Predicting Concrete Strength to Support Better Material and Budget Decisions

2.1 Introduction

2.1.1 Scenario Summary

The construction company is assessing a variety of concrete mix designs to gain a clearer understanding of how different material components influence compressive strength. In this context, the analyst's task (the task in question) is to apply statistical techniques—primarily regression analysis and hypothesis testing—to explore the relationships between mix composition and resulting strength. By identifying significant predictors and quantifying their effects, the analysis aims to support more informed decisions around mix optimisation, quality assurance, and cost-efficient material planning.

2.1.2 Initial Data Exploration Findings

An initial review of the dataset shows that it contains 1,030 observations across nine continuous numerical variables, representing the key components of concrete mix design along with the resulting compressive strength. The structure is clean and consistent, with no categorical fields and no missing values, making it highly suitable for statistical modelling. Because all variables are numeric and measured on continuous scales, the dataset lends itself naturally to regression analysis, where relationships between material composition and strength can be explored without the need for data transformation or imputation at this stage. This provides a strong foundation for developing reliable models and conducting inferential statistical tests.

Next, we went on to install and load all necessary packages. The ggplot2 and dplyr packages form a powerful pairing for data analysis in R. *dplyr* provides an intuitive, grammar-based approach to data manipulation—allowing us to filter, summarise, transform, and organise data efficiently. The former (ggplot2) uses the grammar of graphics to build clear, flexible, and publication-quality visualisations.

```
# ---------- REGRESSION MODELS FOR CONCRETE STRENGTH -------------


# installing and loading necessary libraries
install.packages("corrplot")
install.packages("readxl") # Had to install this necessary library for reading exc
                           # were saved in excel format instead of csv
install.packages('ggplot2')
install.packages('dplyr')
install.packages('car') # for vif function. comes with the car package
install.packages('caret')
```

```r
# load packages
library(readxl)
library(corrplot)
library(ggplot2) # get documentation for usage
library(dplyr)
library(car)
library(caret)
```

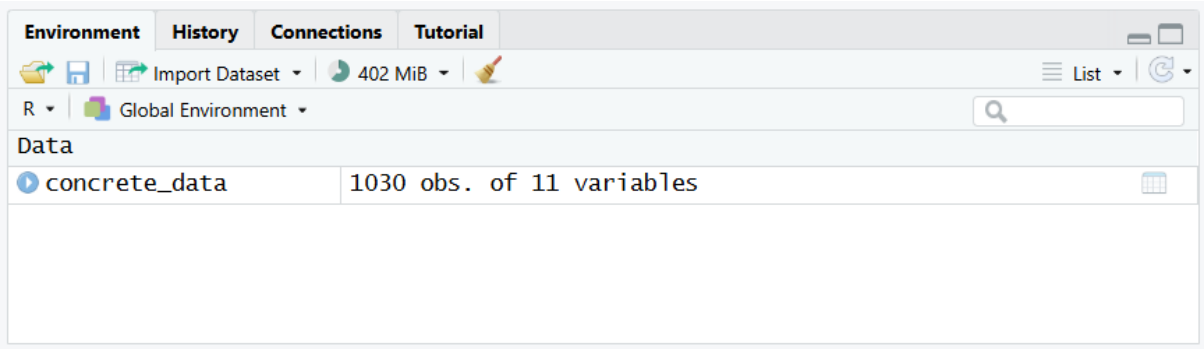
The dataset was then loaded using the readxl package:

```r
# Load dataset
concrete_data = read_excel('concrete compressive strength.xlsx')
```

And then we previewed the dataset.

```
> head(concrete_data)
# A tibble: 6 × 11
  Cement (component 1)(kg …¹ Blast Furnace Slag (…² Fly Ash (component 3…³ Water  (component 4)…⁴
                       <dbl>                  <dbl>                  <dbl>                  <dbl>
1                        540                      0                      0                    162
2                        540                      0                      0                    162
3                        332.                   142.                     0                    228
4                        332.                   142.                     0                    228
5                        199.                   132.                     0                    192
6                        266                    114                      0                    228
# ℹ abbreviated names: ¹`Cement (component 1)(kg in a m^3 mixture)`
```

Could also be viewed from the environment pane at the top right



The column names were also previewed holistically. It's best practice to familiarise oneself with them for transformation and cleaning purposes.

```r
# view columns
colnames(concrete_data) # long names
```

```
R 4.5.2 · C:/Users/brigh/OneDrive - University of Salford/ASDV Workshop/Assessment/Task 2/
 [1] "Cement (component 1)(kg in a m^3 mixture)"
 [2] "Blast Furnace Slag (component 2)(kg in a m^3 mixture)"
 [3] "Fly Ash (component 3)(kg in a m^3 mixture)"
 [4] "Water  (component 4)(kg in a m^3 mixture)"
 [5] "Superplasticizer (component 5)(kg in a m^3 mixture)"
 [6] "Coarse Aggregate  (component 6)(kg in a m^3 mixture)"
 [7] "Fine Aggregate (component 7)(kg in a m^3 mixture)"
 [8] "Age (day)"
 [9] "Concrete Category"
[10] "Contains Fly Ash"
[11] "Concrete compressive strength(MPa, megapascals)"
>
```

It seemed appropriate to rename the columns since they were just so long.

```
# Renaming columns
concrete_data <- concrete_data %>%
  rename(
    cement = `Cement (component 1)(kg in a m^3 mixture)`,
    slag = `Blast Furnace Slag (component 2)(kg in a m^3 mixture)`,
    flyAsh = `Fly Ash (component 3)(kg in a m^3 mixture)`,
    water = `Water  (component 4)(kg in a m^3 mixture)`,
    superPlasticizer = `Superplasticizer (component 5)(kg in a m^3 mixture)`,
    coarseAgg = `Coarse Aggregate  (component 6)(kg in a m^3 mixture)`,
    fineAgg = `Fine Aggregate (component 7)(kg in a m^3 mixture)`,
    age = `Age (day)`,
    concrete_category= `Concrete Category`,
    isFlyAsh = `Contains Fly Ash`,
    concrete_strength = `Concrete compressive strength(MPa, megapascals)`
  )
```

The column names were more concise and memorable now:

```
 [1] "cement"            "slag"              "flyAsh"            "water"
 [5] "superPlasticizer"  "coarseAgg"         "fineAgg"           "age"
 [9] "concrete_category" "isFlyAsh"          "concrete_strength"
>
```

A quick five-number summary snapshot was also made on the dataset:

```
> summary(concrete_data)
     cement           slag          flyAsh           water       superPlasticizer
 Min.   :102.0   Min.   :  0.0   Min.   :  0.00   Min.   :121.8   Min.   : 0.000
 1st Qu.:192.4   1st Qu.:  0.0   1st Qu.:  0.00   1st Qu.:164.9   1st Qu.: 0.000
 Median :272.9   Median : 22.0   Median :  0.00   Median :185.0   Median : 6.350
 Mean   :281.2   Mean   : 73.9   Mean   : 54.19   Mean   :181.6   Mean   : 6.203
 3rd Qu.:350.0   3rd Qu.:142.9   3rd Qu.:118.27   3rd Qu.:192.0   3rd Qu.:10.160
 Max.   :540.0   Max.   :359.4   Max.   :200.10   Max.   :247.0   Max.   :32.200

   coarseAgg         fineAgg           age         concrete_category   isFlyAsh
 Min.   : 801.0   Min.   :594.0   Min.   :  1.00   Length:1030        Mode :logical
 1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00   Class :character   FALSE:566
 Median : 968.0   Median :779.5   Median : 28.00   Mode  :character   TRUE :464
 Mean   : 972.9   Mean   :773.6   Mean   : 45.66
 3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00
 Max.   :1145.0   Max.   :992.6   Max.   :365.00
 concrete_strength
 Min.   : 2.332
 1st Qu.:23.707
 Median :34.443
 Mean   :35.818
 3rd Qu.:46.136
 Max.   :82.599
>
```

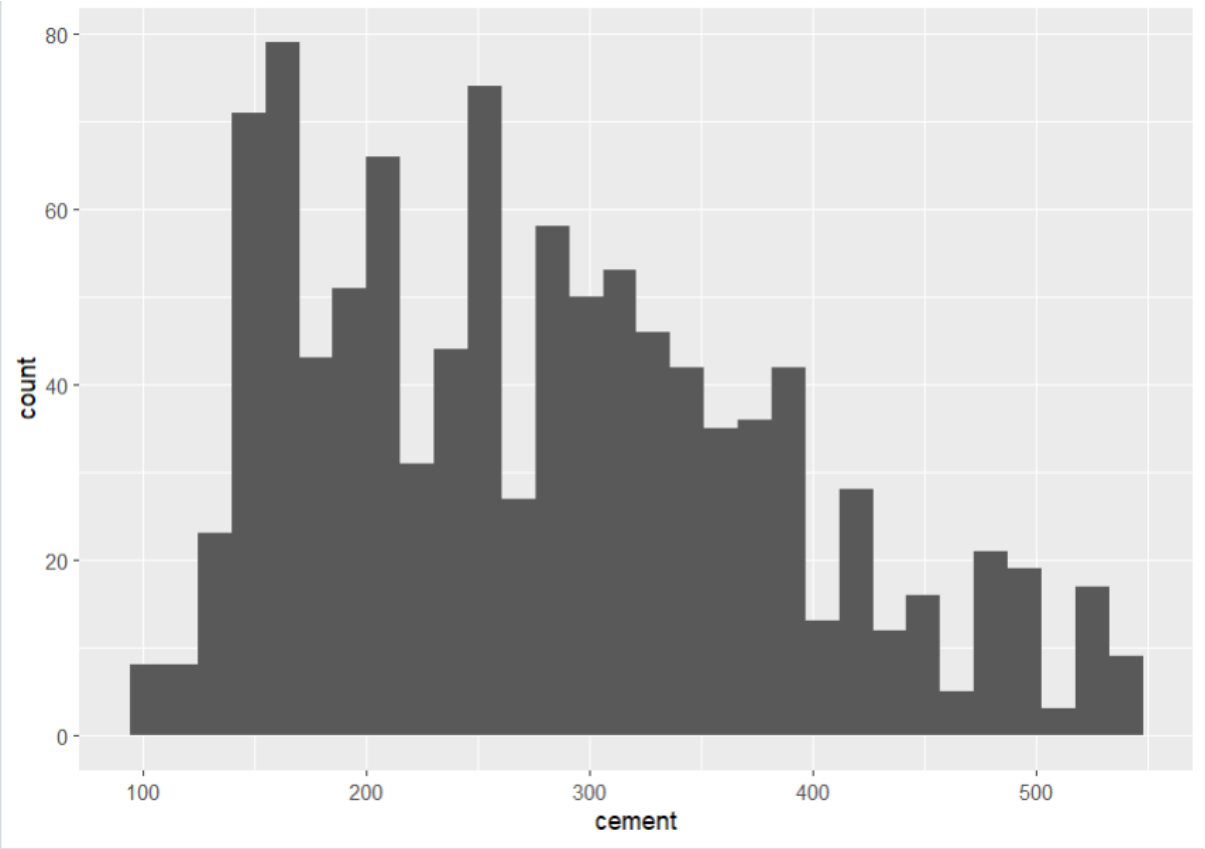
The dataset was checked for missing values as should be done always:

```
> colSums(is.na(concrete_data)) # CHECK FOR MISSING VALUES
          cement             slag            flyAsh            water  superPlasticizer
               0                0                 0                0                 0
       coarseAgg          fineAgg              age concrete_category         isFlyAsh
               0                0                 0                0                 0
concrete_strength
               0
>
```

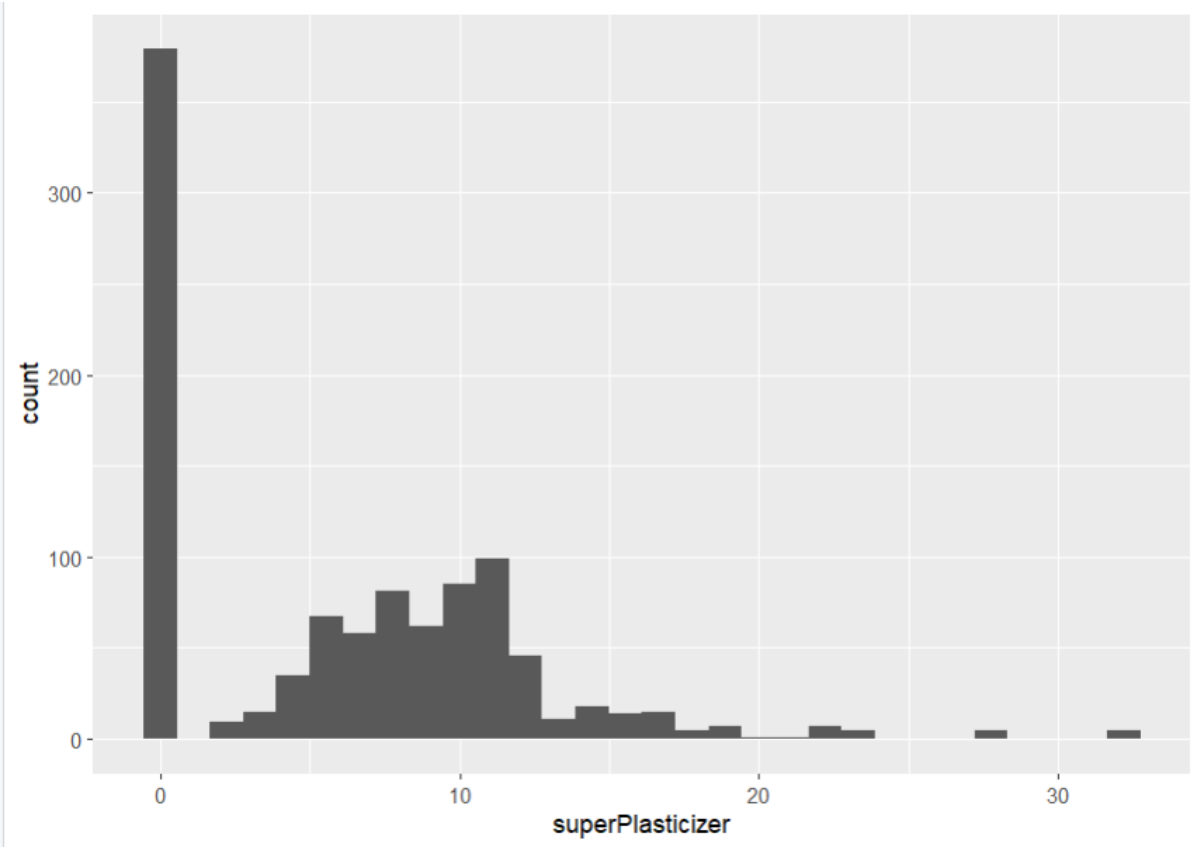
This confirmed that the concrete dataset had no missing values.

After this, key distribution of variables were checked:

```
> ggplot(concrete_data, aes(cement)) + geom_histogram(bins = 30)
>
```

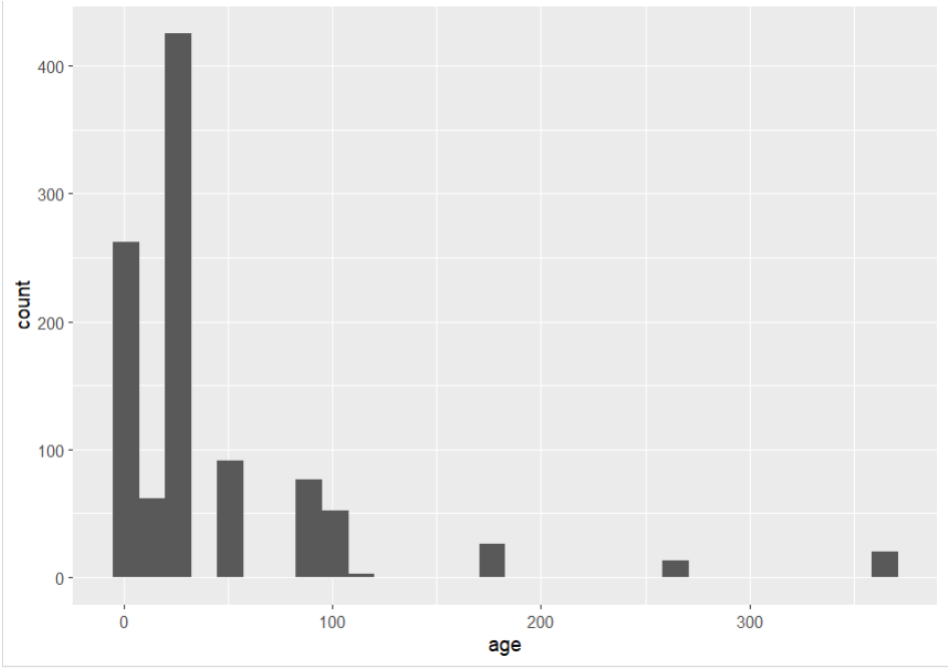



```
> ggplot(concrete_data, aes(superPlasticizer)) + geom_histogram(bins = 30)
>
```



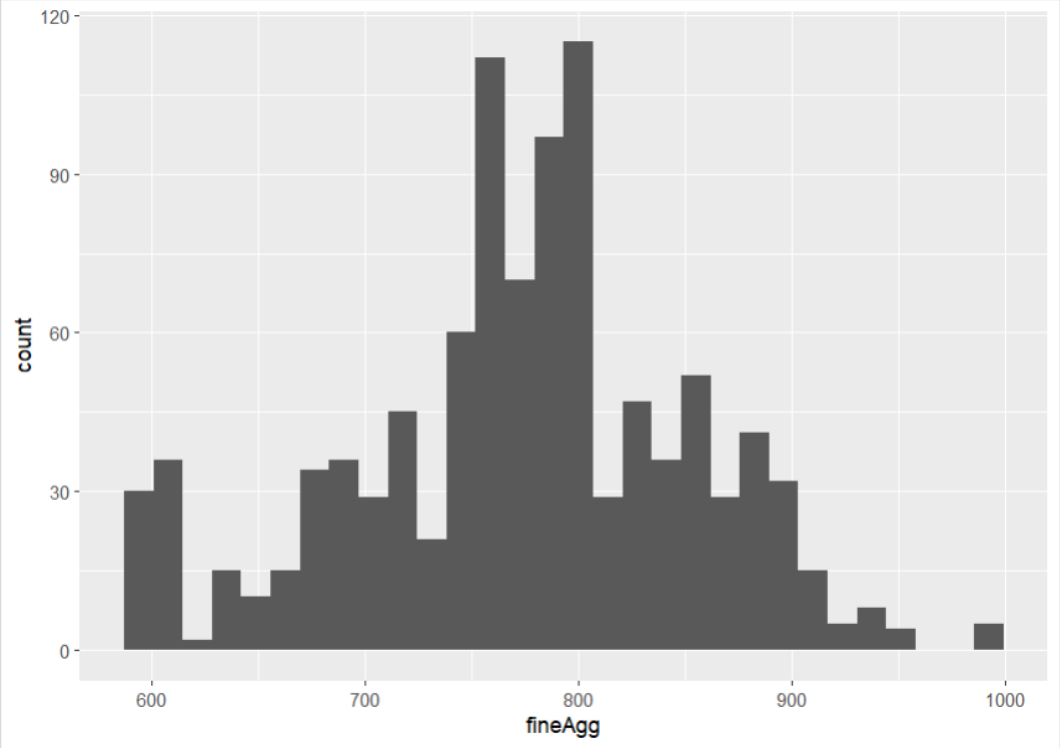
This shows that there is a lot of mix compositions that do not have this component.

```
> ggplot(concrete_data, aes(age)) + geom_histogram(bins = 30)
>
```



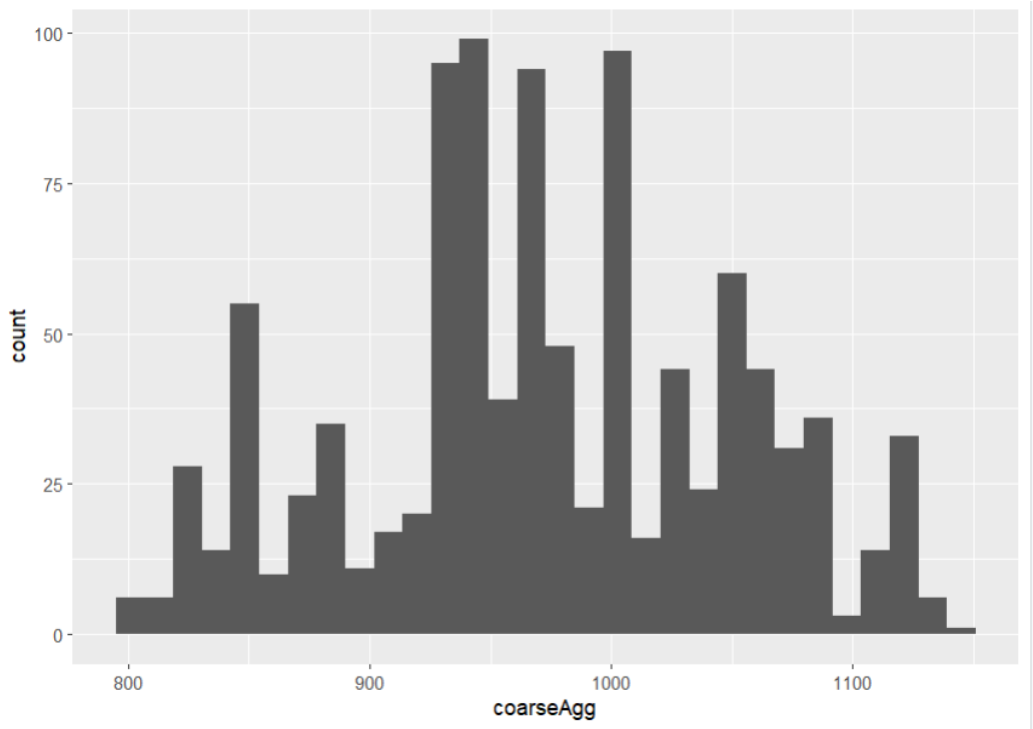
This shows segmentation in distribution

```
> ggplot(concrete_data, aes(fineAgg)) + geom_histogram(bins = 30)
>
```



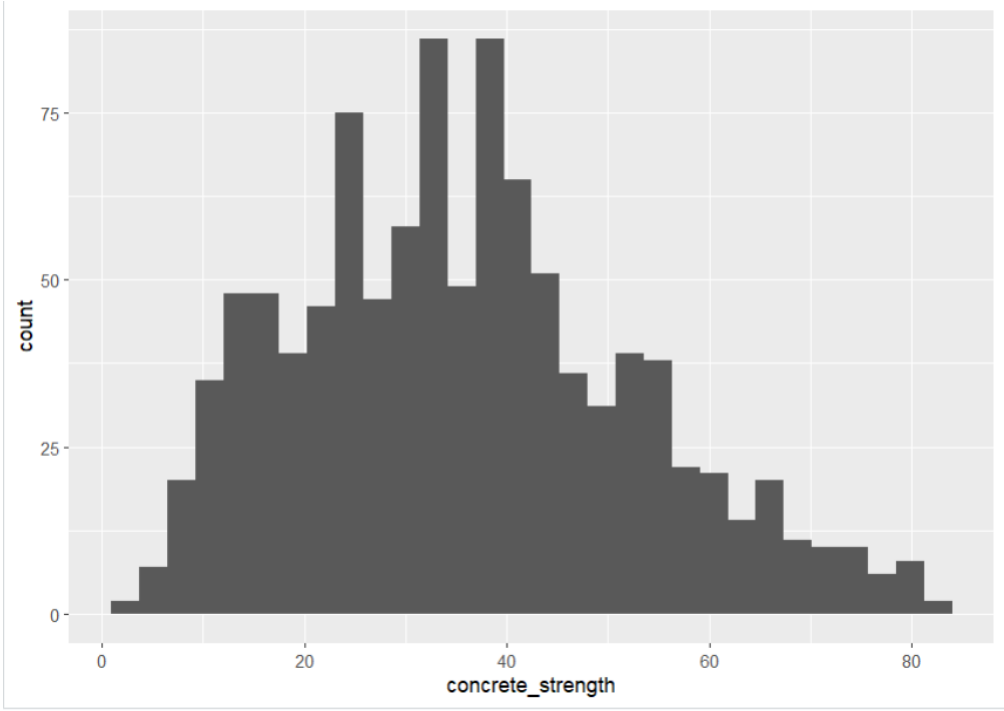
Fairly normal distribution. There is a noticeable concentration of values between roughly 750 and 850 kg, indicating this is the most common range used in the mix designs.

```
> ggplot(concrete_data, aes(coarseAgg)) + geom_histogram(bins = 30)
>
```



Again, the coarseAgg variable is fairly normally distributed

```
> ggplot(concrete_data, aes(concrete_strength)) +
+   geom_histogram(bins = 30)
>
```

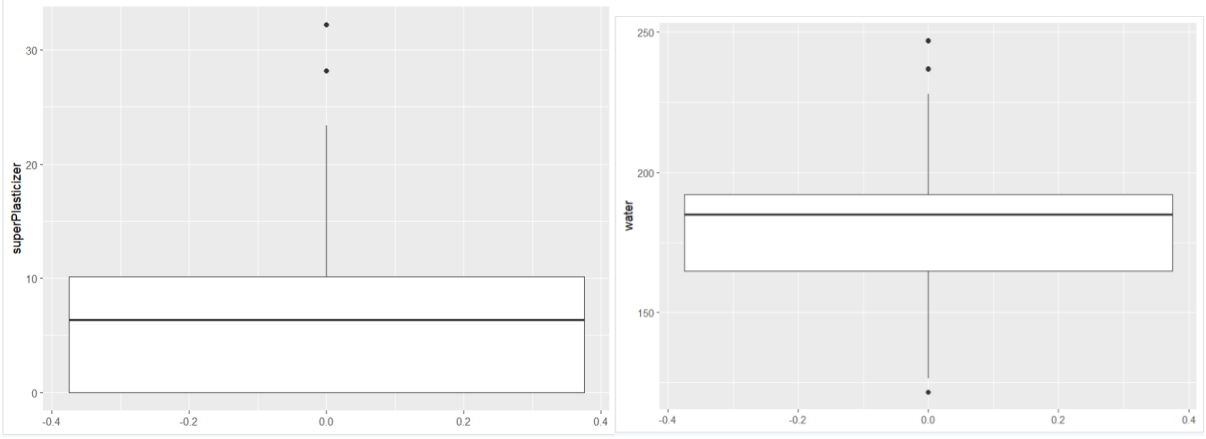


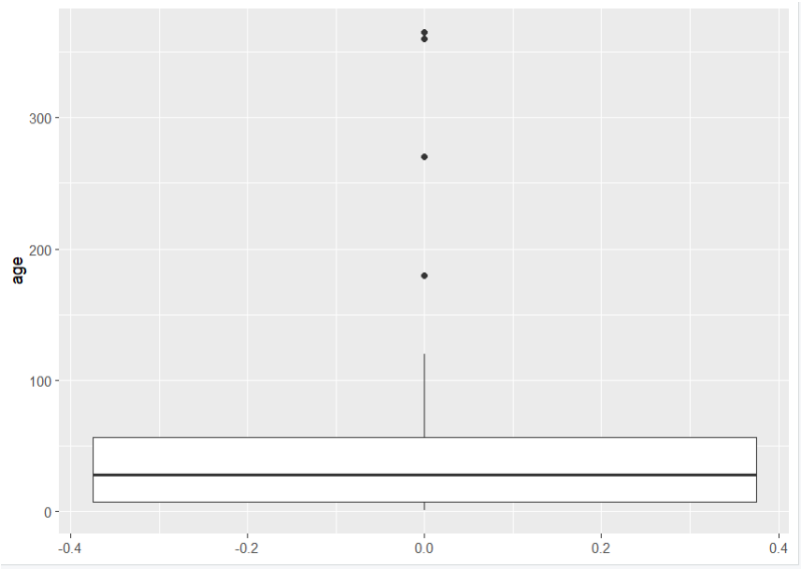
Normally distributed with slight taper on the right-hand side.

Another useful check which was carried out was outlier check

```
# Outliers?
ggplot(concrete_data, aes(y = cement)) + geom_boxplot()
ggplot(concrete_data, aes(y = water)) + geom_boxplot()
ggplot(concrete_data, aes(y = age)) + geom_boxplot()
ggplot(concrete_data, aes(y = concrete_strength)) +
  geom_boxplot()
```

Notable mentions were water, age and superPlasticizer:





Age seems the one with the most significant. Overall, the outlier datapoints seemed few and so we proceeded.

And finally, our objective was defined

```
# ------------------------------------------------------------------------
# Define Objective
# ------------------------------------------------------------------------

# We want to examine the possible linear relation between concrete strength and
# more than one independent variable (predictors)
```

2.2 Implementation of Models

2.2.1 Multiple Linear Regression (MLR) Model

Regression analysis with multiple features was first given attention to.

We started out isolating only variables relevant to the regression model we plan to develop – namely the numerical ones

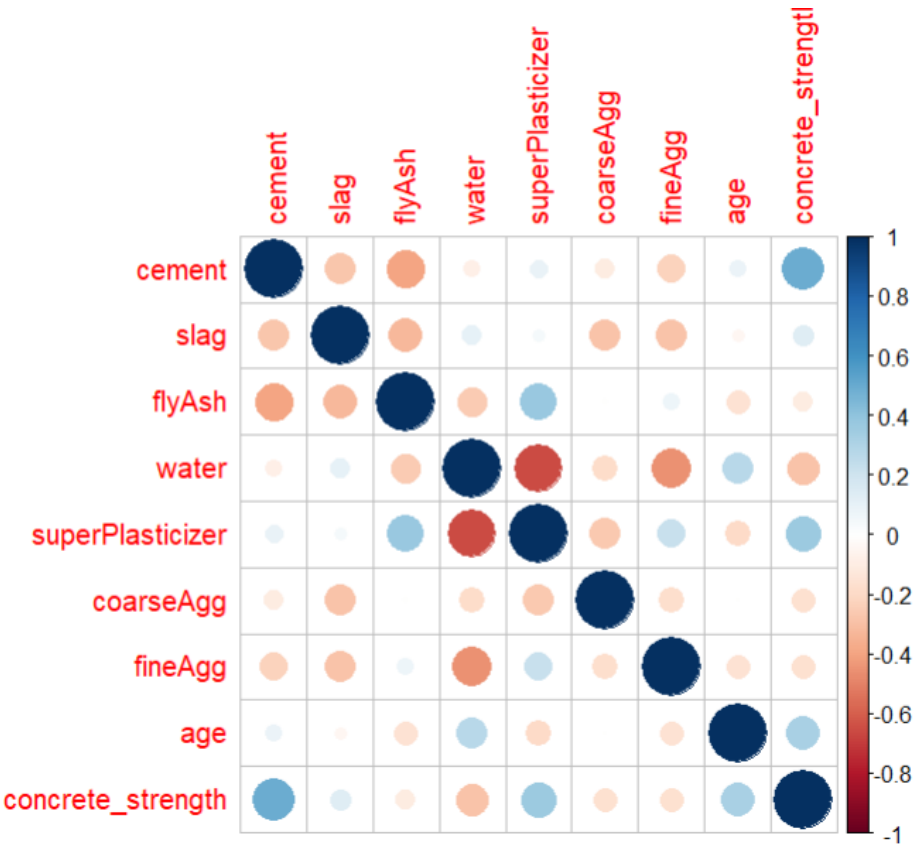
```
# ------------------------------------------------------------------------
# Performing Regression Analysis
# ------------------------------------------------------------------------

concrete_reduced <- concrete_data[, c("cement", "slag", "flyAsh", "water",
                                      "superPlasticizer", "coarseAgg",
                                      "fineAgg", "age", "concrete_strength")]
```

	cement	slag	flyAsh	water	superPlasticizer	coarseAgg	fineAgg	age	concrete_strength
1	540.0	0.0	0	162.0	2.5	1040.0	676.0	28	79.986111
2	540.0	0.0	0	162.0	2.5	1055.0	676.0	28	61.887366
3	332.5	142.5	0	228.0	0.0	932.0	594.0	270	40.269535
4	332.5	142.5	0	228.0	0.0	932.0	594.0	365	41.052780
5	198.6	132.4	0	192.0	0.0	978.4	825.5	360	44.296075
6	266.0	114.0	0	228.0	0.0	932.0	670.0	90	47.029847
7	380.0	95.0	0	228.0	0.0	932.0	594.0	365	43.698299
8	380.0	95.0	0	228.0	0.0	932.0	594.0	28	36.447770
9	266.0	114.0	0	228.0	0.0	932.0	670.0	28	45.854291
10	475.0	0.0	0	228.0	0.0	932.0	594.0	28	39.289790
11	198.6	132.4	0	192.0	0.0	978.4	825.5	90	38.074244
12	198.6	132.4	0	192.0	0.0	978.4	825.5	28	28.021684
13	427.5	47.5	0	228.0	0.0	932.0	594.0	270	43.012960
14	190.0	190.0	0	228.0	0.0	932.0	670.0	90	42.326932

The Boolean (isFlyAsh) and text (concrete_category) columns were no more.

```
> corrplot(cor(concrete_reduced))
>
```

Next, we created a correlation plot for our variables:



This highlighted four major correlated variables in;

- Cement (most correlated)
- Water
- Superplasticizer
- Age

Having them in mind, we began fitting our regression model using the forward stepwise method.

```
#forward stepwise fitting
model_0 <- lm(concrete_strength ~ cement, concrete_reduced)
summary.lm(model_0) # 24.71
```

This was the output summary of the model:

```
Call:
lm(formula = concrete_strength ~ cement, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-40.594 -10.952  -0.572   9.992  43.241

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.442795   1.296925   10.37   <2e-16 ***
cement       0.079580   0.004324   18.41   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.5 on 1028 degrees of freedom
Multiple R-squared:  0.2478,    Adjusted R-squared:  0.2471
F-statistic: 338.7 on 1 and 1028 DF,  p-value: < 2.2e-16

> |
```

The model was only able to capture 24.71% total variability in concrete strength which is poor because the higher the R^2 with significant Pr(>|t|), the better. The Pr(>|t|) value indicates the probability of observing a t-statistic as extreme as the one calculated if the null hypothesis were true and a very small value of it (< 0.05) is strong evidence that the predictor has a statistically significant effect on the outcome.

With this in mind, more highly correlated variables with concrete strength were added individually.

```
model_1 <- lm(concrete_strength ~ cement + superPlasticizer, concrete_reduced)
summary.lm(model_1) # 34.98


Call:
lm(formula = concrete_strength ~ cement + superPlasticizer, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-33.949 -10.032  -0.515   9.117  43.676

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     9.190242   1.250302    7.35 4.03e-13 ***
cement          0.074794   0.004036   18.53  < 2e-16 ***
superPlasticizer 0.902460   0.070603   12.78  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.47 on 1027 degrees of freedom
Multiple R-squared:  0.3511,    Adjusted R-squared:  0.3498
F-statistic: 277.8 on 2 and 1027 DF,  p-value: < 2.2e-16

> |
```

This one captured 34.98% of total variability. One more was added.

```
model_2 <- lm(concrete_strength ~ cement + superPlasticizer
              + age, concrete_reduced)
summary.lm(model_2) # 48.01
```

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    age, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-33.274  -8.077  -0.561   7.526  45.670

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     5.097652   1.146613   4.446 9.71e-06 ***
cement          0.068836   0.003628  18.976  < 2e-16 ***
superPlasticizer 1.111650   0.064459  17.246  < 2e-16 ***
age             0.097900   0.006090  16.077  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 12.05 on 1026 degrees of freedom
Multiple R-squared:  0.4816,    Adjusted R-squared:  0.4801
F-statistic: 317.8 on 3 and 1026 DF,  p-value: < 2.2e-16

> |
```

This captured 48.01% of total variability in concrete strength. The last highly correlated variable was added.

```
model_3 <- lm(concrete_strength ~ cement + superPlasticizer + water
              + age, concrete_reduced)
summary.lm(model_3) # 49.66
```

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + age, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-34.766  -8.182  -0.276   7.200  40.689

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     32.061564   4.716991   6.797 1.81e-11 ***
cement           0.067793   0.003574  18.969  < 2e-16 ***
superPlasticizer  0.803663   0.082218   9.775  < 2e-16 ***
water           -0.138258   0.023484  -5.887 5.32e-09 ***
age              0.105404   0.006126  17.205  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.85 on 1025 degrees of freedom
Multiple R-squared:  0.4986,    Adjusted R-squared:  0.4966
F-statistic: 254.8 on 4 and 1025 DF,  p-value: < 2.2e-16

> |
```

Captured close to 50% of total variability. Curiosity led to the addition of other slightly correlated variables to see if they would improve our model.

```
model_4 <- lm(concrete_strength ~ cement + superPlasticizer + water + fineAgg
              + age, concrete_reduced)
summary.lm(model_4) # 52.92
```

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + fineAgg + age, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-31.667  -8.170  -0.041   7.966  36.366

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      85.861032   7.812529  10.990   <2e-16 ***
cement            0.059050   0.003607  16.372   <2e-16 ***
superPlasticizer  0.736554   0.079905   9.218   <2e-16 ***
water            -0.229091   0.025109  -9.124   <2e-16 ***
fineAgg          -0.044491   0.005245  -8.483   <2e-16 ***
age               0.105076   0.005925  17.735   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.46 on 1024 degrees of freedom
Multiple R-squared:  0.5315,    Adjusted R-squared:  0.5292
F-statistic: 232.4 on 5 and 1024 DF,  p-value: < 2.2e-16
```

All variables are significant and they help the model capture 53% total variability.

coarseAgg was added next:

```
model_5 <- lm(concrete_strength ~ cement + superPlasticizer + water + fineAgg
              + coarseAgg + age, concrete_reduced)
summary.lm(model_5) # 56.81
```

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + fineAgg + coarseAgg + age, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-37.391  -7.261   0.227   7.305  38.002

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      203.078793  14.266100  14.235   <2e-16 ***
cement             0.049397   0.003596  13.735   <2e-16 ***
superPlasticizer   0.189394   0.095248   1.988    0.047 *
water             -0.419490   0.031107 -13.485   <2e-16 ***
fineAgg           -0.070657   0.005709 -12.377   <2e-16 ***
coarseAgg         -0.058044   0.006014  -9.651   <2e-16 ***
age                0.108883   0.005689  19.141   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 10.98 on 1023 degrees of freedom
Multiple R-squared:  0.5706,    Adjusted R-squared:  0.5681
F-statistic: 226.6 on 6 and 1023 DF,  p-value: < 2.2e-16

> |
```

This captured 56.81% total variability. Yet another model was fitted, this time with the addition of the slag variable.

```
model_6 <- lm(concrete_strength ~ cement + superPlasticizer + water + fineAgg
              + coarseAgg + slag + age, concrete_reduced)
summary.lm(model_6) # 59.43

Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + fineAgg + coarseAgg + slag + age, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-30.941  -7.234   0.439   6.912  34.405

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     121.781880  17.018910   7.156 1.59e-12 ***
cement            0.067615   0.004135  16.353  < 2e-16 ***
superPlasticizer  0.370142   0.094912   3.900 0.000103 ***
water            -0.323634   0.032339 -10.007  < 2e-16 ***
fineAgg          -0.038587   0.006778  -5.693 1.63e-08 ***
coarseAgg        -0.027560   0.006915  -3.985 7.22e-05 ***
slag              0.042530   0.005191   8.192 7.60e-16 ***
age               0.109748   0.005514  19.903  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.64 on 1022 degrees of freedom
Multiple R-squared:  0.5971,    Adjusted R-squared:  0.5943
F-statistic: 216.3 on 7 and 1022 DF,  p-value: < 2.2e-16

> |
```

This improved the model a bit more to enable it capture 59.43% total variability.

We tried added the least correlated flyash to the model:

```
model_7 <- lm(concrete_strength ~ cement + superPlasticizer + water + fineAgg
              + coarseAgg + slag + flyAsh + age, concrete_reduced)
summary.lm(model_7) # 61.25 -intercept, fine aggregate and coarse aggregate not


Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + fineAgg + coarseAgg + slag + flyAsh + age, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-28.653  -6.303   0.704   6.562  34.446
```

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      -23.163756  26.588421  -0.871 0.383851
cement             0.119785   0.008489  14.110  < 2e-16 ***
superPlasticizer   0.290687   0.093460   3.110 0.001921 **
water             -0.150298   0.040179  -3.741 0.000194 ***
fineAgg            0.020154   0.010703   1.883 0.059968 .
coarseAgg          0.018030   0.009394   1.919 0.055227 .
slag               0.103847   0.010136  10.245  < 2e-16 ***
flyAsh             0.087943   0.012585   6.988 5.03e-12 ***
age                0.114226   0.005427  21.046  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.4 on 1021 degrees of freedom
Multiple R-squared:  0.6155,    Adjusted R-squared:  0.6125
F-statistic: 204.3 on 8 and 1021 DF,  p-value: < 2.2e-16
```

Although this significantly increased R^2, fineAgg, coarseAgg and intercept were not statistically relevant (p-value > 0.05). Switching up the different variables was tried but did not it did not significantly improve our model. An example is shown below:

```
model_8 <- lm(concrete_strength ~ cement + superPlasticizer + water + fineAgg
              + coarseAgg + flyAsh + age, concrete_reduced)
summary.lm(model_8) # 57.3
```

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + fineAgg + coarseAgg + flyAsh + age, data = concrete_reduced)

Residuals:
    Min      1Q  Median      3Q     Max
-36.083  -7.543   0.116   7.433  36.594

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      210.546339  14.336933  14.686  < 2e-16 ***
cement             0.042423   0.004072  10.418  < 2e-16 ***
superPlasticizer   0.280937   0.098093   2.864 0.004269 **
water             -0.428950   0.031042 -13.818  < 2e-16 ***
fineAgg           -0.074024   0.005754 -12.866  < 2e-16 ***
coarseAgg         -0.058486   0.005981  -9.778  < 2e-16 ***
flyAsh            -0.023675   0.006613  -3.580 0.000359 ***
age                0.108013   0.005661  19.080  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.92 on 1022 degrees of freedom
Multiple R-squared:  0.5759,    Adjusted R-squared:  0.573
F-statistic: 198.3 on 7 and 1022 DF,  p-value: < 2.2e-16

>
```

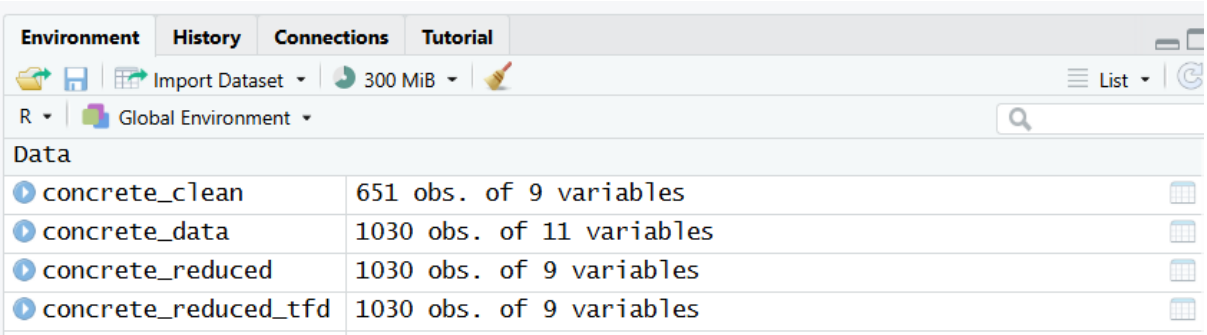
One final thing that was tired though was to log transform the variables with few outliers and segmented distributions, to see if there will improvements. This was how we achieved it.

```
# predictor transformation
# log transforming age and superPlasticizer predictors
concrete_reduced_tfd <- concrete_reduced # create a new df for transformation ops
concrete_reduced_tfd$superPlasticizer <- log(concrete_reduced_tfd$superPlasticizer)
concrete_reduced_tfd$age <- log(concrete_reduced_tfd$age)
head(concrete_reduced_tfd) # contains -ve infinite values in 5th column
# handling -ve infinte values
concrete_reduced_tfd[sapply(concrete_reduced_tfd, is.infinite)] <- NA
concrete_clean <- na.omit(concrete_reduced_tfd)
```

After transformation, it was noticed that the superplasticizer column had some infinite values (from applying log to zero values in the column). So, we replaced them with N/A and omitted rows that had them. This however, significantly reduced the observations we went on to work with.

```
> head(concrete_reduced_tfd) # contains -ve infinite values in 5th column
# A tibble: 6 × 9
  cement  slag flyAsh water superPlasticizer coarseAgg fineAgg   age concrete_strength
   <dbl> <dbl>  <dbl> <dbl>            <dbl>     <dbl>   <dbl> <dbl>            <dbl>
1    540     0      0   162            0.916      1040     676  3.33             80.0
2    540     0      0   162            0.916      1055     676  3.33             61.9
3    332.  142.     0   228           -Inf         932     594  5.60             40.3
4    332.  142.     0   228           -Inf         932     594  5.90             41.1
5    199.  132.     0   192           -Inf         978.    826.  5.89             44.3
6    266   114      0   228           -Inf         932     670  4.50             47.0
>
```

```
# A tibble: 6 × 9
  cement  slag flyAsh water superPlasticizer coarseAgg fineAgg   age concrete_strength
   <dbl> <dbl>  <dbl> <dbl>            <dbl>     <dbl>   <dbl> <dbl>            <dbl>
1    540     0      0   162            0.916      1040     676  3.33             80.0
2    540     0      0   162            0.916      1055     676  3.33             61.9
3    332.  142.     0   228           NA          932     594  5.60             40.3
4    332.  142.     0   228           NA          932     594  5.90             41.1
5    199.  132.     0   192           NA          978.    826.  5.89             44.3
6    266   114      0   228           NA          932     670  4.50             47.0
>
```



379 rows were lost unfortunately, but since they were all zero values, it is still satisfactory.

Another model was fitted with the new transformed dataset and the results proved that it was worth it.

```
model_9 <- lm(concrete_strength ~ cement + superPlasticizer + water
              + age + slag, concrete_clean)
summary.lm(model_9)# 81.35
```

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + age + slag, data = concrete_clean)

Residuals:
     Min       1Q   Median       3Q      Max
-22.8710  -4.9575  -0.2671   5.1530  25.1112

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     23.913945   4.235381   5.646 2.46e-08 ***
cement           0.097445   0.002929  33.271  < 2e-16 ***
superPlasticizer -2.545074   0.762155  -3.339 0.000888 ***
water           -0.237388   0.017417 -13.630  < 2e-16 ***
age              9.758547   0.283360  34.439  < 2e-16 ***
slag             0.068334   0.003673  18.606  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.217 on 645 degrees of freedom
Multiple R-squared:  0.8149,    Adjusted R-squared:  0.8135
F-statistic: 567.9 on 5 and 645 DF,  p-value: < 2.2e-16

> |
```

A whooping 81.35% of total variability captured by the model now, and with minimal features as well. Effort was still put in getting more improvements. Added more features like coarseAgg and fineAgg, but it didn’t improve the model significantly.

```
model_10 <- lm(concrete_strength ~ cement + superPlasticizer + water
               + age + slag + fineAgg, concrete_clean)
summary.lm(model_10) # doesn't improve model significantly
```

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + age + slag + fineAgg, data = concrete_clean)

Residuals:
     Min       1Q   Median       3Q      Max
-22.8934  -4.7969  -0.4484   4.9732  25.0495

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     34.208403   6.464445   5.292 1.66e-07 ***
cement           0.096598   0.002949  32.760  < 2e-16 ***
superPlasticizer -2.492532   0.760550  -3.277   0.0011 **
water           -0.251192   0.018569 -13.527  < 2e-16 ***
age              9.756139   0.282613  34.521  < 2e-16 ***
slag             0.065728   0.003867  16.999  < 2e-16 ***
fineAgg         -0.009686   0.004604  -2.104   0.0358 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.198 on 644 degrees of freedom
Multiple R-squared:  0.8162,    Adjusted R-squared:  0.8144
F-statistic: 476.5 on 6 and 644 DF,  p-value: < 2.2e-16

> |
```

```
model_11 <- lm(concrete_strength ~ cement + superPlasticizer + water
               + age + slag + fineAgg + coarseAgg, concrete_clean)
summary.lm(model_11) # doesn't improve model significantly

Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + age + slag + fineAgg + coarseAgg, data = concrete_clean)

Residuals:
     Min       1Q   Median       3Q      Max
-23.2000  -4.6757  -0.2944   4.7747  26.3312

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      11.916767  14.273318   0.835   0.4041
cement            0.100275   0.003616  27.730   <2e-16 ***
superPlasticizer -1.999039   0.809948  -2.468   0.0138 *
water            -0.220071   0.025682  -8.569   <2e-16 ***
age               9.760153   0.282170  34.590   <2e-16 ***
slag              0.069507   0.004423  15.716   <2e-16 ***
fineAgg          -0.003415   0.005827  -0.586   0.5581
coarseAgg         0.010016   0.005720   1.751   0.0804 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.186 on 643 degrees of freedom
Multiple R-squared:  0.817,     Adjusted R-squared:  0.815
F-statistic: 410.2 on 7 and 643 DF,  p-value: < 2.2e-16

> |
```

Barely any improvement and the last model contained some statistically insignificant variables. So, we stuck with **model_9** and moved to testing the statistic assumptions with it.

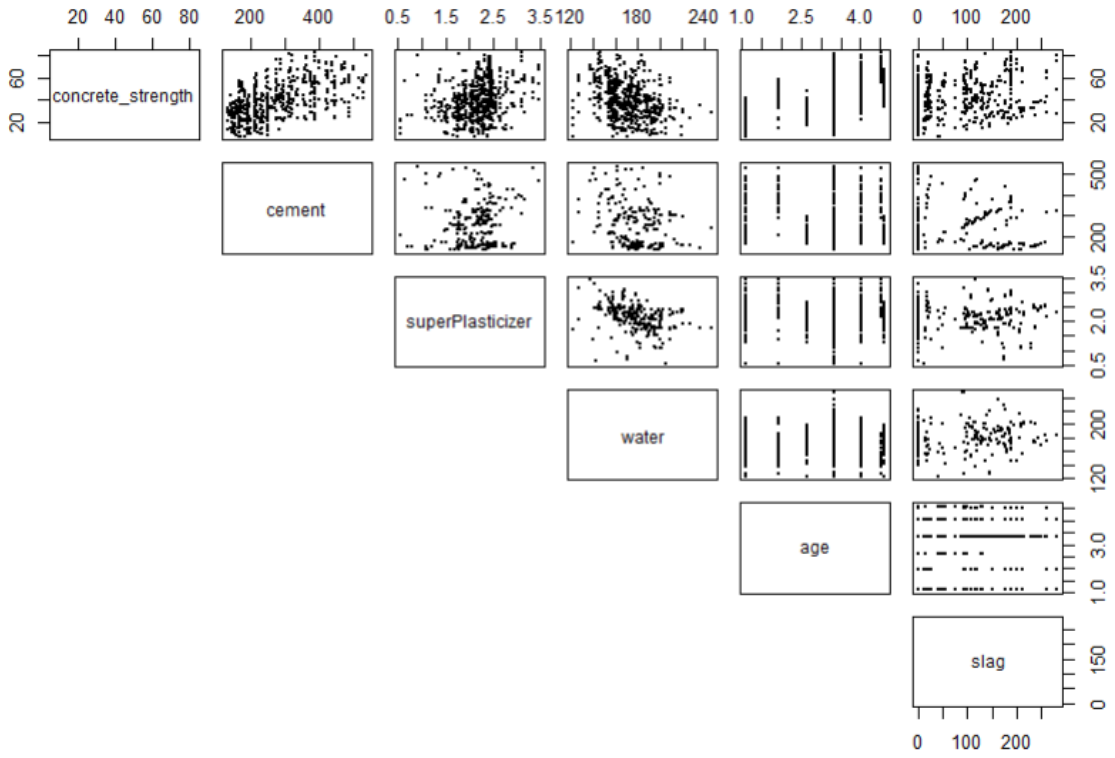
2.2.2 MLR Assumptions Testing

2.2.2.1 Linearity

```
# retrying linearity assumption
data.frame(colnames(concrete_clean))

  colnames.concrete_clean.
1                   cement
2                     slag
3                   flyAsh
4                    water
5          superPlasticizer
6                 coarseAgg
7                   fineAgg
8                      age
9         concrete_strength
> |
```

This will help us get the index of each column accurately

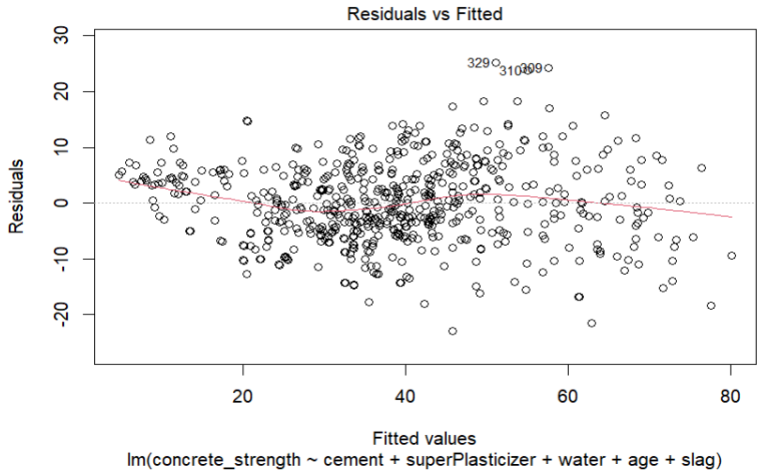
```
pairs(concrete_clean[,c(9,1,5,4,8,2)], lower.panel = NULL, pch = 19,cex = 0.2) # passed
```



Notable linearity between the variables. Cement and age (logged) show clear upward trends, indicating that higher cement content and longer curing times are both associated with stronger concrete. Water shows the opposite pattern, with strength generally decreasing as water content increases. For slag and superplasticizer the relationship is present but less clear from first glance.

2.2.2.2 Residuals Independence

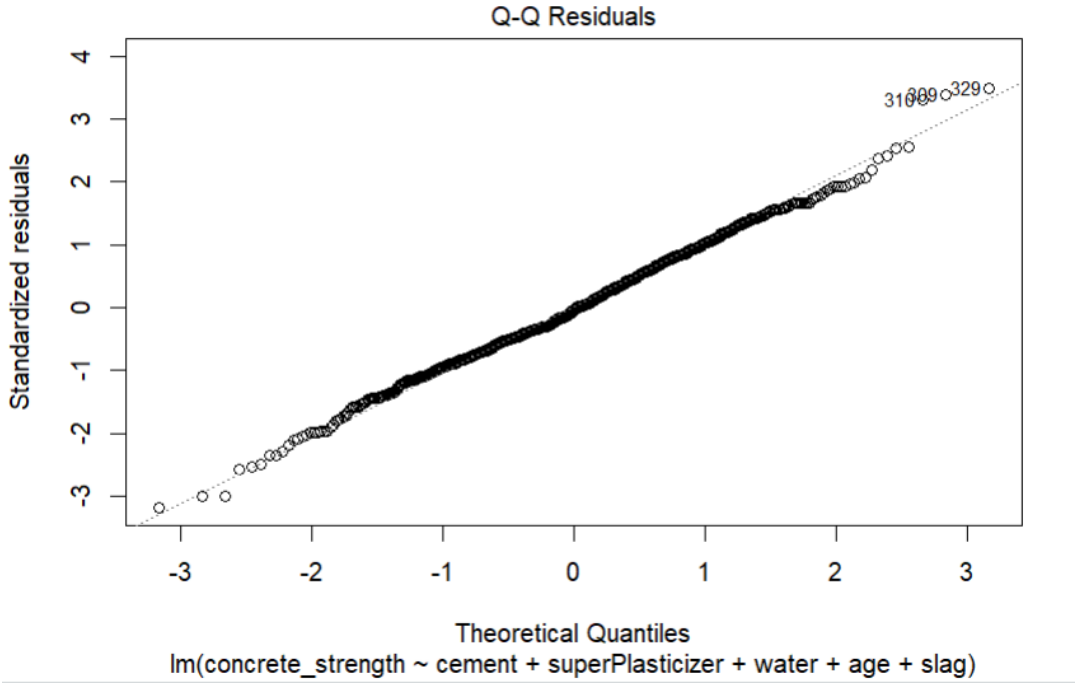
```
# 2. Residuals independence
plot(model_9, 1) # roughly lies on the zero line. passed
```



In this case, the residuals are mostly scattered around zero, which is what we want to see. The smooth red line is mostly flat but bends gently at the higher fitted values, hinting at a tiny degree of non-linearity that the model may not fully capture.

2.2.2.3 Normality of Residuals

```
# 3. normality of residuals
plot(model_9, 2) # all fitted to the diagonal. passed
```

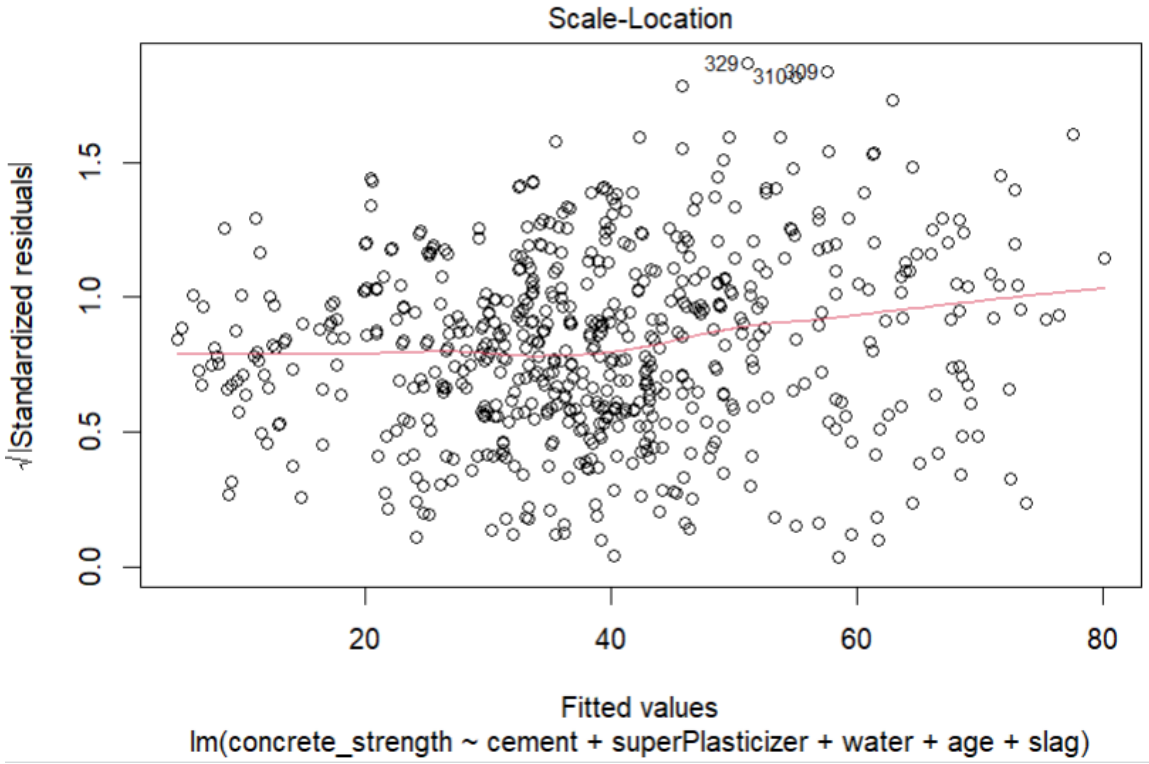


The plot shows the residuals falling almost perfectly along the straight reference line. This means that the residuals follow a roughly normal distribution with no major deviations at the tails. Because majority of the points closely track the line rather than curving away from it, there's no strong evidence of skewness or heavy-tailed behaviour.

So, this suggests that the normality assumption is well satisfied.

2.2.2.4 Homscedasticity

```
# 4. homoscedasticity
plot(model_9, 3) # randomly scattered around the red line. passed
```



The residuals-versus-fitted plot shows points that are randomly scattered around zero with no clear pattern, funnel shape, or systematic structure. It suggests that the assumption of homoscedasticity (constant variance of residuals) is being met.

When residuals behave like this, it means the model is not systematically over- or under-estimating the response for certain ranges of fitted values, which makes the regression estimates more trustworthy and the standard errors more accurate.

2.2.2.5 Multicollinearity

```
# 5. multicollinearity

vif(model_9) # all between 1 - 5. passed
```

```
> vif(model_9) # all between 1 - 5. passed
         cement superPlasticizer           water             age            slag
       1.119467         1.488695        1.449971        1.013149        1.054305
>
```

The model appears to have passed the multicollinearity check, with all VIF values falling well below the commonly used thresholds (typically 5 or 10). This indicates that the predictors are not excessively correlated with one another. In practical terms, it means each variable is contributing its own independent information to the model, and the coefficient estimates are stable and reliable.

Because multicollinearity is not a concern here, we can be more confident in interpreting the effects of each predictor without worrying that overlapping relationships are distorting the results.

2.2.3 Final MLR Model

```
Call:
lm(formula = concrete_strength ~ cement + superPlasticizer +
    water + age + slag, data = concrete_clean)

Residuals:
     Min       1Q   Median       3Q      Max
-22.8710  -4.9575  -0.2671   5.1530  25.1112

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     23.913945   4.235381   5.646 2.46e-08 ***
cement           0.097445   0.002929  33.271  < 2e-16 ***
superPlasticizer -2.545074   0.762155  -3.339 0.000888 ***
water           -0.237388   0.017417 -13.630  < 2e-16 ***
age              9.758547   0.283360  34.439  < 2e-16 ***
slag             0.068334   0.003673  18.606  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The final regression model for predicting concrete strength is:

$$\text{concrete\_strength} = 23.914 + 0.0974(\text{cement}) - 2.545(\ln(\text{superplasticizer})) - 0.2374(\text{water}) + 9.759(\ln(\text{age})) + 0.0683(\text{slag})$$

We used natural log (ln) in the final formula because we log transformed age and superplasticizer.

So if we have variable values as follows:

Cement	520
Superplasticizer	3.0
Water	125
Slag	200
Age	97

Concrete strength = 23.914 + (0.0974 * 520) – (2.545 * ln(3)) – (0.2374 * 125) + (9.759 * ln(97)) + (0.0683 * 200)

Concrete strength = 23.914+50.648−2.798−29.675+44.64+13.66

Concrete strength = **100.389 MPa**

2.2.4 Logistic Regression Model

As usual, the objective was first defined:

```
# ------------------------ Logistic Regression ----------------------------
# Define objective
# We want to fit a possible Logistic regression between isFlyAsh dependent variable
# and multiple possible independent variables. Basically, We want our possible model
# to correctly predict the likelihood that our concrete mix contains fly ash
```

First order of business was to create a new dataframe for our logistic regression procedures as a new structure was required. The new dataframe was also preprocessed for ease of model development.

```
# new df for logistic regression operations
concrete_lr <- concrete_data
concrete_lr$superPlasticizer <- log(concrete_lr$superPlasticizer)
concrete_lr$age <- log(concrete_lr$age)
# handling -ve infinte values
concrete_lr[sapply(concrete_lr, is.infinite)] <- NA
concrete_lr <- na.omit(concrete_lr)
```

Next was to encode the Boolean target column – isFlyAsh.

```
concrete_lr <- concrete_lr %>%
  mutate(isFlyAsh = ifelse(isFlyAsh == "FALSE", 0, 1))
# FALSE = 0 and TRUE = 1
```

The value of 1 was assigned to TRUE and 0 for FALSE.

lag	flyAsh	water	superPlasticizer	coarseAgg	fineAgg	age	concrete_category	isFlyAsh	concrete_strength
189.00	0.00	145.90	3.091042	944.70	755.80	4.510860	Fine	0	82.599225
189.00	0.00	164.90	2.451005	944.70	755.80	4.510860	Fine	0	79.296635
189.00	0.00	174.90	2.251292	944.70	755.80	4.510860	Fine	0	67.796175
0.00	96.67	189.29	1.495149	967.08	870.32	1.098612	Fine	1	11.576302
0.00	96.67	189.29	1.495149	967.08	870.32	2.639057	Fine	1	24.448819
0.00	96.67	189.29	1.495149	967.08	870.32	3.332205	Fine	1	24.890084
0.00	96.67	189.29	1.495149	967.08	870.32	4.025352	Fine	1	29.447520
0.00	96.67	189.29	1.495149	967.08	870.32	4.605170	Fine	1	40.713558
0.00	94.58	197.89	1.518857	947.04	852.16	1.098612	Fine	1	10.383509
0.00	94.58	197.89	1.518857	947.04	852.16	2.639057	Fine	1	22.139074

Showing 113 to 123 of 651 entries, 11 total columns

A model tried to be fitted using the backward stepwise method, starting first with all the independent variables.

```
model_logistic_1 <- glm(isFlyAsh ~ cement + superPlasticizer + water + flyAsh +
                          age + slag + fineAgg + coarseAgg + concrete_category
                        + concrete_strength, data = concrete_lr, family = "binomial")
summary(model_logistic_1) # perfect separation or quasi-complete separation problem
```

This was the output:

```
Call:
glm(formula = isFlyAsh ~ cement + superPlasticizer + water +
    flyAsh + age + slag + fineAgg + coarseAgg + concrete_category +
    concrete_strength, family = "binomial", data = concrete_lr)

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -5.979e+02  6.450e+05  -0.001    0.999
cement                 9.867e-02  4.630e+02   0.000    1.000
superPlasticizer       1.155e+01  3.258e+04   0.000    1.000
water                  4.669e-01  1.182e+03   0.000    1.000
flyAsh                 1.305e+00  7.409e+02   0.002    0.999
age                    2.159e+00  9.687e+03   0.000    1.000
slag                   6.358e-02  5.929e+02   0.000    1.000
fineAgg                2.681e-01  5.400e+02   0.000    1.000
coarseAgg              2.302e-01  2.201e+02   0.001    0.999
concrete_categoryFine -2.324e+01  1.042e+05   0.000    1.000
concrete_strength     -2.004e-01  6.109e+02   0.000    1.000
```

The model showed clear signs of perfect separation, which means the predictor variables can almost perfectly distinguish between the two outcome groups. This is reflected in the output by the extremely large standard errors, very small z-values, and p-values close to 1 for nearly all predictors. These patterns indicate that the maximum likelihood estimates cannot be reliably computed because the model struggles to converge on stable coefficient values.

Although the residual deviance is extremely low and the AIC appears small, these values are misleading under perfect separation—the model is effectively **overfitting** by assigning extreme coefficients rather than estimating meaningful effects. As a result, the coefficient estimates are not interpretable, and no conclusions should be drawn from this model, since it's not reliable.

Other models were fitted too, first without collinear predictors:

```
model_logistic_2 <- glm(isFlyAsh ~ cement + superPlasticizer + water +
                          age + slag + fineAgg + coarseAgg,
                        data = concrete_lr, family = "binomial") # remove collin
summary(model_logistic_2) # AIC 143.05
```

With output:

```
Call:
glm(formula = isFlyAsh ~ cement + superPlasticizer + water +
    age + slag + fineAgg + coarseAgg, family = "binomial", data = concrete_lr)

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      97.293765  13.894115   7.003 2.51e-12 ***
cement           -0.050873   0.006097  -8.343  < 2e-16 ***
superPlasticizer  0.956084   0.633728   1.509  0.13138
water            -0.065112   0.021802  -2.986  0.00282 **
age              -0.495199   0.286285  -1.730  0.08368 .
slag             -0.076690   0.008528  -8.992  < 2e-16 ***
fineAgg          -0.042630   0.006365  -6.697 2.13e-11 ***
coarseAgg        -0.031457   0.004939  -6.370 1.90e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 791.41  on 650  degrees of freedom
Residual deviance: 127.05  on 643  degrees of freedom
AIC: 143.05

Number of Fisher Scoring iterations: 8

>
```

Although AIC value was great (143.05), two variables (age and superplasticizer) were not statistically relevant.

Using the Imp() method available with the RandomForest package, we were able to see which features have the strongest influence on the model's predictions

```
Imp <- varImp(model_logistic_2, scale=FALSE)
Imp
```

```
                 Overall
cement           8.343380
superPlasticizer 1.508666
water            2.986473
age              1.729743
slag             8.992292
fineAgg          6.697110
coarseAgg        6.369528
>
```

It highlighted that the model relied most heavily on slag and cement levels to make predictions, with aggregate components also playing a meaningful role. Variables like age and superplasticizer still matter, but they influence the model far less compared to the major materials.

This was the other model fitted removing least influential variables (age and superplasticizer):

```
model_logistic_3 <- glm(isFlyAsh ~ cement + slag + fineAgg + water + coarseAgg,
                        data = concrete_lr, family = "binomial") # remove least
summary(model_logistic_3) # AIC 145.12
```

With output:

```
Call:
glm(formula = isFlyAsh ~ cement + slag + fineAgg + water + coarseAgg,
    family = "binomial", data = concrete_lr)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 99.014308  13.051799   7.586 3.29e-14 ***
cement      -0.049371   0.005826  -8.475  < 2e-16 ***
slag        -0.074949   0.008318  -9.011  < 2e-16 ***
fineAgg     -0.041243   0.006153  -6.703 2.04e-11 ***
water       -0.080696   0.018949  -4.259 2.06e-05 ***
coarseAgg   -0.031723   0.004718  -6.723 1.78e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 791.41  on 650  degrees of freedom
Residual deviance: 133.12  on 645  degrees of freedom
AIC: 145.12

Number of Fisher Scoring iterations: 8

>
```

The final logistic regression model fits so well and shows no signs of separation or instability, making it appropriate for interpretation. All five predictors are statistically significant. The model shows a strong reduction in deviance, meaning these variables collectively explain the majority of the difference between fly-ash and non–fly-ash concrete mixes.

Water variable which is now the least influential was removed to see if the model would improve:

```
model_logistic_4 <- glm(isFlyAsh ~ cement + slag + fineAgg + coarseAgg,
                        data = concrete_lr, family = "binomial") # keep most imp variables
summary(model_logistic_4) # AIC 165.69

Call:
glm(formula = isFlyAsh ~ cement + slag + fineAgg + coarseAgg,
    family = "binomial", data = concrete_lr)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 62.594752   7.744859   8.082 6.37e-16 ***
cement      -0.040721   0.004775  -8.528  < 2e-16 ***
slag        -0.070189   0.007868  -8.921  < 2e-16 ***
fineAgg     -0.027457   0.004252  -6.457 1.07e-10 ***
coarseAgg   -0.022824   0.003706  -6.160 7.30e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 791.41  on 650  degrees of freedom
Residual deviance: 155.69  on 646  degrees of freedom
AIC: 165.69

Number of Fisher Scoring iterations: 8

>
```

Although all variables are statistically relevant, the model wasn't improved because AIC increased from 145.12 to 165.69.

So the third model (model_logistic_3) was chosen as the final logistic regression model. With this, assumptions were now tested.

2.2.5 Assumptions Testing – Logistic Regression Model

2.2.5.1 Linearity

π values for each data row were calculated and added to the concrete_lr dataframe

```
# calculate pi values as probs
probs <- predict(model_logistic_3, data = concrete_lr, type = "response")
concrete_lr$probs <- probs
```

Then Logit (π) values were calculated for each π row and the result added to the concrete dataframe as a new column.

```
# calculate logit(pi) values as logits
logits <- log(probs/(1-probs))
concrete_lr$logits <- logits
```

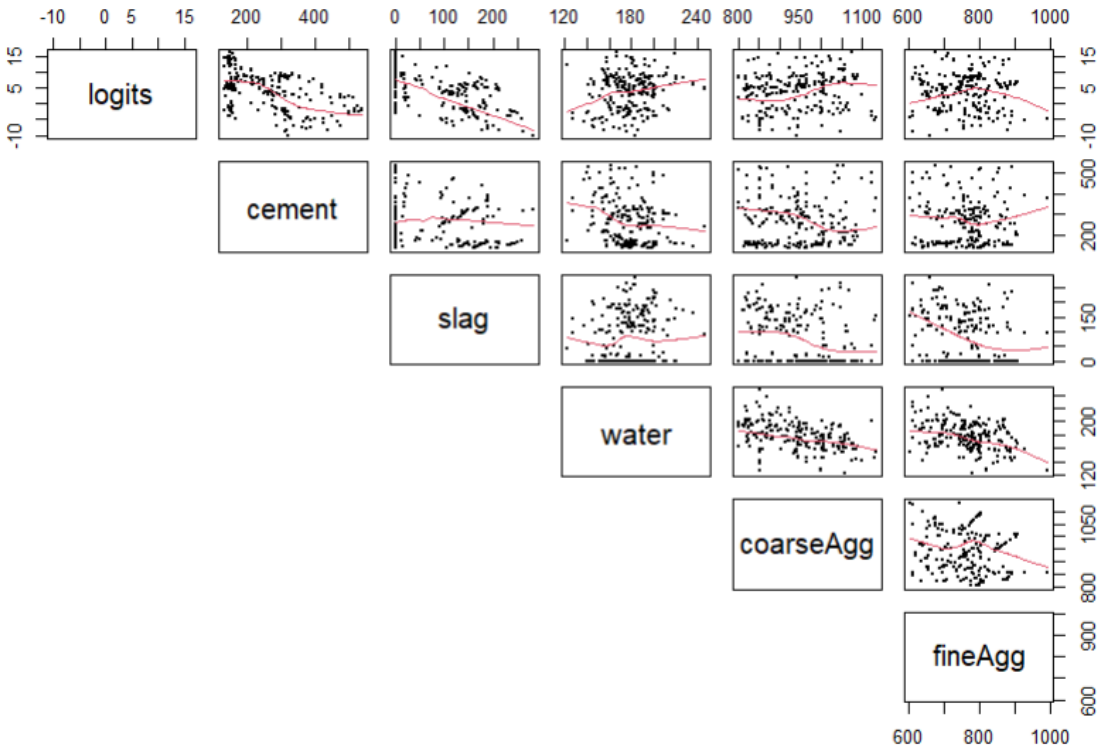
We confirmed this by checking the column names in our dataframe.

```
> data.frame(colnames(concrete_lr))
   colnames.concrete_lr.
1                 cement
2                   slag
3                flyAsh
4                  water
5       superPlasticizer
6              coarseAgg
7                fineAgg
8                    age
9      concrete_category
10              isFlyAsh
11     concrete_strength
12                  probs
13                 logits
> |
```

Then a scatterplot matrix was plotted based on the index c(13, 1, 2, 4, 6, 7):

```
pairs(concrete_lr[,c(13,1,2,4,6,7)], lower.panel = NULL, upper.panel = panel.smooth,
      pch = 19,cex = 0.2)
```

With output:

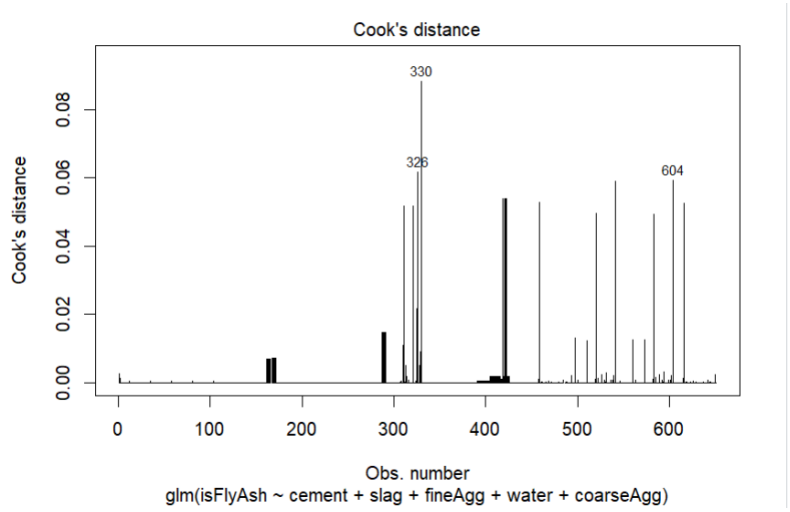


As already noted earlier, there significant approximate linear relationships between logits and the predictors in the model. This relationship is stronger with greater model influencers, like cement and slag and continuously weaker until the least influencer, which is water. It is satisfactory to say the model passed this assumption.

2.2.5.2 Influential Values

The Cook's distance plot was used to check this assumption.

```
# Influential values?
plot(model_logistic_3, which = 4, id.n = 3)
```



The Cook's distance plot showed that all observations had values well below the common cutoff of 1, indicating that no single data point had an unusually strong influence on the regression model. This means the model is not being driven or distorted by outliers or overly influential cases. In practical terms, the regression results can be considered stable and reliable because no single concrete mix is disproportionately affecting the estimated relationships. We can consider this assumption passed.

2.2.5.3 Multicollinearity

The VIF method was used in this regard:

```
> vif(model_logistic_3)
   cement      slag   fineAgg     water coarseAgg
 9.755235  6.456131  2.920276  2.081277  2.630970
> |
```

The VIF values for the reduced logistic model are all below 10 (ideally we want them below 5), indicating that multicollinearity is not a serious concern. Cement (VIF ≈ 9.76) and slag (VIF ≈ 6.46) show moderate correlation with other predictors, but still remain within acceptable limits. Overall, since > 50% of predictors pass the assumption, this result suggest that the predictors can be included together without destabilising the model or inflating the coefficient estimates.

2.2.6 Final Logistic Regression Model

This is the final model formula:

$$\text{logit}(\pi) = 99.014308 - 0.049371(\text{cement}) - 0.074949(\text{slag}) - 0.041243(\text{fineAgg}) - 0.080696(\text{water}) - 0.031723(\text{coarseAgg})$$

Where

$$\pi = P(\text{FlyAsh} = 1)$$

and

$$\pi = \frac{1}{1 + e^{-logit(\pi)}}$$

So if we have variable values as follows:

Cement	350
fineAgg	730
Water	170
Slag	150
coarseAgg	828

$$\text{logit}(\pi) = 99.014308 - 0.049371(350) - 0.074949(150) - 0.041243(730) - 0.080696(170) - 0.031723(828)$$

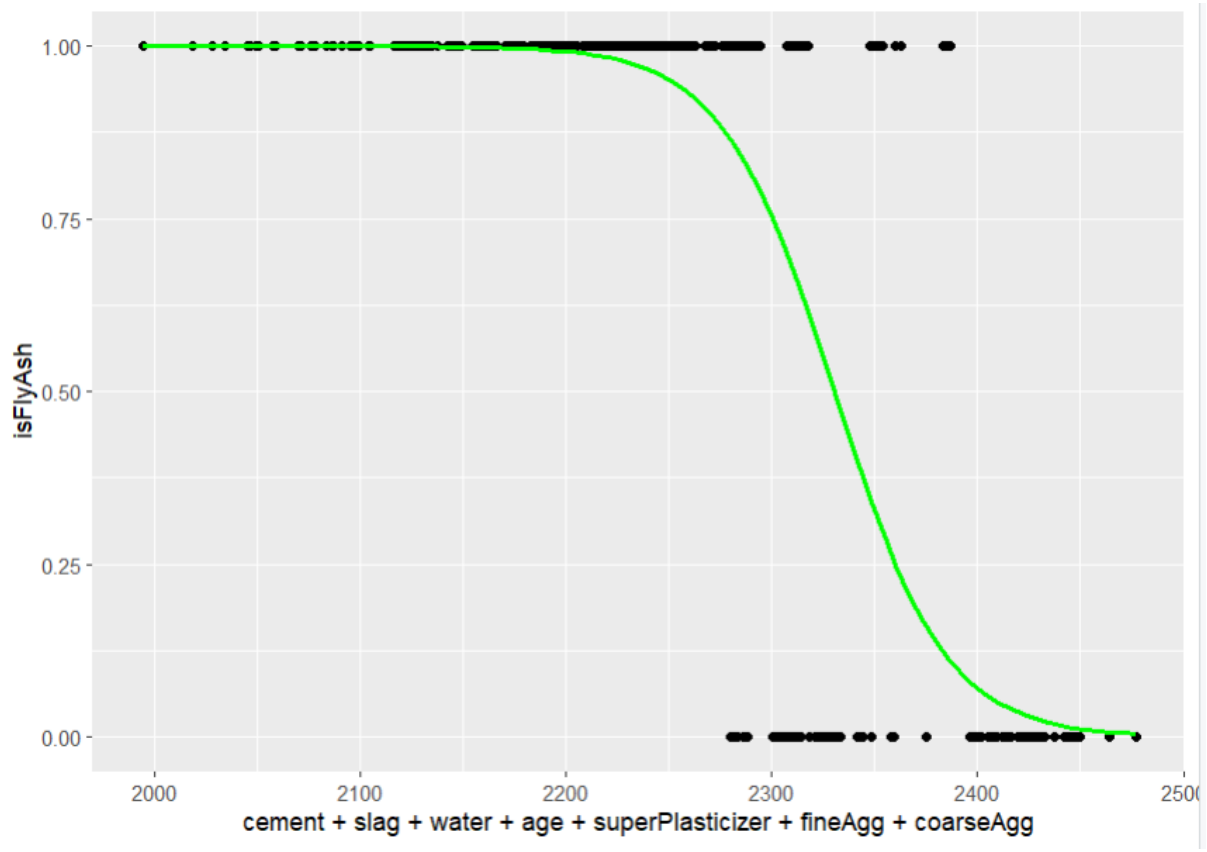
$$\text{logit}(\pi) = 0.3998$$

$$\pi \approx 0.5986$$

For this concrete mix, the model predicts about a *59.9% probability* that the concrete contains fly ash.

We can now plot the fitted logistical curve.

```
# plotting fitted regression curve
ggplot(concrete_lr, aes(x=cement+slag+water+age+superPlasticizer+fineAgg+coarseAgg, y=isFlyAsh
  stat_smooth(method="glm", color="green", se=FALSE,
              method.args = list(family=binomial))
```



2.3 Hypothesis Testing

The three main hypothesis tests we intended to carry out were:

- Does a concrete category being fine or coarse affect concrete strength?
- Does having fly ash in a concrete's composition mix affect concrete strength?
- Is there any association between concrete categories and the use of fly ash in a composition mix?

2.3.1 Concrete Strength vs Concrete Category

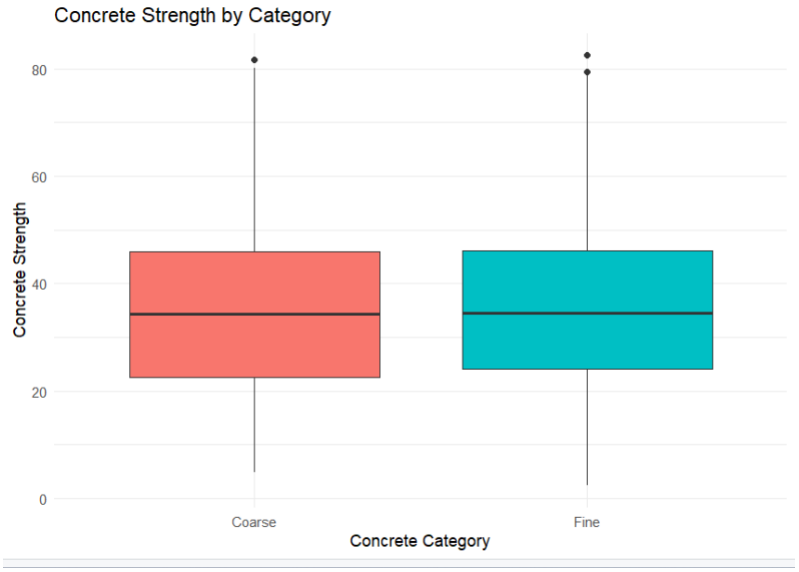
ANOVA Test was chosen to carry out this hypothesis test because our dataset structure of the columns in question perfectly passes half of the required assumptions for ANOVA (dependent variable is continuous, independent variable is categorical with two or more categories, and observations are independent of one another). The one-way test of ANOVA was to be used because only one independent variable was being considered.

This was our null and alternative hypothesis:

```
# H0: μcoarse = μfine
# H1: μcoarse ≠ μfine
# one way ANOVA p = 0.05
# Assumptions 1-3 passed by dataset design
```

As can be seen, our significance level was 5%. Assumption 4 was then tested.

```
# Assumption 4
ggplot(concrete_data, aes(x = concrete_category, y = concrete_strength, fill = concrete_cate
  geom_boxplot() +
  labs(title = "Concrete Strength by Category",
       x = "Concrete Category",
       y = "Concrete Strength") +
  theme_minimal() +
  theme(legend.position = "none")
```



The outliers aren't significant, so the assumption is satisfactorily passed.

```
# Assumption 5
# H0: the variable follows a normal distribution
# H1: the variable does NOT follow a normal distribution

install.packages("RVAideMemoire")   # install and load required library
library(RVAideMemoire)

byf.shapiro(concrete_strength ~ concrete_category, data=concrete_data) # assumption failed
```

```
        Shapiro-Wilk normality tests

data:  concrete_strength by concrete_category

             W   p-value
Coarse 0.9821 3.530e-06 ***
Fine   0.9712 3.101e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

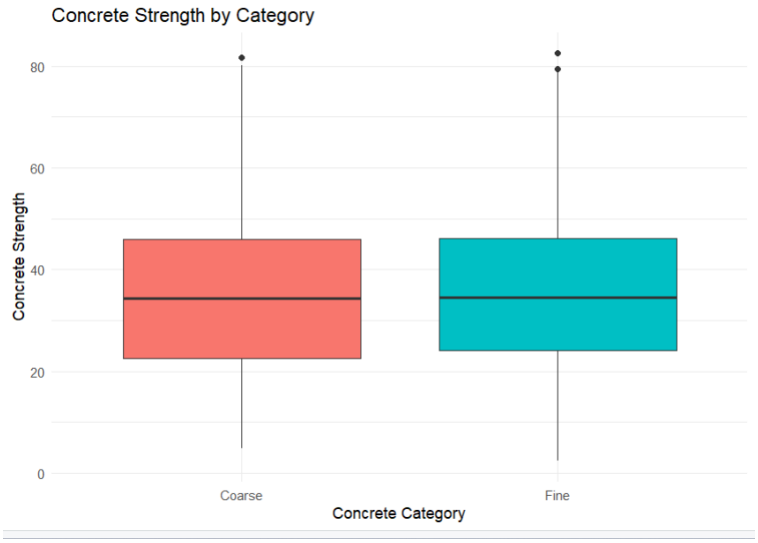
This assumption failed (p-value < 0.05 meant we reject H_0), the Shapiro-Wilk test showed that the various values of concrete strength for categories of concrete were not normally distributed. To perform the test, we needed to also install and load the RVAdeMemoire package.

This meant the needed use of a non-parametric version of the one-way test.. Kruskal-Wallis was settled on.

```
# Non-parametric test - Kruskal-Wallis
kruskal.test(concrete_strength ~ concrete_category, data=concrete_data)

# p-value of 0.3364 > 0.05 for kruskal test. we fail to reject H0
# No difference between both categories
```

```
        Kruskal-Wallis rank sum test

data:  concrete_strength by concrete_category
Kruskal-Wallis chi-squared = 0.92415, df = 1, p-value = 0.3364
```

H_0 was failed to be rejected because p-value (0.3364) was > 0.05. It meant that there was no evidence to suggest that different texture categories of concrete mix affected concrete strength. The boxplot shown earlier also supports this.



2.3.2 Concrete Strength vs Fly Ash Composition

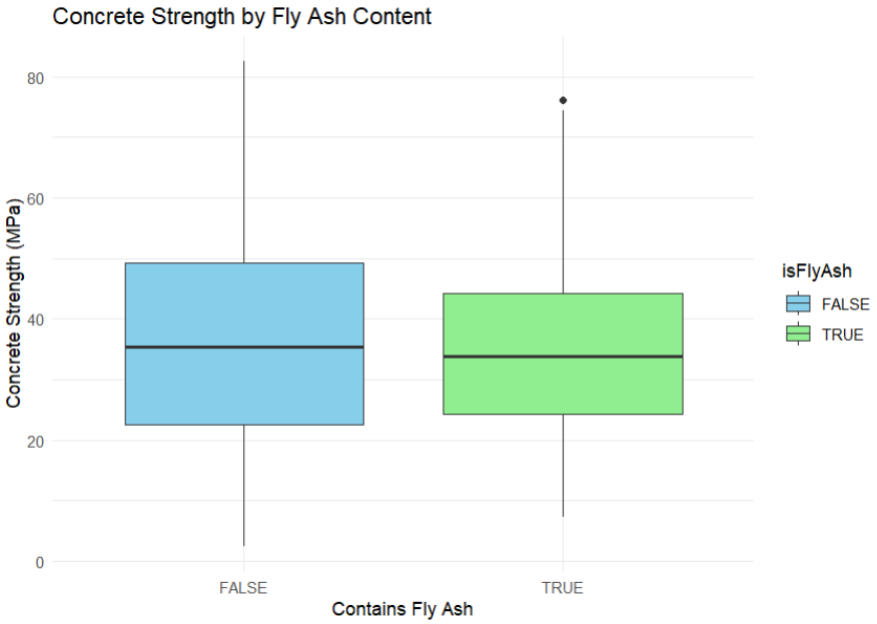
The one-way ANOVA test was used here also for the same reasons (dependent variable is continuous, independent variable is categorical with two or more categories, and observations are independent of one another).

These were the hypotheses:

```
# H0: μnoflyash = μflyash
# H1: μnoflyash ≠ μflyash
# ANOVA one-way test p = 0.05
# Assumptions 1-3 passed by dataset design
```

Assumption 4 was carried out:

```
# Assumption 4
ggplot(concrete_data, aes(x = isFlyAsh, y = concrete_strength, fill = isFlyAsh)) +
  geom_boxplot() +
  labs(title = "Concrete Strength by Fly Ash Content",
       x = "Contains Fly Ash",
       y = "Concrete Strength (MPa)") +
  scale_fill_manual(values = c("FALSE" = "skyblue", "TRUE" = "lightgreen")) +
  theme_minimal()
```



Passed with flying colors. Assumption 5 was carried out next.

```
# Assumption 5
# H0: the variable follows a normal distribution
# H1: the variable does NOT follow a normal distribution

byf.shapiro(concrete_strength ~ isFlyAsh, data=concrete_data) # Assumption failed
```

```
        Shapiro-Wilk normality tests

data:  concrete_strength by isFlyAsh

           W   p-value
FALSE 0.9734 1.285e-08 ***
TRUE  0.9863 0.0002347 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

This assumption failed (p-value < 0.05 meant we reject H_0), the Shapiro-Wilk test showed that the various values of concrete strength for fly ash categories of concrete were not normally distributed. So again, we reverted to the Kruskal-Wallis test.

```
# Non-parametric test - Kruskal-Wallis
kruskal.test(concrete_strength ~ isFlyAsh, data=concrete_data)

# p-value of 0.2324 > 0.05 for kruskal test. we fail to reject H0
# No difference between both categories
```

```
        Kruskal-Wallis rank sum test

data:  concrete_strength by isFlyAsh
Kruskal-Wallis chi-squared = 1.4263, df = 1, p-value = 0.2324

>
```

H_0 was failed to be rejected because p-value (0.3364) was > 0.05. It meant that there was no evidence to suggest that different fly ash compositions of concrete mix affected concrete strength.

2.3.3 Concrete Categories vs Use of Fly Ash

Here we made use of Chi-square test of independence. The goal was to determine if there was an association between fly ash compositions and the texture of a concrete mix.

First, a unique table was created with the two columns unpivoted:

```
# Test of independence between both categories (isFlyAsh and concrete_category)
chiq_table <- table(concrete_data$concrete_category,
                    concrete_data$isFlyAsh)
chiq_table
```

```
> chiq_table

         FALSE TRUE
  Coarse   296  243
  Fine     270  221
>
```

Both rows have very similar proportions.

Next, the test was carried out:

```
chisq.test(chiq_table, correct = FALSE)
```

```
> chisq.test(chiq_table, correct = FALSE)

        Pearson's Chi-squared test

data:  chiq_table
X-squared = 0.00055775, df = 1, p-value = 0.9812

>
```

The interpretation is straight-forward looking at the p-value (which is far greater than 0.05). There is no evidence of any association between concrete category (Coarse vs Fine) and the use of fly ash. In simpler terms: Coarse and Fine concrete are equally likely to use fly ash.

This is meaningful insight because now we know that we didn't miss out on any impactful categorical predictor when developing our regression model.

2.4 Conclusion

The analysis consistently showed how mix composition influences both compressive strength and the likelihood of using fly ash. The multiple linear regression indicated that higher cement, slag and age increase strength, while more water and superplasticizer (logged) reduce it. Residual checks showed that the model provided realistic and acceptable strength predictions.

Non-parametric tests showed that strength distributions for the different labels of categorical variables were non-normal, but importantly, there was no significant difference in strength between fly-ash and non–fly-ash mixes, and also between texture categories. The chi-square test confirmed no association between concrete category and fly-ash use. The logistic regression model performed well, with cement, slag, water, fine aggregate and coarse aggregate significantly predicting fly-ash inclusion, and VIF values confirming no serious multicollinearity.

These results suggest that mix proportions—especially cement and slag—are the key drivers of both strength and fly-ash usage, while fly ash itself does not affect performance in this dataset. These insights could help in budget allocation, thereby improving operational processes for construction companies. The models however, could be further strengthened through cross-validation or exploring non-linear relationships.