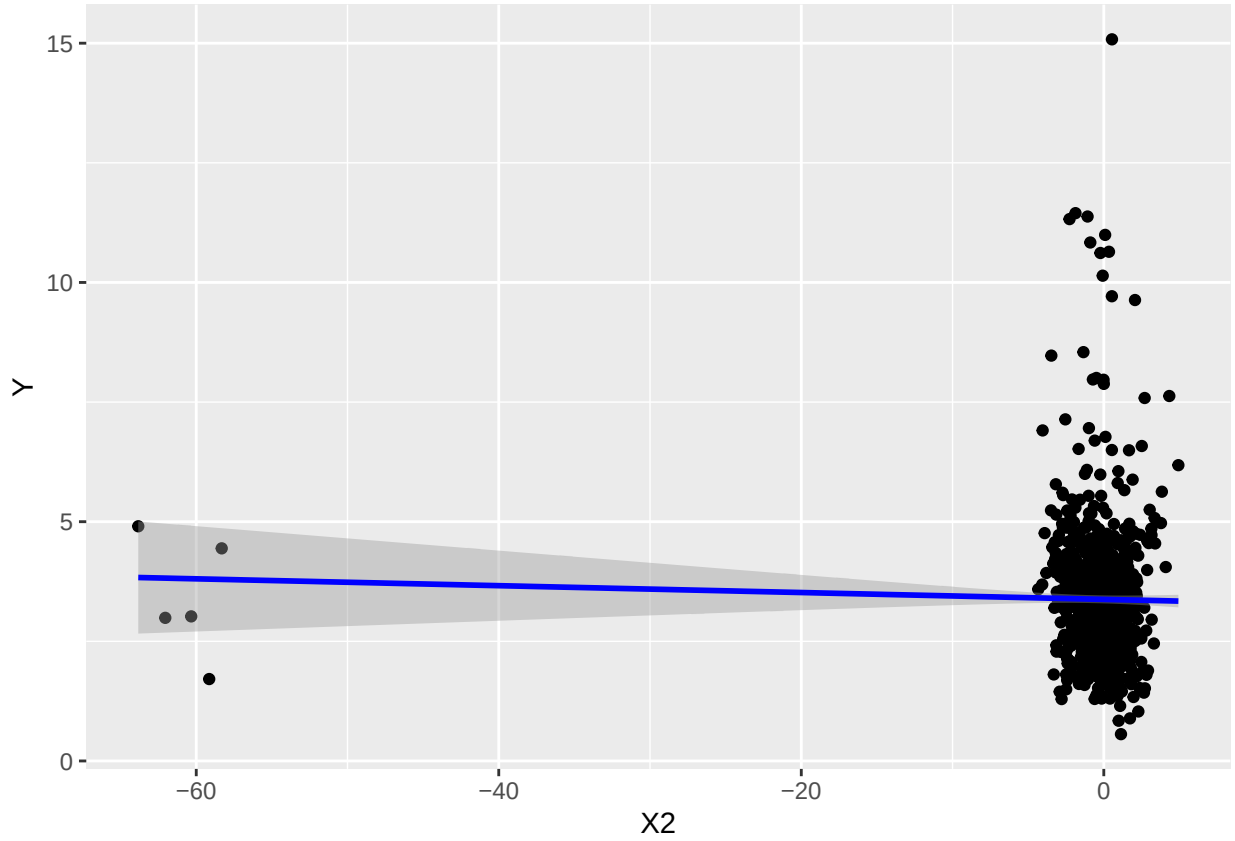
```
data %>% ggplot(aes(x=X1,y=Y)) + geom_point() +
  geom_smooth(col="blue",method = "lm")
```



X1 and Y has a weak, positive, linear relationship. Its weak because the slope of the distribution is not steep.

X2 and Y

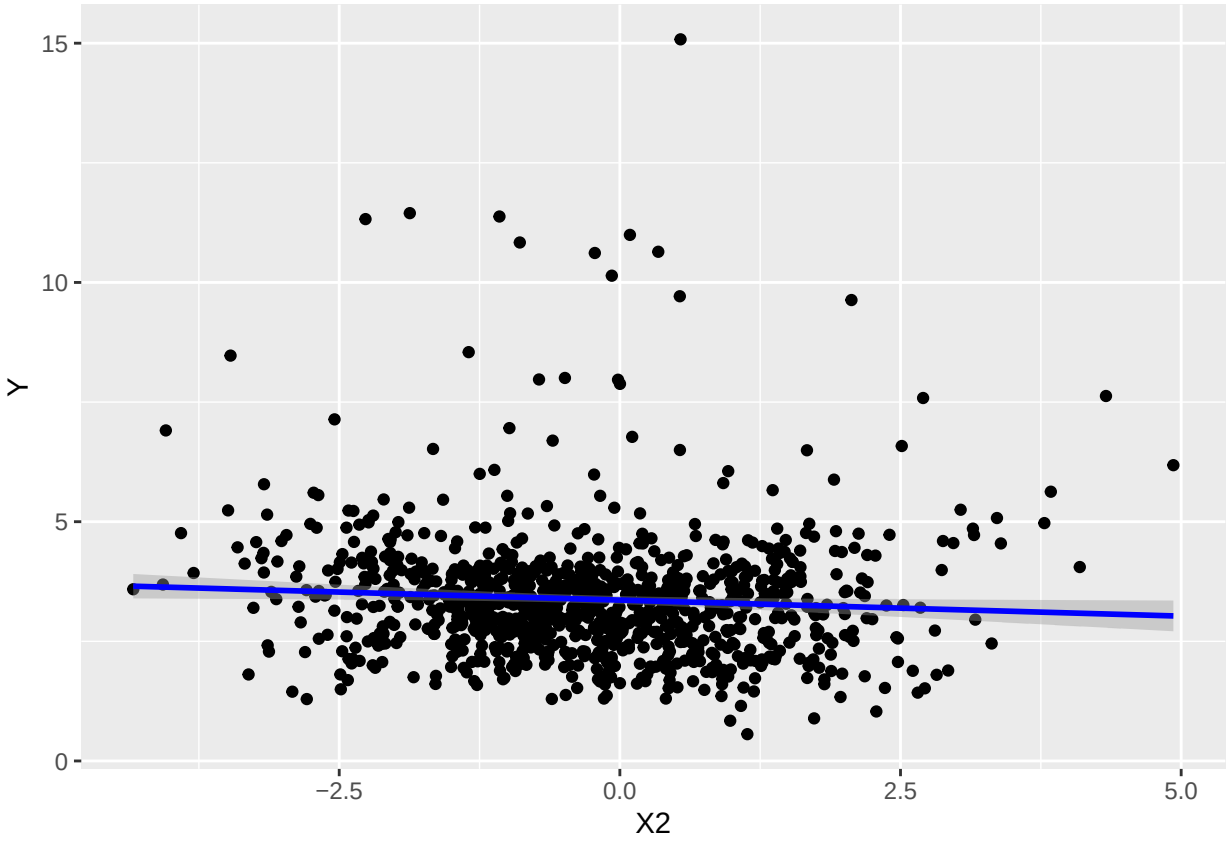
```
data %>% ggplot(aes(x=X2,y=Y)) + geom_point() +
  geom_smooth(col="blue",method = "lm")
```



X2 and Y has a weak, negative, linear relationship. Its weak because the slope of the distribution is not steep.

X2 and Y

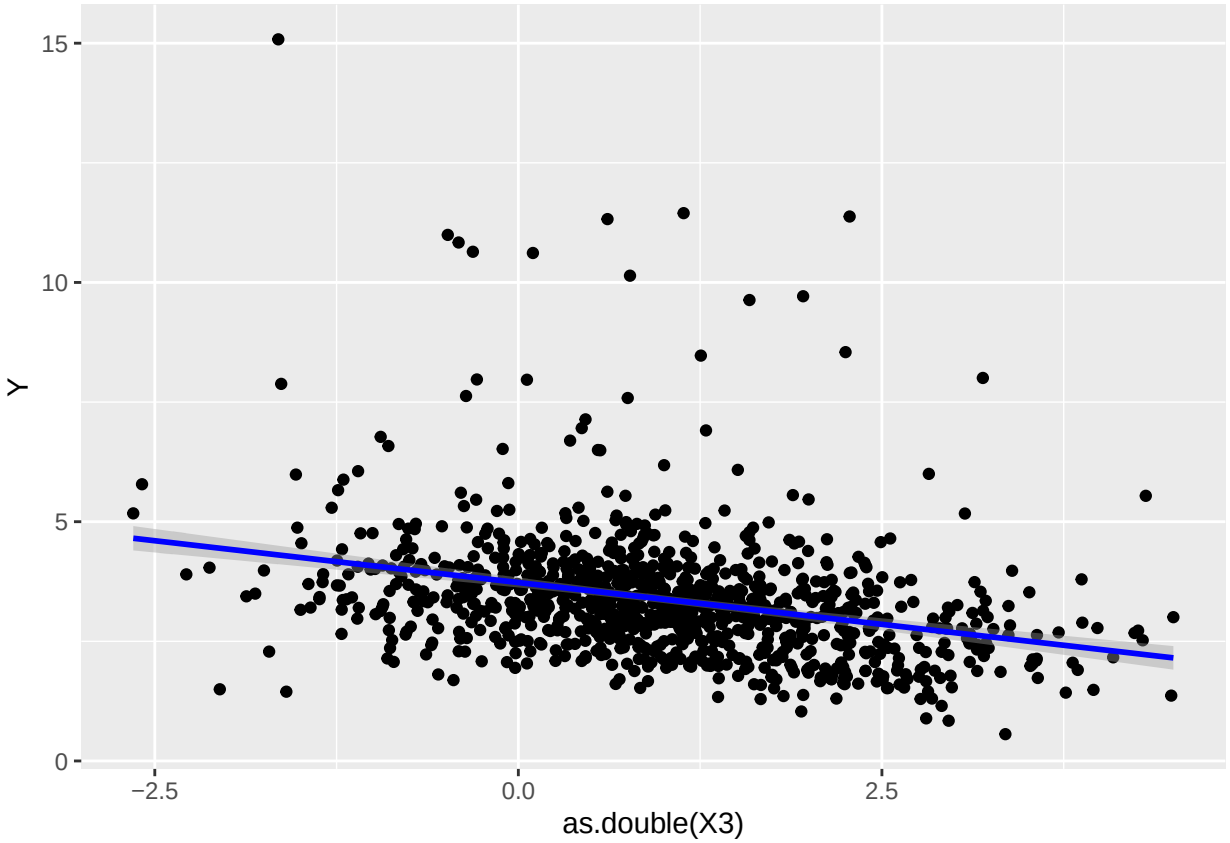
```
data %>% filter (X2 > -20) %>% ggplot(aes(x=X2,y=Y)) + geom_point() +
  geom_smooth(col="blue",method = "lm")
```



Another angle we can see this X2 vs Y distribution after we had filtered the X2 > -20 value. The distribution seems reasonable but still hold the same notion with X2 and Y has a weak, negative, linear relationship.

X3 and Y

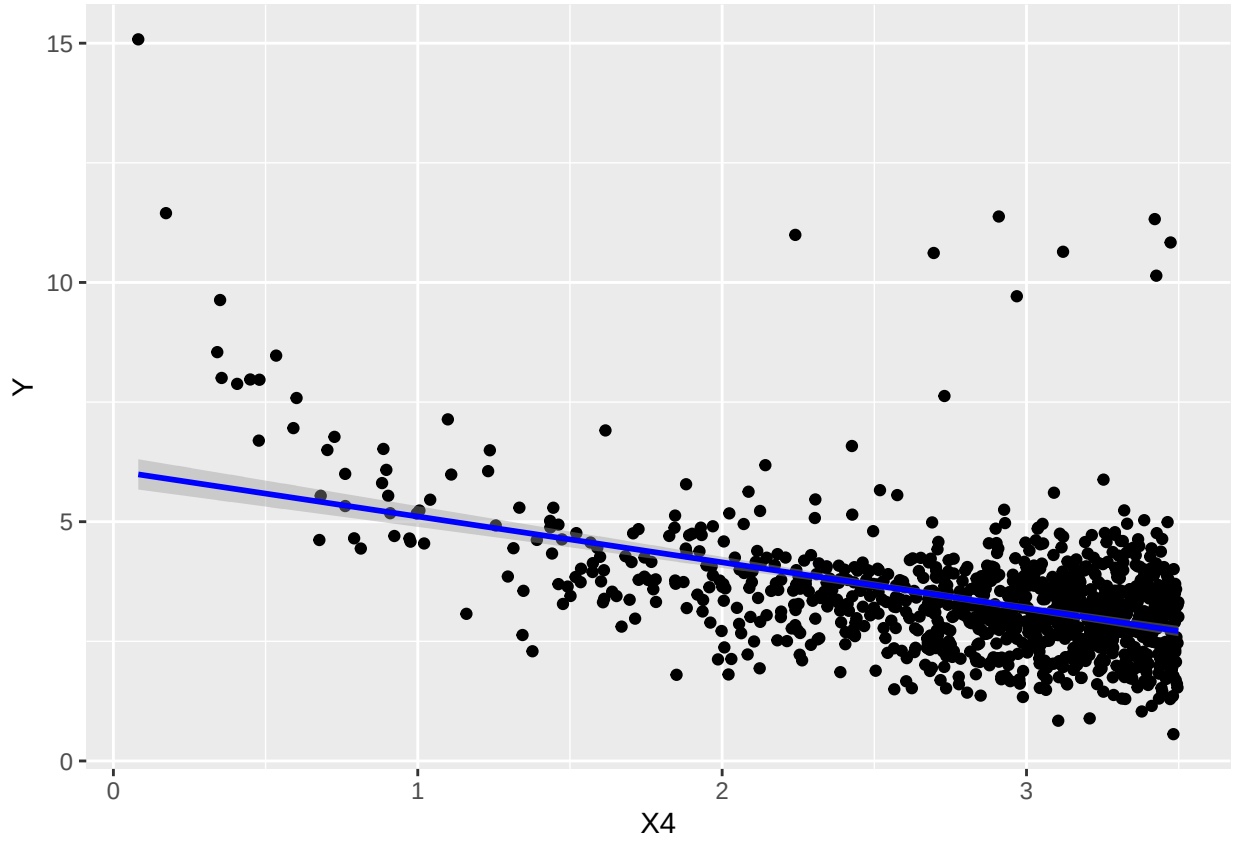
```
data %>% ggplot(aes(x=as.double(X3),y=Y)) + geom_point() +
  geom_smooth(col="blue",method = "lm")
```



X3 and Y has a moderate, negative, linear relationship.

X4 and Y

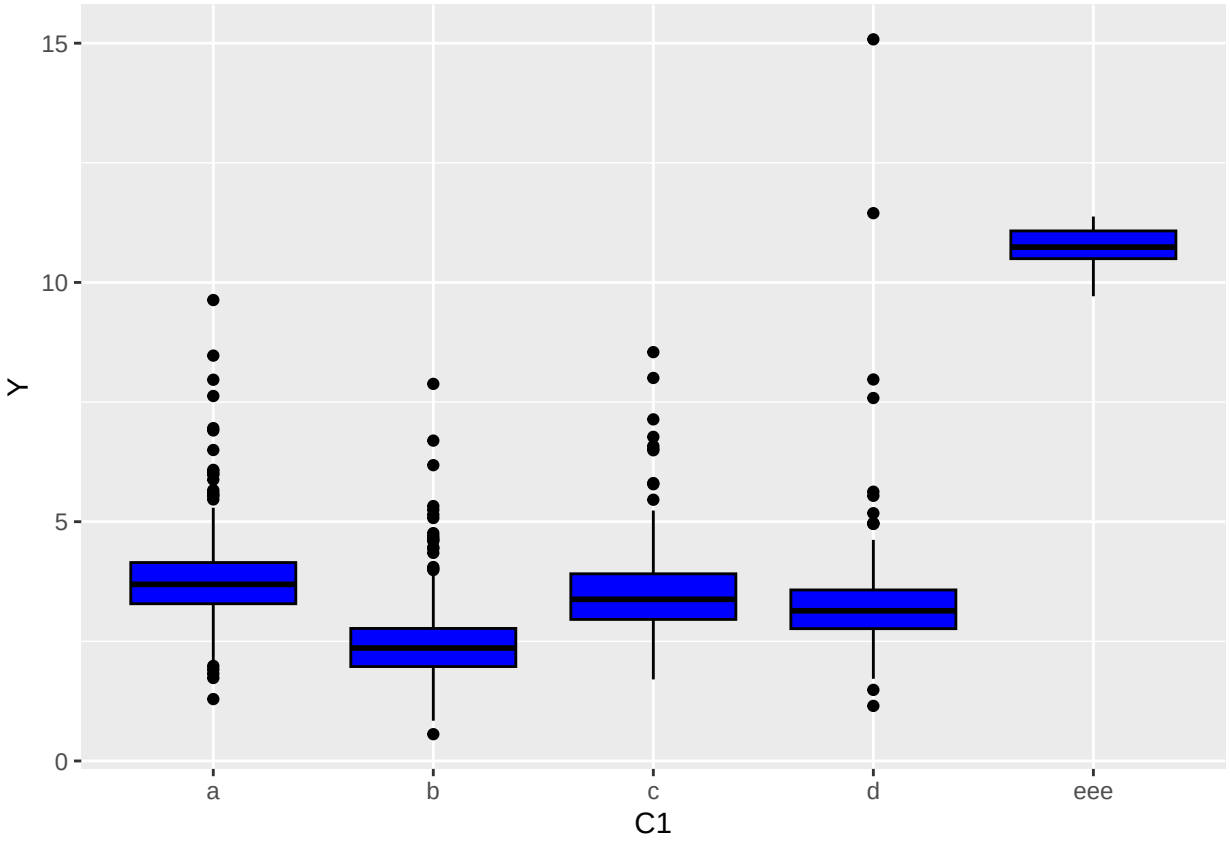
```
data %>% ggplot(aes(x=X4,y=Y)) + geom_point() +
  geom_smooth(col="blue",method = "lm")
```



X4 and Y has a moderate, negative, linear relationship.

C1 and Y

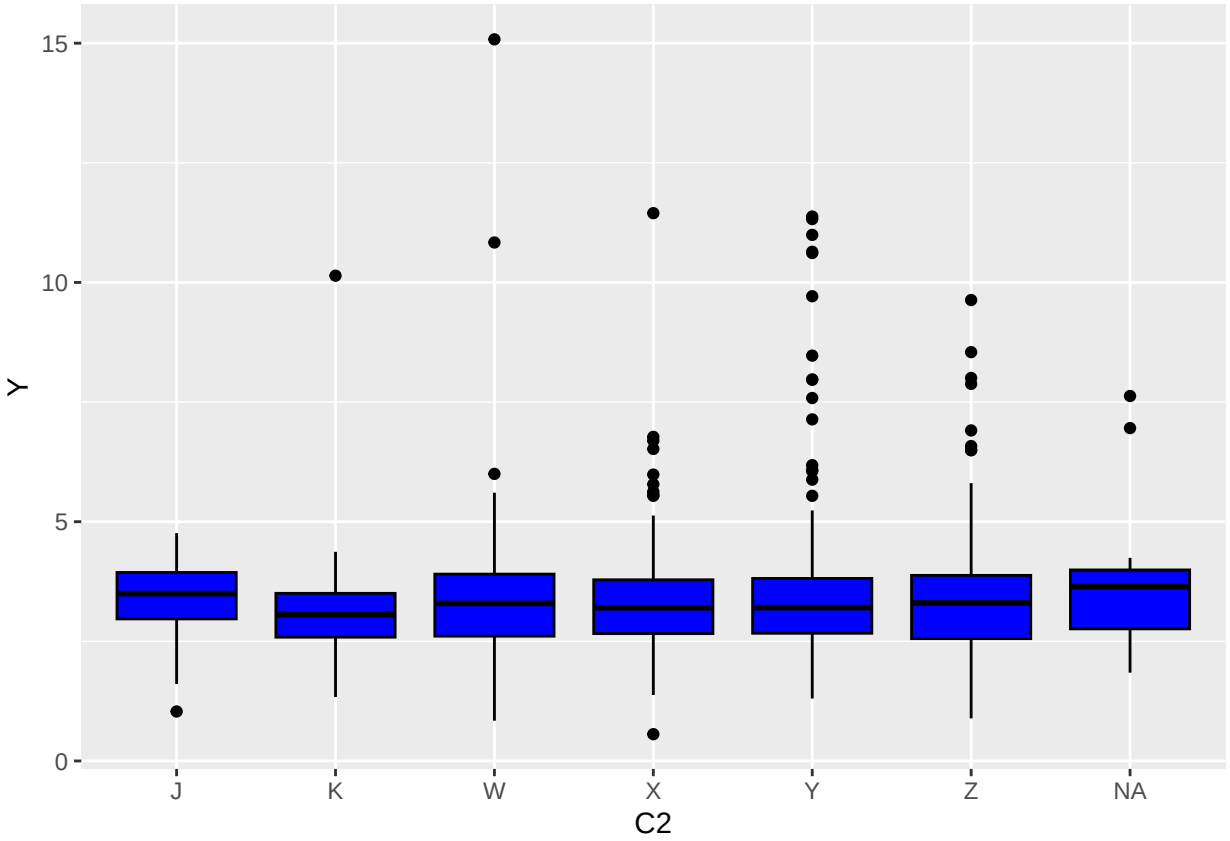
```
data %>% ggplot(aes(x=C1,y=Y,group=C1)) + geom_boxplot(col="black",fill='blue')
```



Level **eee** seems to have the highest mean followed by **a**, **c** ,**d** and lastly **b**. It appears all levels have outliers except for level **eee**. The width of each boxplot is roughly the same indicating interquartile range (IQR) is relatively close to Q1 and Q3 in the distribution.

C2 and Y

```
data %>% ggplot(aes(x=C2,y=Y,group=C2)) + geom_boxplot(col="black",fill='blue')
```

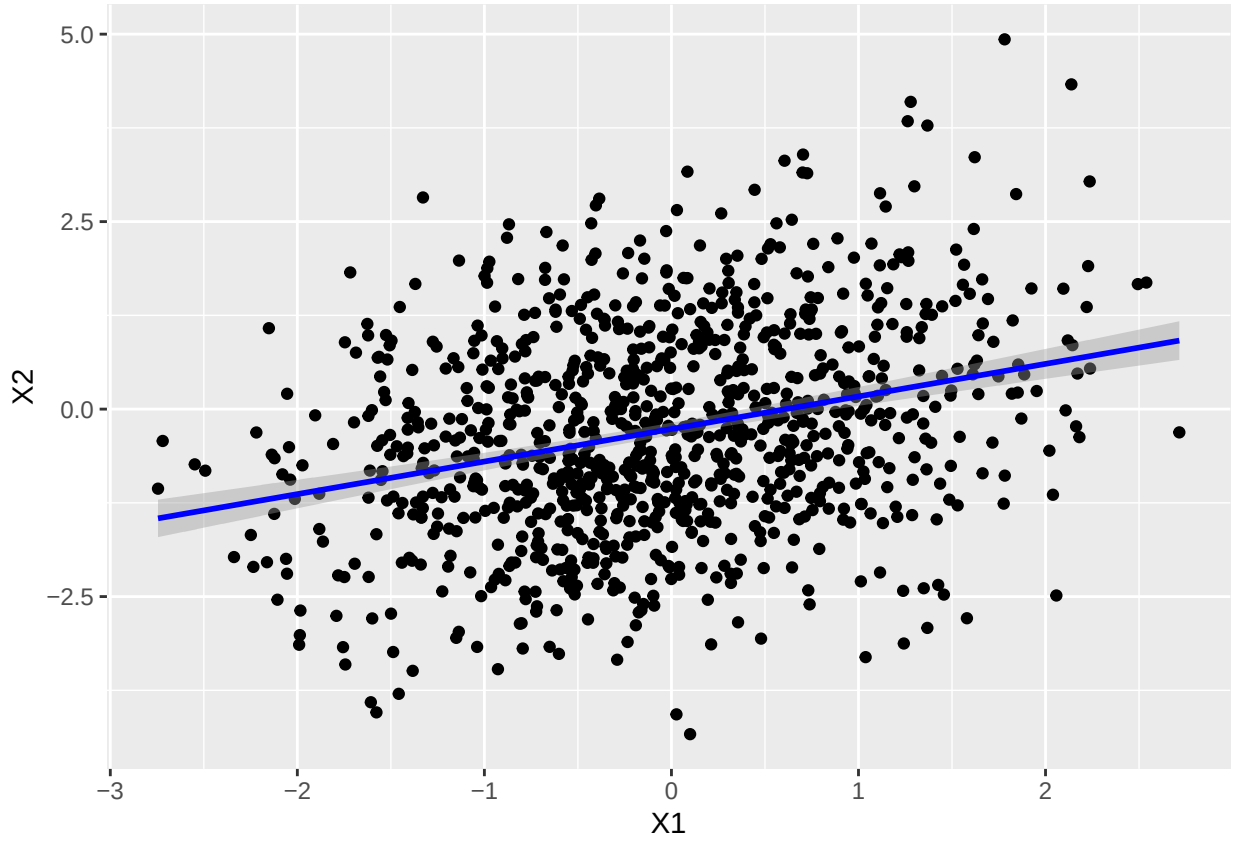


The highest mean is level **NA** while the rest have roughly the same median in the distribution. It appears all levels have outliers.

There is no strong relationship between features and target variables. Perhaps, there is interaction between the features.

X1 and X2

```
data %>% filter (X2 > -20) %>% ggplot(aes(x=X1,y=X2)) + geom_point() +
  geom_smooth(col="blue",method = "lm")
```



It appears X1 and X2 have positive and linear relationship after filtering out the X2 outliers. Could be worth to explore their interaction.

2. Cleaning the dataset

Cleaning the level names

Changing the level **'eee'** to **e**.

```
data <- data %>%
  mutate(C1 = ifelse(C1 == "eee", "e", C1))
data
```

```
## # A tibble: 1,000 x 7
##        Y     X1      X2 X3                    X4 C1    C2
##    <dbl>  <dbl>   <dbl> <chr>              <dbl> <chr> <chr>
##  1  3.98 -0.150 -2.60   -1.74935165817275  3.19 c     W
##  2  2.86  1.11  -0.0636 0.685874962343099  2.06 b     W
##  3  1.89 -0.344 -1.07   2.86342100595413   2.50 b     X
##  4  3.07 NA     -0.164  0.592887995497871  3.06 c     Y
##  5  1.91 -1.53   0.988  2.76778554575478   3.48 a     Z
##  6  3.06 -0.777  0.921  1.17375123517006   3.46 a     Y
##  7  2.36 -1.56   0.697  2.75561166563404   3.47 a     W
##  8  3.41  1.10   1.36   1.24742213386541   2.64 d     Y
##  9  1.88  0.266  2.61   2.38658284906148   3.31 b     Y
```

```
## 10  2.80 -0.100 -0.902  2.21486518034953   3.34 c     X
## # i 990 more rows
```

Changing the type.

Changing the **X3** column type from **character** to **double**.

```
data <- data %>%
  mutate(X3 = as.double(X3))
data
```

```
## # A tibble: 1,000 x 7
##        Y     X1      X2     X3    X4 C1    C2
##    <dbl>  <dbl>   <dbl>  <dbl> <dbl> <chr> <chr>
##  1  3.98 -0.150 -2.60   -1.75   3.19 c     W
##  2  2.86  1.11  -0.0636  0.686  2.06 b     W
##  3  1.89 -0.344 -1.07    2.86   2.50 b     X
##  4  3.07 NA     -0.164   0.593  3.06 c     Y
##  5  1.91 -1.53   0.988   2.77   3.48 a     Z
##  6  3.06 -0.777  0.921   1.17   3.46 a     Y
##  7  2.36 -1.56   0.697   2.76   3.47 a     W
##  8  3.41  1.10   1.36    1.25   2.64 d     Y
##  9  1.88  0.266  2.61    2.39   3.31 b     Y
## 10  2.80 -0.100 -0.902   2.21   3.34 c     X
## # i 990 more rows
```

Removing outliers

Removing outlier based on Y and X2 columns.

```
data  <- data %>%
filter(Y < 12.5)

data <- data %>%
filter(X2 > -20)
data
```

```
## # A tibble: 994 x 7
##        Y     X1      X2     X3    X4 C1    C2
##    <dbl>  <dbl>   <dbl>  <dbl> <dbl> <chr> <chr>
##  1  3.98 -0.150 -2.60   -1.75   3.19 c     W
##  2  2.86  1.11  -0.0636  0.686  2.06 b     W
##  3  1.89 -0.344 -1.07    2.86   2.50 b     X
##  4  3.07 NA     -0.164   0.593  3.06 c     Y
##  5  1.91 -1.53   0.988   2.77   3.48 a     Z
##  6  3.06 -0.777  0.921   1.17   3.46 a     Y
##  7  2.36 -1.56   0.697   2.76   3.47 a     W
##  8  3.41  1.10   1.36    1.25   2.64 d     Y
##  9  1.88  0.266  2.61    2.39   3.31 b     Y
## 10  2.80 -0.100 -0.902   2.21   3.34 c     X
## # i 984 more rows
```

3. Data Splitting

Split the data into training and testing sets.

Ensure that our training and testing sets are balanced according to Y, so we specify **strata = Y** in the function initial_split().

```
set.seed(2023)
# splitting the data into testing and training set
data_split <- initial_split(data, strata = Y)
data_train <- training(data_split)
data_test <- testing(data_split)
data_cv <- vfold_cv(data_train,strata=Y)
data_cv
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [668/76]> Fold01
##  2 <split [668/76]> Fold02
##  3 <split [668/76]> Fold03
##  4 <split [668/76]> Fold04
##  5 <split [668/76]> Fold05
##  6 <split [668/76]> Fold06
##  7 <split [672/72]> Fold07
##  8 <split [672/72]> Fold08
##  9 <split [672/72]> Fold09
## 10 <split [672/72]> Fold10
```

4. Recipe

Pre-processing the data using recipe.

Creating the recipe.

```
library(recipes)
data_recipe <- recipe (Y ~., data = data_train) %>%
  step_impute_median(all_numeric_predictors())%>% # impute the missing value with median of each column
  step_impute_mode(all_nominal_predictors()) %>%  # impute the missing value with the highest mode ie l
  step_other(C2) %>%  #consolidate J and K.
  step_interact(terms=~X1:X2) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

data_recipe %>% prep %>% bake(new_data = NULL)
```

```
## # A tibble: 744 x 14
##          X1    X2      X3     X4     Y X1_x_X2   C1_b   C1_c   C1_d    C1_e
##       <dbl> <dbl>   <dbl>  <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>
##  1 -1.59    0.677  1.47    1.07   2.36  -1.05  -0.633 -0.494 -0.456 -0.0636
```

```
##  2  0.323   2.03   1.16    0.811  1.88   0.182  1.58  -0.494 -0.456 -0.0636
##  3 -0.408   0.688  0.0227  0.488  2.09  -0.509  1.58  -0.494 -0.456 -0.0636
##  4 -0.483   0.151  0.269   1.01   1.99  -0.281  1.58  -0.494 -0.456 -0.0636
##  5 -0.0798  0.766  1.04   -0.655  1.86  -0.364  1.58  -0.494 -0.456 -0.0636
##  6 -0.699   0.489  0.146   0.788  1.99  -0.508  1.58  -0.494 -0.456 -0.0636
##  7 -0.00248 0.866  1.05    0.186  1.76  -0.326  1.58  -0.494 -0.456 -0.0636
##  8  0.414   1.63  -0.366   0.692  2.59   0.201  1.58  -0.494 -0.456 -0.0636
##  9 -1.48    1.15   0.431   0.713  2.13  -1.66  -0.633  2.02  -0.456 -0.0636
## 10  0.619   0.928  0.657   0.943  2.19   0.101  1.58  -0.494 -0.456 -0.0636
## # i 734 more rows
## # i 4 more variables: C2_X <dbl>, C2_Y <dbl>, C2_Z <dbl>, C2_other <dbl>
```

5. Model

Specify the Lasso Regression model.

In defining Lasso Regression model, we need to specify **mixture = 1**, so the Linear Regression will set up our model to Lasso mode. Since we are going to find the best lambda, the **penalty** will be **tuned**.

```
data_lasso_model <- linear_reg(mixture = 1, penalty = tune()) %>%
  set_engine("glmnet") %>%
  set_mode("regression")
```

6. Workflow

Combining the model and and recipe into one workflow

```
data_workflow <- workflow() %>%
  add_recipe(data_recipe) %>%
  add_model(data_lasso_model)
data_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 6 Recipe Steps
##
## * step_impute_median()
## * step_impute_mode()
## * step_other()
## * step_interact()
## * step_dummy()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Linear Regression Model Specification (regression)
##
```

```
## Main Arguments:
##   penalty = tune()
##   mixture = 1
##
## Computational engine: glmnet
```

7.Tuning

Tuning lambda using grid search

To find the best lambda for the Lasso model, we will utilise the grid search function.

```
data_grid <- tibble(penalty = 10^seq(-6,2,0.2))
data_grid
```

```
## # A tibble: 41 x 1
##       penalty
##         <dbl>
##  1 0.000001
##  2 0.00000158
##  3 0.00000251
##  4 0.00000398
##  5 0.00000631
##  6 0.00001
##  7 0.0000158
##  8 0.0000251
##  9 0.0000398
## 10 0.0000631
## # i 31 more rows
```

Model Tuning

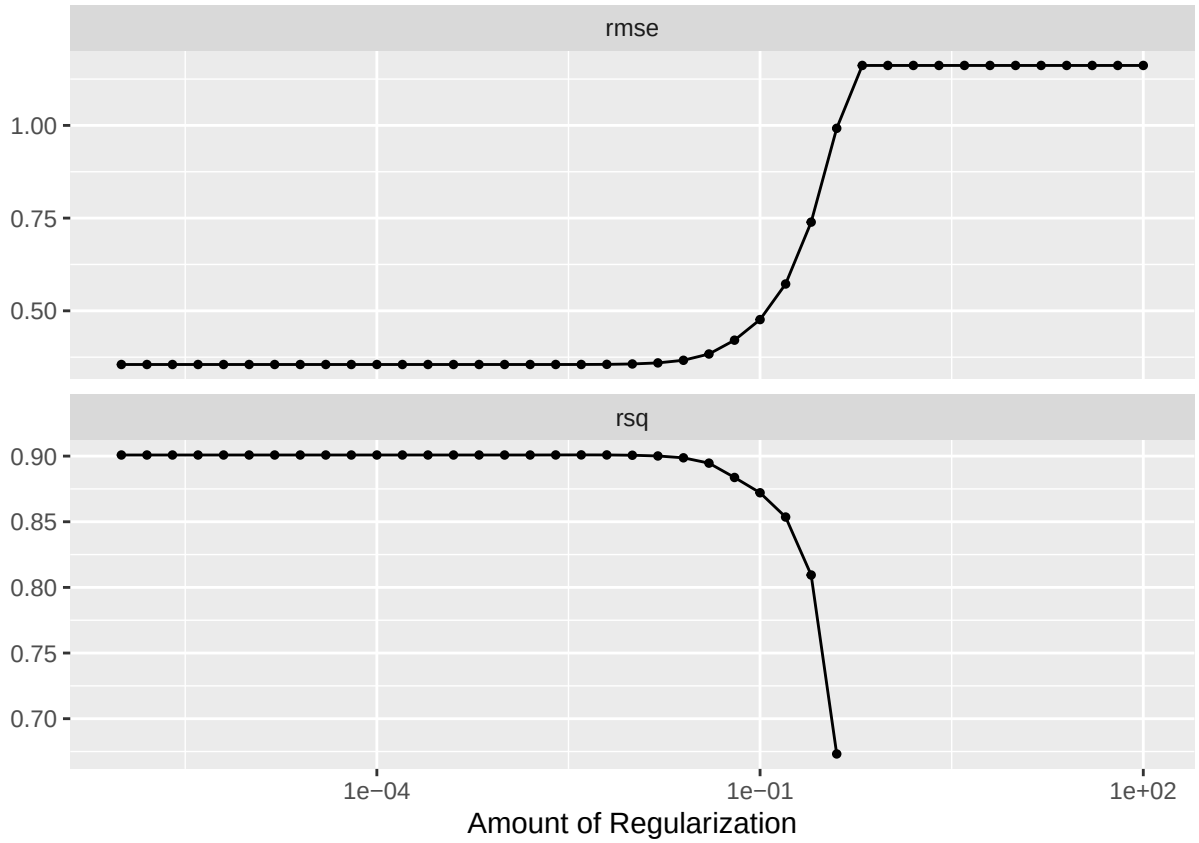
```
doParallel::registerDoParallel()
data_tune_results <- tune_grid(data_workflow,
                               resamples = data_cv,
                               grid = data_grid)

write_rds(data_tune_results,"data_tune.rds")
```

8.Result

Getting the result

Choose the best model.

```
data_tune_results %>% autoplot()
```




```
show_best(data_tune_results, metric = "rmse")
```

```
## # A tibble: 5 x 7
##      penalty .metric .estimator  mean     n std_err .config
##        <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 0.000001   rmse    standard   0.355    10  0.0458 Preprocessor1_Model01
## 2 0.00000158 rmse    standard   0.355    10  0.0458 Preprocessor1_Model02
## 3 0.00000251 rmse    standard   0.355    10  0.0458 Preprocessor1_Model03
## 4 0.00000398 rmse    standard   0.355    10  0.0458 Preprocessor1_Model04
## 5 0.00000631 rmse    standard   0.355    10  0.0458 Preprocessor1_Model05
```

```
data_hyperparams <- select_best(data_tune_results, metric = "rmse")
data_hyperparams
```

```
## # A tibble: 1 x 2
##    penalty .config
##      <dbl> <chr>
## 1 0.000001 Preprocessor1_Model01
```

Finalising the model

```
data_final <- data_workflow %>%
finalize_workflow(data_hyperparams) %>%
fit(data_train)
data_final %>% tidy() %>% print(n = Inf)
```

```
## Warning: package 'glmnet' was built under R version 4.2.3
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.2.3
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-7
```

```
## # A tibble: 14 x 3
##    term        estimate  penalty
##    <chr>          <dbl>    <dbl>
##  1 (Intercept)  3.33    0.000001
##  2 X1          -0.0180  0.000001
##  3 X2          -0.00796 0.000001
##  4 X3          -0.395   0.000001
##  5 X4          -0.527   0.000001
##  6 X1_x_X2      0.588   0.000001
##  7 C1_b        -0.578   0.000001
##  8 C1_c        -0.135   0.000001
##  9 C1_d        -0.222   0.000001
## 10 C1_e         0.460   0.000001
## 11 C2_X         0.00687 0.000001
## 12 C2_Y         0       0.000001
## 13 C2_Z         0.00195 0.000001
## 14 C2_other    -0.00406 0.000001
```

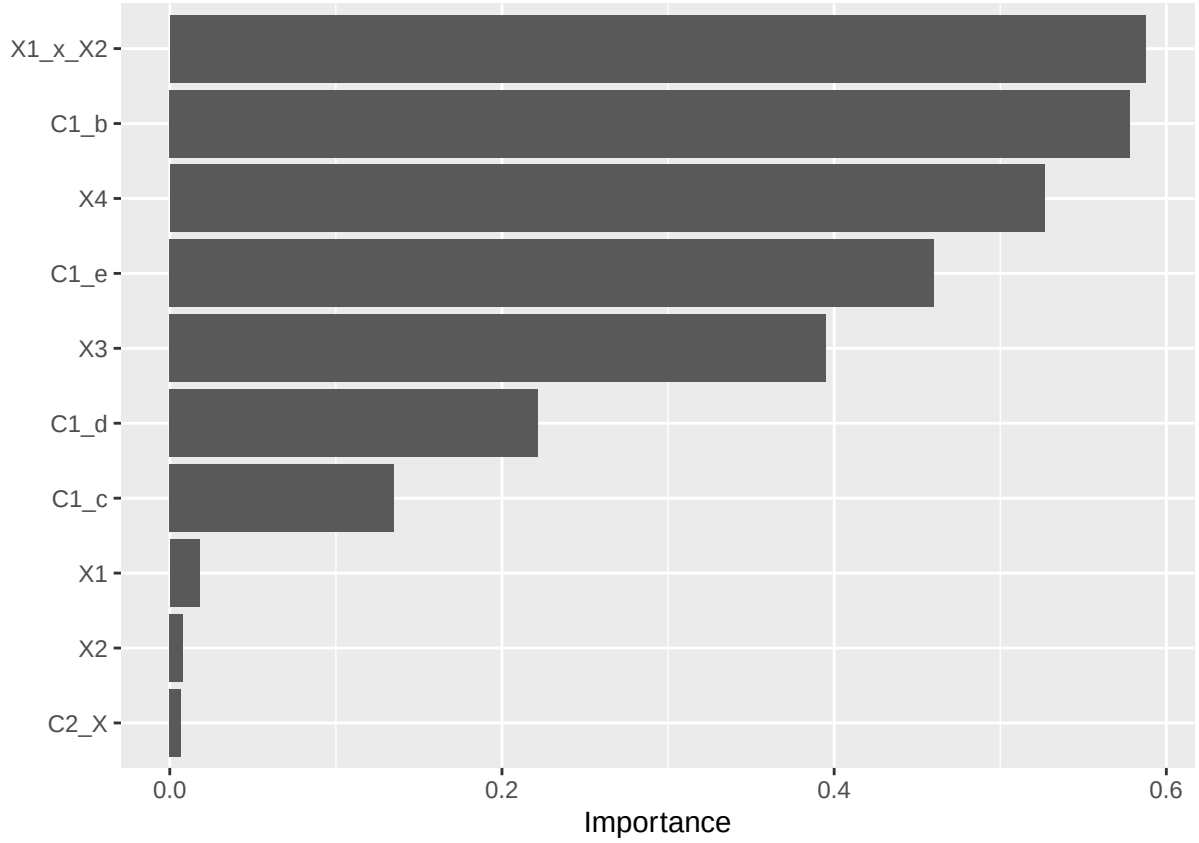
Check last fit

```
data_fit <- data_final %>%
last_fit(data_split)

data_fit %>% collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0.434 Preprocessor1_Model1
## 2 rsq     standard       0.922 Preprocessor1_Model1
```

Variable Importance

```
data_fit %>% extract_fit_engine() %>% vip::vip()
```



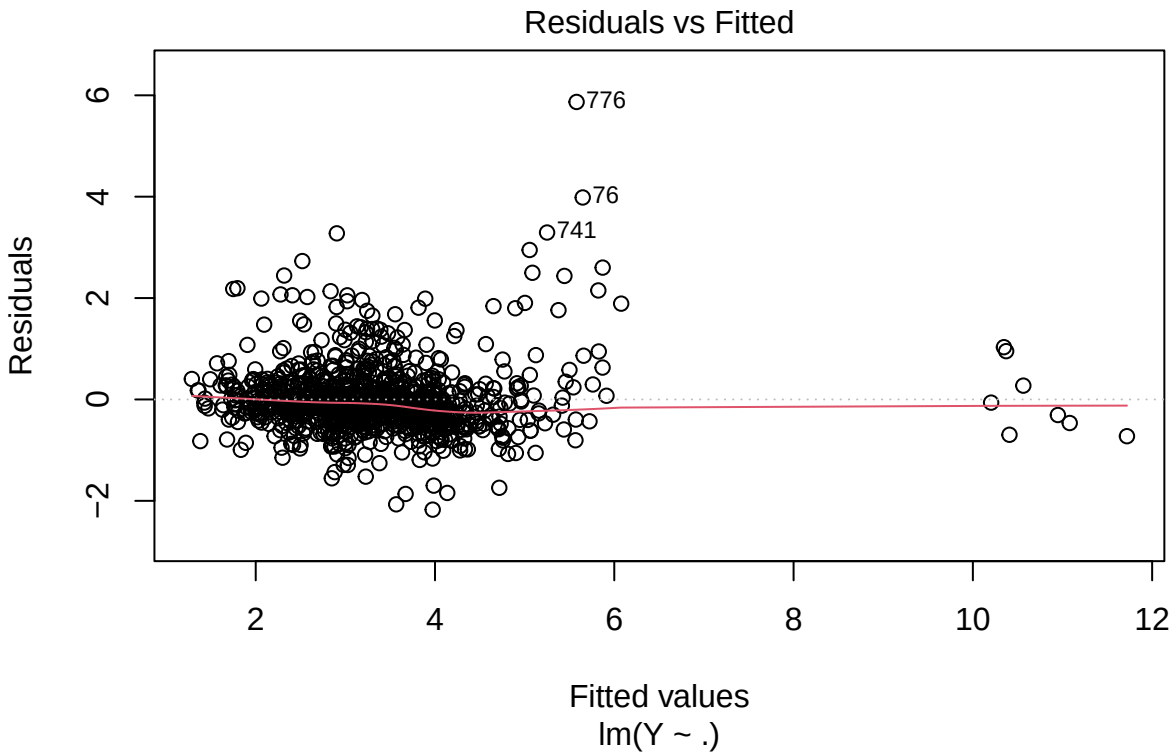
Seems there is interaction between **X1 and X2** features making it has the large effect on Y as does **C1**. **X4** is also contributing the effect on Y. However, **X2** alone and **C2** have negligible effect on Y.

9. Assumption Checking

Linearity

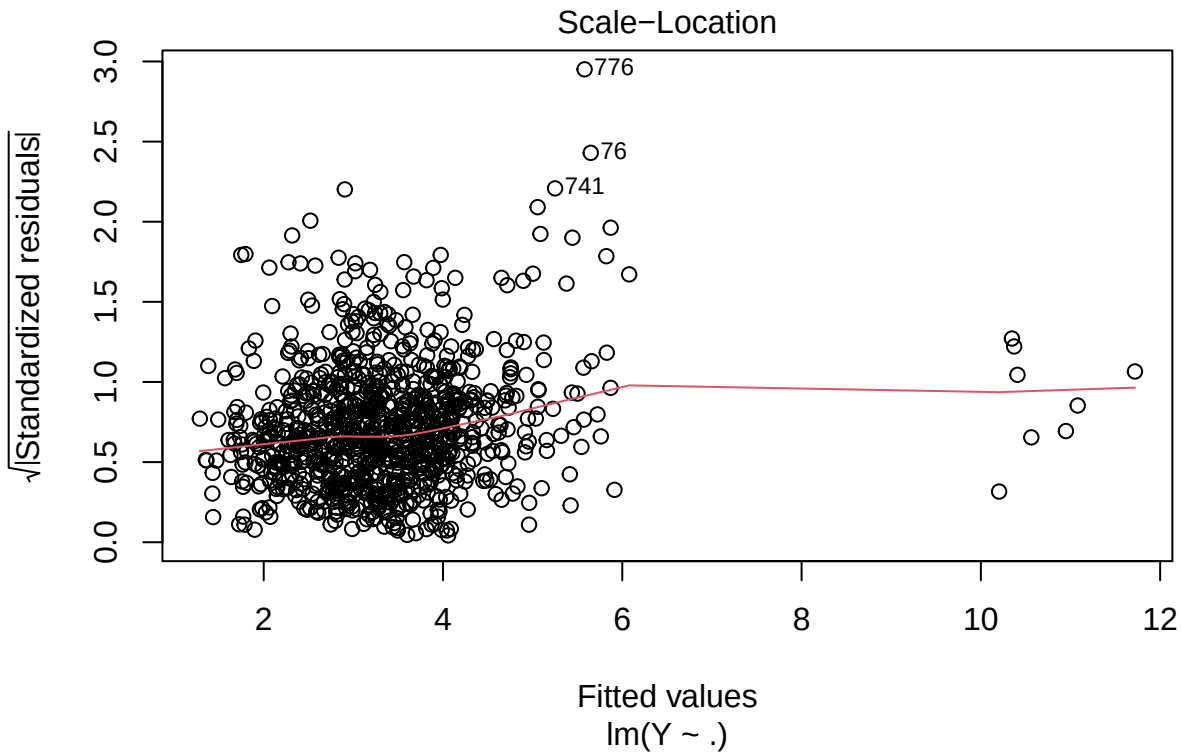
It appears the dataset has a good linearity pattern because the residuals are randomly scattered around the zero line without any systematic trends, curvatures, or sudden changing in spreads.

```
lm_1 <- lm(Y~.,data = data)
plot(lm_1, which = 1)
```



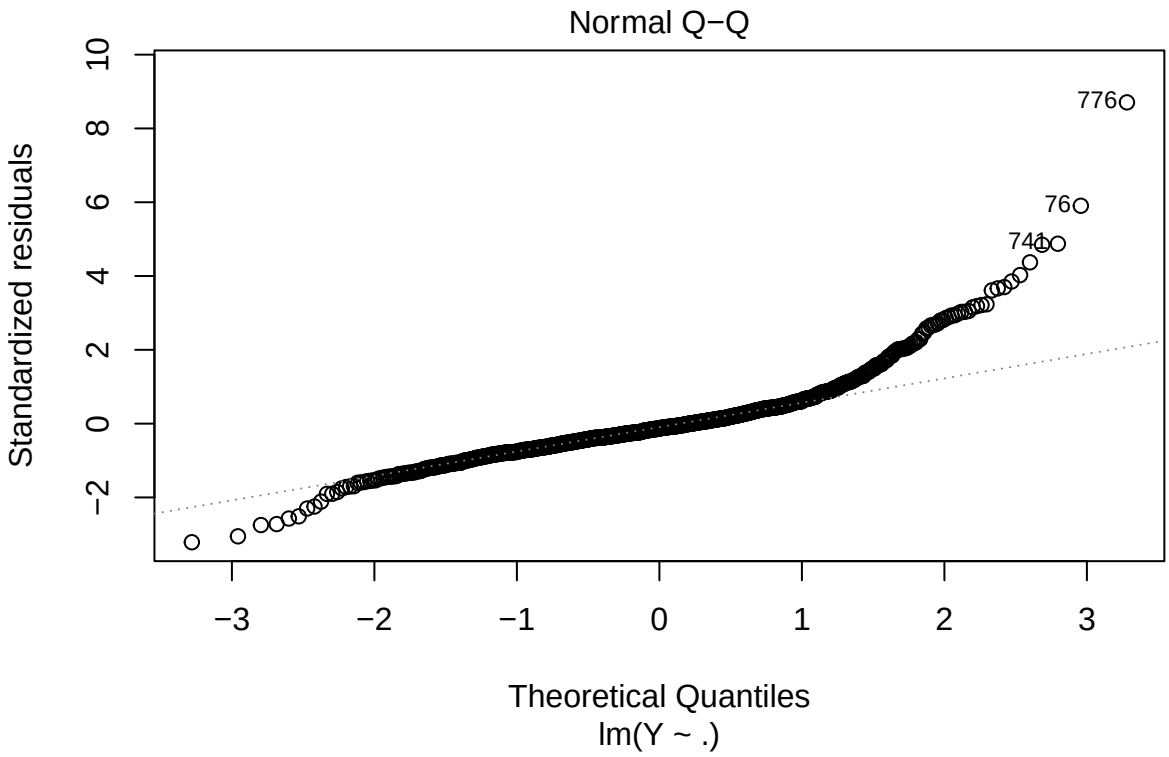
Homoscedasticity It observes a slight increase in the spread of residuals followed by a relatively constant spread around 1 (in the range of -0.5 to 1). The dataset still abide to homoscedasticity assumption. The residuals have a roughly constant spread around zero with few data been flagged (741,76 and 776).

```
lm_1 <- lm(Y~.,data = data)
plot(lm_1, which = 3)
```



Normality of residuals. The normality assumption is evaluated based on the residuals and can be evaluated using a QQ-plot by comparing the residuals to "ideal" normal observations along the 45-degree line. It has flagged those same 3 data points that have large residuals (observations 741,76 and 776). However, aside from those 3 data points, observations lie well along the 45-degree line in the QQ-plot, so we may assume that normality holds here.

```
lm_1 <- lm(Y~.,data = data)
plot(lm_1, which = 2)
```



10. Predict

To evaluate the model is performing well, new data point is given as input into the final model.

```
new_data <- tibble(X1 = -1,
                   X2= 0.5,
                   X3 = 2,
                   X4 =1,
                   C1 ="c",
                   C2 = "X")
data_fit %>%
  extract_fit_parsnip() %>%
  predict(data_recipe %>%
            prep() %>%
            bake(new_data=new_data))
```

```
## # A tibble: 1 x 1
##   .pred
##   <dbl>
## 1  4.32
```

Plotting the Predicted values against observed (Y) values.

```
data_fit %>% collect_predictions() %>%
  ggplot(aes(Y, .pred)) +
  geom_point() +
  geom_smooth(col = "#1f5f3f", method = "lm") +
  geom_abline(intercept = 0, slope = 1) +
  ggtitle("Predicted vs Observed") +
  labs(x = "Observed", y = "Predicted")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```