Modelling

Importing the libraries necessary for the Modelling part of the report.

```
library(broom)
library(tidyr)
library(R.utils)
```

```
## Loading required package: R.oo

## Loading required package: R.methodsS3

## R.methodsS3 v1.8.2 (2022-06-13 22:00:14 UTC) successfully loaded. See ?R.methodsS3 for help.

## R.oo v1.27.0 (2024-11-01 18:00:02 UTC) successfully loaded. See ?R.oo for help.

##
## Attaching package: 'R.oo'

## The following object is masked from 'package:R.methodsS3':
##
##     throw

## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods

## The following objects are masked from 'package:base':
##
##     attach, detach, load, save

## R.utils v2.13.0 (2025-02-24 21:20:02 UTC) successfully loaded. See ?R.utils for help.

##
## Attaching package: 'R.utils'

## The following object is masked from 'package:tidyr':
##
##     extract

## The following object is masked from 'package:utils':
##
##     timestamp

## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, isOpen, nullfile, parse, use, warnings
```

```
library(GGally)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:R.utils':
##
##     wrap
```

```
library(ggplot2)
library(MASS)
library(foreign)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x R.utils::extract() masks tidyr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x dplyr::select()    masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readstata13)
library(haven)
library(ISLR)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(survival)
library(knitr)
library(readxl)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
```

```
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(Amelia)
```

```
## Loading required package: Rcpp
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.8.3, built: 2024-11-07)
## ## Copyright (C) 2005-2025 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
##
## The following object is masked from 'package:R.oo':
##
##     ll
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
```

```
## The following objects are masked from 'package:Metrics':
##
##     precision, recall
##
## The following object is masked from 'package:survival':
##
##     cluster
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
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(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(caret)
library(dplyr)
```

Initial Data Cleaning

Now we will use the dataframe final_previous_merged. This is the dataframe that was made that consisted of the previous Financial Helath Ratings that we are utilizing to make models for predicting the Financial Health Ratings.

I am naming this dataframe df and making a very simple filtering on NA values to be equal to 0s as mentioned by the client.

Moreover upon furthur discussion with out group we decided to remove the following coulumns as well to simplify our dataset

```
#Defining the dataframe as df
df <- read_csv("final_previous_merged.csv")
```

```
## Rows: 2591 Columns: 63
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (1): prev_currency
## dbl  (61): prev_accountsPayable, prev_accountsReceivable, prev_bankCashBalan...
## date  (1): prev_financialDate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Converting NA values to 0
df[is.na(df)] <- 0

#Removing the columns in the dataframe
df<-df %>% select(-c("prev_inf_factor","prev_X.Other.Currency.to.USD","prev_financialDate"))
df
```

```
## # A tibble: 2,591 x 60
##    prev_accountsPayable prev_accountsReceivable prev_bankCashBalances
##                   <dbl>                   <dbl>                  <dbl>
##  1                 3.20                    6.71                   3.15
##  2                 1.68                   10.4                    2.71
##  3                 1.37                    8.01                   6.68
##  4                 0                       0                      1.49
##  5                 0                       0                     67.6
##  6                 0                       0                     79.6
##  7                 0                       0                    130.
##  8                 0                       0                    109.
##  9                 0                       0                    158.
## 10                 0                       0                    198.
## # i 2,581 more rows
## # i 57 more variables: prev_bankOverdraft <dbl>,
## #   prev_debitOwnedWithinOneYear <dbl>, prev_financialAssets <dbl>,
## #   prev_fixedAssets <dbl>, prev_intangibleAssets <dbl>,
## #   prev_otherCurrentAssets <dbl>, prev_otherCurrentLiabilities <dbl>,
## #   prev_otherEquity <dbl>, prev_otherTermAssets <dbl>,
## #   prev_otherTermLiabilities <dbl>, prev_prepayments <dbl>, ...
```

Random Forest Model

Now with the initial datacleaning done I am not making a Random Forest model for predicting FHR.

Random Forest models are a collection of decisions trees where each tree is trained on a random subset of data.

Our model then aggregates the predictions from the trees to give us a final prediction.

We are using random forest model as it can understand and predict our scores based on the complex relationships we have present in our fiscal data. Moreover it can handle overfitting as well as different types of data suhc as categorical and numerical as well.

Through the Random Forest we can also compute feature importance which shows which variables are the most influential in predicting our FHR scores.

For our RF(Random Forest) Model we are creating an initial train-test split. This split is done such that 80% of the data is considered for training and the rest of the 20% is considered for testing the model.

```r
set.seed(222)
#Creating the partition for the split
split <- createDataPartition(df$FHR, p = 0.8, list = FALSE)

#Creating a 80-20 split for the train and test data
train_data <- df[split, ]
test_data <- df[-split, ]

#The X_train and X_test are the defined predictor variables
#The y_train and y_test are the defined target variables
X_train <- train_data[, -which(names(train_data) == "FHR")]
y_train <- train_data$FHR

X_test <- test_data[, -which(names(test_data) == "FHR")]
y_test <- test_data$FHR

#Defining the RF model
rf_model <- randomForest(x = X_train, y = y_train, ntree = 500, importance = TRUE)

#Getting the predicitions for the RF model
y_prediction_rf <- predict(rf_model, newdata = test_data)

#Getting the metrics, in this case we are using MSE,RMSE,RSS,Total RSS, R^2

mse_rf <- (sum((y_prediction_rf - y_test)^2)/length(y_prediction_rf))
rmse_rf <- sqrt(mse_rf)

#Displaying the RMSE of the model
cat("RMSE of the RF Model: ",rmse_rf)
```

```
## RMSE of the RF Model:  9.191136
```

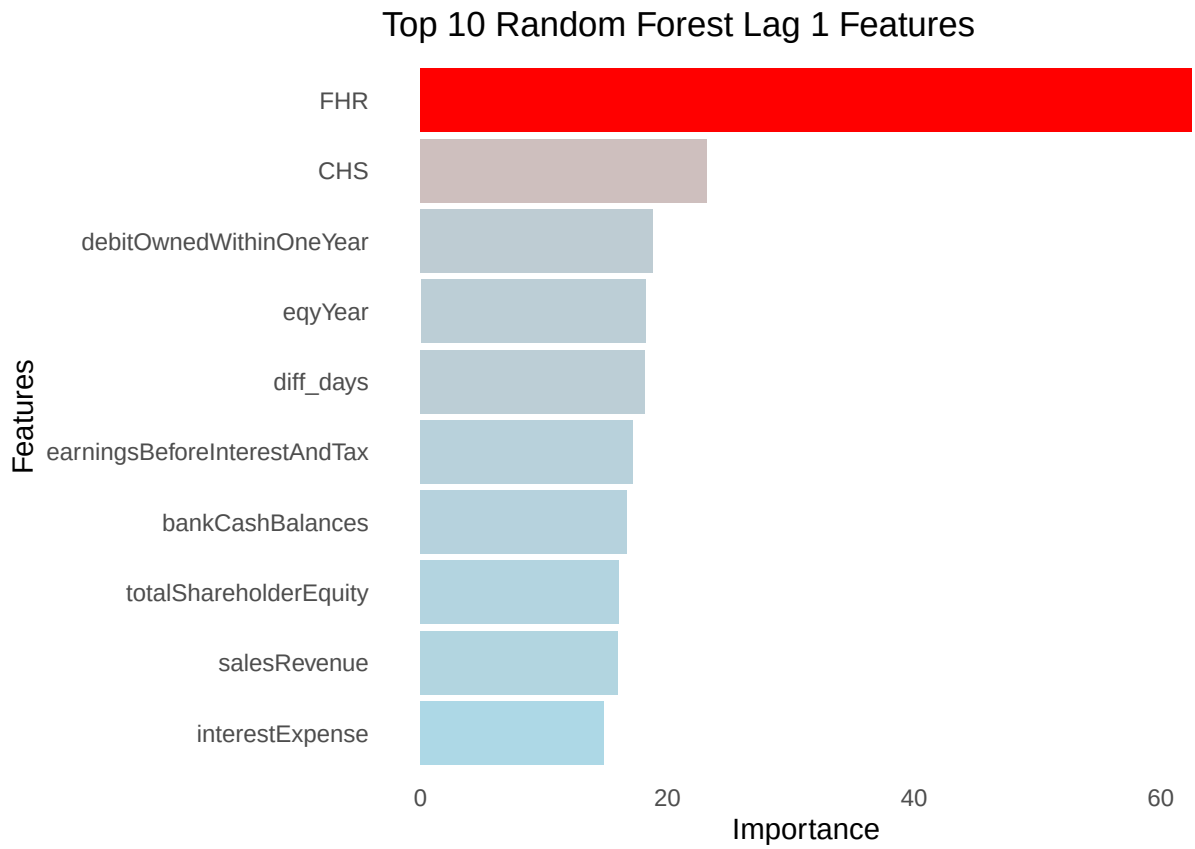
```r
cat("\n")
```

```r
ss_res <- sum((y_test - y_prediction_rf)^2)
ss_tot <- sum((y_test - mean(y_prediction_rf))^2)
r2 <- 1 - ss_res / ss_tot
cat("R^2 value",r2)
```

```
## R^2 value 0.8122503
```

```r
#Getting the variable importance to see which variables are #important for predicting the target variab
imp <- importance(rf_model)

#Sorting the importance in descending order of importance
sorted_importance <- sorted_importance <- imp[order(-imp[, 1]), ]

#Displaying the top 10 variables that are important
sorted_df <- as.data.frame(sorted_importance) %>%
  head(10)  # Get top 10 important features

rownames(sorted_df) <- gsub("prev_", "", rownames(sorted_df))
```

```
ggplot(sorted_df, aes(x = reorder(rownames(sorted_df), sorted_df[, 1]),
                      y = sorted_df[, 1],
                      fill = sorted_df[, 1])) +
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_minimal() +
  scale_fill_gradient(low = "lightblue", high = "red") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "none") +
  labs(x = "Features", y = "Importance", title = "Top 10 Random Forest Lag 1 Features")
```



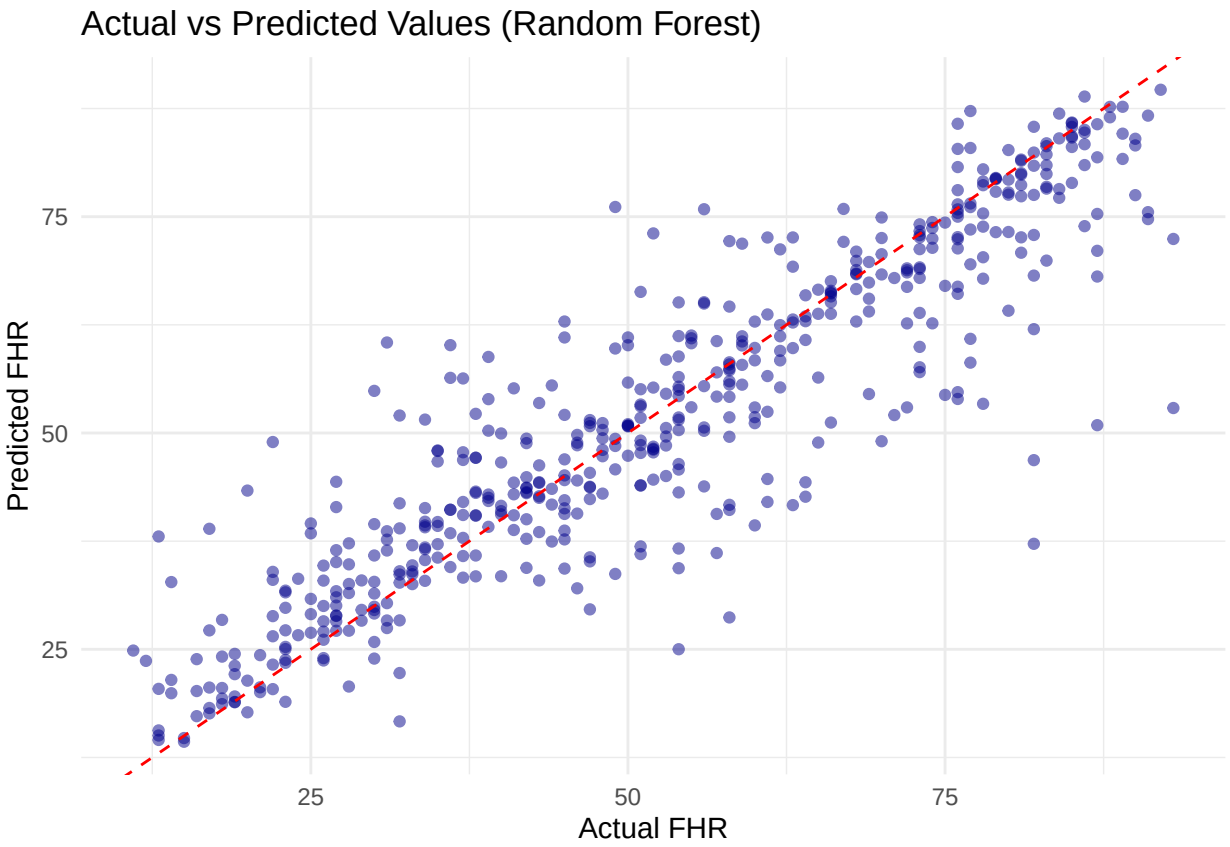
As we can see here the following are of importance in predicting the FHR score: Previous FHR Previous CHS Previous debit Owned within One year

and other fiscal factors..

Now to look at a visual representation of how well fitted the RF Model is we plot an actual versus predicted graph

```
actual_vs_pred <- data.frame(
  Actual = y_test,
  Predicted = y_prediction_rf
)

ggplot(actual_vs_pred, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.5, color = "darkblue") +
```

```
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  theme_minimal() +
  labs(
    title = "Actual vs Predicted Values (Random Forest)",
    x = "Actual FHR",
    y = "Predicted FHR"
  )
```



From above we can see that while there are some outliers the fit described above showcases a model that has pretty good fit in correspondance with the fit line.

Now let us look at a model utilizing Ridge Regression

Ridge Regression Model

Now I am doing a similar process to the one that I did for the RF Model for the Ridge Regression Model

Ridge Regression is a type of linear regression, this is useful to see if there are any interesting linear relationships within the data given.

I am uploading the dataset and doing the usual data cleaning as mentioned in the portion for the RF MOdel

```
finaldf <- read_csv("final_previous_merged.csv")
```

```
## Rows: 2591 Columns: 63
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr   (1): prev_currency
## dbl  (61): prev_accountsPayable, prev_accountsReceivable, prev_bankCashBalan...
## date  (1): prev_financialDate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
finaldf_ridge<-finaldf %>% select(-c('prev_X.Other.Currency.to.USD','prev_inf_factor',
                                     'prev_financialDate'))
```

Now I am creating the train-test split as mentioned above in the RF model and getting the summary statistics to see how well the Ridge Regression Model performed

```
set.seed(222)
#Creating the partition for the split
split <- createDataPartition(finaldf_ridge$FHR, p = 0.8, list = FALSE)

#Creating a 80-20 split for the train and test data
train_data <- finaldf_ridge[split, ]
test_data <- finaldf_ridge[-split, ]

#The X_train and X_test are the defined predictor variables
#The y_train and y_test are the defined target variables
#Since for Ridge Regression modelling we make these matrices
X_train <- as.matrix(train_data[, -which(names(train_data) == "FHR")])
y_train <- train_data$FHR
X_test <- as.matrix(test_data[, -which(names(test_data) == "FHR")])
y_test <- test_data$FHR

#Ridge Regression model
CV_ridge <- cv.glmnet(X_train,y_train,alpha = 0)
```

```
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
```

```
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```
#Seeing goodness of fit
best_lambda <- CV_ridge$lambda.min

#Getting the predicitions for the Ridge Regression model
y_predicted_ridge <- predict(CV_ridge,newx = X_test, s= "lambda.min")
```

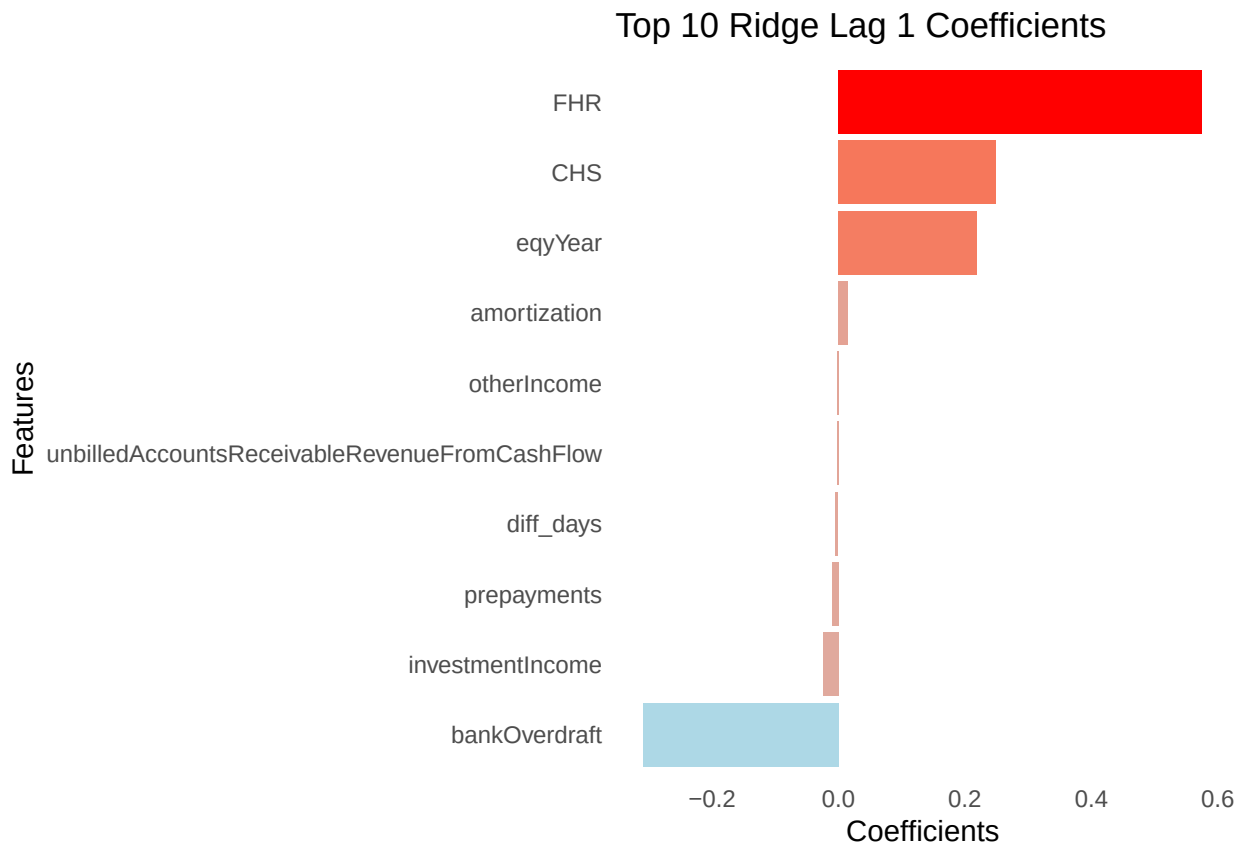
```
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```
#Getting the metrics, in this case we are using MSE,RMSE,RSS,Total RSS, R^2
mse_ridge <- (sum((y_predicted_ridge - y_test)^2)/length(y_predicted_ridge))
rmse_ridge <- sqrt(mse_ridge)
cat("Ridge Regression model RMSE: ",rmse_ridge)
```

```
## Ridge Regression model RMSE:  10.33241
```

```
cat("\n")
```

```
#Extracting coefficients at best lambda
ridge_coef <- coef(CV_ridge, s = "lambda.min")

#Converting to a dataframe
ridge_coef_df <- as.data.frame(as.matrix(ridge_coef))
ridge_coef_df$Feature <- rownames(ridge_coef_df)  # Add feature names
colnames(ridge_coef_df) <- c("Coefficient", "Feature")
ridge_coef_df <- ridge_coef_df[-1, ]
ridge_coef_df$Abs_Coefficient <- abs(ridge_coef_df$Coefficient)
ridge_coef_df <- ridge_coef_df[order(-ridge_coef_df$Abs_Coefficient), ]

ridge_coef_df$Feature <- gsub("prev_", "", ridge_coef_df$Feature)

top10 <- ridge_coef_df %>%
  arrange(desc(Abs_Coefficient)) %>%
  head(10)

ggplot(top10, aes(x = reorder(Feature, Coefficient), y = Coefficient, fill = Coefficient)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_minimal() +
  scale_fill_gradient(low = "lightblue", high = "red") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "none") +
  labs(x = "Features", y = "Coefficients", title = "Top 10 Ridge Lag 1 Coefficients")
```



We can see the the Previous FHR and the Previous CHS is still some of the most important predictor variables similar to the Random Forest.

But compared to the Random Forest we can see that the RMSE value is higher at approximately 10.33. This means that the model predictions are about 10.33 units from the actual on average.

Lasso Model

Now we will look into modelling for Lasso which is a linear regression which prevents overfitting and also does variable selection. This helps again with seeing which variables can be selected to predict FHR. It is also easier to interpret in comparison to the other models such as Random Forest.

Here again I am redefining the df for the Lasso Model but referencing the same data for all the models. Datacleaning as usual is converting the NAs to 0.

Below I am also looking at which columns have character values to see if there any non-numerical columns as linear models like LASSO do not take non-numerical values for modelling

```
library(glmnet)
finaldf <- read.csv("final_previous_merged.csv")
finaldf[is.na(finaldf)] <- 0
for(col in names(finaldf)) {
  if(any(is.character(finaldf[[col]]))) {
    print(paste("Column", col, "contains character values"))
  }
}
```

```
## [1] "Column prev_currency contains character values"
## [1] "Column prev_financialDate contains character values"
```

Since these columns upon discussion are not necessary to extract useful insights from, we decided to remove the columns instead along with other columns.

```
finaldf_lasso<-finaldf %>% select(-c('prev_X.Other.Currency.to.USD','prev_inf_factor',
                                     'prev_financialDate'))
```

Now I am doing the test-train split for the Lasso Model using the metrics similar to the RF and Ridge Regression (80/20 split on the data for training and testing respectively). Also I am defining the Lasso Model as well.

```
set.seed(222)
#Creating the partition for the split
split <- createDataPartition(finaldf_lasso$FHR, p = 0.8, list = FALSE)

#Creating a 80-20 split for the train and test data
train_data <- finaldf_lasso[split, ]
test_data <- finaldf_lasso[-split, ]

#The X_train and X_test are the defined predictor variables
#The y_train and y_test are the defined target variables
X_train <- as.matrix(train_data[, -which(names(train_data) == "FHR")])
y_train <- train_data$FHR
X_test <- as.matrix(test_data[, -which(names(test_data) == "FHR")])
y_test <- test_data$FHR

#Defining the lasso model
CV_lasso <- cv.glmnet(X_train,y_train,alpha = 1)
```

```
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
```

```
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```r
#Goodness of fit
best_lambda <- CV_lasso$lambda.min

#Getting the predictions from the model
y_predicted_lasso <- predict(CV_lasso,newx = X_test, s= "lambda.min")
```

```
## Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion
```

```r
#Getting the metrics, in this case we are using MSE,RMSE,RSS,Total RSS, R^2
mse_lasso <- (sum((y_predicted_lasso - y_test)^2)/length(y_predicted_lasso))
rmse_lasso <- sqrt(mse_lasso)
lasso_coef <- coef(CV_lasso, s = "lambda.min")
ss_res <- sum((y_test - y_predicted_lasso)^2)
ss_tot <- sum((y_test - mean(y_predicted_lasso))^2)
r2 <- 1 - ss_res / ss_tot
cat("The Lasso Model R^2:",r2)
```

```
## The Lasso Model R^2: 0.7930211
```

```r
cat("\n")
```

```r
cat("The Lasso Model RMSE:",rmse_lasso)
```

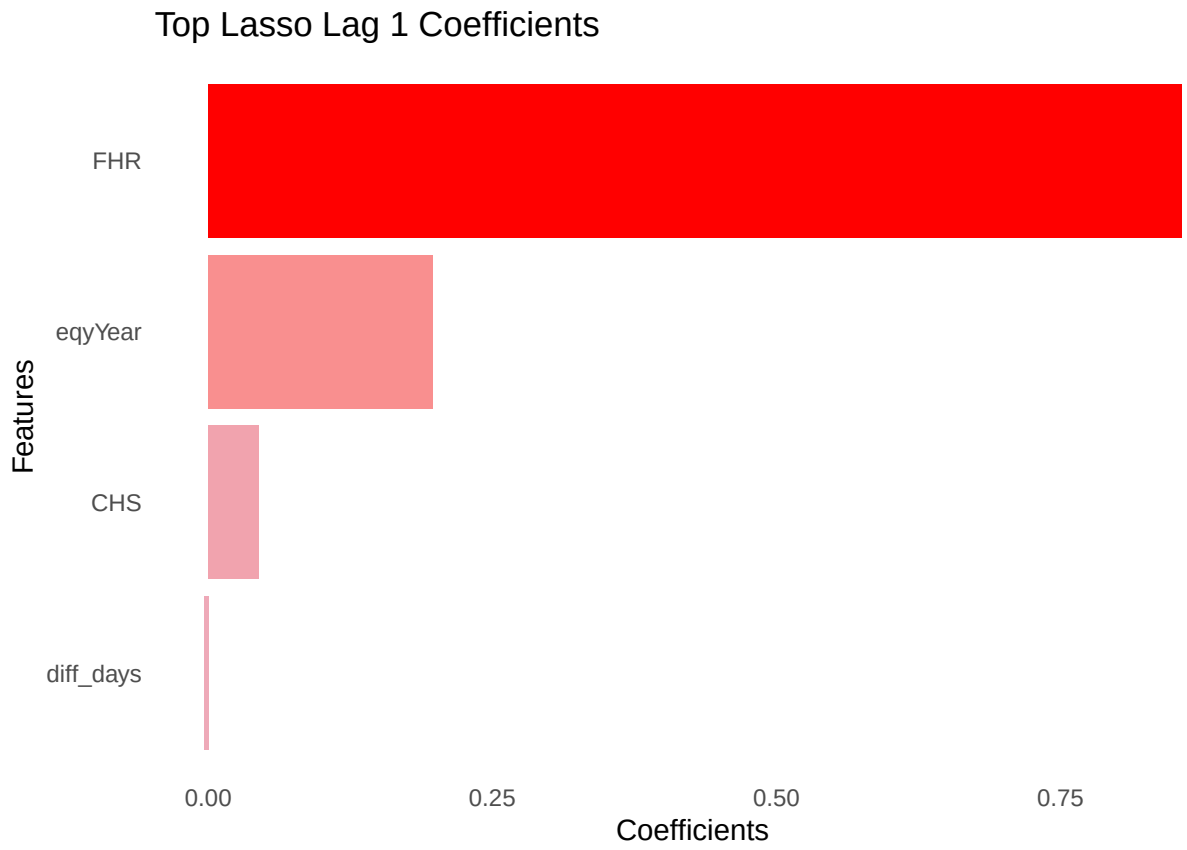
```
## The Lasso Model RMSE: 9.652856
```

We can see that the Lasso does better that the Ridge in terms of the RMSE value which is 9.65 but not as good as the Random Forest model. Using these values we can get a sense that the Random Forest has the best RMSE.

Now looking at the importance of variables for which Lasso does variable selection on:

```r
lasso_coef_df <- as.data.frame(as.matrix(lasso_coef))  # Convert to DataFrame
lasso_coef_df$Feature <- rownames(lasso_coef_df)  # Add feature names
colnames(lasso_coef_df) <- c("Coefficient", "Feature")
lasso_coef_df <- lasso_coef_df[-1, ]  # Remove intercept
lasso_coef_df$Abs_Coefficient <- abs(lasso_coef_df$Coefficient)  # Get absolute values
lasso_coef_df <- lasso_coef_df[order(-lasso_coef_df$Abs_Coefficient), ]

lasso_coef_df$Feature <- gsub("prev_", "", lasso_coef_df$Feature)

lasso_coef_df <- lasso_coef_df[lasso_coef_df$Abs_Coefficient != 0, ]

#Taking only the top 10 values to get the variable importance
top10 <- lasso_coef_df %>%
  arrange(desc(Abs_Coefficient)) %>%
  head(10)


ggplot(top10, aes(x = reorder(Feature, Coefficient), y = Coefficient, fill = Coefficient)) +
  geom_bar(stat = "identity") +
  coord_flip() +
```

```
  theme_minimal() +
  scale_fill_gradient(low = "pink2", high = "red") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "none") +
  labs(x = "Features", y = "Coefficients", title = "Top Lasso Lag 1 Coefficients")
```



For additional modelling I decided to explore XGBoost.

XGBOOST

This is gradient boosting model where the method creates an ensemble of decision trees where each new tree it creates it corrects the error made by the previous trees, hence minimizing errors through gradient descent. The decision trees are done sequentially where each new tree reduces the error of the previous ensemble of trees.

Now modelling using XGBoost for predicting the previous FHR

```
set.seed(222)
#Defining the data frame for the XGBoost similar to the other models and doing some data cleaning throu
finaldf <- read.csv("final_previous_merged.csv")

finaldf[is.na(finaldf)] <- 0

finaldf_xgboost<-finaldf %>% select(-c('prev_X.Other.Currency.to.USD','prev_inf_factor','prev_financiall
```

```r
#Creating a 80-20 split for the train and test data
split_index <- createDataPartition(finaldf_xgboost$FHR, p = 0.8, list = FALSE)
train_data <- finaldf_xgboost[split_index, ]
test_data <- finaldf_xgboost[-split_index, ]

#The X_train and X_test are the defined predictor variables
#The y_train and y_test are the defined target variables
X_train <- as.matrix(train_data %>% select(-FHR))
y_train <- train_data$FHR
X_test <- as.matrix(test_data %>% select(-FHR))
y_test <- test_data$FHR

#For processing need to convert it to a XGBoost DMatrix
dtrain <- xgb.DMatrix(data = X_train, label = y_train)
dtest <- xgb.DMatrix(data = X_test, label = y_test)

#Defining the hyperparameters for the XGBoost model
params <- list(
  objective = "reg:squarederror",  #For regression
  eta = 0.1,            #Learning rate
  max_depth = 6,        #Max. depth of trees
  min_child_weight = 1,#Minimum sum of instance weights for a child
  subsample = 0.8,   #Subsample ratio for training instances
  colsample_bytree = 0.8     #Subsample ratio of columns when constructing each tree
)

#Cross validation to find optimal number of boosting rounds
cv_results <- xgb.cv(
  params = params,
  data = dtrain,
  nrounds = 1000,      #Max. # of boosting rounds
  nfold = 5,           #5 fold Cross validation
  early_stopping_rounds = 50, #Early stopping is to indicat to stop if there is no
                              #improvements for 50 rounds
  verbose = 0
)

#Optimale number of rounds based on the cross validation
best_n_rounds <- which.min(cv_results$evaluation_log$test_rmse_mean)

#XGBoost model for optimal number of rounds
xgb_model <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = best_n_rounds,
  watchlist = list(train = dtrain, test = dtest),
  verbose = 0
)

#Predictions from the model
y_pred_xgb <- predict(xgb_model, dtest)

#Calculating the metrics of RMSE,MSE
```

```r
xbg_mse <- mean((y_pred_xgb - y_test)^2)
xgb_rmse <- sqrt(xbg_mse)

cat(paste("RMSE:", xgb_rmse))
```

```
## RMSE: 9.52703214351681
```

```r
cat("\n")
```

As we can see while this does better that the Linear models,as the data contains complex fiscal relationships that are better captured by trees and tree based-models through either bagging or boosting, we can see that in this case RF Models tend to perform better than the gradient boosting model with XGBoost.

Hence we will use the RF Model for final modelling for the future FHR score as is performed better in terms of metrics in comparison to the other models.

Now we are using the model to predict the future FHR scores.

```r
#Now we are using a dataframe that have the previous FHR scores and using this we will get predictions
suppliers <- read_csv("final_future_merged.csv")
```

```
## Rows: 164 Columns: 63
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (2): prev_Supplier.Number, prev_currency
## dbl  (60): prev_accountsPayable, prev_accountsReceivable, prev_bankCashBalan...
## date  (1): prev_financialDate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#Doing some data cleaning on the dataframe
suppliers[is.na(suppliers)] <- 0
suppliers<-suppliers %>% select(-c("prev_inf_factor","prev_X.Other.Currency.to.USD",
                                   "prev_financialDate"))

#Predicting the model for the data given
pred <- predict(rf_model, newdata = suppliers)

results <- data.frame(supplierID = suppliers$prev_Supplier.Number, predicted_FHR = pred)

#Saving it to a CSV for further analysis
write.csv(results, "predicted_FHR_supplier.csv", row.names = FALSE)
```

We will now utilize the predictions to do further analysis on the results gained.

```r
#Getting the predictions we got from running the model in the cell before
final_predictions<-read_csv("predicted_FHR_supplier.csv")
```

```
## Rows: 164 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr (1): supplierID
## dbl (1): predicted_FHR
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
supplier_info<-read_csv("final_merged_updated.csv")
```

```
## New names:
## Rows: 2925 Columns: 69
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (9): Supplier.Number, currency, financialDate, period, RRID, id, Group,... dbl
## (60): ...1, accountsPayable, accountsReceivable, bankCashBalances, bankO...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
head(final_predictions)
```

```
## # A tibble: 6 x 2
##   supplierID predicted_FHR
##   <chr>              <dbl>
## 1 021320P             78.2
## 2 032380P             73.1
## 3 061043P             38.3
## 4 061090P             86.4
## 5 070740P             55.9
## 6 082610P             33.5
```

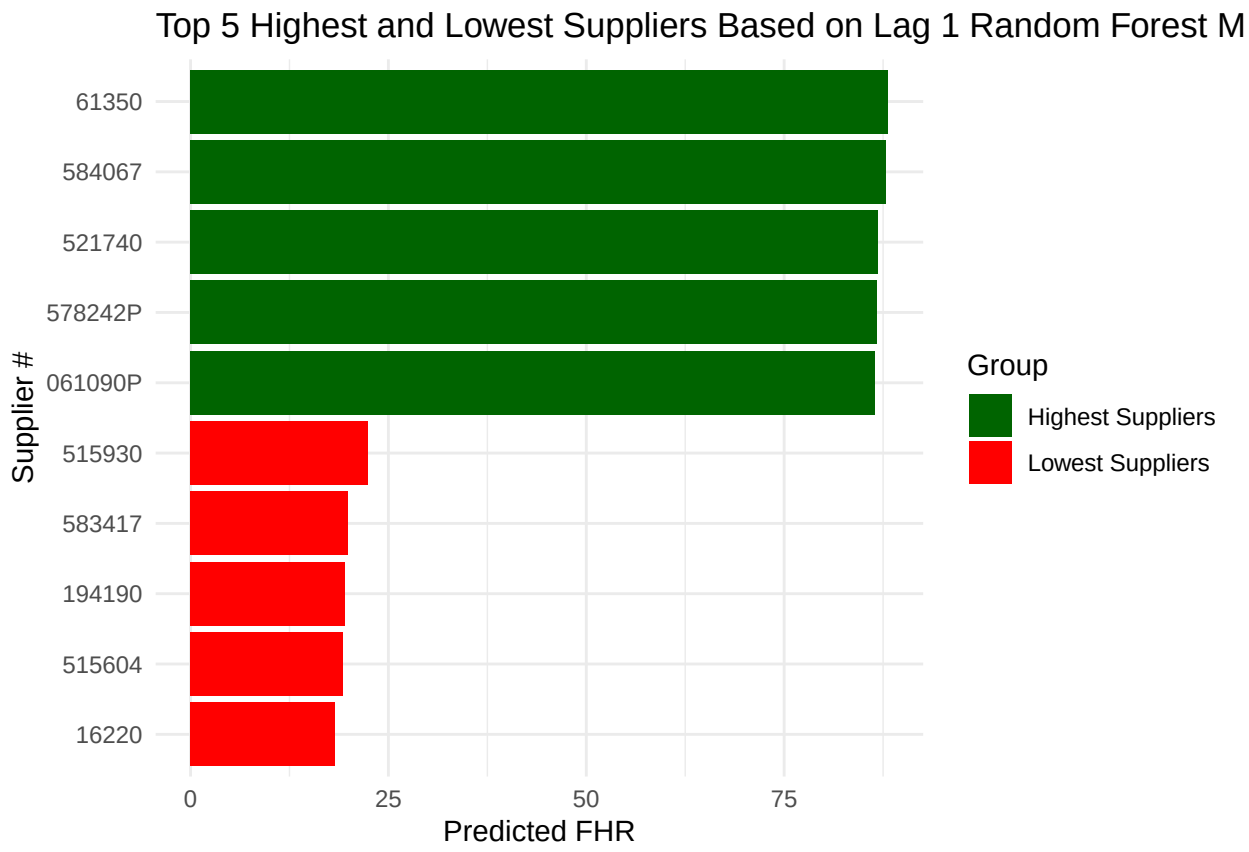
Now we will look at the top 5 and bottom 5 suppliers in terms of the FHR score

Further we are seeing which countries of origin based on the currency the supplier uses.

Plotting the top and bottom currencies to see which countries perform the best or perform poorly

```
lowest_suppliers <- final_predictions %>%
  arrange(predicted_FHR) %>%
  head(5) %>%
  mutate(Group = "Lowest Suppliers")

# Get the highest 5 suppliers and mark them as "Highest"
highest_suppliers <- final_predictions %>%
  arrange(desc(predicted_FHR)) %>%
  head(5) %>%
  mutate(Group = "Highest Suppliers")

# Combine the two datasets into one data frame
combined_suppliers <- bind_rows(lowest_suppliers, highest_suppliers)

# Create a bar chart with ggplot2
ggplot(combined_suppliers, aes(x = reorder(supplierID, predicted_FHR), y = predicted_FHR, fill = Group))
  geom_col() +
```

```
  scale_fill_manual(values = c("Highest Suppliers" = "darkgreen", "Lowest Suppliers" = "red")) +
  labs(title = "Top 5 Highest and Lowest Suppliers Based on Lag 1 Random Forest Model",
       x = "Supplier #", y = "Predicted FHR") +
  coord_flip() +
  theme_minimal()
```



Now I will look into the supplier data and try to find out their country of origin

```
supplier_info
```

```
## # A tibble: 2,925 x 69
##     ...1 Supplier.Number accountsPayable accountsReceivable bankCashBalances
##    <dbl> <chr>                     <dbl>              <dbl>            <dbl>
##  1     1 508398                       NA                 NA            1435.
##  2     2 508398                       NA                 NA            1491.
##  3     3 508398                       NA                 NA            2284.
##  4     4 508398                       NA                 NA            1750.
##  5     5 508398                       NA                 NA            1832.
##  6     6 508398                       NA                 NA            1884.
##  7     7 508398                       NA                 NA            2218.
##  8     8 508398                       NA                 NA            2054.
##  9     9 508398                       NA                 NA            2007.
## 10    10 508398                       NA                 NA            2317.
## # i 2,915 more rows
## # i 64 more variables: bankOverdraft <dbl>, debitOwnedWithinOneYear <dbl>,
```

```
## #   financialAssets <dbl>, fixedAssets <dbl>, intangibleAssets <dbl>,
## #   otherCurrentAssets <dbl>, otherCurrentLiabilities <dbl>, otherEquity <dbl>,
## #   otherTermAssets <dbl>, otherTermLiabilities <dbl>, prepayments <dbl>,
## #   retainedEarnings <dbl>, subscribedCapital <dbl>, termLoans <dbl>,
## #   totalAssets <dbl>, totalCurrentAssets <dbl>, ...
```

```
#Analysis on the perspective of currency on the two dataframes
latest_currency_data <- supplier_info %>%
  group_by(Supplier.Number) %>%
  arrange(desc(financialDate)) %>%
  dplyr::slice(1) %>%
  select(Supplier.Number, currency, financialDate)

result <- final_predictions %>%
  left_join(latest_currency_data,
            by = c("supplierID" = "Supplier.Number"))

head(result)
```

```
## # A tibble: 6 x 4
##   supplierID predicted_FHR currency financialDate
##   <chr>              <dbl> <chr>    <chr>
## 1 021320P             78.2 USD      9/30/2023
## 2 032380P             73.1 JPY      9/30/2023
## 3 061043P             38.3 JPY      9/30/2023
## 4 061090P             86.4 JPY      9/30/2023
## 5 070740P             55.9 JPY      9/30/2023
## 6 082610P             33.5 JPY      6/30/2024
```

```
result
```

```
## # A tibble: 164 x 4
##    supplierID predicted_FHR currency financialDate
##    <chr>              <dbl> <chr>    <chr>
##  1 021320P             78.2 USD      9/30/2023
##  2 032380P             73.1 JPY      9/30/2023
##  3 061043P             38.3 JPY      9/30/2023
##  4 061090P             86.4 JPY      9/30/2023
##  5 070740P             55.9 JPY      9/30/2023
##  6 082610P             33.5 JPY      6/30/2024
##  7 092210P             56.8 JPY      9/30/2023
##  8 100380              62.4 USD      9/30/2023
##  9 100520P             86.4 JPY      9/30/2023
## 10 11330               35.2 JPY      9/30/2023
## # i 154 more rows
```

It might be interesting to see the country of origin for the suppliers to see if we see similar trends to what we saw during EDA where Canadian and Japanese suppliers perform better than the rest.

```
#Top 20 suppliers
top_20 <- result%>%
  arrange(desc(predicted_FHR)) %>%
  slice_head(n = 20)
```

```r
#Bottom 10 suppliers
bottom_20 <- result %>%
  arrange(predicted_FHR) %>%
  slice_head(n = 20)
```

Further we are seeing which countries of origin based on the currency the supplier uses.

```r
#Top countries of origin
top_currencies <- top_20 %>%
  group_by(currency) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

#Bottom countries of origin
bottom_currencies <- bottom_20 %>%
  group_by(currency) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
```

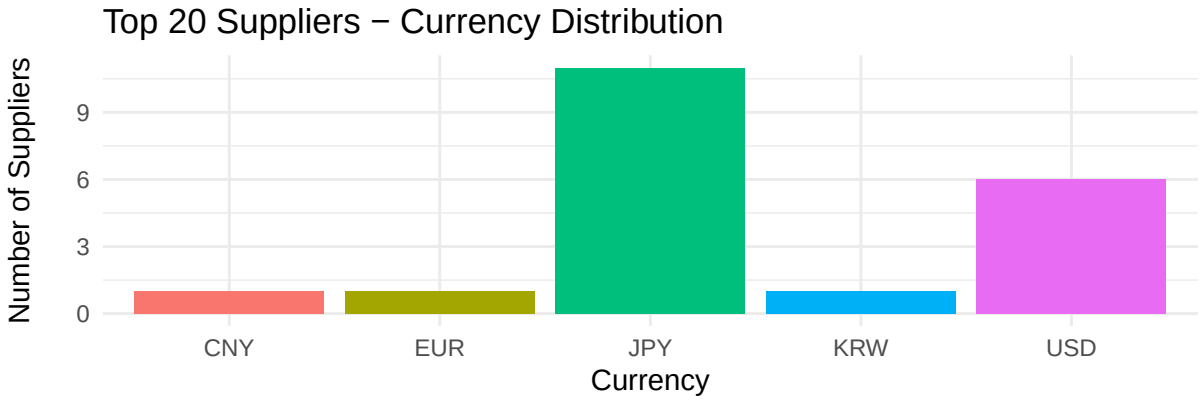
Plotting the top and bottom currencies to see which countries perform the best or perform poorly

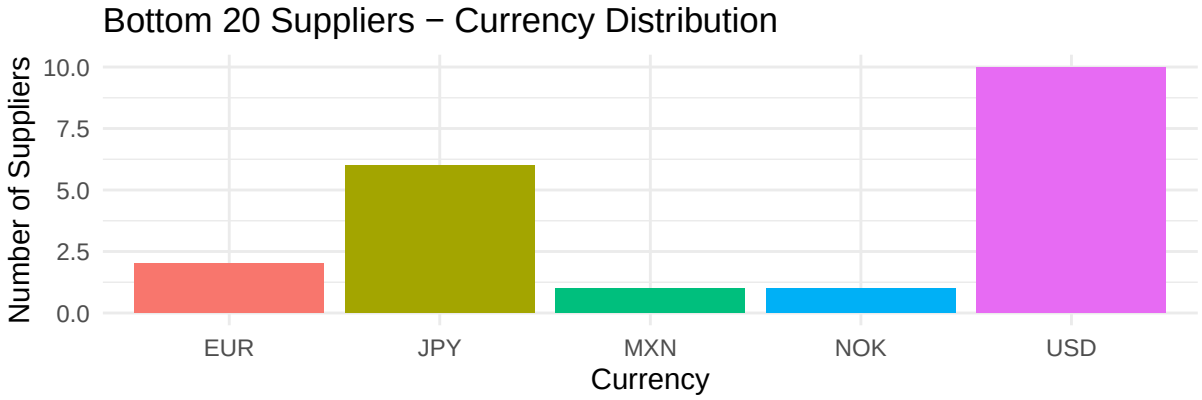
```r
#Best performing countries
top_plot <- ggplot(top_currencies, aes(x = currency, y = count, fill = currency)) +
  geom_bar(stat = "identity") +
  labs(title = "Top 20 Suppliers - Currency Distribution",
       x = "Currency",
       y = "Number of Suppliers") +
  theme_minimal() +
  theme(legend.position = "none")

#Worst performing countries
bottom_plot <- ggplot(bottom_currencies, aes(x = currency, y = count, fill = currency)) +
  geom_bar(stat = "identity") +
  labs(title = "Bottom 20 Suppliers - Currency Distribution",
       x = "Currency",
       y = "Number of Suppliers") +
  theme_minimal() +
  theme(legend.position = "none")

library(patchwork)
```

```
##
## Attaching package: 'patchwork'
```

```
## The following object is masked from 'package:MASS':
##
##     area
```

```r
top_plot/bottom_plot  # vertically stacked
```





These charts show an interesting currency distribution pattern.

We can see how the suppliers with countries of origin that have the top number of suppliers show variability in terms of the predicted FHR scores. This can be identified with Japanese and US suppliers being high in number in terms of count.

But it is interesting to note that Japanese suppliers are top performing while US has some underperformers.

It will be interesting to consider Japanese based suppliers and analyzing what they offer better in terms of quality for such high performance.

Moreover the performance of Japanese suppliers can also be seen as high even in our EDA process and hence we can say that in general suppliers from Japan could be advantageous to have more trade with.

It is interesting to see that one supplier from Korea is also a high performer along with a supplier from China. This could maybe correlate with the East Asian region havinghigh performers in terms of supplier FHR scores.