# Modeling using PCR

We want to see if using principal components help with building the predictive model. The random forest model using original variables is the best we got so far. ## Random Forest Model using Principal Components

```r
set.seed(222)
final_previous_merged <- read.csv("final_previous_merged.csv") # load new dataset

final_previous_merged_updated<- final_previous_merged %>% mutate(risk= ifelse(FHR >= 80, "very low risk

# make sure all na values are converted to 0
final_previous_merged_updated[is.na(final_previous_merged_updated)] <- 0

# remove the currency conversion columns
valid_cols <- final_previous_merged_updated %>%
  select(where(is.numeric)) %>%
  select(-prev_X.Other.Currency.to.USD, -prev_inf_factor) %>%
  summarise(across(everything(), ~ mean(!is.na(.)))) %>%
  pivot_longer(everything(), names_to = "col", values_to = "non_na_ratio") %>%
  pull(col)

# also remove FHR to avoid it predicts itself
numeric_data <- final_previous_merged_updated %>%
  select(all_of(valid_cols)) %>%
  select(-FHR) %>%
  drop_na()

nrow(numeric_data)
```

```
## [1] 2591
```

```r
# use Principal Component for modeling, use prcomp function to find number of PCs being used
pca_result <- prcomp(numeric_data, scale. = TRUE)
summary(pca_result)
```

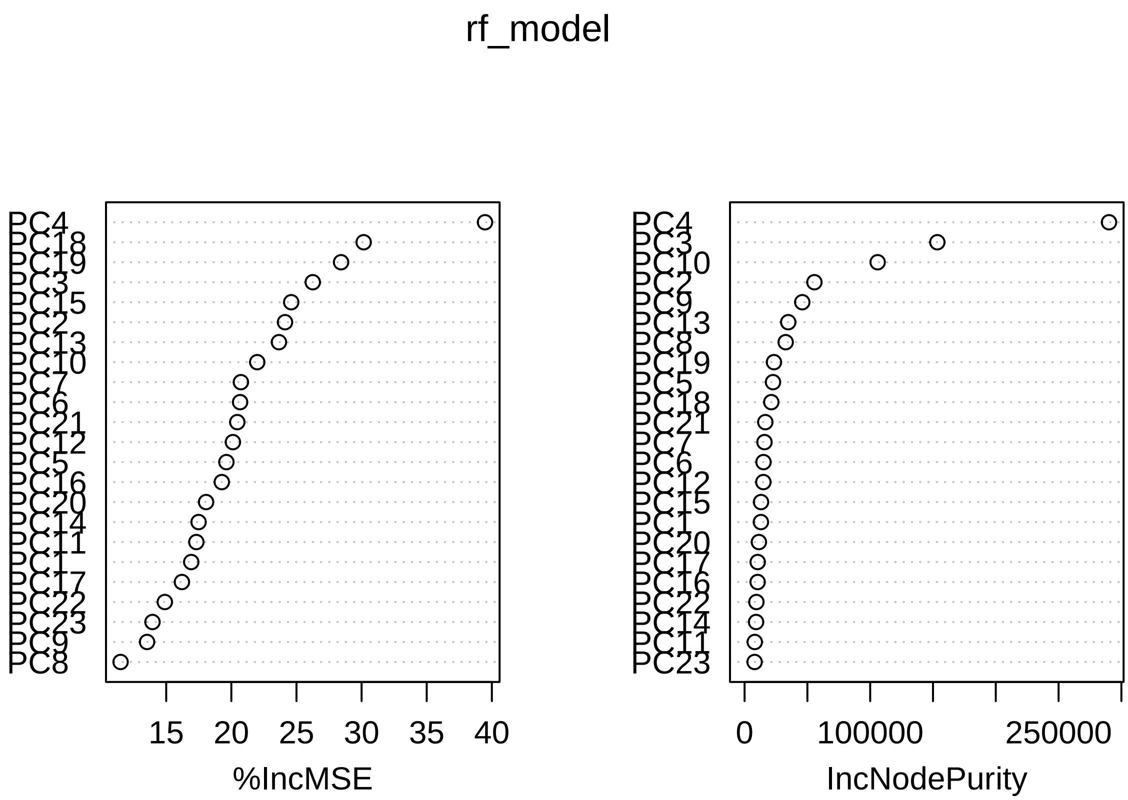
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     5.6330 3.1808 1.38322 1.36807 1.27247 1.25719 1.09385
## Proportion of Variance 0.5471 0.1744 0.03299 0.03227 0.02792 0.02725 0.02063
## Cumulative Proportion  0.5471 0.7215 0.75451 0.78678 0.81469 0.84194 0.86257
##                            PC8     PC9    PC10    PC11   PC12    PC13    PC14
## Standard deviation     1.05610 1.02067 0.99751 0.89910 0.8913 0.74740 0.68511
## Proportion of Variance 0.01923 0.01796 0.01716 0.01394 0.0137 0.00963 0.00809
## Cumulative Proportion  0.88180 0.89977 0.91692 0.93086 0.9446 0.95419 0.96228
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.65552 0.64473 0.56741 0.43240 0.41759 0.37859 0.34766
## Proportion of Variance 0.00741 0.00717 0.00555 0.00322 0.00301 0.00247 0.00208
## Cumulative Proportion  0.96969 0.97686 0.98241 0.98563 0.98864 0.99111 0.99319
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.30195 0.25850 0.20352 0.19311 0.17775 0.15506 0.14966
## Proportion of Variance 0.00157 0.00115 0.00071 0.00064 0.00054 0.00041 0.00039
## Cumulative Proportion  0.99476 0.99592 0.99663 0.99727 0.99782 0.99823 0.99862
```

```
##                             PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.12137 0.11653 0.09765 0.08924 0.08023 0.07513 0.06934
## Proportion of Variance 0.00025 0.00023 0.00016 0.00014 0.00011 0.00010 0.00008
## Cumulative Proportion  0.99887 0.99911 0.99927 0.99941 0.99952 0.99962 0.99970
##                             PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.05421 0.05304 0.05013 0.04552 0.04282 0.04216 0.03526
## Proportion of Variance 0.00005 0.00005 0.00004 0.00004 0.00003 0.00003 0.00002
## Cumulative Proportion  0.99975 0.99980 0.99984 0.99988 0.99991 0.99994 0.99996
##                             PC43    PC44    PC45   PC46    PC47    PC48    PC49
## Standard deviation     0.02559 0.01968 0.01761 0.0163 0.01298 0.01263 0.01171
## Proportion of Variance 0.00001 0.00001 0.00001 0.0000 0.00000 0.00000 0.00000
## Cumulative Proportion  0.99997 0.99998 0.99998 1.0000 0.99999 0.99999 1.00000
##                             PC50     PC51     PC52     PC53     PC54     PC55
## Standard deviation     0.008489 0.006888 0.006081 0.004068 0.003316 0.002259
## Proportion of Variance 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
## Cumulative Proportion  1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
##                             PC56      PC57      PC58
## Standard deviation     0.0003631 2.072e-06 3.985e-10
## Proportion of Variance 0.0000000 0.000e+00 0.000e+00
## Cumulative Proportion  1.0000000 1.000e+00 1.000e+00
```

```r
# we use 23 Principal Components not the elbow point because we want to maximize the accuracy
pca_df <- as.data.frame(pca_result$x[, 1:23])
pca_df$FHR <- final_previous_merged_updated$FHR
set.seed(222)
train_index <- sample(nrow(pca_df), 0.8 * nrow(pca_df))
train <- pca_df[train_index, ]
test <- pca_df[-train_index, ]

rf_model <- randomForest(FHR ~ ., data = train, ntree = 500, importance = TRUE)
rf_pred <- predict(rf_model, newdata = test)
# RMSE
rmse <- sqrt(mean((rf_pred - test$FHR)^2))
cat("RMSE:", round(rmse, 2))
```

```
## RMSE: 9.53
```

```r
varImpPlot(rf_model)
```




```r
top_pcs <- c("PC4", "PC19", "PC18", "PC3")

actual <- test$FHR
# R^2
ss_res <- sum((actual - rf_pred)^2)          # residual sum of squares
ss_tot <- sum((actual - mean(actual))^2)     # total sum of squares
r_squared <- 1 - (ss_res / ss_tot)
cat("R-squared:", round(r_squared, 4))
```

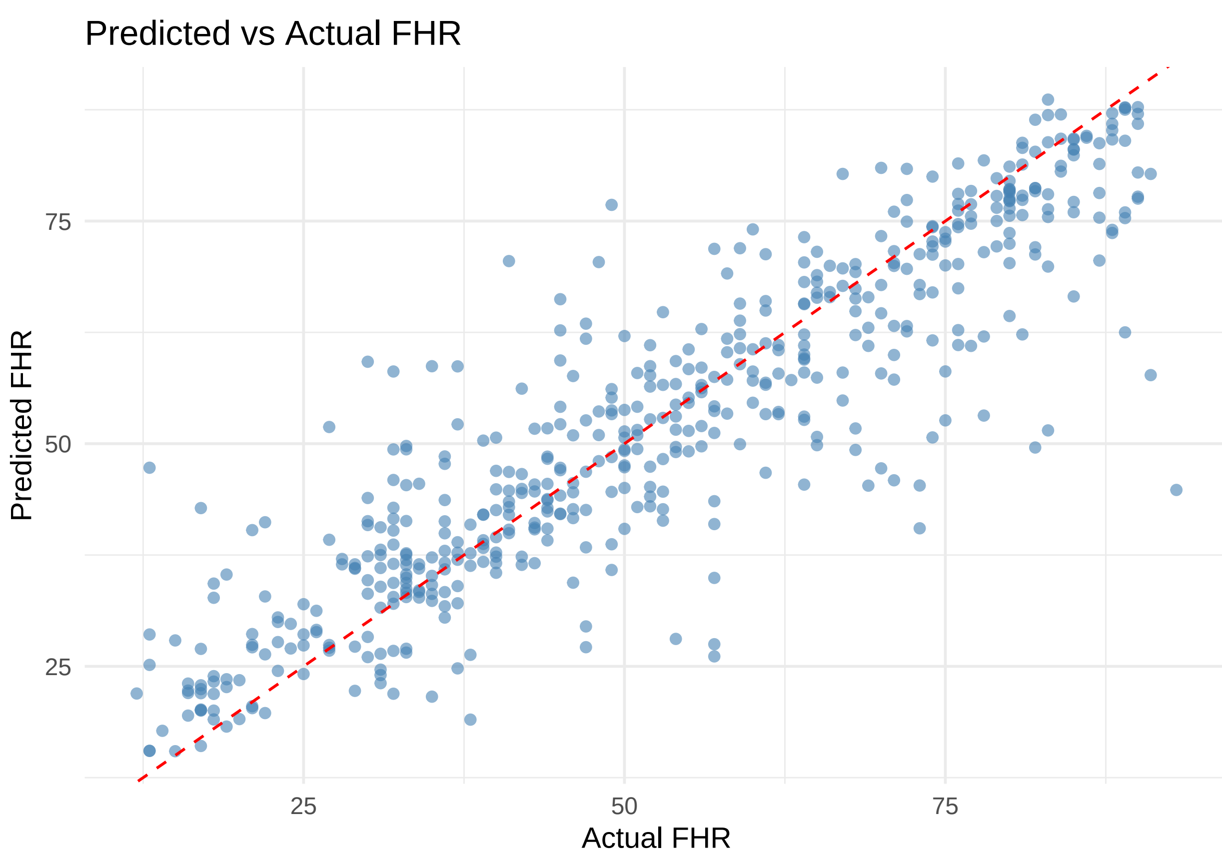
```
## R-squared: 0.7977
```

```r
# this table shows how each top 4 components are consisted by different variables
pc_loadings <- pca_result$rotation[, top_pcs]

loading_table <- as.data.frame(pc_loadings) %>%
  tibble::rownames_to_column("Variable") %>%
  tidyr::pivot_longer(cols = all_of(top_pcs), names_to = "PC", values_to = "Loading") %>%
  mutate(abs_loading = abs(Loading)) %>%
  group_by(PC) %>%
  slice_max(order_by = abs_loading, n = 10) %>%   # top 10 per PC
  arrange(PC, desc(abs_loading))

print(loading_table)
```

```
## # A tibble: 40 x 4
```

```
## # Groups:   PC [4]
##    Variable                         PC    Loading abs_loading
##    <chr>                            <chr>   <dbl>       <dbl>
##  1 prev_debitOwnedWithinOneYear     PC18    0.547       0.547
##  2 prev_financialAssets             PC18   -0.464       0.464
##  3 prev_CHS                         PC18    0.365       0.365
##  4 prev_FHR                         PC18   -0.329       0.329
##  5 prev_interestExpense             PC18   -0.271       0.271
##  6 prev_totalCurrentLiabilities     PC18    0.209       0.209
##  7 prev_netProfitAfterTax           PC18   -0.192       0.192
##  8 prev_earningsBeforeInterestAndTax PC18  -0.152       0.152
##  9 prev_totalCurrentAssets          PC18    0.136       0.136
## 10 prev_salesRevenue                PC18    0.103       0.103
## # i 30 more rows
```

```r
# Show how the prediction fit visually
ggplot(data = NULL, aes(x = test$FHR, y = rf_pred)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Predicted vs Actual FHR", x = "Actual FHR", y = "Predicted FHR") +
  theme_minimal()
```

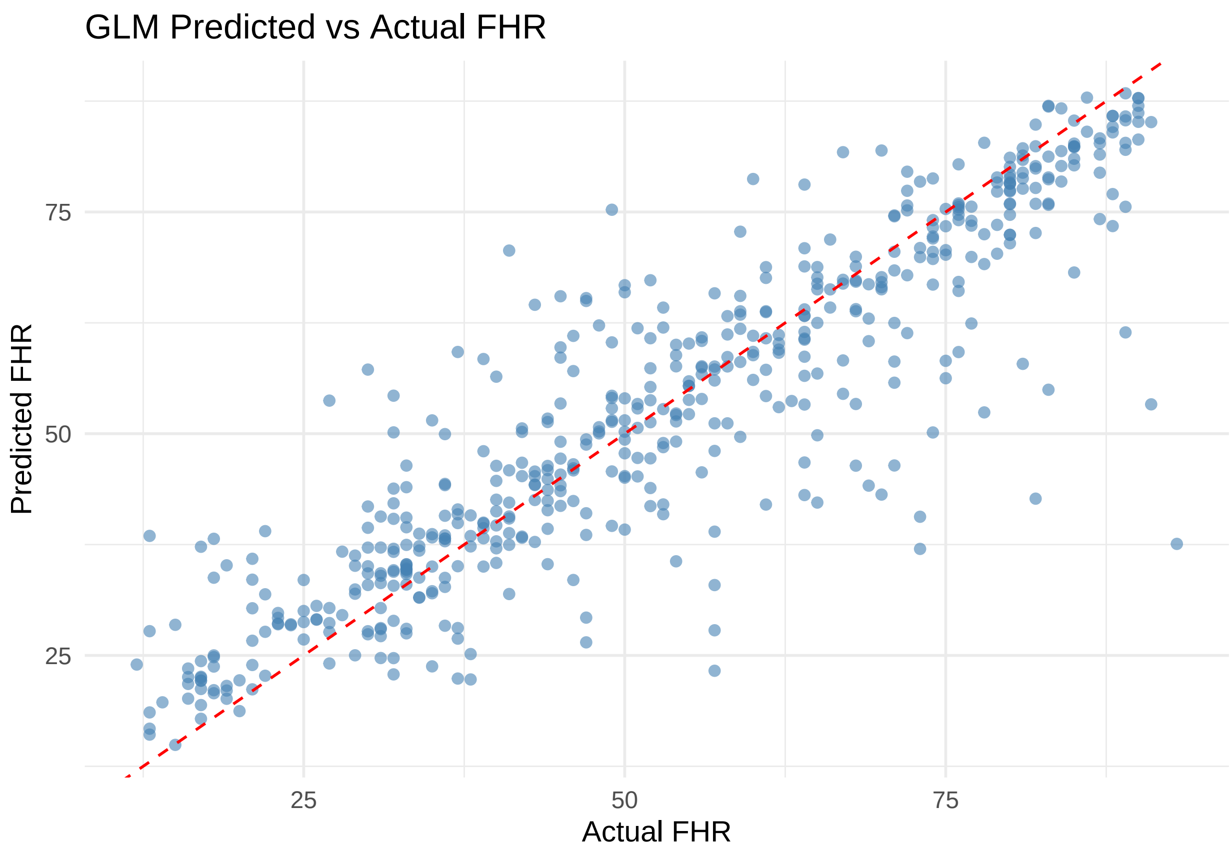



First Random Forest model using PCs shows R^2: 0.7977, RMSE: 9.5322, which shows better than the baseline model, but it is not better than the random forest model using variables.

For the rest of the models, We use the similar code as the first random forest model. ## GLM Model using PCs

```
## RMSE: 9.51
```

```
## R-squared: 0.7988
```



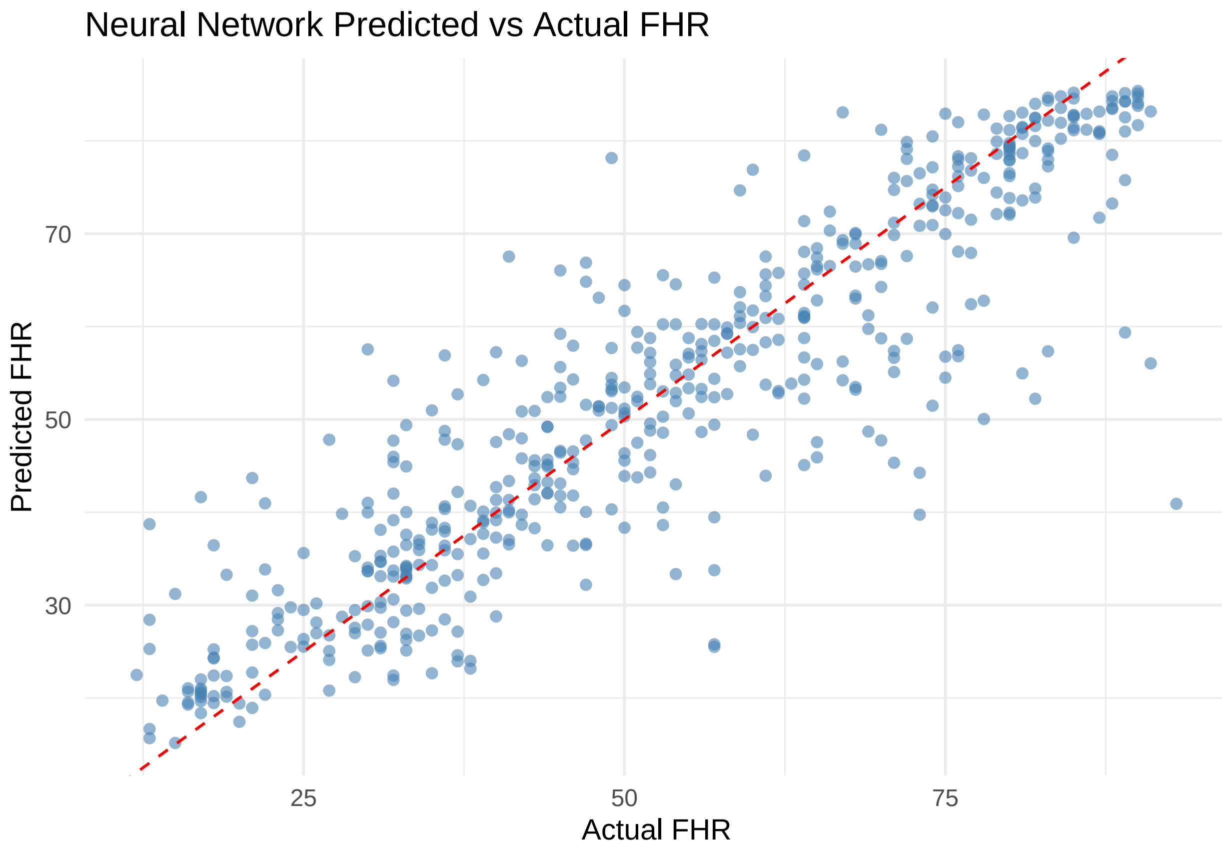

The GLM model using PCs shows R^2: 0.7988, RMSE: 9.51, which can not beat the random forest model using original variables.

### Neural Network Model using PCs

```
## # weights:  126
## initial  value 8406633.917903
## iter  10 value 261667.996904
## iter  20 value 231167.170027
## iter  30 value 227040.640039
## iter  40 value 222663.056038
## iter  50 value 221651.813293
## iter  60 value 220766.718154
## iter  70 value 220489.005995
## iter  80 value 220158.788195
## iter  90 value 219461.476704
## iter 100 value 218279.768828
## final  value 218279.768828
## stopped after 100 iterations
```

```
## Neural Network RMSE: 9.34
```

```
## R-squared: 0.8059
```




The Neural Network Model shows R^2: 0.8059, RMSE: 9.34, which is still not better than the Random Forest Model using the original variables.

In conclusion, after testing different models using PCs, all those PCR models cannot beat the random forest model as predictive models, thus we may want to use Random Forest Model using the original variables to do the next step.