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

```
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 ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(dplyr)
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(tree)
library(lubridate)
library(grid)
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
```

```r
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```r
library(tibble)
library(nnet)
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
sqrt(mean((rf_pred - test$FHR)^2))
```

```
## [1] 9.532225
```

```r
print(rf_model)
```

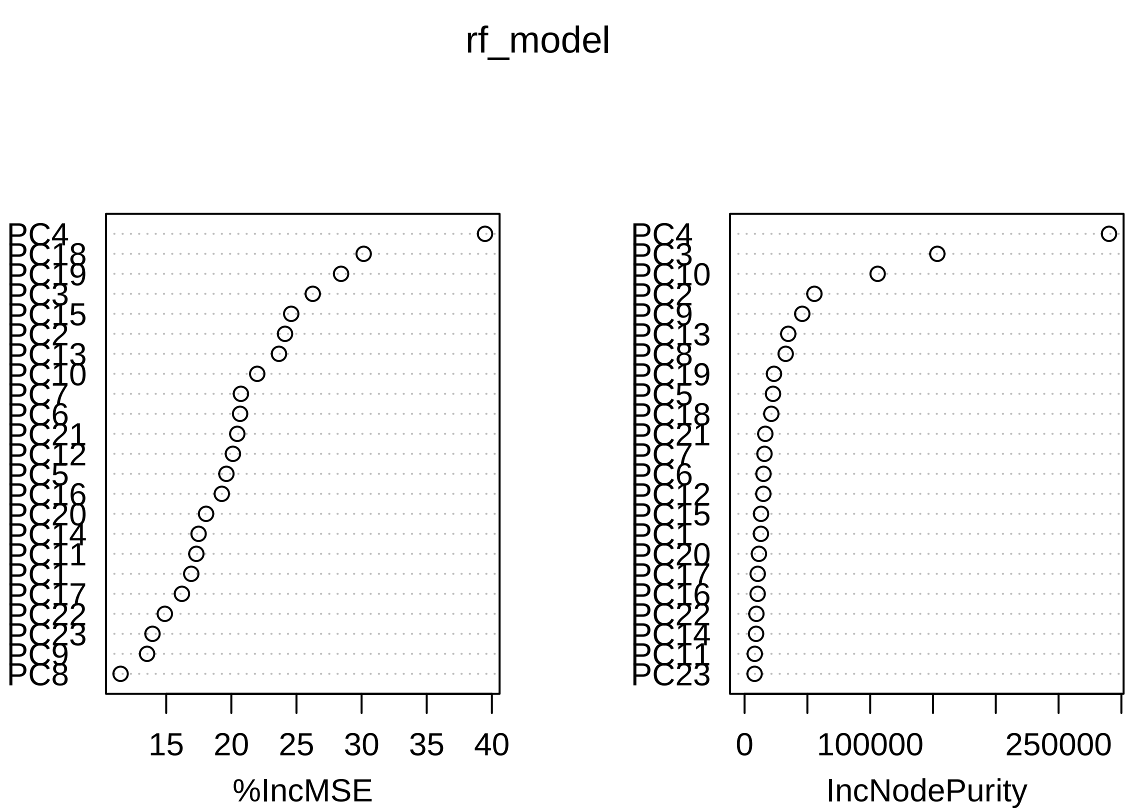
```
##
## Call:
##  randomForest(formula = FHR ~ ., data = train, ntree = 500, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 88.03068
##                     % Var explained: 80.84
```

```r
importance(rf_model)
```

```
##       %IncMSE IncNodePurity
```

```
## PC1  16.92293     13008.293
## PC2  24.12035     55589.880
## PC3  26.25106    153464.108
## PC4  39.47553    290154.957
## PC5  19.61978     22634.064
## PC6  20.67286     15065.539
## PC7  20.73368     15853.371
## PC8  11.49598     32732.631
## PC9  13.52912     45922.756
## PC10 21.98642    105946.854
## PC11 17.31341      8117.065
## PC12 20.11906     14950.694
## PC13 23.65488     34766.783
## PC14 17.48543      9165.862
## PC15 24.59712     13085.395
## PC16 19.26811     10400.721
## PC17 16.20630     10459.969
## PC18 30.15869     21290.156
## PC19 28.42380     23393.330
## PC20 18.06106     11412.169
## PC21 20.45543     16511.912
## PC22 14.89348      9402.842
## PC23 13.94559      7992.357
```

```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
```

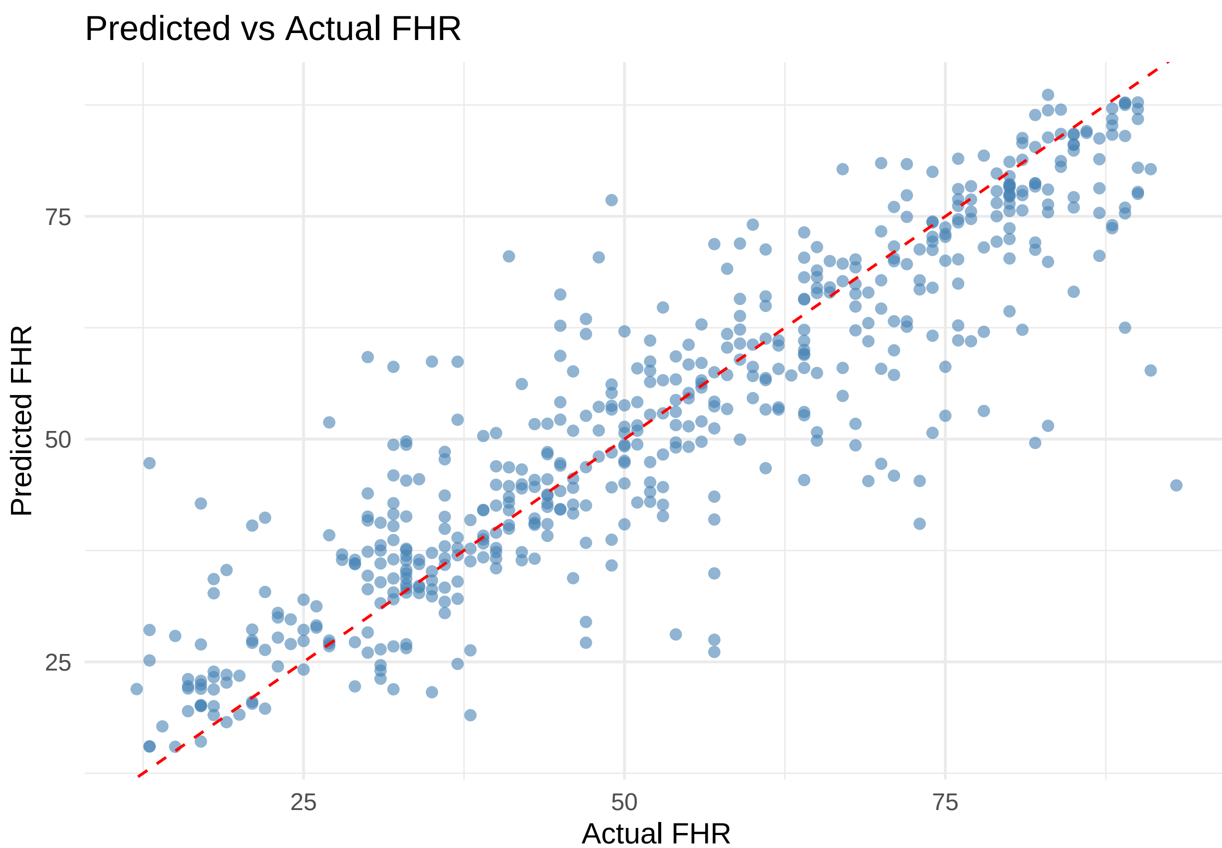
```r
# this table shows how each top 4 components are consisted by different variables
pc_loadings <- pca_result$rotation[, top_pcs]

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.

GLM Model using PCs

```
set.seed(222)
model_glm <- glm(FHR ~ ., data = train, family = gaussian)
summary(model_glm)
```

```
##
## Call:
## glm(formula = FHR ~ ., family = gaussian, data = train)
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 52.49156    0.20899 251.167  < 2e-16 ***
## PC1         -0.03111    0.03325  -0.936  0.34963
## PC2          1.07768    0.06391  16.862  < 2e-16 ***
## PC3         -4.27523    0.15044 -28.417  < 2e-16 ***
## PC4         11.30062    0.15225  74.222  < 2e-16 ***
## PC5          3.87515    0.16411  23.613  < 2e-16 ***
## PC6         -2.79336    0.16308 -17.129  < 2e-16 ***
## PC7         -1.55778    0.18303  -8.511  < 2e-16 ***
## PC8          2.15327    0.19056  11.300  < 2e-16 ***
```

```
## PC9         -2.73956    0.21258 -12.887  < 2e-16 ***
## PC10         1.31722    0.18967   6.945 5.06e-12 ***
## PC11        -0.43307    0.22863  -1.894  0.05834 .
## PC12        -1.13578    0.22846  -4.971 7.20e-07 ***
## PC13         0.95164    0.44863   2.121  0.03402 *
## PC14        -0.84169    0.32464  -2.593  0.00959 **
## PC15         0.61832    0.31233   1.980  0.04787 *
## PC16        -0.64194    0.31519  -2.037  0.04181 *
## PC17         0.21135    0.40769   0.518  0.60422
## PC18        -6.58001    0.46871 -14.039  < 2e-16 ***
## PC19        -9.12188    0.49431 -18.454  < 2e-16 ***
## PC20         1.05648    0.57249   1.845  0.06512 .
## PC21         4.31424    0.58258   7.405 1.90e-13 ***
## PC22        -0.62442    0.65655  -0.951  0.34169
## PC23         1.20714    0.81562   1.480  0.13902
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 90.29505)
##
##     Null deviance: 952116  on 2071  degrees of freedom
## Residual deviance: 184924  on 2048  degrees of freedom
## AIC: 15236
##
## Number of Fisher Scoring iterations: 2
```

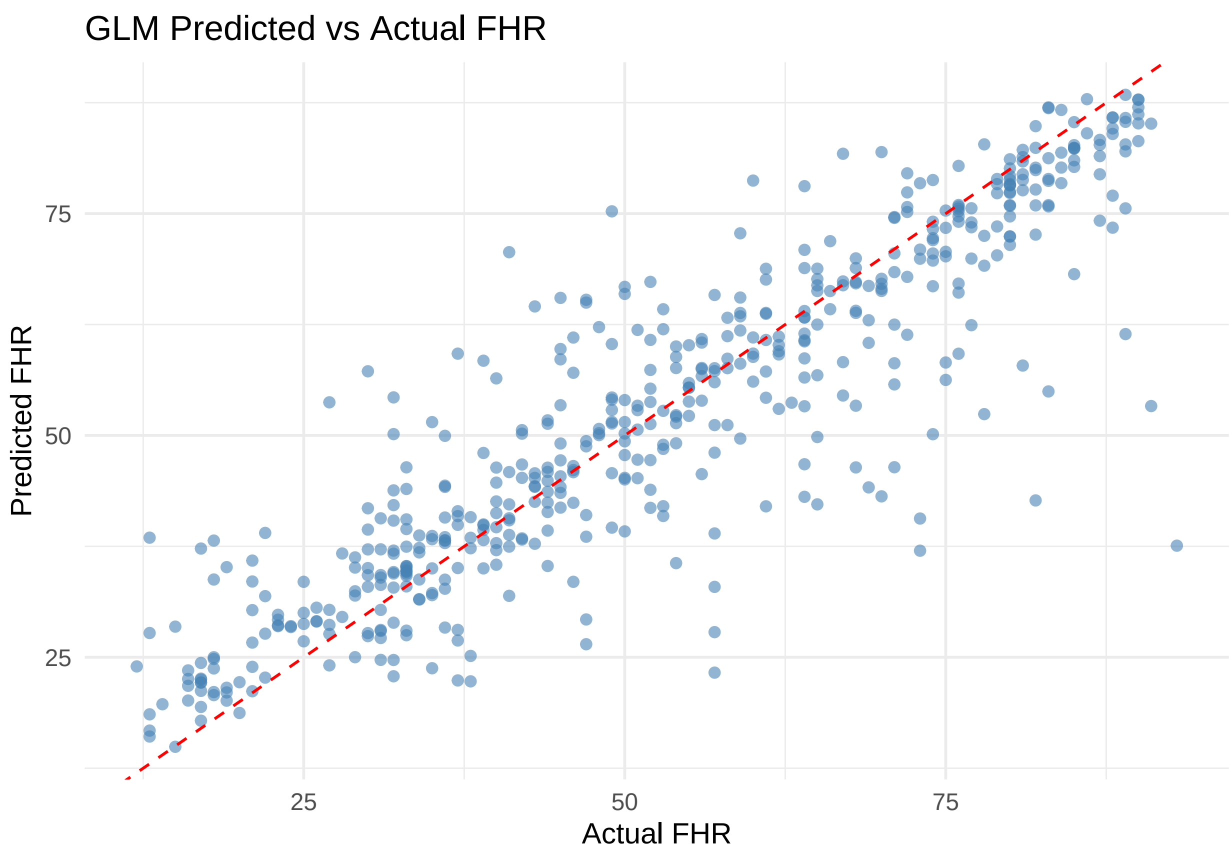
```r
glm_preds <- predict(model_glm, newdata = test)
rmse <- sqrt(mean((glm_preds - test$FHR)^2))
cat("RMSE:", round(rmse, 2))
```

```
## RMSE: 9.51
```

```r
actual <- test$FHR
ss_res2 <- sum((actual - glm_preds)^2)
ss_tot2 <- sum((actual - mean(actual))^2)
r_squared2 <- 1 - (ss_res2 / ss_tot2)
cat("R-squared:", round(r_squared2, 4))
```

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

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



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

```
set.seed(222)
nn_model <- nnet(FHR ~ ., data = pca_df, size = 5, linout = TRUE)
```

```
## # 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
```

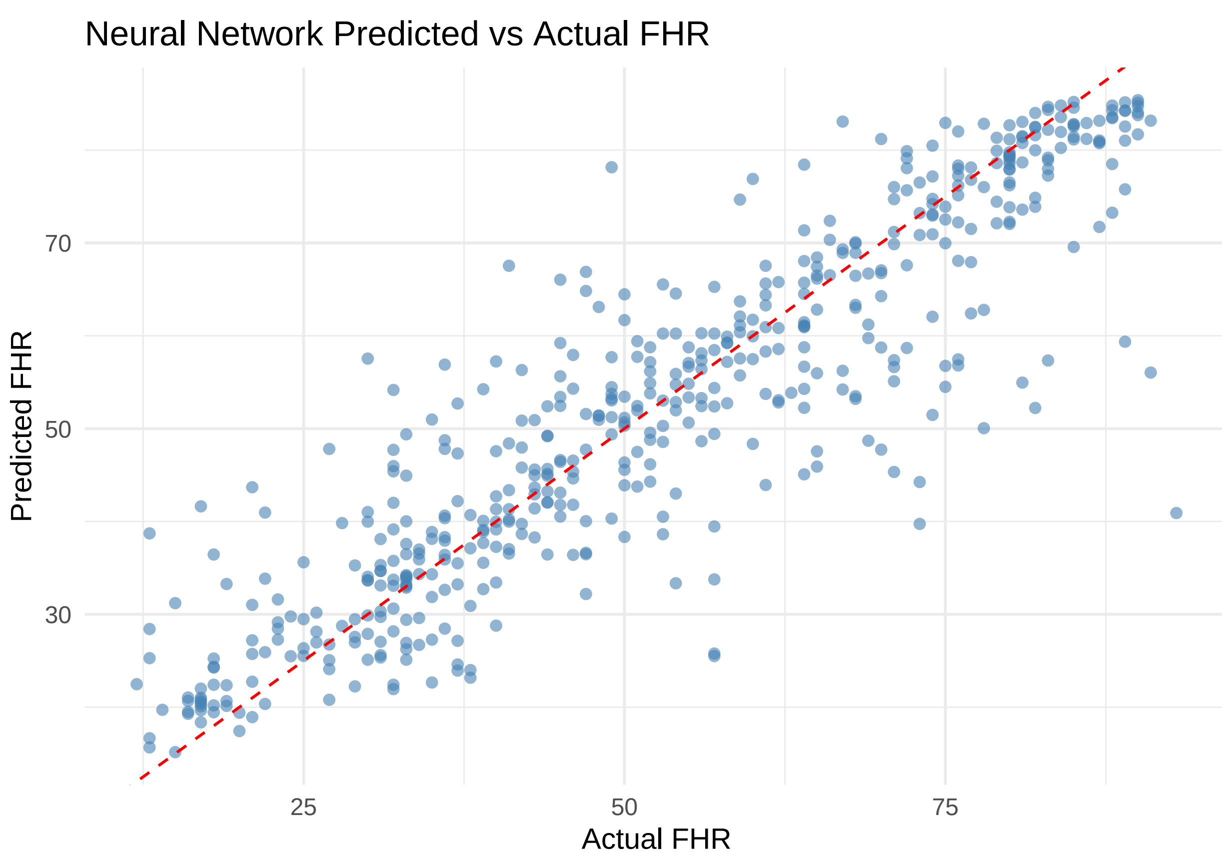
```r
nn_preds <- predict(nn_model, newdata = test)
rmse_nn <- sqrt(mean((nn_preds - test$FHR)^2))
cat("Neural Network RMSE:", round(rmse_nn, 2))
```

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

```r
actual <- test$FHR
ss_res3 <- sum((actual - nn_preds)^2)
ss_tot3 <- sum((actual - mean(actual))^2)
r_squared2 <- 1 - (ss_res3 / ss_tot3)
cat("R-squared:", round(r_squared2, 4))
```

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

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



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.