Lab 10: Regression Tress, Random Forest (Problem 45)

AUTHOR
Aaron Younger

Business Understanding

Merrick Stevens is a sports analyst working for ACE Sports Management, a sports agency that represents over 200 athletes. He is interested in understanding the relationship between an NBA player's salary and his physicality and performance statistics.

Data Understanding

R Version

```
options(scipen = 999)
suppressWarnings(RNGversion("3.5.3"))
```

Libraries

```
library(janitor)
library(DataExplorer)
library(readxl)
library(tidyverse)
library(ggplot2)
library(dplyr)
library(e1071)
library(dlookr)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
```

Import Modeling and Score Dataset

```
nba_data <- read_excel("jaggia_ba_2e_ch13_data.xlsx",
    sheet = "NBA_Data")
nba_data <- clean_names(nba_data)
View(nba_data)

nba_score <- read_excel("jaggia_ba_2e_ch13_data.xlsx",
    sheet = "NBA_Score")
nba_score <- clean_names(nba_score)
```

Data Set Key:
- player_number: Represents the identification number assigned to each NBA player.
- salary (Dependent Variable) : Indicates the annual salary earned by the NBA player (in dollars).
- age: Shows the player’s age in years.
- height: Represents the player’s height (typically measured in inches).
- weight: Represents the player’s weight (typically measured in pounds).
- games_played: The total number of games the player has participated in during the season.
- games_started: The total number of games in which the player was part of the starting lineup.
- minutes_per_game: The average number of minutes the player spends on the court per game.
- FG_made: Total number of field goals successfully made by the player.
- FG_attempted: Total number of field goal attempts taken by the player.
- FG_percent: Field goal shooting percentage (FG_made ÷ FG_attempted × 100).
- 3P_made: Total number of three-point shots successfully made by the player.
- 3P_attempted: Total number of three-point shot attempts taken by the player.
- 3P_percent: Three-point shooting percentage (3P_made ÷ 3P_attempted × 100).
- FT_made: Total number of free throws successfully made by the player.
- FT_attempted: Total number of free throw attempts taken by the player.
- FT_percent: Free throw shooting percentage (FT_made ÷ FT_attempted × 100).
- offensive_rebounds: The total number of rebounds collected on the offensive side of the court.
- defensive_rebounds: The total number of rebounds collected on the defensive side of the court.
- assists: The number of times a player passes the ball leading directly to a made basket.
- blocks: The total number of opponent shots blocked by the player.
- steals: The total number of times the player takes the ball away from an opponent.
- personal_fouls: The total number of personal fouls committed by the player.
- turnovers: The total number of times the player loses possession of the ball to the opposing team.
- points: The total number of points scored by the player throughout the season.

Dataset Exploration

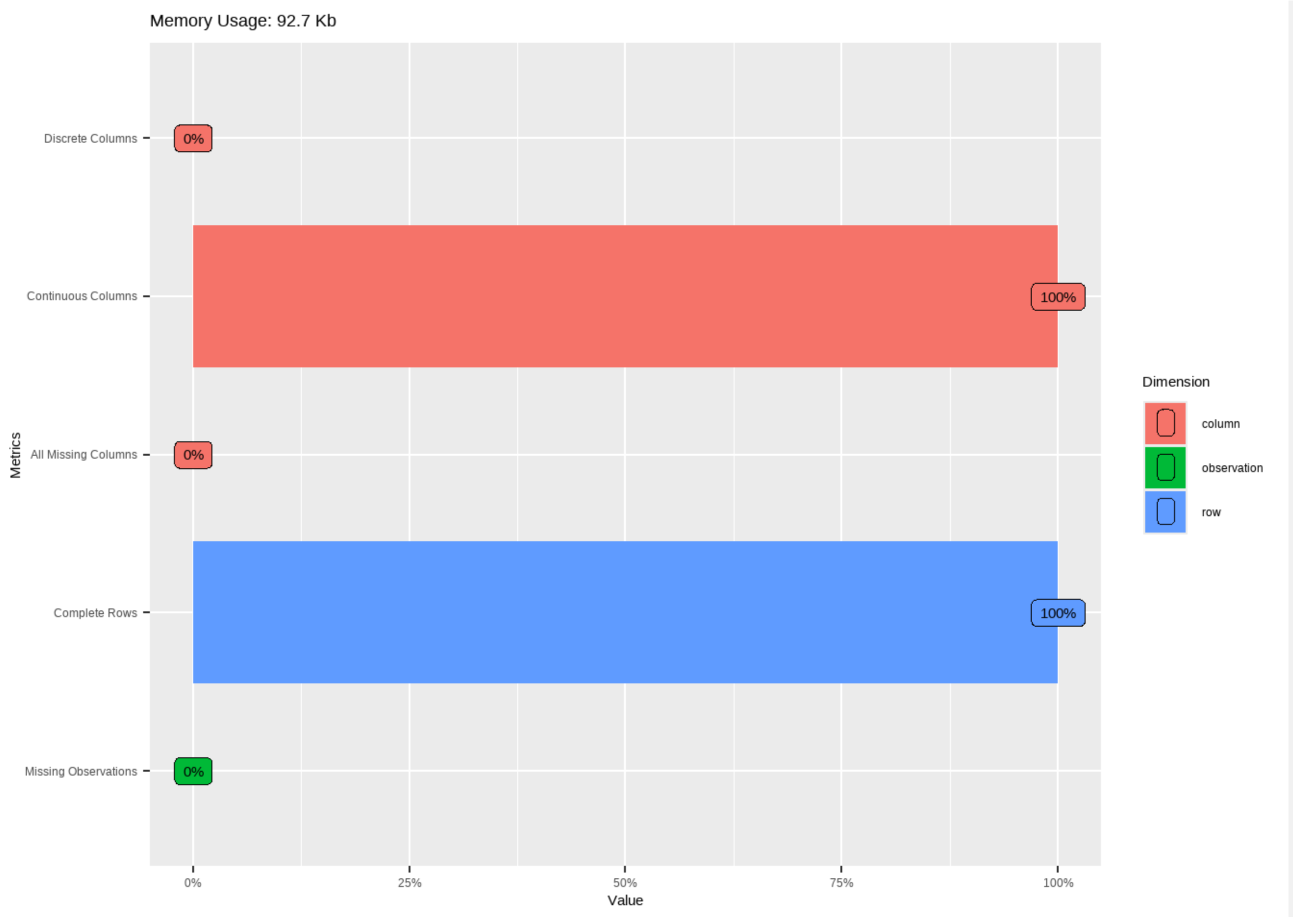
```
nba_data %>% head()
```

```
# A tibble: 6 × 25
  player_number   salary   age height weight games_played games_started
          <dbl>    <dbl> <dbl>  <dbl>  <dbl>        <dbl>         <dbl>
1             1   947276    36     79    260          966           838
2             2 25000000    37     78    212         1346          1198
3             3  4088019    39     78    220         1274           954
4             4  5675000    36     77    195         1100           432
5             5  5250000    40     83    250         1392          1389
6             6  8500000    39     83    240         1462          1425
# ℹ 18 more variables: minutes_per_game <dbl>, fg_made <dbl>,
#   fg_attempted <dbl>, fg_percent <dbl>, x3p_made <dbl>, x3p_attempted <dbl>,
#   x3p_percent <dbl>, ft_made <dbl>, ft_attempted <dbl>, ft_percent <dbl>,
#   offensive_rebounds <dbl>, defensive_rebounds <dbl>, assists <dbl>,
#   blocks <dbl>, steals <dbl>, personal_fouls <dbl>, turnovers <dbl>,
#   points <dbl>
```

```
nba_data %>% tail()
```

```
# A tibble: 6 × 25
  player_number  salary   age height weight games_played games_started
          <dbl>   <dbl> <dbl>  <dbl>  <dbl>        <dbl>         <dbl>
1           440 5103120    20     77    195           80            48
2           441 1733040    19     78    202           70             6
3           442 1140240    21     83    200           24             4
4           443 1131960    20     81    220            5             0
5           444 3102240    20     77    200           68            66
6           445  525093    23     79    185           12             0
# ℹ 18 more variables: minutes_per_game <dbl>, fg_made <dbl>,
#   fg_attempted <dbl>, fg_percent <dbl>, x3p_made <dbl>, x3p_attempted <dbl>,
#   x3p_percent <dbl>, ft_made <dbl>, ft_attempted <dbl>, ft_percent <dbl>,
#   offensive_rebounds <dbl>, defensive_rebounds <dbl>, assists <dbl>,
#   blocks <dbl>, steals <dbl>, personal_fouls <dbl>, turnovers <dbl>,
#   points <dbl>
```

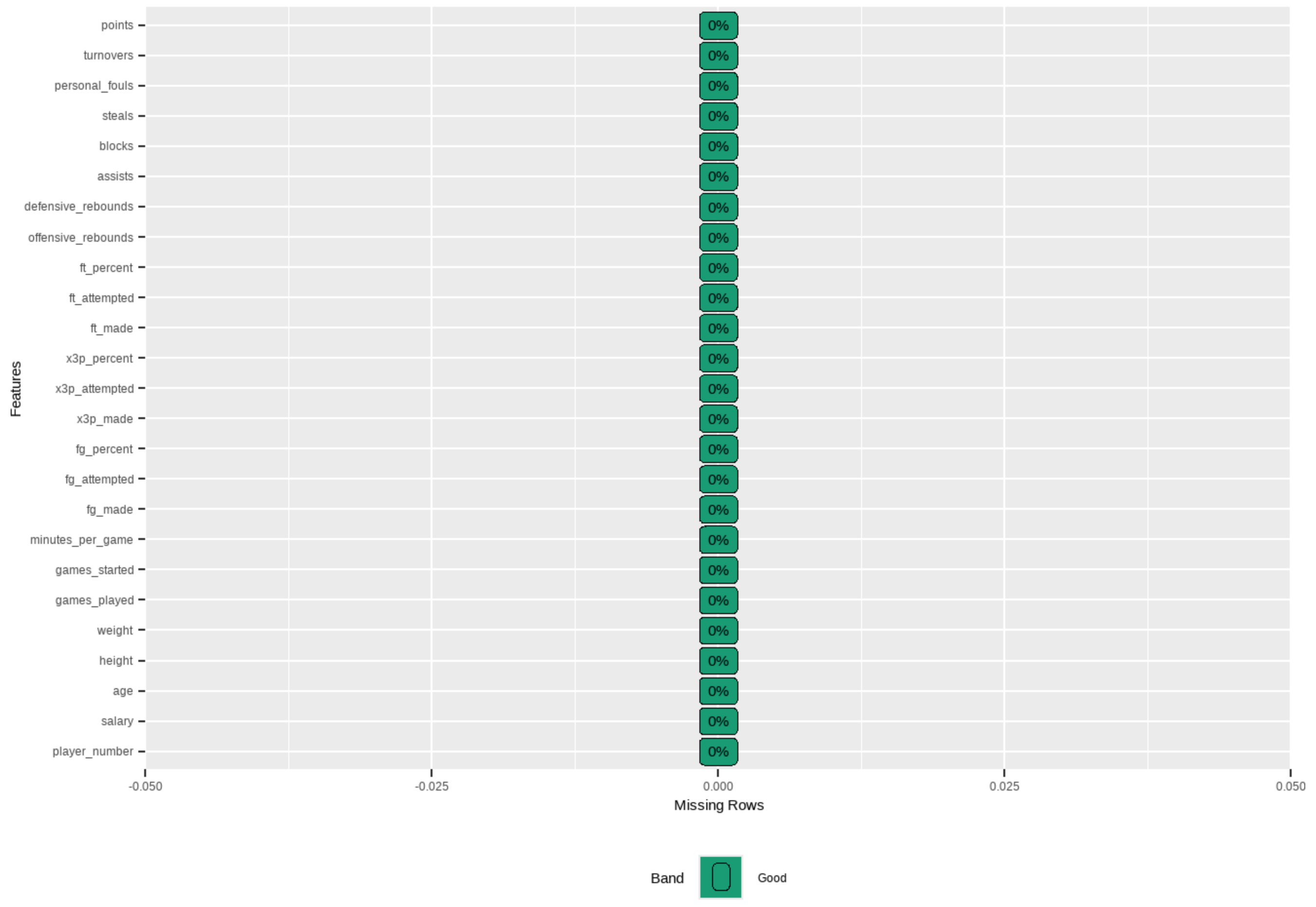
```
nba_data %>% plot_intro()
```



```r
nba_data %>% glimpse()
```

```
Rows: 445
Columns: 25
$ player_number      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, …
$ salary             <dbl> 947276, 25000000, 4088019, 5675000, 5250000, 850000…
$ age                <dbl> 36, 37, 39, 36, 40, 39, 38, 40, 36, 38, 37, 38, 38,…
$ height             <dbl> 79, 78, 78, 77, 83, 83, 78, 75, 80, 82, 84, 79, 74,…
$ weight             <dbl> 260, 212, 220, 195, 250, 240, 205, 200, 218, 250, 2…
$ games_played       <dbl> 966, 1346, 1274, 1100, 1392, 1462, 923, 1304, 1012,…
$ games_started      <dbl> 838, 1198, 954, 432, 1389, 1425, 349, 939, 569, 354…
$ minutes_per_game   <dbl> 32.4, 36.1, 32.4, 30.9, 34.0, 34.5, 26.3, 30.9, 27.…
$ fg_made            <dbl> 4.8, 8.7, 6.8, 5.4, 7.4, 7.2, 4.6, 4.6, 4.0, 2.4, 7…
$ fg_attempted       <dbl> 11.6, 19.5, 15.5, 13.1, 14.6, 14.5, 10.1, 10.0, 8.6…
$ fg_percent         <dbl> 0.415, 0.447, 0.439, 0.410, 0.506, 0.497, 0.450, 0.…
$ x3p_made           <dbl> 1.2, 1.4, 1.5, 1.8, 0.0, 0.1, 1.5, 0.1, 1.6, 0.0, 1…
$ x3p_attempted      <dbl> 3.5, 4.1, 4.1, 5.0, 0.1, 0.4, 3.9, 0.7, 3.8, 0.0, 3…
$ x3p_percent        <dbl> 0.340, 0.329, 0.373, 0.349, 0.179, 0.275, 0.369, 0.…
$ ft_made            <dbl> 2.7, 6.2, 3.7, 3.0, 4.2, 3.3, 3.4, 3.1, 1.3, 1.0, 5…
$ ft_attempted       <dbl> 3.7, 7.4, 4.6, 3.5, 6.1, 4.2, 4.2, 3.8, 1.7, 1.6, 5…
$ ft_percent         <dbl> 0.716, 0.837, 0.800, 0.861, 0.696, 0.789, 0.827, 0.…
$ offensive_rebounds <dbl> 1.2, 1.1, 1.2, 0.4, 2.8, 2.2, 0.7, 1.1, 0.6, 1.8, 1…
$ defensive_rebounds <dbl> 3.4, 4.1, 3.4, 2.0, 8.1, 7.8, 3.0, 2.6, 3.7, 2.9, 6…
$ assists            <dbl> 2.7, 4.7, 3.4, 3.6, 3.0, 3.7, 4.0, 6.5, 2.6, 0.4, 2…
$ blocks             <dbl> 0.5, 0.5, 0.6, 0.2, 2.2, 1.4, 0.3, 0.2, 0.2, 0.6, 0…
$ steals             <dbl> 1.8, 1.4, 1.1, 1.0, 0.7, 1.3, 1.4, 1.2, 0.6, 0.4, 0…
$ personal_fouls     <dbl> 2.7, 2.5, 2.8, 1.6, 2.4, 2.4, 2.1, 2.2, 2.0, 2.1, 2…
$ turnovers          <dbl> 1.8, 3.0, 1.9, 2.0, 2.4, 2.2, 2.1, 2.4, 1.5, 0.9, 1…
$ points             <dbl> 13.5, 25.0, 18.8, 15.5, 19.0, 17.8, 14.0, 12.5, 10.…
```

```r
nba_data %>% plot_missing()
```



```r
nba_data %>% str()
```

```
tibble [445 × 25] (S3: tbl_df/tbl/data.frame)
 $ player_number     : num [1:445] 1 2 3 4 5 6 7 8 9 10 ...
 $ salary            : num [1:445] 947276 25000000 4088019 5675000 5250000 ...
 $ age               : num [1:445] 36 37 39 36 40 39 38 40 36 38 ...
 $ height            : num [1:445] 79 78 78 77 83 83 78 75 80 82 ...
 $ weight            : num [1:445] 260 212 220 195 250 240 205 200 218 250 ...
 $ games_played      : num [1:445] 966 1346 1274 1100 1392 ...
 $ games_started     : num [1:445] 838 1198 954 432 1389 ...
 $ minutes_per_game  : num [1:445] 32.4 36.1 32.4 30.9 34 34.5 26.3 30.9 27.3 15.8 ...
 $ fg_made           : num [1:445] 4.8 8.7 6.8 5.4 7.4 7.2 4.6 4.6 4 2.4 ...
 $ fg_attempted      : num [1:445] 11.6 19.5 15.5 13.1 14.6 14.5 10.1 10 8.6 4.9 ...
 $ fg_percent        : num [1:445] 0.415 0.447 0.439 0.41 0.506 0.497 0.45 0.461 0.459 0.486 ...
 $ x3p_made          : num [1:445] 1.2 1.4 1.5 1.8 0 0.1 1.5 0.1 1.6 0 ...
 $ x3p_attempted     : num [1:445] 3.5 4.1 4.1 5 0.1 0.4 3.9 0.7 3.8 0 ...
 $ x3p_percent       : num [1:445] 0.34 0.329 0.373 0.349 0.179 0.275 0.369 0.217 0.407 0 ...
 $ ft_made           : num [1:445] 2.7 6.2 3.7 3 4.2 3.3 3.4 3.1 1.3 1 ...
 $ ft_attempted      : num [1:445] 3.7 7.4 4.6 3.5 6.1 4.2 4.2 3.8 1.7 1.6 ...
 $ ft_percent        : num [1:445] 0.716 0.837 0.8 0.861 0.696 0.789 0.827 0.807 0.769 0.64 ...
 $ offensive_rebounds: num [1:445] 1.2 1.1 1.2 0.4 2.8 2.2 0.7 1.1 0.6 1.8 ...
 $ defensive_rebounds: num [1:445] 3.4 4.1 3.4 2 8.1 7.8 3 2.6 3.7 2.9 ...
 $ assists           : num [1:445] 2.7 4.7 3.4 3.6 3 3.7 4 6.5 2.6 0.4 ...
 $ blocks           : num [1:445] 0.5 0.5 0.6 0.2 2.2 1.4 0.3 0.2 0.2 0.6 ...
 $ steals           : num [1:445] 1.8 1.4 1.1 1 0.7 1.3 1.4 1.2 0.6 0.4 ...
 $ personal_fouls   : num [1:445] 2.7 2.5 2.8 1.6 2.4 2.4 2.1 2.2 2 2.1 ...
 $ turnovers        : num [1:445] 1.8 3 1.9 2 2.4 2.2 2.1 2.4 1.5 0.9 ...
 $ points           : num [1:445] 13.5 25 18.8 15.5 19 17.8 14 12.5 10.8 5.8 ...
```

```
nba_data %>% ncol()
```

```
[1] 25
```

```
nba_data %>% nrow()
```

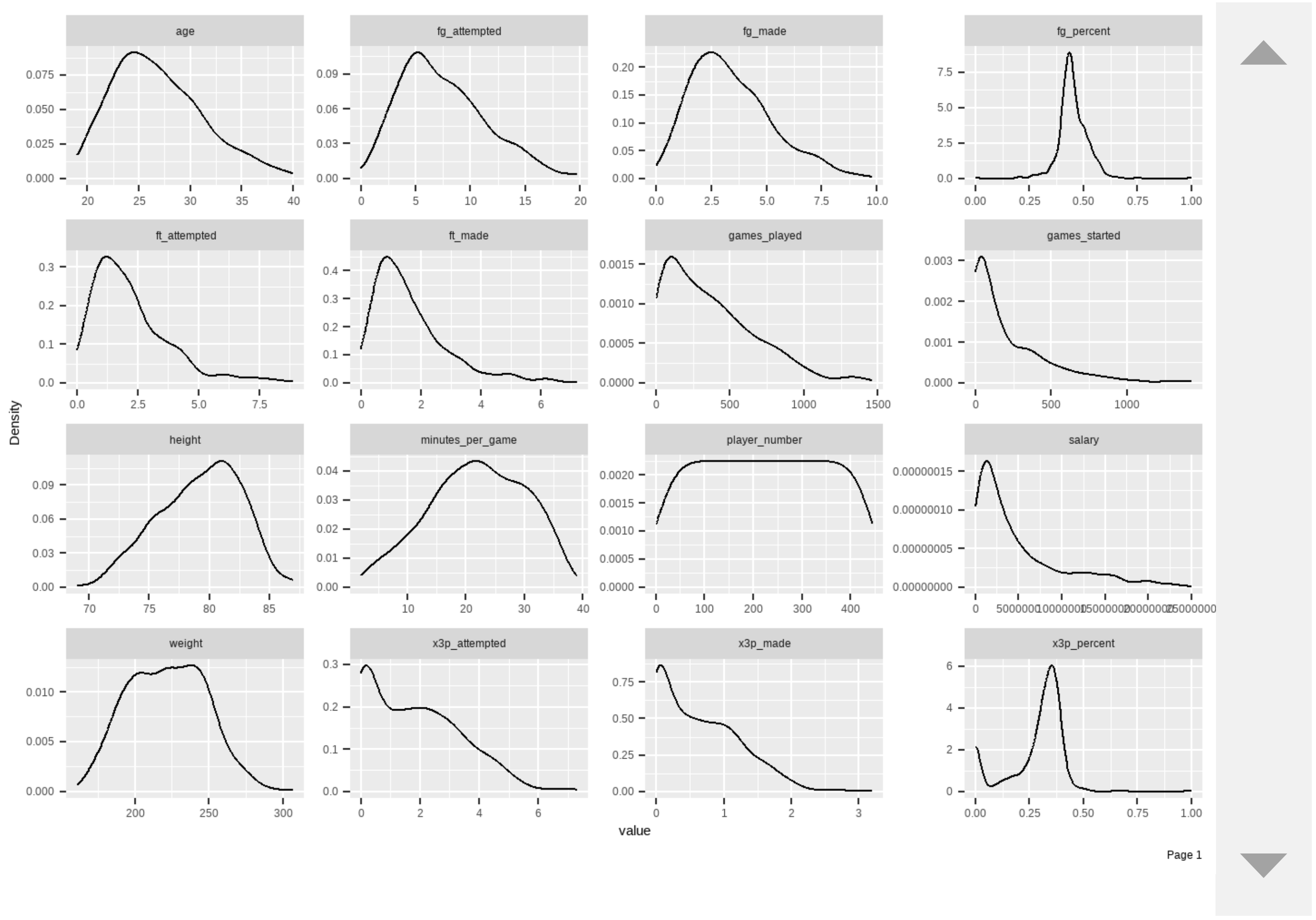
```
[1] 445
```

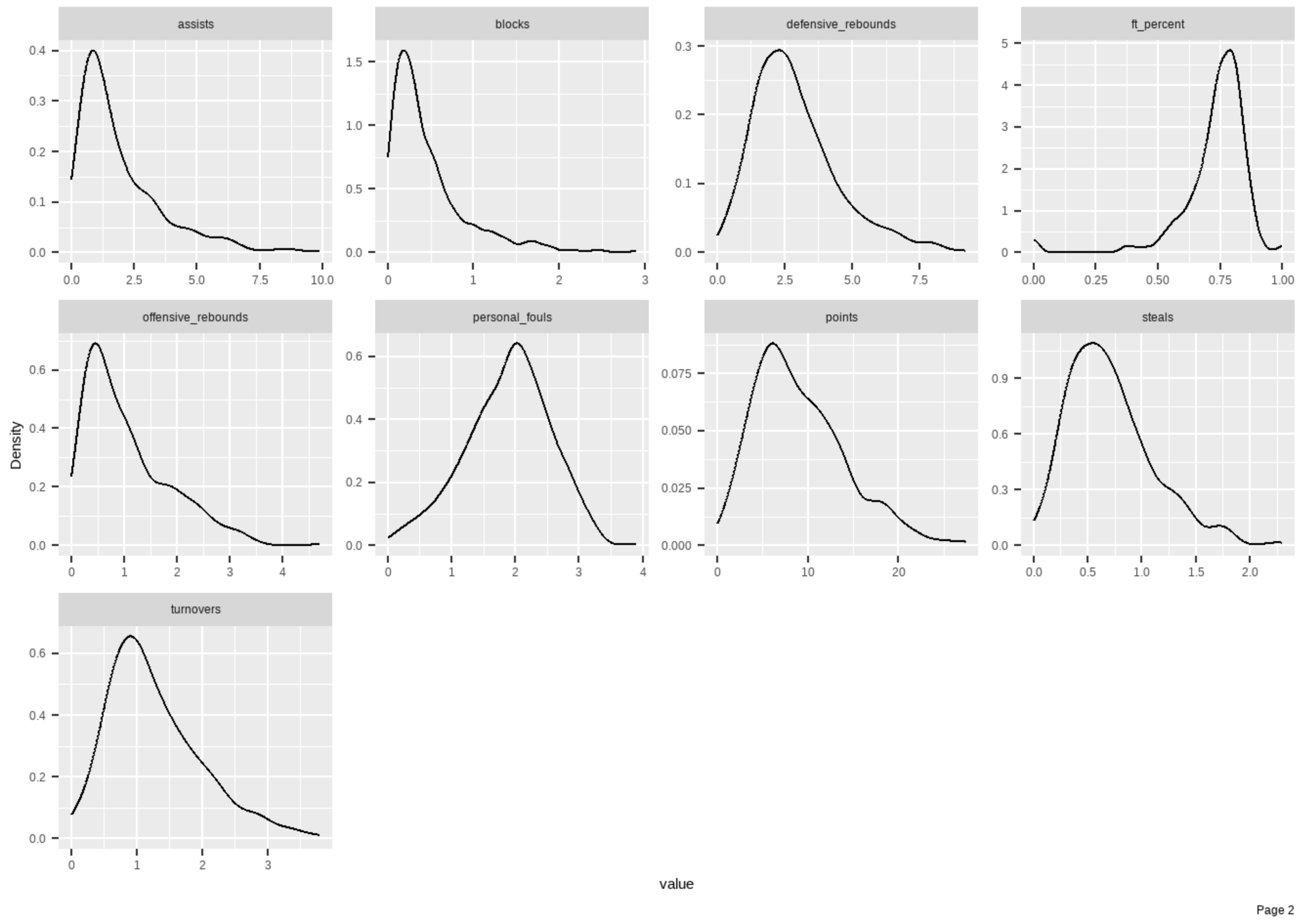
Comments on Dataset Exploration:

This dataset is made of only numeric variables, contains no categorical variables. This dataset also contains no missing values.

Variable EDA

```
nba_data %>% plot_density()
```

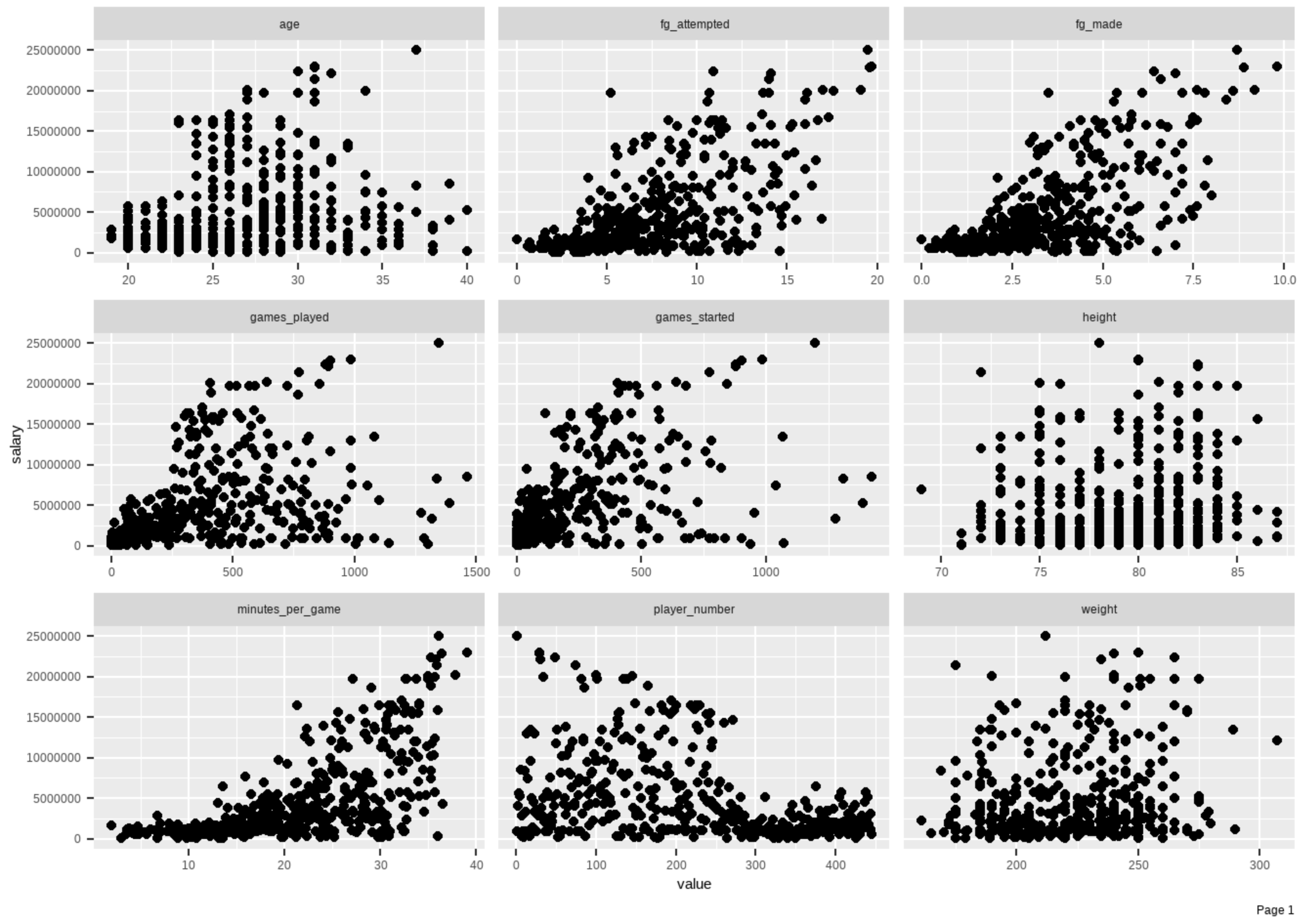


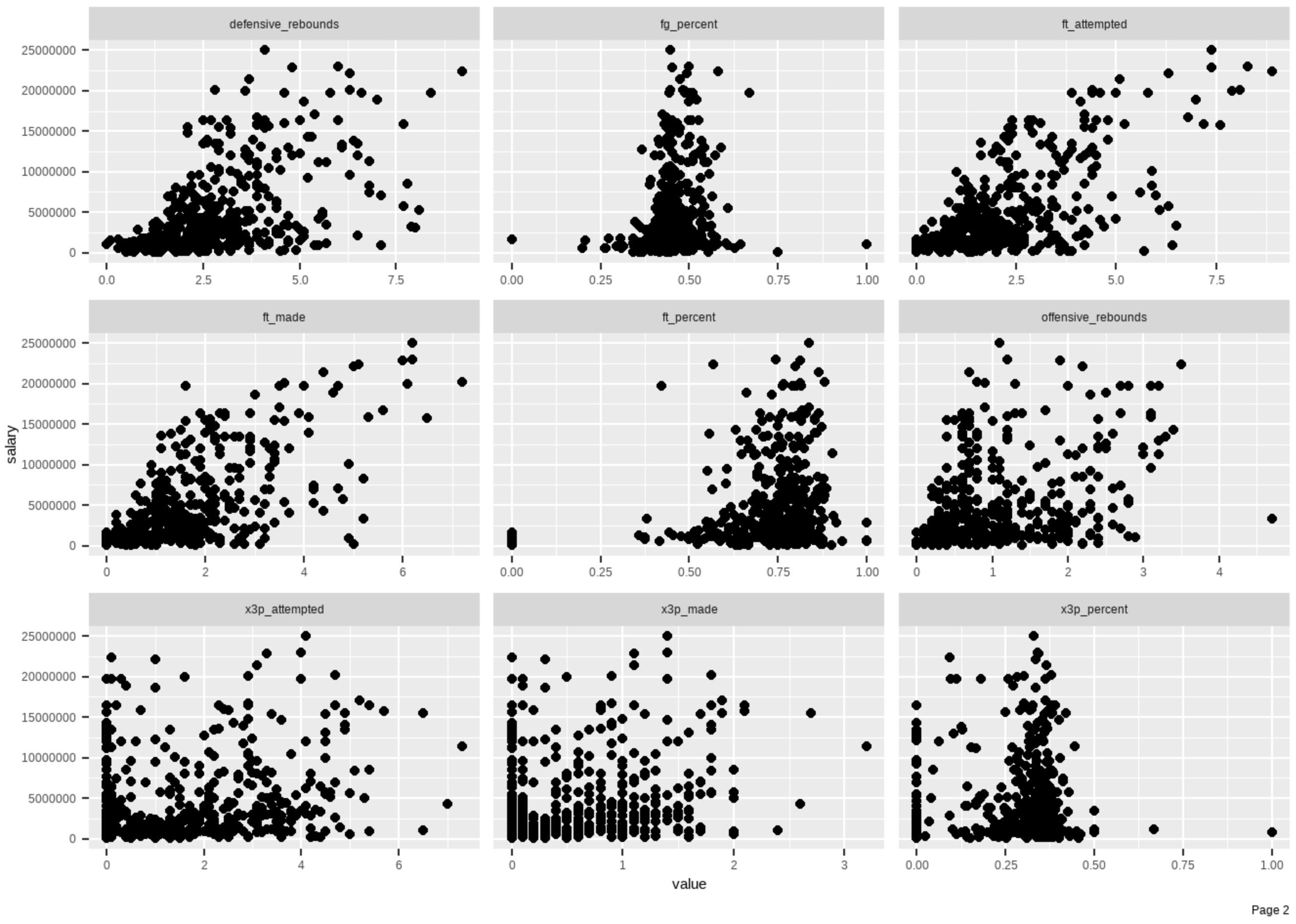


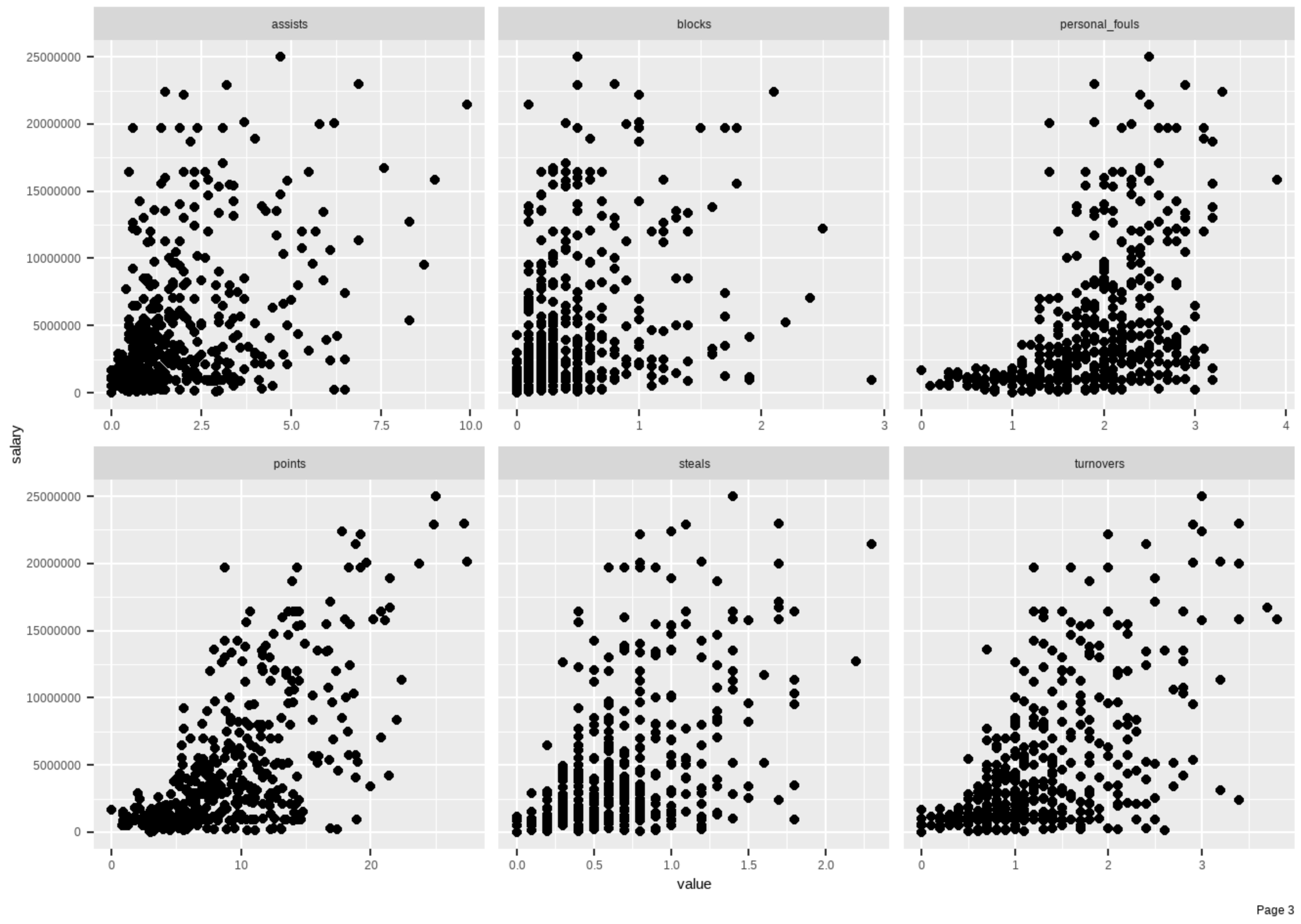
```
diagnose_outlier(nba_data) ## Although there are outlier counts in almost all variables it would
        not be ideal to remove people that are outliers in categories because that could
        determine salary, ex if a person is an outlier because he has more blocks but block
        influence salary that player would earn more salary.
```

```
# A tibble: 25 × 6
   variables    outliers_cnt outliers_ratio outliers_mean with_mean without_mean
   <chr>               <int>          <dbl>         <dbl>     <dbl>        <dbl>
 1 player_numb…            0          0             NaN     2.23e2        223
 2 salary                 35          7.87    18164924.     4.84e6    3705946.
 3 age                     2          0.449         40      2.69e1         26.8
 4 height                  1          0.225         69      7.92e1         79.2
 5 weight                  1          0.225        307      2.22e2        221.
 6 games_played            8          1.80        1340.     3.70e2        352.
 7 games_start…           20          4.49        1015.     2.12e2        174.
 8 minutes_per…            0          0             NaN     2.19e1         21.9
 9 fg_made                 6          1.35           8.93   3.41e0          3.33
10 fg_attempted            4          0.899         19.5    7.52e0          7.41
# ℹ 15 more rows
```

```
nba_data %>% plot_scatterplot(by="salary")
```


age
fg_attempted
fg_made
games_played
games_started
height
minutes_per_game
player_number
weight
salary
value
25000000
20000000
15000000
10000000
5000000
0
Page 1


defensive_rebounds
fg_percent
ft_attempted
ft_made
ft_percent
offensive_rebounds
x3p_attempted
x3p_made
x3p_percent
salary
value
25000000
20000000
15000000
10000000
5000000
0
Page 2



```r
skewness_values <- sapply(nba_data, skewness, na.rm = TRUE)
skew_table <- data.frame(
  Variable = names(skewness_values),
  Skewness = round(skewness_values, 3)
)
print(skew_table) ## Prints skew values of all numeric variables
```

```
                        Variable Skewness
player_number      player_number    0.000
salary                    salary    1.569
age                          age    0.601
height                    height   -0.301
weight                    weight    0.122
games_played        games_played    0.981
games_started      games_started    1.810
minutes_per_game minutes_per_game   -0.215
fg_made                  fg_made    0.738
fg_attempted        fg_attempted    0.648
fg_percent            fg_percent    0.556
x3p_made                x3p_made    0.845
x3p_attempted      x3p_attempted    0.642
x3p_percent          x3p_percent   -0.697
ft_made                  ft_made    1.502
```

```
ft_attempted             ft_attempted    1.398
ft_percent                 ft_percent   -2.777
offensive_rebounds offensive_rebounds    1.121
defensive_rebounds defensive_rebounds    1.063
assists                       assists    1.668
blocks                         blocks    1.918
steals                         steals    0.890
personal_fouls         personal_fouls   -0.308
turnovers                   turnovers    0.881
points                         points    0.829
```

Simple Random Forest To Find Variables with low importance

```r
rf_model <- randomForest(salary ~ ., data = nba_data, importance = TRUE)
imp <- importance(rf_model)

imp_sorted <- imp[order(imp[, "%IncMSE"]), , drop = FALSE]

# if you want both columns
imp_table <- data.frame(
  Variable       = rownames(imp_sorted),
  `%IncMSE`      = imp_sorted[, "%IncMSE"],
  IncNodePurity  = imp_sorted[, "IncNodePurity"],
  row.names = NULL,
  check.names = FALSE
)
print(imp_table)
```

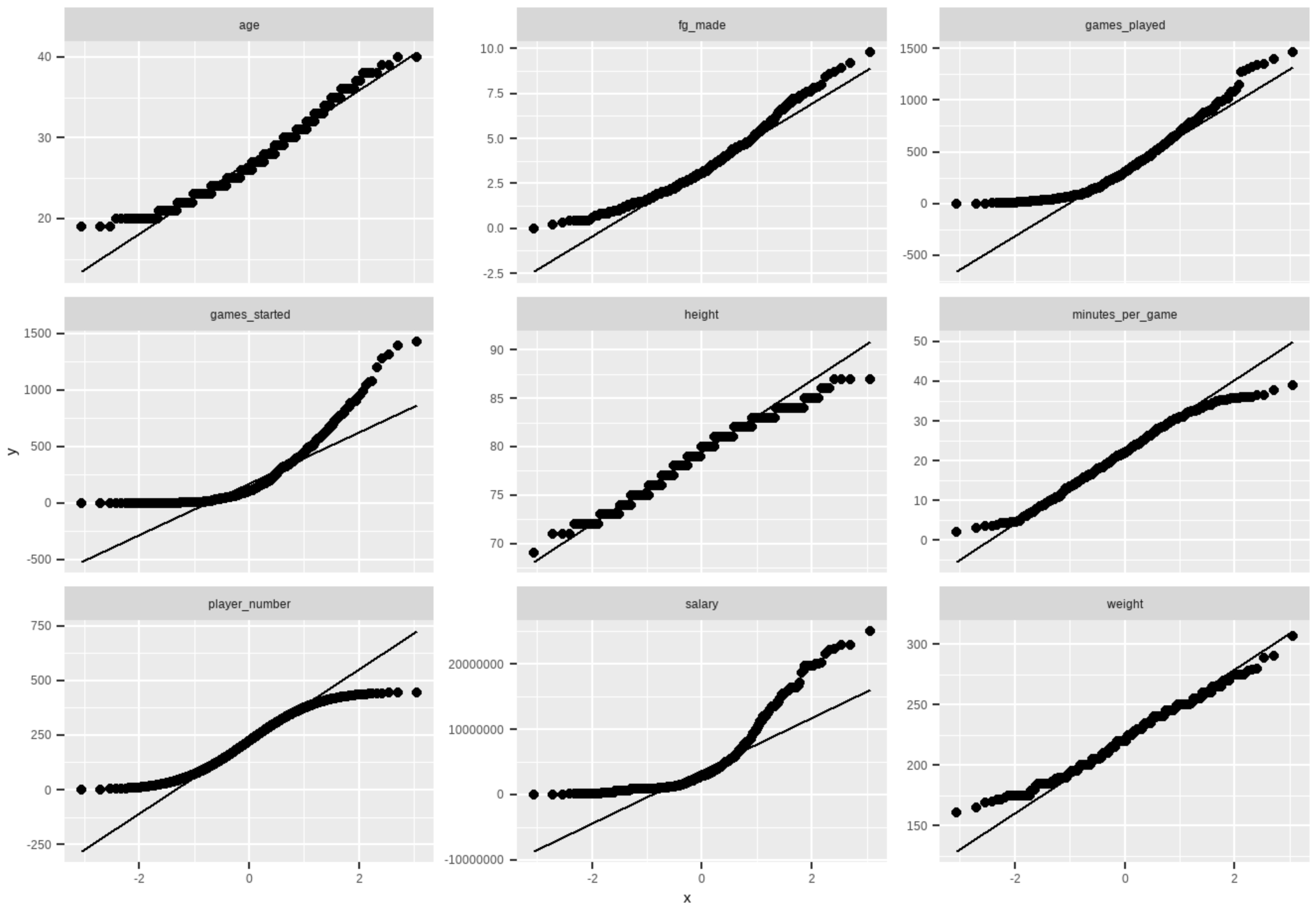
```
             Variable   %IncMSE    IncNodePurity
1       x3p_attempted  1.368133  104998860067654
2              weight  2.359012  153857781823841
3              height  2.589529  100537444272033
4         x3p_percent  2.790541  139173589374633
5          ft_percent  3.496960  185432723489313
6             assists  3.714161  137649255719861
7           turnovers  4.388886  230125973562826
8            x3p_made  4.403810   78264698949576
9      personal_fouls  5.449440  163991577429311
10             steals  5.807849  140729659538850
11       ft_attempted  6.799977  836759317636665
12         fg_percent  7.406890  266522088603520
13             blocks  7.486030  191505252733787
14 defensive_rebounds  7.887766  355276626549753
15            ft_made  8.034190  766716840070486
16 offensive_rebounds  9.128378  298226729291832
17            fg_made  9.202119  770673532152055
18       fg_attempted  9.391082  433064503004834
19             points 10.291604 1120464340915471
20    minutes_per_game 14.179956 1426834891603088
```

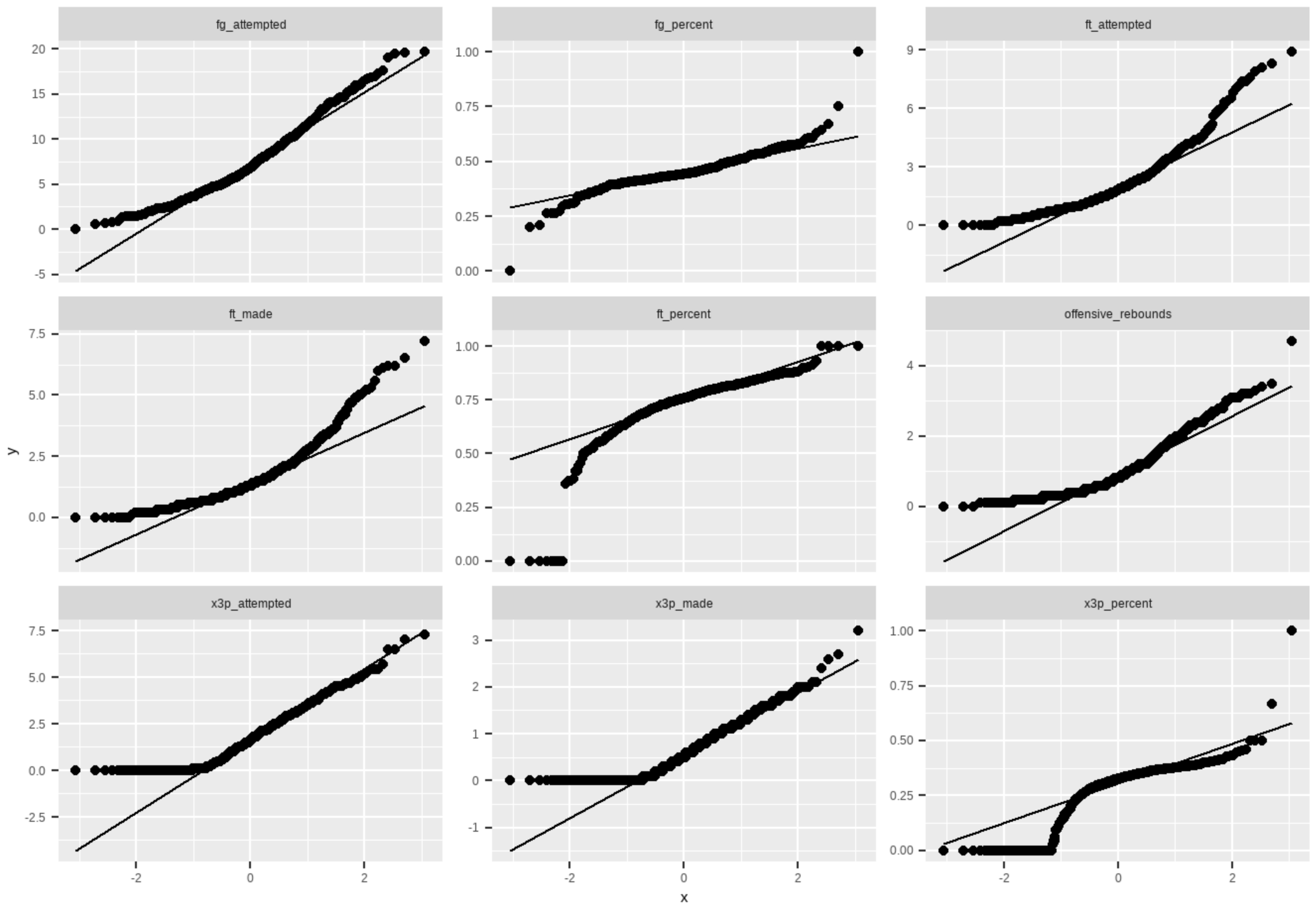
```
21              age 16.430643  526371897102351
22    player_number 17.654092  734368361264811
23     games_played 18.157244  742989010145924
24    games_started 18.636092 2057591595861610
```

Comments on Simple Random Forest:
The variable importance feature of the random forest can help initially identify variables that are not important when compared to salary, the dependent variable. This helps get insight into potential variables to drop due to their lack of importance. Any variable that has a value of <5 for %IncMSE can be safely dropped.

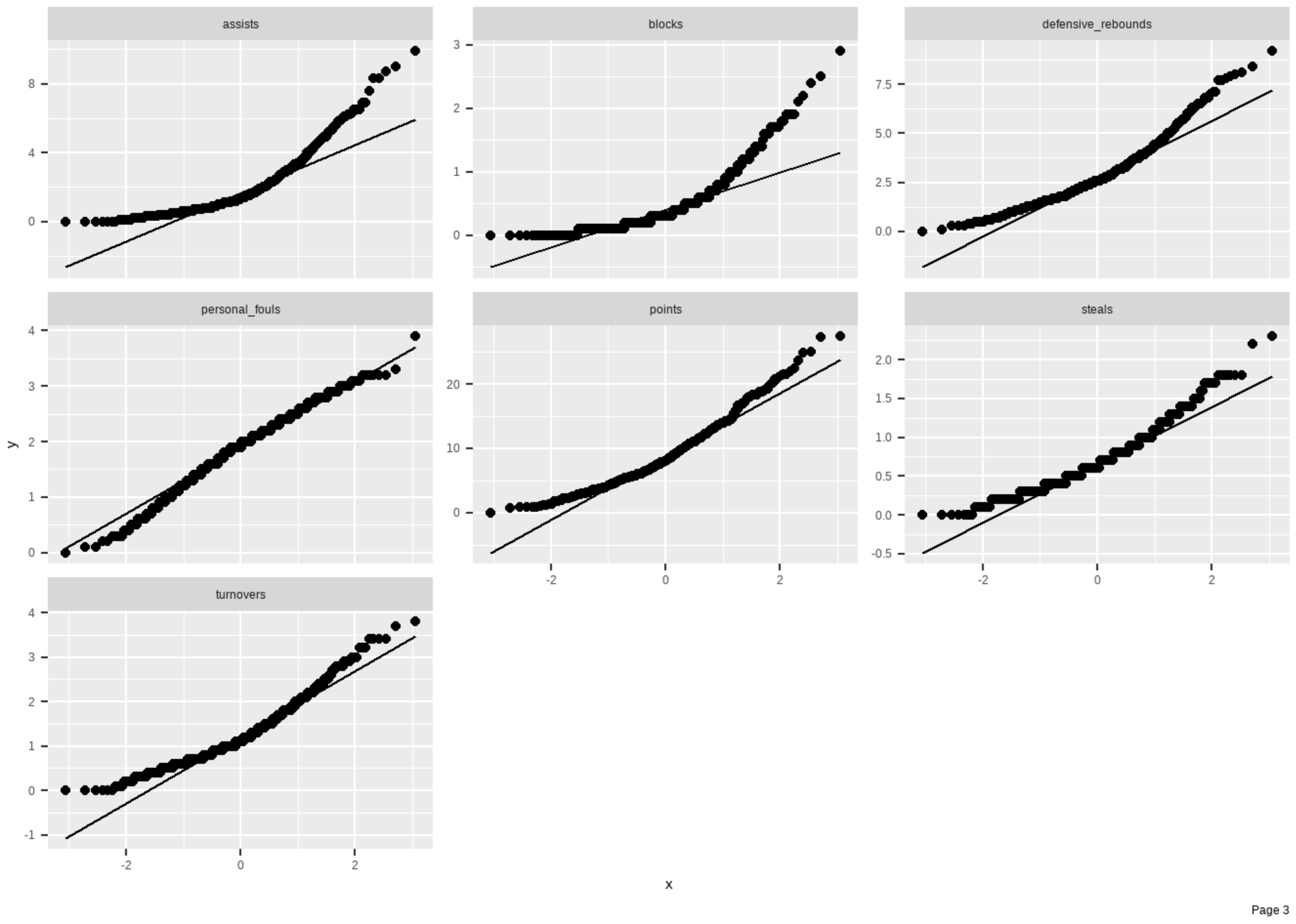
Q-Q Plots

```
nba_data %>% plot_qq()
```



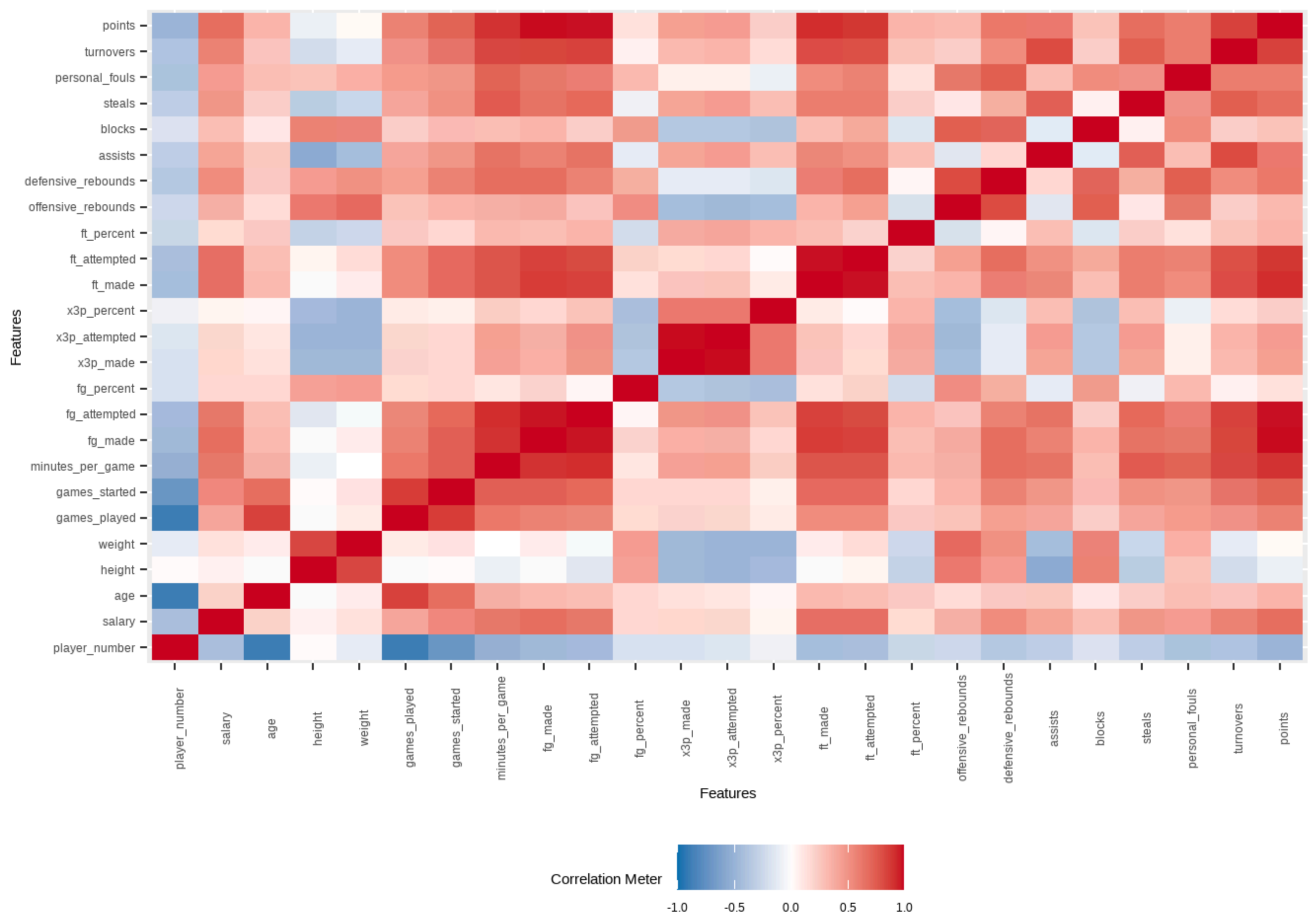

fg_attempted
fg_percent
ft_attempted
ft_made
ft_percent
offensive_rebounds
x3p_attempted
x3p_made
x3p_percent
y
x

Page 2



Correlation Matrix

```r
nba_data %>% plot_correlation()
```



Data Prepartion

Drop Player Number as a predictor

```
nba_data <- nba_data %>%
  select(-player_number)
```

Partition

```
myindex <- createDataPartition(nba_data$salary, p=0.7, list = FALSE)
trainSet <- nba_data[myindex,]
testSet <- nba_data[-myindex,]

cat("Mean of Dependent Varaible \n")
```

```
Mean of Dependent Varaible
```

```
mean(nba_data$salary)
```

```
[1] 4843169
```

```
cat("Mean of trainset dependent variable \n")
```

```
Mean of trainset dependent variable
```

```
mean(trainSet$salary)
```

```
[1] 4814817
```

```
cat("Mean of testset dependent variable \n")
```

```
Mean of testset dependent variable
```

```
mean(testSet$salary)
```

```
[1] 4910398
```

Comments on Data Partitioning:
The modeling dataset was partitioned into a 70/30 split, 70% of the data will be used to train the model and 30% of the dataset will be used to test the model. The means of salary for the train and test dataset were both close which means both have similar distributions of the dependent variable.

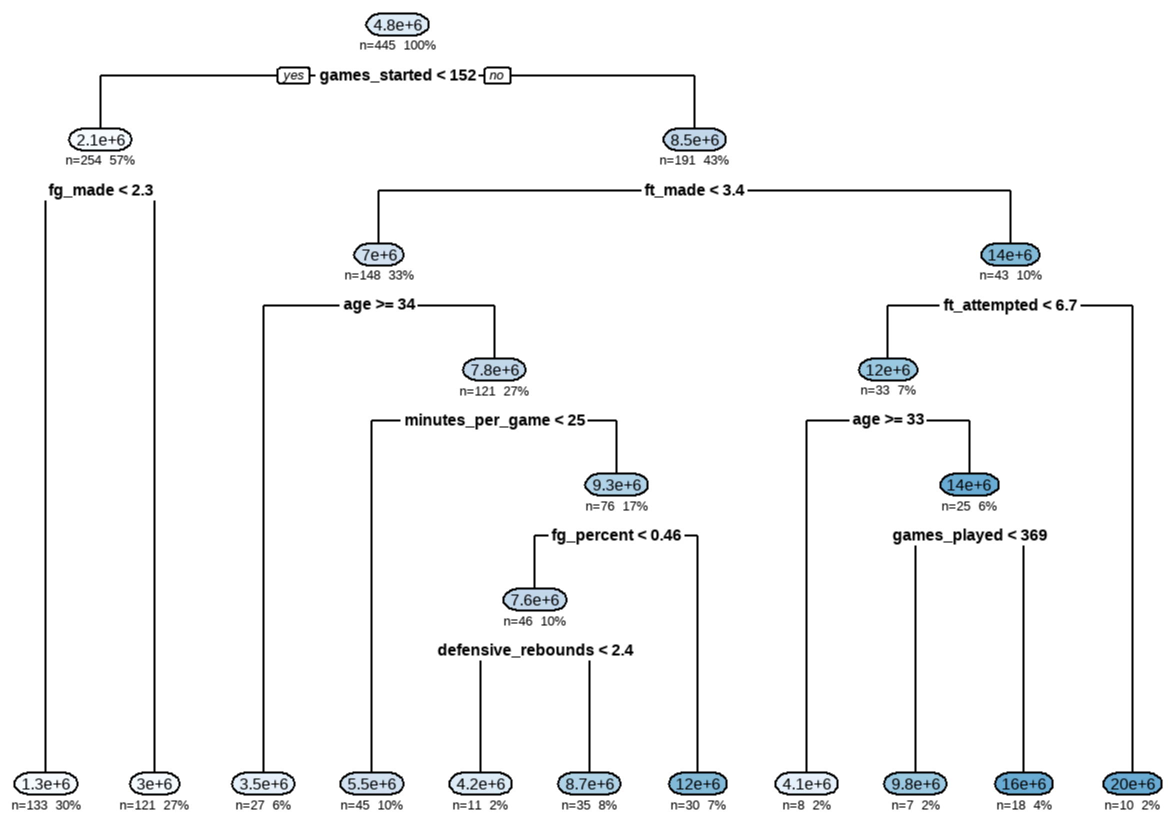
Modeling (Rpart)

Cross Validation

```
myCtrl <- trainControl(method = "cv", number = 10)
```

Default Tree (Rpart)

```
set.seed(1)
default_tree <- rpart(salary ~., data = nba_data, method = "anova")
rpart.plot(default_tree, type = 2, extra = 101, under = TRUE, fallen.leaves = TRUE)
```



Answer to Question 45 Part A:
A) What are the predictor variable and split value for the first split of the default regression tree?

The Predictor Variable for the first split is games started. The split values are for people that have less than 152 games started their salary is 2,100,000 whereas people with more than 152 games started their salary is 8,500,000.

Full Tree (Rpart)

```
set.seed(1)
full_tree <- rpart(salary ~., data = nba_data, control = rpart.control(cp = 0, minsplit = 2,
         minbucket = 1, maxdepth = 30, xval = 10))

#xval = 10 is cross fold validation of 10, this is how to do it using rpart, used rpart to get
         correct values in the cp table

head(full_tree$cptable)
```

```
          CP nsplit rel error    xerror       xstd
1 0.36913609      0 1.0000000 1.0046536 0.09285005
2 0.11584050      1 0.6308639 0.6867124 0.06355926
3 0.04723599      2 0.5150234 0.6693572 0.06324465
4 0.03493893      4 0.4205514 0.6011018 0.06073049
```

```
5 0.03326144      5 0.3856125 0.5903899 0.06014571
6 0.02580501      6 0.3523511 0.5424423 0.05800483
```

```
## Where to Find Answers to Part B

min_error <- full_tree$cptable[which.min(full_tree$cptable[,"xerror"]),]
min_error
```

```
        CP     nsplit  rel error     xerror       xstd
0.01786872 7.00000000 0.32654605 0.53163737 0.06063138
```

```
## Find number of leaf nodes in the minimum error tree
num_leaves <- min_error["nsplit"]
cat("Number of Leaf Nodes in the Minimum Tree: ")
```

```
Number of Leaf Nodes in the Minimum Tree:
```

```
print(num_leaves)
```

```
nsplit
     7
```

Answer to Question 45 Part B:
B) Build a full-grown tree. Which cp value is associated with the lowest cross-validation error? How many leaf nodes are in the minimum-error tree?

The CP value associated with the lowest cross-validation error is 0.01786872. There are seven leaf nodes in the minimum-error tree.

Best Pruned Tree

```
cpt <- full_tree$cptable

imin  <- which.min(cpt[,"xerror"]) ## Grabs the row that the minimum error is on and defines it
         with a variable
xerr_min  <- cpt[imin,"xerror"] ## Gets the xerror
xstd_min  <- cpt[imin,"xstd"] ## Gets the xstd
threshold <- xerr_min + xstd_min ## Creates the xerror threshold for best pruned tree.
threshold ## lowest amount of splits while staying below threshold xerror value
```

```
[1] 0.5922688
```

```
within_threshold <- which(cpt[,"xerror"] <= threshold)

best_pruned_index <- within_threshold[which.min(cpt[within_threshold, "nsplit"])]

best_pruned_cp <- cpt[best_pruned_index, "CP"]
```

```
best_pruned_xerror <- cpt[best_pruned_index, "xerror"]
best_pruned_nsplits <- cpt[best_pruned_index, "nsplit"]

cat("Best-Pruned Tree (1-SE Rule)\n")
```

```
Best-Pruned Tree (1-SE Rule)
```

```
cat("CP value:", best_pruned_cp, "\n")
```

```
CP value: 0.03326144
```

```
cat("Cross-validation error:", best_pruned_xerror, "\n")
```

```
Cross-validation error: 0.5903899
```

```
cat("Number of splits:", best_pruned_nsplits, "\n")
```

```
Number of splits: 5
```

Answer to Question 45 Part C:
C) Is there a simpler tree with a cross-validation error that is within one standard error of the minimum error? If there is, then which cp value is associated with the best-pruned tree?

Yes, there is a simpler tree with a cross-validation error that is within one standard error of the minimum error. This tree has a cp value of 0.03326144.

Best Tree

```
best_pruned_tree <- prune(full_tree, cp=0.03326144)
```

Variable Importance

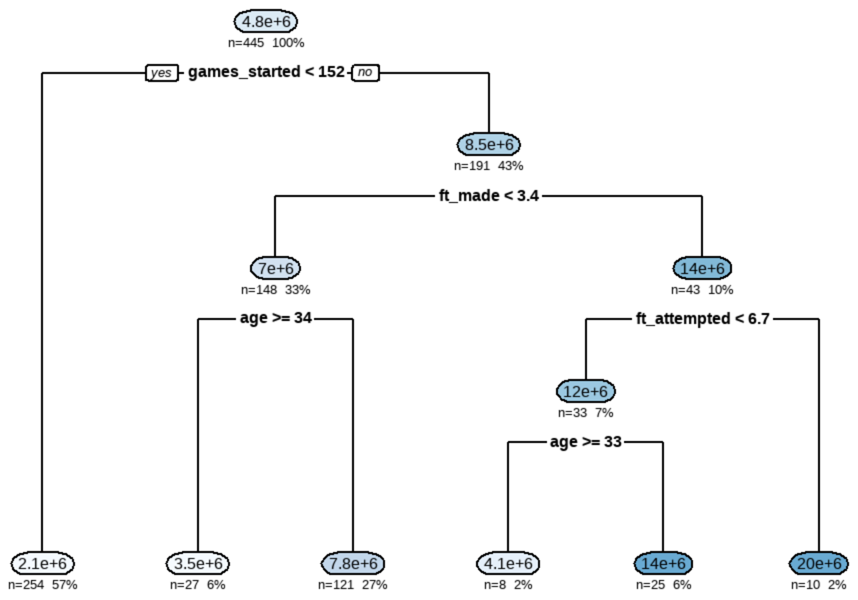
```
caret::varImp(best_pruned_tree)
```

```
                      Overall
age                0.78148610
defensive_rebounds 0.07855415
fg_attempted       0.23861331
fg_made            0.72078128
ft_attempted       0.48212458
ft_made            0.50742141
games_played       0.31387854
games_started      0.51839552
minutes_per_game   0.69766452
offensive_rebounds 0.09162487
personal_fouls     0.08197401
```

```
points             0.73324452
x3p_percent        0.16038949
height             0.00000000
weight             0.00000000
fg_percent         0.00000000
x3p_made           0.00000000
x3p_attempted      0.00000000
ft_percent         0.00000000
assists            0.00000000
blocks             0.00000000
steals             0.00000000
turnovers          0.00000000
```

Show Best Pruned Tree

```r
rpart.plot(best_pruned_tree, type = 2, extra = 101, under = TRUE, fallen.leaves = TRUE)
```



```r
cat("Number of Leaf Nodes in the Default Tree: ")
```

```
Number of Leaf Nodes in the Default Tree:
```

```
sum(best_pruned_tree$frame$var == "<leaf>")
```

```
[1] 6
```

Answer to Question 45 Part D:

D. Prune the full tree to the best-pruned tree or the minimum-error tree if the answer to part (c) is “No.” Display the tree. What are the rules that can be derived from the pruned tree?

From the best pruned tree, it can be derived that games started, ft made, age, and ft attempted are important predictor variables.

- Path 1: Players who started fewer than 152 games have an average predicted salary of about $2.1 million.
- Path 2: Players who started 152 or more games, made fewer than 3.4 free throws, and are aged 34 or older have an average salary of about $3.5 million.
- Path 3: Players who started 152 or more games, made fewer than 3.4 free throws, and are younger than 34 have an average salary of about $7.8 million.
- Path 4: Players who started 152 or more games, made at least 3.4 free throws, attempted fewer than 6.7 free throws, and are aged 33 or older have an average salary of about $4.1 million.
- Path 5: Players who started 152 or more games, made at least 3.4 free throws, attempted fewer than 6.7 free throws, and are younger than 33 have an average salary of about $12 million.
- Path 6: Players who started 152 or more games, made at least 3.4 free throws, and attempted 6.7 or more free throws have the highest predicted salary of about $20 million.

Evaluation (Rpart)

Evaluation Metrics

```
predict_bpt <- predict(best_pruned_tree, testSet)
bpt_metrics <- round(forecast::accuracy(predict_bpt, testSet$salary),2)

## For inline code
ME_value   <- bpt_metrics["Test set", "ME"]
RMSE_value <- bpt_metrics["Test set", "RMSE"]
MAE_value  <- bpt_metrics["Test set", "MAE"]
MPE_value  <- bpt_metrics["Test set", "MPE"]
MAPE_value <- bpt_metrics["Test set", "MAPE"]

rownames(bpt_metrics) <- "Rpart Regression Tree"
print(bpt_metrics)
```

```
                             ME    RMSE     MAE     MPE   MAPE
Rpart Regression Tree -150347.9 3210484 2345165 -231.71 256.97
```

Answer to Question 45 Part E:
E. What are the ME, RMSE, MAE, MPE, and MAPE of the pruned tree on the validation data?

- **ME Value:** ‘r ME_value’.
- **RMSE Value:** ‘r RMSE_value’.
- **MAE Value:** ‘r MAE_value’.
- **MPE Value:** ‘r MPE_value’%.
- **MAPE Value:** ‘r MAPE_value’%.

Measures of Accuracy:
- RMSE (Root Mean Square Error = 894.68) and MAE (Mean Absolute Error = 713.39) indicate that on average, the pruned tree’s salary predictions deviate by roughly 700–900 from the true values.
- MAPE (Mean Absolute Percentage Error = 34.74%) means predictions are off by about 35% on average relative to the actual salary level.
- Overall, the model shows moderate prediction accuracy, capturing general salary trends but still leaving room for improvement in precision.

Measures of Bias:
- ME (Mean Error = -1.35) is extremely close to zero, showing the model is essentially unbiased — there is no meaningful tendency to systematically over- or under-predict salary.
- MPE (Mean Percentage Error = -14.75%) is slightly negative, indicating a mild tendency to under-predict salaries on average, but the bias is not large relative to the scale of the values.

Score (Rpart)

```r
rpart_score <- predict(best_pruned_tree, nba_score)
rpart_pip <- cbind(nba_score, rpart_score)
rpart_pip
```

```
   player age height weight games_played games_started minutes_per_game fg_made
1 Player1  36     81    260          984           820             31.8     6.8
2 Player2  32     76    185          714           504             30.2     5.4
3 Player3  27     85    235          636           472             32.0     5.2
  fg_attempted fg_percent x3p_made x3p_attempted x3p_percent ft_made
1         14.3      0.473      0.1           0.5       0.260     3.3
2         11.0      0.437      1.6           4.2       0.384     5.0
3         11.8      0.438      1.0           3.0       0.339     2.6
  ft_attempted ft_percent offensive_rebounds defensive_rebounds assists blocks
1          4.4      0.765                3.1                6.3     1.8    0.3
2          5.7      0.870                0.6                2.6     2.1    0.1
3          3.4      0.786                1.0                3.9     1.7    0.6
  steals personal_fouls turnovers points rpart_score
1    0.8            2.4       2.1   18.0     3454649
2    0.9            1.9       1.7   16.5    13977446
3    0.8            2.4       1.7   15.0     7843275
```

Answer to Question 45 Part F:
F) Score the three NBA players Merrick is trying to sign as ACE Sports Management clients in the NBA_Score worksheet using the pruned tree. What is the average predicted salary of the three players?

- Player 1: The Average salary for Player 1 was 3,454,649.
- Player 2: The Average salary for Player 2 was 13,977,446.
- Player 3: The Average salary for Player 3 is 7,843,275.

Modeling (Caret)

Best Tree (Caret)

```
caret_bp <- train(salary~., data = trainSet,
                  method = "rpart",
                  trControl = myCtrl,
                  tuneLength =25,
                  metric = "RMSE",
                  control = rpart::rpart.control(minsplit = 2, minbucket = 1, cp = 0))

caret_bp$resample
```

```
      RMSE    Rsquared     MAE Resample
1  4120411 0.266250283 3023060   Fold01
2  2995713 0.510020593 2212342   Fold07
3  4109285 0.378319536 3241325   Fold08
4  4078978 0.474508720 2778811   Fold05
5  5373464 0.004145221 3786076   Fold04
6  3970711 0.391088715 3059824   Fold06
7  3357730 0.425822028 2351667   Fold03
8  4799444 0.182573225 3247050   Fold02
9  4628543 0.360334374 3420274   Fold10
10 4719672 0.452011479 2873042   Fold09
```

CP Values

```
caret_bp$results
```

```
           cp    RMSE  Rsquared     MAE   RMSESD RsquaredSD    MAESD
1  0.00000000 4883057 0.3040928 3096999 695785.0  0.1439206 357985.9
2  0.01653605 4639461 0.3209965 3000470 611877.9  0.1522815 385906.4
3  0.03307209 4270504 0.3443940 2964679 452755.4  0.1333859 361348.6
4  0.04960814 4223961 0.3526245 2930154 507358.0  0.1340030 408134.3
5  0.06614419 4252855 0.3421655 2941652 522804.0  0.1318271 399558.0
6  0.08268023 4215783 0.3562352 2926215 537380.2  0.1658159 414744.7
7  0.09921628 4412471 0.3001825 3061849 650790.4  0.1465602 471881.0
8  0.11575233 4402258 0.2979999 3072796 642946.2  0.1424450 476584.6
```

```
9  0.13228838 4330241 0.3219284 3043569 758346.9  0.1567372 522090.4
10 0.14882442 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
11 0.16536047 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
12 0.18189652 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
13 0.19843256 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
14 0.21496861 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
15 0.23150466 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
16 0.24804070 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
17 0.26457675 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
18 0.28111280 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
19 0.29764884 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
20 0.31418489 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
21 0.33072094 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
22 0.34725699 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
23 0.36379303 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
24 0.38032908 4215395 0.3445074 2999347 701522.1  0.1542800 474190.3
25 0.39686513 4737126 0.2637852 3557586 738561.4  0.1501069 466476.1
```

```r
cat("\nBest Tuned cp Value: ", caret_bp$bestTune$cp)
```

```
Best Tuned cp Value:  0.3803291
```

Variable Importance

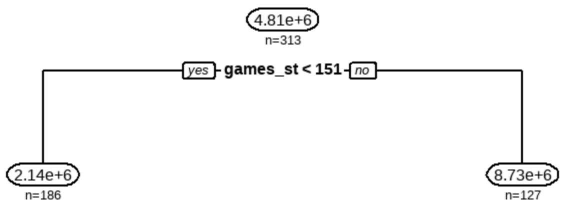
```r
caret::varImp(caret_bp)
```

```
rpart variable importance

  only 20 most important variables shown (out of 23)

                   Overall
games_started       100.00
minutes_per_game     90.49
points               85.58
fg_made              83.88
ft_attempted         81.20
personal_fouls        0.00
fg_percent            0.00
turnovers             0.00
x3p_attempted         0.00
weight                0.00
blocks                0.00
height                0.00
x3p_made              0.00
fg_attempted          0.00
offensive_rebounds    0.00
steals                0.00
x3p_percent           0.00
```

```
ft_percent            0.00
games_played          0.00
ft_made               0.00
```

Show Best Tree (Caret)

```
prp(caret_bp$finalModel, type = 2, extra = 1, under = TRUE, digits = 3, fallen.leaves = TRUE)
```



```
cat("Number of Leaf Nodes in the Default Tree: ")
```

```
Number of Leaf Nodes in the Default Tree:
```

```
sum(caret_bp$finalModel$frame$var == "<leaf>")
```

```
[1] 2
```

Comments on Best Tree (Caret):
The CP value for Caret's best tree is lower than rparts best pruned tree, so caret's best tree will have more splits and thus more leaf nodes. Caret's best tree has 12 leaf nodes whereas rpart's best tree has 6 leaf nodes. The Variables that the models deem important are also different. Caret's top three most important variables are age, defensive rebounds, and fg attempted whereas rpart's top three most important variables

are games started, games played, and minutes per game. Due to the increase in splits there are more paths in caret's model all prompted by different predictor variables. For example both models top average salary is 20e+6 however the path of variables to get to that average salary is different.
For Rpart's model players who started 152 or more games, made at least 3.4 free throws, and attempted 6.7 or more free throws will have the average salary of about $20 million.
For Caret's model players who started at least 185 games, made at least 2.85 free throws, are 33 years of age or older, weigh 233 pounds or more, and block at least 2.25 shots per game have an average salary of about $20 million.

Evaluation metrics will be used to see if Caret's model performs better than rpart's model.

Evaluation (caret)

```r
cbp_predict <- predict(caret_bp, testSet)
caret_bp_metrics <- round(forecast::accuracy(cbp_predict, testSet$salary),2)
rownames(caret_bp_metrics) <- "Caret Regression Tree"
print(caret_bp_metrics)
```

```
                             ME    RMSE     MAE     MPE   MAPE
Caret Regression Tree -474796.5 4585161 3254818 -295.88 321.03
```

```r
rbind(caret_bp_metrics, bpt_metrics)
```

```
                             ME    RMSE     MAE     MPE   MAPE
Caret Regression Tree -474796.5 4585161 3254818 -295.88 321.03
Rpart Regression Tree -150347.9 3210484 2345165 -231.71 256.97
```

Comments Comparing Evaluation across both regression trees:
The Rpart Regression Tree performs better than the Caret Regression Tree, due to its lower RMSE (3,372,027 vs. 4,283,480) and MAE (2,432,908 vs. 2,765,978), meaning its predictions are closer to the true salary values on average. Additionally, the Rpart model has smaller percentage errors suggesting it is more accurate and less biased.

Score (Caret)

```r
caret_score <- predict(caret_bp, nba_score)
rf_pip <- cbind(nba_score, caret_score)
rf_pip
```

```
   player age height weight games_played games_started minutes_per_game fg_made
1 Player1  36     81    260          984           820             31.8     6.8
2 Player2  32     76    185          714           504             30.2     5.4
3 Player3  27     85    235          636           472             32.0     5.2
  fg_attempted fg_percent x3p_made x3p_attempted x3p_percent ft_made
1         14.3      0.473      0.1           0.5       0.260     3.3
```

```
2         11.0     0.437     1.6           4.2       0.384     5.0
3         11.8     0.438     1.0           3.0       0.339     2.6
  ft_attempted ft_percent offensive_rebounds defensive_rebounds assists blocks
1          4.4      0.765                3.1                6.3     1.8    0.3
2          5.7      0.870                0.6                2.6     2.1    0.1
3          3.4      0.786                1.0                3.9     1.7    0.6
  steals personal_fouls turnovers points caret_score
1    0.8            2.4       2.1   18.0     8725432
2    0.9            1.9       1.7   16.5     8725432
3    0.8            2.4       1.7   15.0     8725432
```

Data Partition (Random Forest)

```
p <- ncol(trainSet)-1

mtry_center <- max(1, round(p/3))
mtry_grid <- data.frame(mtry = sort(unique(pmax(1, c(mtry_center-1, mtry_center, mtry_center+1,
         2, p)))))
```

Modeling (Random Forest)

```
set.seed(1)
rf_model <- train(salary ~ .,
                                         data = trainSet,
                                         method = "rf",
                                         trControl = myCtrl,
                                       tuneGrid = mtry_grid,
                                       ntree = 1000,
                                       importance = TRUE)
```

Resample

```
rf_model
```

```
Random Forest

313 samples
 23 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 282, 281, 281, 282, 282, 282, ...
Resampling results across tuning parameters:

  mtry  RMSE     Rsquared   MAE
```

```
   2    3472405  0.5658555  2389507
   7    3432674  0.5766471  2333652
   8    3408193  0.5865120  2310666
   9    3393808  0.5871320  2297864
  23    3441298  0.5746062  2317649

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 9.
```

Variable Importance

```
caret::varImp(rf_model)
```

```
rf variable importance

  only 20 most important variables shown (out of 23)

                   Overall
games_started      100.000
games_played        94.822
minutes_per_game    65.573
age                 60.315
points              56.444
fg_made             44.587
defensive_rebounds  39.359
ft_made             31.713
offensive_rebounds  31.387
fg_percent          31.167
ft_attempted        30.327
fg_attempted        28.804
personal_fouls      17.223
steals              15.845
turnovers           15.300
assists             13.600
x3p_attempted       12.898
x3p_percent          8.699
blocks               6.380
ft_percent           2.828
```

Comments on Random Forest Variable Importance:
The top four variables by variable importance is games started, games played, age, and minutes per game.

Evaluation (Random Forest)

```
predicted_rf <- predict(rf_model, testSet)
test_rf <- round(forecast::accuracy(predicted_rf, testSet$salary),2)
```

```
rownames(test_rf) <- "Random Forest"
test_rf
```

```
                     ME    RMSE     MAE     MPE   MAPE
Random Forest -292188.9 3259179 2243796 -224.08 246.02
```

Evaluation Metrics Across All Trees

```
rbind(test_rf, caret_bp_metrics, bpt_metrics)
```

```
                             ME    RMSE     MAE     MPE   MAPE
Random Forest         -292188.9 3259179 2243796 -224.08 246.02
Caret Regression Tree -474796.5 4585161 3254818 -295.88 321.03
Rpart Regression Tree -150347.9 3210484 2345165 -231.71 256.97
```

Comments on Evaluation Metrics:
The Rpart Regression Tree performs best overall, with the lowest RMSE (3,372,027) and MAE (2,432,908), indicating the smallest average and squared prediction errors among the models. It also has the lowest MAPE (158.93%), meaning its salary predictions are the closest to the true values. While the Random Forest performs similarly, its higher percentage errors suggest slightly less consistent prediction accuracy.

Score (Random Forest)

```
score_rf <- predict(rf_model, nba_score)
rf_pip <- cbind(nba_score, score_rf, caret_score, rpart_score)
rf_pip
```

```
   player age height weight games_played games_started minutes_per_game fg_made
1 Player1  36     81    260          984           820             31.8     6.8
2 Player2  32     76    185          714           504             30.2     5.4
3 Player3  27     85    235          636           472             32.0     5.2
  fg_attempted fg_percent x3p_made x3p_attempted x3p_percent ft_made
1         14.3      0.473      0.1           0.5       0.260     3.3
2         11.0      0.437      1.6           4.2       0.384     5.0
3         11.8      0.438      1.0           3.0       0.339     2.6
  ft_attempted ft_percent offensive_rebounds defensive_rebounds assists blocks
1          4.4      0.765                3.1                6.3     1.8    0.3
2          5.7      0.870                0.6                2.6     2.1    0.1
3          3.4      0.786                1.0                3.9     1.7    0.6
  steals personal_fouls turnovers points score_rf caret_score rpart_score
1    0.8            2.4       2.1   18.0  7240941     8725432     3454649
2    0.9            1.9       1.7   16.5 11604523     8725432    13977446
3    0.8            2.4       1.7   15.0 10345555     8725432     7843275
```

Comments on Scoring Across Tree Models:
We trust the Rpart score predictions more because the evaluation metrics clearly show that the Rpart

Regression Tree achieved the lowest RMSE (3,372,027) and lowest MAE (2,432,908) among all models. These values indicate that its predictions are, on average, closest to the true player salaries, with smaller overall errors and less variation.