HarvardX Professional Certificate in Data Science (PH125.9x) - CYO NBA Capstone Assignment

Rob Costello

04/06/2021

Contents

1 Overview

This report artifact details the analysis of the NBA Games dataset as part of the second capstone assignment in the HarvardX Professional Certificate in Data Science (PH125.9x) course. In this report I will discuss the goal of the project, the approach taken to perform the dataset analysis, followed by the final results of the analysis and a conclusion.

1.1 Introduction

Using pre-existing data we are able to construct a model of events (such as a basketball game), and based on this model we can generate predictions on future events. In the case of this project, we will be modeling game statistics for the NBA from seasons 2004 to 2020, based on a Kaggle dataset.

The goal of this project is to create a **model** that is optimized to predict the final spread of points based on the existing dataset, taking into account various effects/biases in the dataset. The final spread of points is the difference between the points scored by the Home team and the Away team, such that a positive spread indicates a home team win, and a negative spread indicates an away team win.

For more details on the NBA Games dataset please refer to the Description located here: https://www.kaggle.com/nathanlauga/nba-games.

To measure and compare accuracy of the models that are produced, the Root Mean Squared Error (RMSE) method is used, which provides a measure of the deviation between actual values from the dataset and predicted values from a model. The larger the RMSE, the more our predictions deviate from actual data.

The following formula is used in this analysis to calculate the RMSE for each model:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u}(\hat{y}_u - y_u)^2}$$

Here we are defining $\hat{y}_u$ as the predicted points spread for a game u.

This formula is represented as the following in our R code:

```
RMSE <- function(true_spread, predicted_spread){
  sqrt(mean((true_spread - predicted_spread)^2))
}
```

1.2 Approach Summary

The approach used for this analysis consists of the following steps, which are represented in the accompanying R script as Sections One through Eight:

1. Data sourcing and preparation - This step loads all the required libraries, acquires the dataset in csv format, performs some preparation of columns, splits the dataset into a working dataframe called **nba** and a validation dataframe called **validation**. The Validation dataframe is not used until the final step of generating predictions using the most optimal model. The **nba** dataset is further split into **train** and **test** dataframes for use during analysis.
2. Data exploration - In this step some basic dataset exploration is performed to help build awareness of the data.
3. Construct RMSE function and generate Naive analysis - This section creates the RMSE function that will be used throughout the analysis, and generates a Naive model based on the data in the **train** set that is used to generate predictions that are compared with the **test** set. Finally a tibble is constructed which will ultimately store all the RMSE results from the subsequent model analysis.
4. Regression on different factors to find out which are important, measured by RMSE
5. Regularise the previous models.
6. Use Simple Linear Model to evaluate effectiveness
7. Test other built-in functions to construct models and evaluate RMSE
8. Run predictions on Validation set for final output RMSE - in this final section we run our final optimized model on the **Validation** set to generate our final RMSE.

2 Method

In this section we discuss the process and techniques used in the analysis and model generation. Each section below aligns with a section in the corresponding R script submitted with this report.

2.1 SECTION ONE - Data sourcing and preparation

This step loads all the required libraries, acquires the dataset, performs some preparation of columns, splits the dataset into a working dataframe called **nba** and a validation dataframe called **validation**. Also the **nba** dataset is further split into **train** and **test** dataframes to ensure the **validation** set is not used for model creation.

As Kaggle requires users to be registered to download datasets, the NBA dataset has been placed in a separate location to ensure this analysis can be re-run by anyone.

To enable analysis further in this report, the dataset is also mutated to include additional columns for the "spread" of various statistics. The spread indicates the difference between the home team and the away team. These fields will be used as predictors to assess their impact on the Points Spread for a game.

2.2 SECTION TWO - Explore Data

Here we perform some basic exploration to help build awareness of the dataset.

```
## [1] "Number of rows in the NBA dataset"
```

```
## [1] 543904
```

```
## [1] "Size of the training dataset"
```

```
## [1] 489512
```

```
## [1] "Size of the test dataset"
```

```
## [1] 54392
```

```
## [1] "Size of the validation dataset"
```

```
## [1] 60437
```

```
## [1] "How many seasons are covered in the dataset?"
```

```
## [1] 18
```

```
## [1] "How many teams are there?"
```

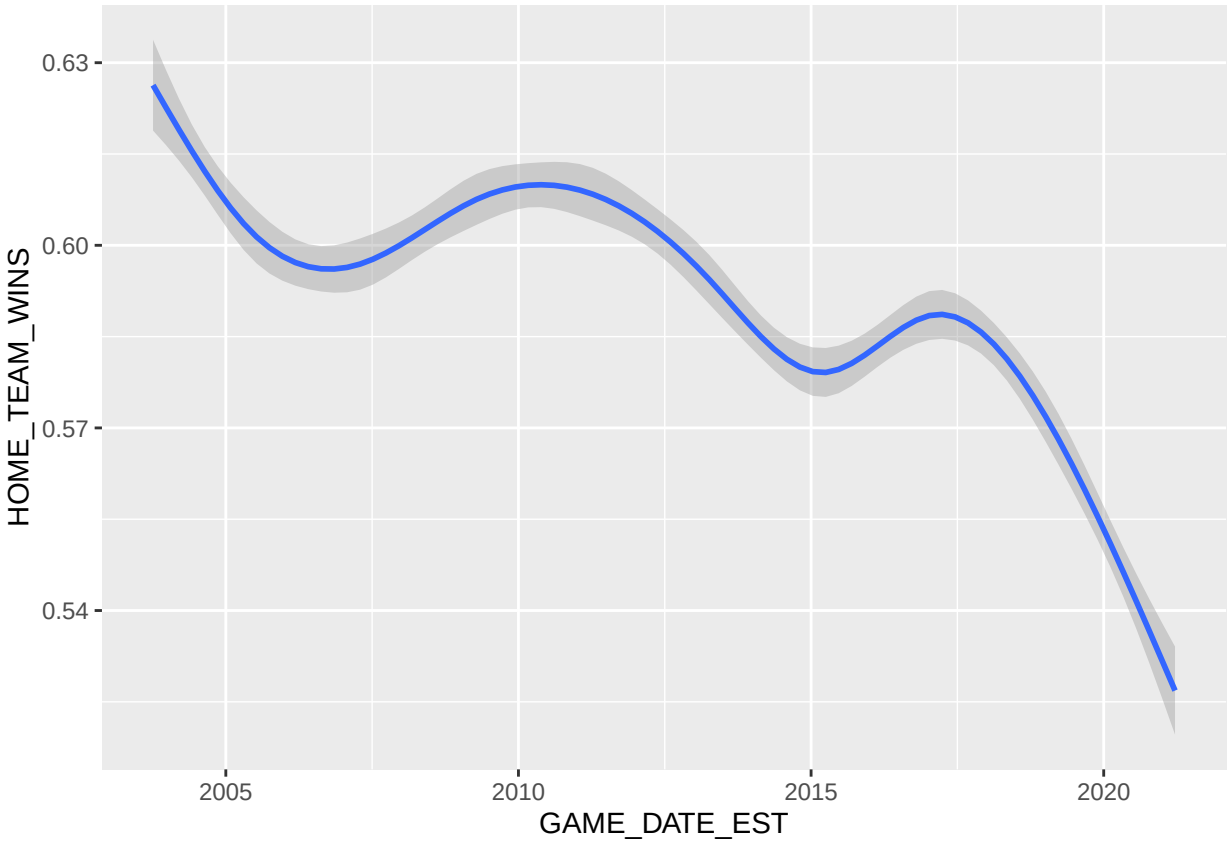
```
## [1] 30
```

```
## [1] "What was the Average points-spread and the Standard Deviation?"
```

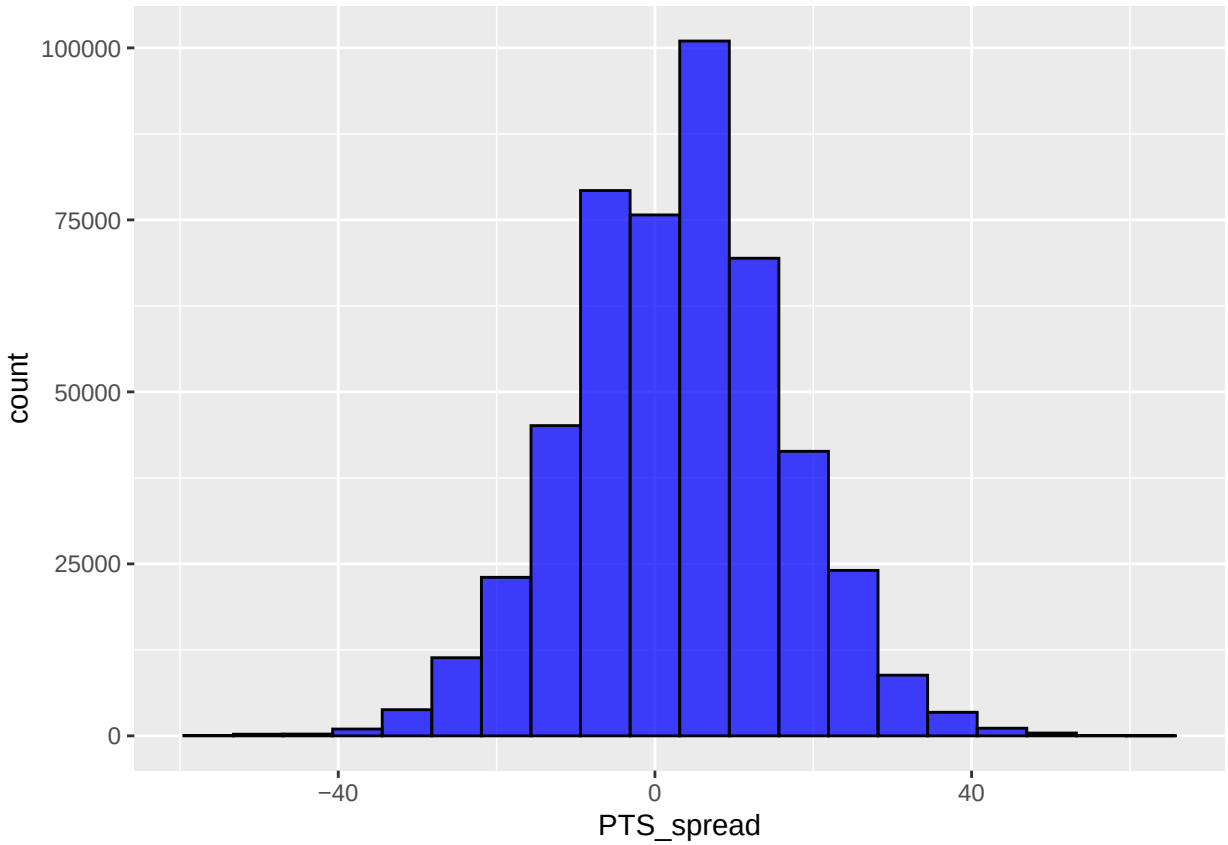
```
## [1] 2.868655
```

```
## [1] 13.37658
```

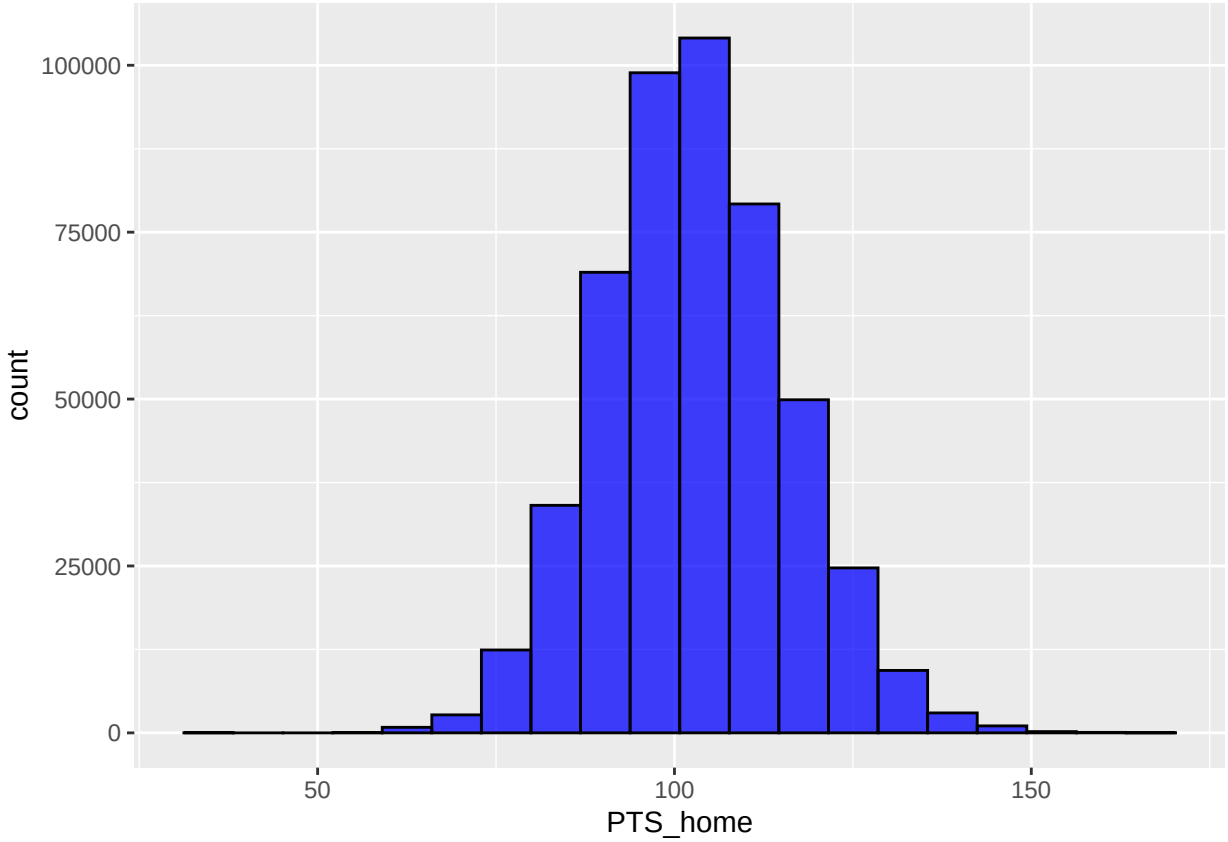
It is often assumed a team playing at their home ground/stadium has an advantage. Here we can see the data validates this assumption as the number of Wins is over 50%, however it's interesting to note this rate is dropping over time. Perhaps this indicates home-court advantage is reducing as the teams are collectively getting better. Regardless, we should consider this effect when creating a model for predictions.

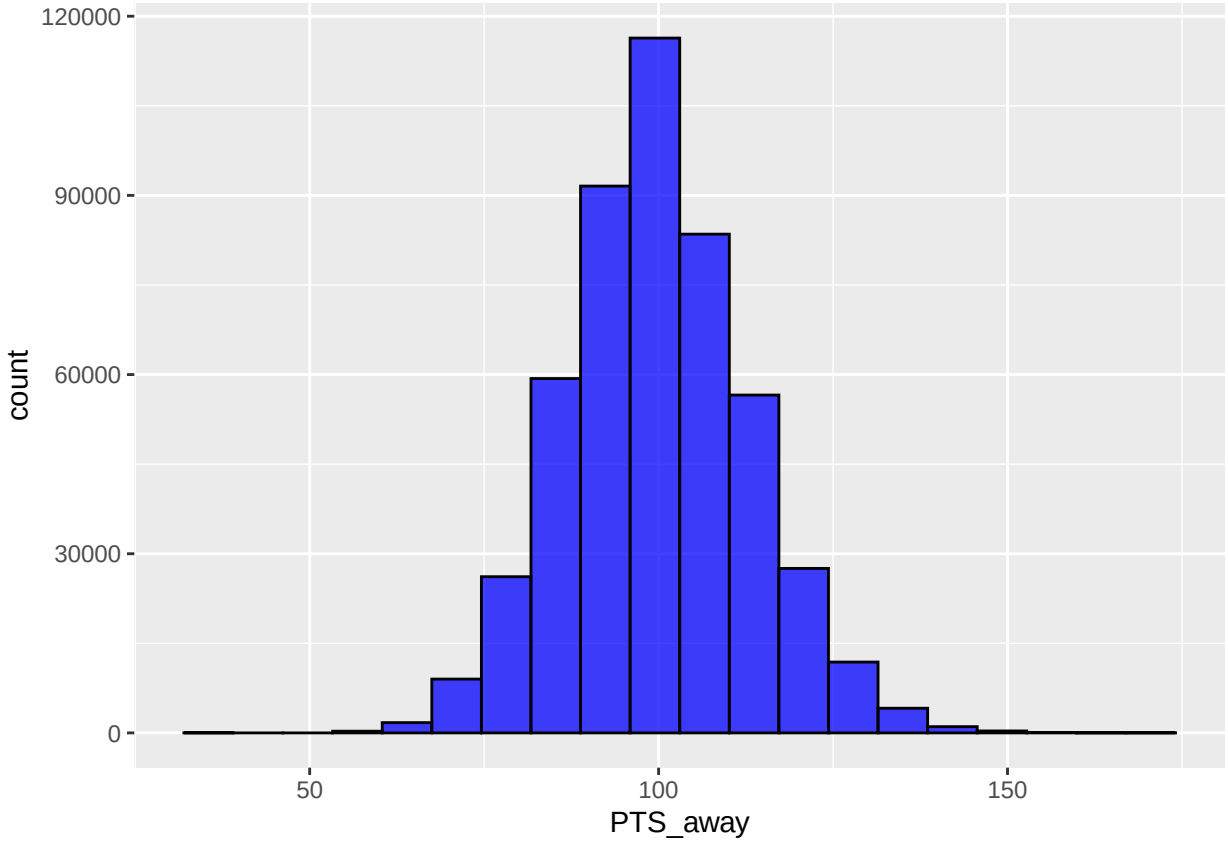


Here we plot the distribution of the Points Spread across all games. Interestingly this looks to be a normal distribution.



Lets now check the distribution of Home and Away teams to cross check that Home teams average scores are higher (i.e. they win more).



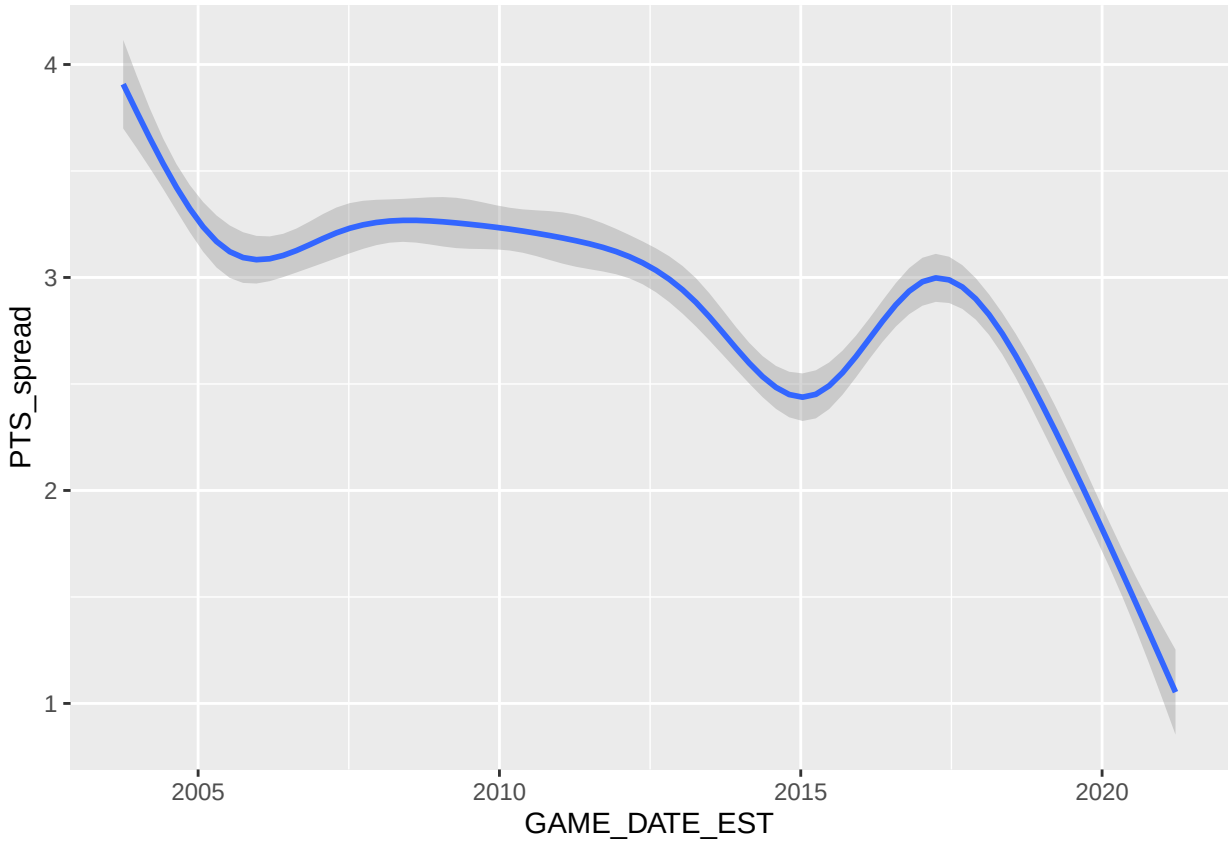


And now confirm this with Average scores:

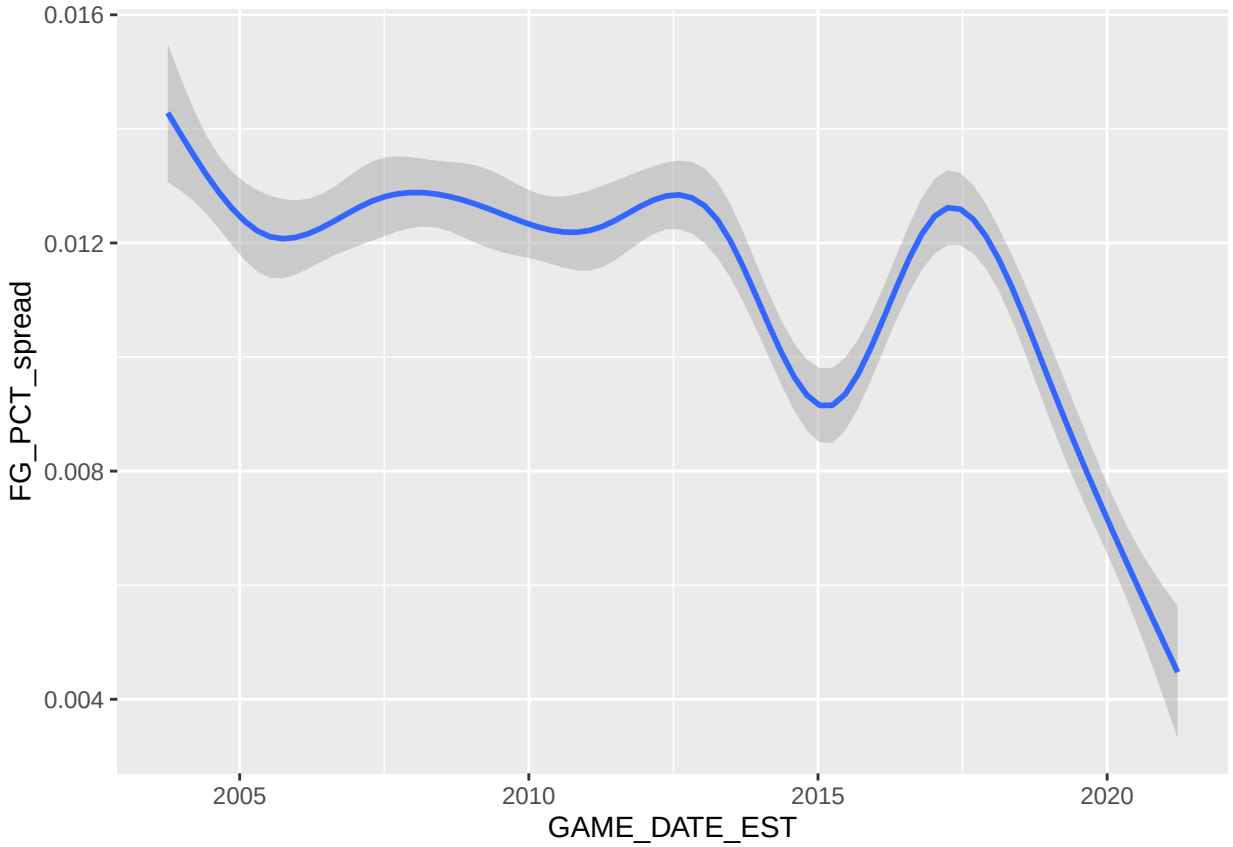
```
## [1] "Home team average scores and SD"

## [1] "Mean:  102.659940920754"

## [1] "SD:  13.0440686328096"

## [1] "Away team average scores and SD"

## [1] "Mean:  99.7912860154603"

## [1] "SD:  13.163975039852"
```

Indeed we see home teams do score more and validate the advantage.

Similar to the declining home court advantage we saw earlier, we can see here that the Points Spread (i.e. the difference between opponents scores in a game) is declining over time.



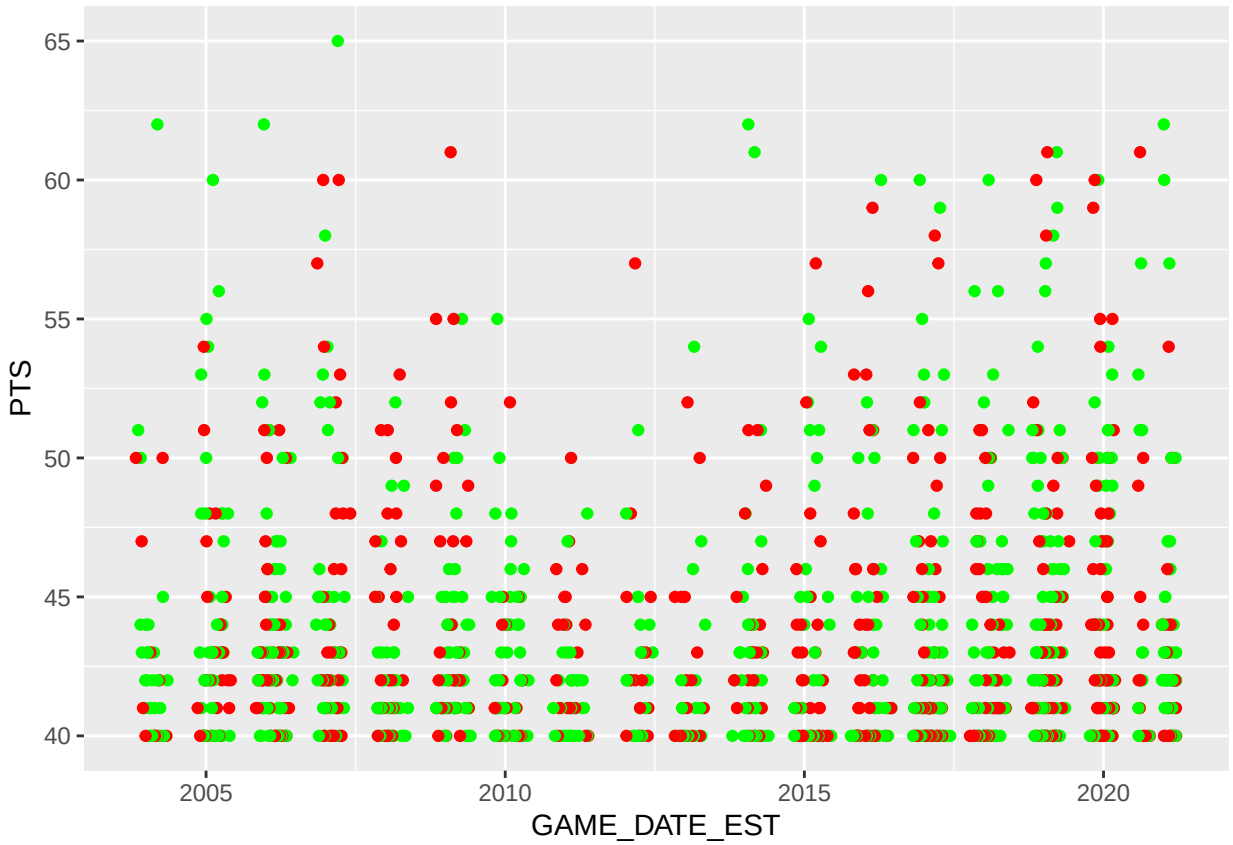
The same time-effect can be seen for Field Goal Percentages



Averages across other statistics like Free-Throw and 3 Point Percentages is also consistent with this Home-Court and time effect. We will consider these other statistics in our models going forward.

We also need to consider individual performance by a player and the effect they have on the outcome of a game.

Here we find games where a single player scored more than 40 points in a game and compare this with Home Team Wins.




```
## [1] "Wins vs Losses where a player scores more than 40 points."
```

```
##   win loss
## 1 761  562
```

2.3 SECTION THREE - Naive analysis

This section creates the RMSE function that will be used throughout the analysis, and generates a Naive model based on the data in the **train** set that is used to generate predictions that are compared with the **test** set. Finally a tibble is constructed which will ultimately store all the RMSE results from the subsequent model analysis.

For the Naive RMSE baseline, we predict based on the average true points spread, μ, and ϵ_u for the independent errors across the distribution of games.

$$Y_u = \mu + \epsilon_u$$

```
# Define RMSE algorithm to be used for evaluating prediction performance
RMSE <- function(true_spread, predicted_spread){
  sqrt(mean((true_spread - predicted_spread)^2))
}

# Determine naive RMSE by comparing mean spread with the training set
mu <- mean(train_set$PTS_spread)
naive_rmse <- RMSE(test_set$PTS_spread, mu)

# Create tibble to store results and capture naive rmse
rmse_results <- tibble(method = "Naive RMSE", RMSE = naive_rmse)
```

Method	RMSE
Naive RMSE	13.44394

Here we can see our RMSE is quite high. In our next stages we begin to address the effects we've found and attempt to lower the RMSE for our model.

2.4 SECTION FOUR - Regression on different factors

2.4.1 Home Court Advantage

Now that we have a Naive RMSE baseline, we will extend the model to address various effects by generating a model that takes the Home Court Advantage into consideration. This will be represented as an additional weighting called b_i in our model.

$$Y_u = \mu + b_i + \epsilon_u$$

Here we consider the potential effect of home court advantage. Can we lower the model RMSE when considering Home Team Wins?

```
game_avgs <- train_set %>%
  group_by(HOME_TEAM_WINS) %>%
  summarize(b_i = mean(PTS_spread - mu))

predicted_spread <- mu + test_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  pull(b_i)

home_effect_rmse <- RMSE(test_set$PTS_spread, predicted_spread)

# Update results tibble with Home Effect RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Home Team Effect RMSE", RMSE = home_effect_rmse
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476

Taking into account the the home team effect, we can see a considerable drop in our RMSE already, which is a good sign, but we can better this number by addressing other effects.

2.4.2 Field Goal Percentage

Here we now consider the potential effect of Field Goal Percentage on wins, can we lower the model RMSE more?

Again, we extend the model to address the Field Goal Percentage effect. This will be represented as an additional weighting called b_{fg} in our model.

$$Y_u = \mu + b_i + b_{fg} + \epsilon_u$$

```
# Define helper function to replace NAs with mu.
na.mu <- function (x) {
  x[is.na(x)] <- mu
  return(x)
}

na.zero <- function (x) {
  x[is.na(x)] <- 0
  return(x)
}

fg_avgs <- train_set %>%
```

```
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  group_by(FG_PCT_spread) %>%
  summarize(b_fg = mean(PTS_spread - mu - b_i))

predicted_spread <- test_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  mutate(pred = mu + b_i + b_fg) %>%
  pull(pred)

predicted_spread <- na.mu(predicted_spread)

home_fg_effect_rmse <- RMSE(test_set$PTS_spread, predicted_spread)

# Update results tibble with Home Game & FG Effect RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Home Game & FG Effect RMSE", RMSE = home_fg_effe
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607

Indeed the RMSE improves when considering Field Goal Percentage. Lets now add additional team statistics to the model to see if they have a meaningful effect.

2.4.3 Three Point Percentage

Again, we extend the model to address the Three Point Percentage effect. This will be represented as an additional weighting called b_{fg3} in our model.

$$Y_u = \mu + b_i + b_{fg} + b_{fg3} + \epsilon_u$$

```
fg3_avgs <- train_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  group_by(FG3_PCT_spread) %>%
  summarize(b_fg3 = mean(PTS_spread - mu - b_i - b_fg))

predicted_spread <- test_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  mutate(pred = mu + b_i + b_fg + b_fg3) %>%
  pull(pred)

predicted_spread <- na.mu(predicted_spread)

home_fg3_effect_rmse <- RMSE(test_set$PTS_spread, predicted_spread)

# Update results tibble with Home Game & FG & FG3 Effect RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Home Game & FG & FG3 Effect RMSE", RMSE = home_f
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157

2.4.4 Free Throw Percentage

Again, we extend the model to address the Free Throw Percentage effect. This will be represented as an additional weighting called b_{ft} in our model.

$$Y_u = \mu + b_i + b_{fg} + b_{fg3} + b_{ft} + \epsilon_u$$

```
ft_avgs <- train_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  group_by(FT_PCT_spread) %>%
  summarize(b_ft = mean(PTS_spread - mu - b_i - b_fg - b_fg3))

predicted_spread <- test_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  mutate(pred = mu + b_i + b_fg + b_fg3 + b_ft) %>%
  pull(pred)

predicted_spread <- na.mu(predicted_spread)

home_ft_effect_rmse <- RMSE(test_set$PTS_spread, predicted_spread)

# Update results tibble with Home Game & FG & FG3 & FT Effect RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Home Game & FG & FG3 & FT Effect RMSE", RMSE = ho
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482

2.4.5 Assists

Again, we extend the model to address the Assist effect. This will be represented as an additional weighting called b_{ast} in our model.

$$Y_u = \mu + b_i + b_{fg} + b_{fg3} + b_{ft} + b_{ast} + \epsilon_u$$

```
ast_avgs <- train_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
```

```
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  group_by(AST_spread) %>%
  summarize(b_ast = mean(PTS_spread - mu - b_i - b_fg - b_fg3 - b_ft))

predicted_spread <- test_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  left_join(ast_avgs, by='AST_spread') %>%
  mutate(pred = mu + b_i + b_fg + b_fg3 + b_ft + b_ast) %>%
  pull(pred)

predicted_spread <- na.mu(predicted_spread)

ast_effect_rmse <- RMSE(test_set$PTS_spread, predicted_spread)

# Update results tibble with Home Game & FG & FG3 & FT & AST Effect RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Home Game & FG & FG3 & FT & AST Effect RMSE", RMS
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907

2.4.6 Rebounds

Again, we extend the model to address the Rebound effect. This will be represented as an additional weighting called b_{reb} in our model.

$$Y_u = \mu + b_i + b_{fg} + b_{fg3} + b_{ft} + b_{ast} + b_{reb} + \epsilon_u$$

```
reb_avgs <- train_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  left_join(ast_avgs, by='AST_spread') %>%
  group_by(REB_spread) %>%
  summarize(b_reb = mean(PTS_spread - mu - b_i - b_fg - b_fg3 - b_ft - b_ast))

predicted_spread <- test_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  left_join(ast_avgs, by='AST_spread') %>%
  left_join(reb_avgs, by='REB_spread') %>%
  mutate(pred = mu + b_i + b_fg + b_fg3 + b_ft + b_ast + b_reb) %>%
  pull(pred)
```

```
predicted_spread <- na.mu(predicted_spread)

reb_effect_rmse <- RMSE(test_set$PTS_spread, predicted_spread)

# Update results tibble with Home Game & FG & FG3 & FT & AST & REB Effect RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Home Game & FG & FG3 & FT & AST & REB Effect RMSE
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233

Inclusion of additional team statistics does indeed improve our model RMSE gradually, so we will include these effects in the final model we run on the validation set.

2.4.7 Individual Performance

Lastly we consider the impact an individual has on our model, by considering the effect of someone scoring more than 40 points in a game.

Finally we extend the model to address the Individual Performance effect. This will be represented as an additional weighting called b_{ifg} in our model.

$$Y_u = \mu + b_i + b_{fg} + b_{fg3} + b_{ft} + b_{ast} + b_{reb} + b_{ifg} + \epsilon_u$$

```
ifg_avgs <- train_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  left_join(ast_avgs, by='AST_spread') %>%
  group_by(PTS) %>%
  filter(PTS >= 40, HOME_TEAM_WINS == 1) %>%
  summarize(b_ifg = mean(PTS_spread - mu - b_i - b_fg - b_fg3 - b_ft - b_ast))

predicted_spread <- test_set %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  left_join(ast_avgs, by='AST_spread') %>%
  left_join(reb_avgs, by='REB_spread') %>%
  left_join(ifg_avgs, by='PTS') %>%
  filter(PTS >= 40, HOME_TEAM_WINS == 1) %>%
  mutate(pred = mu + b_i + b_fg + b_fg3 + b_ft + b_ast + b_reb + b_ifg) %>%
  pull(pred)

#predicted_spread <- na.zero(predicted_spread)
```

```r
# filter the test set so we only compare appropriate values
f_test_set <- test_set %>%
  filter(PTS >= 40, HOME_TEAM_WINS == 1)

home_ifg_effect_rmse <- RMSE(f_test_set$PTS_spread, predicted_spread)

# Update results tibble with Home Game & FG & FG3 & FT & AST & REB & Player Effect RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Home Game & FG & FG3 & FT & AST & REB & Player >
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136

We can see this does indeed have an impact on the RMSE of our model. However given this approach limits the scope of the model to very few games, it will likely skew predictions on the majority of games that don't have a high scoring player. As such we will not carry forward this approach into our final model.

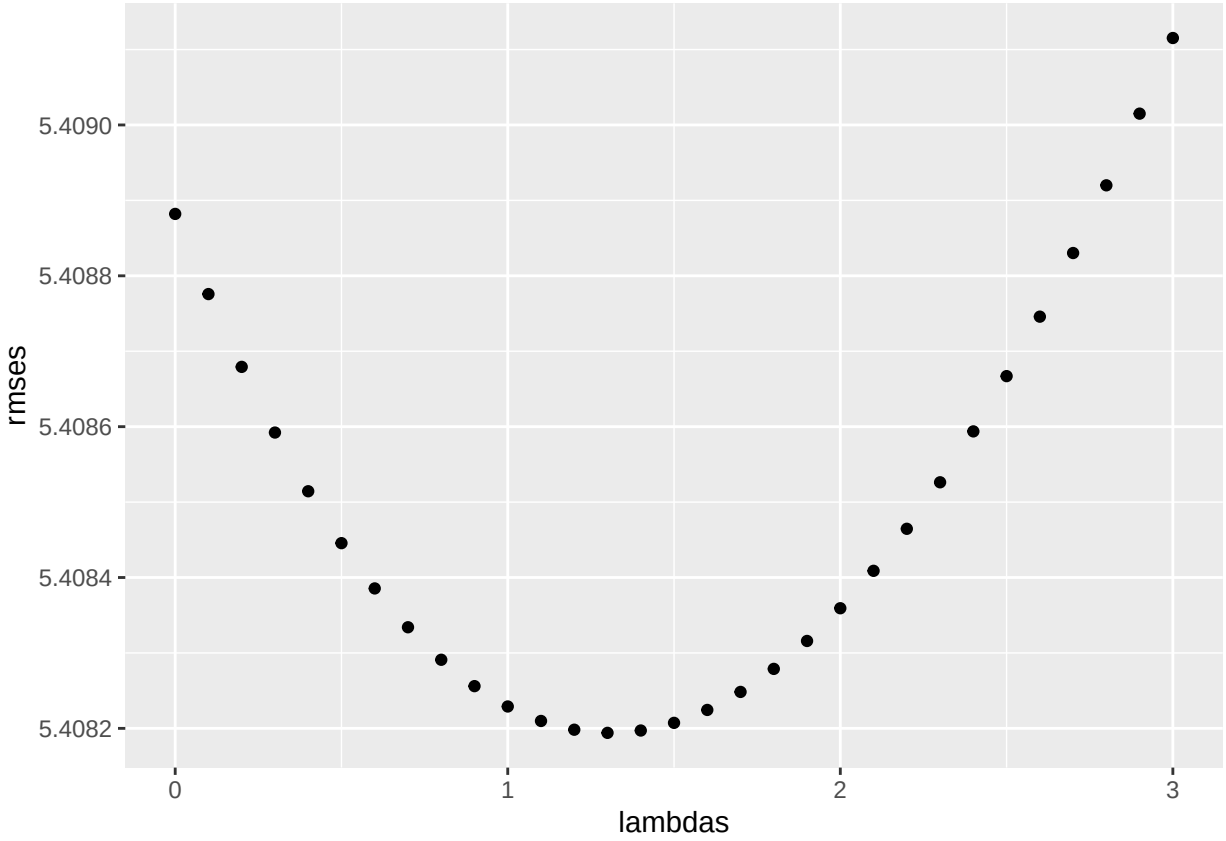
2.5 SECTION FIVE - Regularization

Here we seek to constrain variability of the various effects in the dataset by regularizing the data and penalizing data points that are noise/outliers. We do this by adding an additional cost value into the model, which we calculate using cross-validation to find the optimal value.

```
lambdas <- seq(0, 3, 0.1)

rmses <- sapply(lambdas, function(l){

  game_avgs <- train_set %>%
    group_by(HOME_TEAM_WINS) %>%
    summarize(b_i = sum(PTS_spread - mu)/(n()+l))

  fg_avgs <- train_set %>%
    left_join(game_avgs, by="HOME_TEAM_WINS") %>%
    group_by(FG_PCT_spread) %>%
    summarize(b_u = sum(PTS_spread - b_i - mu)/(n()+l))

  fg3_avgs <- train_set %>%
    left_join(game_avgs, by='HOME_TEAM_WINS') %>%
    left_join(fg_avgs, by='FG_PCT_spread') %>%
    group_by(FG3_PCT_spread) %>%
    summarize(b_fg3 = sum(PTS_spread - b_i - b_u - mu)/(n()+l))

  ft_avgs <- train_set %>%
    left_join(game_avgs, by='HOME_TEAM_WINS') %>%
    left_join(fg_avgs, by='FG_PCT_spread') %>%
    left_join(fg3_avgs, by='FG3_PCT_spread') %>%
    group_by(FT_PCT_spread) %>%
    summarize(b_ft = sum(PTS_spread - b_i - b_u - b_fg3 - mu)/(n()+l))

  ast_avgs <- train_set %>%
    left_join(game_avgs, by='HOME_TEAM_WINS') %>%
    left_join(fg_avgs, by='FG_PCT_spread') %>%
    left_join(fg3_avgs, by='FG3_PCT_spread') %>%
    left_join(ft_avgs, by='FT_PCT_spread') %>%
    group_by(AST_spread) %>%
    summarize(b_ast = sum(PTS_spread - b_i - b_u - b_fg3 - b_ft - mu)/(n()+l))

  reb_avgs <- train_set %>%
    left_join(game_avgs, by='HOME_TEAM_WINS') %>%
    left_join(fg_avgs, by='FG_PCT_spread') %>%
    left_join(fg3_avgs, by='FG3_PCT_spread') %>%
    left_join(ft_avgs, by='FT_PCT_spread') %>%
    left_join(ast_avgs, by='AST_spread') %>%
    group_by(REB_spread) %>%
    summarize(b_reb = sum(PTS_spread - b_i - b_u - b_fg3 - b_ft - b_ast - mu)/(n()+l))


  ifg_avgs <- train_set %>%
    left_join(game_avgs, by='HOME_TEAM_WINS') %>%
    left_join(fg_avgs, by='FG_PCT_spread') %>%
    left_join(fg3_avgs, by='FG3_PCT_spread') %>%
```

```
    left_join(ft_avgs, by='FT_PCT_spread') %>%
    left_join(ast_avgs, by='AST_spread') %>%
    left_join(reb_avgs, by='REB_spread') %>%
    group_by(PTS) %>%
    filter(PTS >= 40, HOME_TEAM_WINS == 1) %>%
    summarize(b_ifg = sum(PTS_spread - b_i - b_u - b_fg3 - b_ft - b_ast - b_reb - mu)/(n()+1))

  predicted_spread <- test_set %>%
    left_join(game_avgs, by='HOME_TEAM_WINS') %>%
    left_join(fg_avgs, by='FG_PCT_spread') %>%
    left_join(fg3_avgs, by='FG3_PCT_spread') %>%
    left_join(ft_avgs, by='FT_PCT_spread') %>%
    left_join(ast_avgs, by='AST_spread') %>%
    left_join(reb_avgs, by='REB_spread') %>%
    left_join(ifg_avgs, by='PTS') %>%
    filter(PTS >= 40, HOME_TEAM_WINS == 1) %>%
    mutate(pred = mu + b_i + b_u + b_fg3 + b_ft + b_ast + b_reb + b_ifg) %>%
    pull(pred)

  # filter the test set so we only compare appropriate values
  f_test_set <- test_set %>%
    filter(PTS >= 40, HOME_TEAM_WINS == 1)

  return(RMSE(f_test_set$PTS_spread, predicted_spread))
})

final_lambda <- lambdas[which.min(rmses)]

reg_effect_rmse <- min(rmses)

# Update results tibble with Regularization RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Regularization of all Effect RMSE", RMSE = reg_e
```

We can see in the graph below the lambda is optimized at around 1.25, however the impact this has on the RMSE is quite small.



Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194

By regularizing the model we have only achieved a minor improvement in RMSE, likely not enough to have a meaningful impact on the final model we run on the validation set.

2.6 SECTION SIX - Simple Linear Mode

Until now we have constructed a model manually, however it might be interesting to see if the built in models in R produce similar results. We will use the original **games_df** dataset for the experiment, as the joined dataset takes considerable time to process for some of the built-in models.

Here we use the LM model type to produce a basic linear model and calculate its RMSE on the **test_set**. We can compare this to our third model that considered Home Court Advantage and Field Goal Percentage.

```
fit <- lm(PTS_spread ~ HOME_TEAM_WINS + FG_PCT_spread, data=games_df)

y_hat <- fit$coef[1] + fit$coef[2]*test_set$FG_PCT_spread

lm_rmse <- sqrt(mean((y_hat - test_set$PTS_spread)^2))

rmse_results <- bind_rows(rmse_results,tibble(method ="Linear Model RMSE", RMSE = lm_rmse ))
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194
Linear Model RMSE	15.563903

Here we can see this didn't perform as well as our manually constructed model. In the next section we will experiment with other built in modeling functions to see if they produce similar or better results.

2.7 SECTION SEVEN - Test other built-in functions

We will continue to use the original **games_df** dataset for the following experiments.

2.7.1 GLM

We expect the LM and GLM functions to give similar results with no tuning parameters applied, however we should test just to be sure.

```
fit <- train(PTS_spread ~ HOME_TEAM_WINS + FG_PCT_spread, method = "glm", data = games_df)

y_hat <- fit$finalModel$coef[1] + fit$finalModel$coef[2]*test_set$FG_PCT_spread

glm_rmse <- sqrt(mean((y_hat - test_set$PTS_spread)^2))

rmse_results <- bind_rows(rmse_results,tibble(method ="GLM RMSE", RMSE = glm_rmse ))
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194
Linear Model RMSE	15.563903
GLM RMSE	15.563903

As expected, a similar result.

2.7.2 SVM

Now we test using the Support Vector Machine model type.

```
fit <- train(PTS_spread ~ HOME_TEAM_WINS + FG_PCT_spread, method = "svmLinear", data = games_df)

svm_rmse <- fit$results$RMSE

rmse_results <- bind_rows(rmse_results,tibble(method ="SVM Linear RMSE", RMSE = svm_rmse ))
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194
Linear Model RMSE	15.563903
GLM RMSE	15.563903
SVM Linear RMSE	6.581140

Similar result to our third manual model (Home Game & FG Effect).

2.7.3 KNN

Now we test using the kNN model type.

```
fit <- train(PTS_spread ~ HOME_TEAM_WINS + FG_PCT_spread, method = "knn", data = games_df)

knn_rmse <- min(fit$results$RMSE)

rmse_results <- bind_rows(rmse_results,tibble(method ="kNN RMSE", RMSE = knn_rmse ))
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194
Linear Model RMSE	15.563903
GLM RMSE	15.563903
SVM Linear RMSE	6.581140
kNN RMSE	6.634824

Similar result to our third manual model (Home Game & FG Effect).

2.7.4 GamLoess

Now we test using the gamLoess model type.

```
fit <- train(PTS_spread ~ HOME_TEAM_WINS + FG_PCT_spread, method = "gamLoess", data = games_df)

gam_rmse <- fit$results$RMSE

rmse_results <- bind_rows(rmse_results,tibble(method ="gamLoess RMSE", RMSE = gam_rmse ))
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194
Linear Model RMSE	15.563903
GLM RMSE	15.563903
SVM Linear RMSE	6.581140
kNN RMSE	6.634824
gamLoess RMSE	6.384514

Interestingly it does perform slightly better than our manual model.

2.7.5 Random Forest

Now we test using the gamLoess model type.

```
fit <- train(PTS_spread ~ HOME_TEAM_WINS + FG_PCT_spread, method = "rf", data = games_df)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```
rf_rmse <- fit$results$RMSE

rmse_results <- bind_rows(rmse_results,tibble(method ="Random Forest RMSE", RMSE = rf_rmse ))
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194
Linear Model RMSE	15.563903
GLM RMSE	15.563903
SVM Linear RMSE	6.581140
kNN RMSE	6.634824
gamLoess RMSE	6.384514
Random Forest RMSE	6.850299

Similar result to our third manual model (Home Game & FG Effect).

2.8 SECTION EIGHT - Run the best model on the Validation set

In this final section we run our final optimized model on the **Validation** set to generate our final RMSE.

```
predicted_spread <- validation %>%
  left_join(game_avgs, by='HOME_TEAM_WINS') %>%
  left_join(fg_avgs, by='FG_PCT_spread') %>%
  left_join(fg3_avgs, by='FG3_PCT_spread') %>%
  left_join(ft_avgs, by='FT_PCT_spread') %>%
  left_join(ast_avgs, by='AST_spread') %>%
  left_join(reb_avgs, by='REB_spread') %>%
  mutate(pred = mu + b_i + b_fg + b_fg3 + b_ft + b_ast + b_reb + final_lambda) %>%
  pull(pred)

predicted_spread <- na.mu(predicted_spread)

final_validation_rmse <- RMSE(validation$PTS_spread, predicted_spread)

# Update results tibble with Final RMSE
rmse_results <- bind_rows(rmse_results,tibble(method ="Final Test on Validation Set RMSE", RMSE = final
```

Method	RMSE
Naive RMSE	13.443942
Home Team Effect RMSE	8.054476
Home Game & FG Effect RMSE	6.548607
Home Game & FG & FG3 Effect RMSE	6.083157
Home Game & FG & FG3 & FT Effect RMSE	5.793482
Home Game & FG & FG3 & FT & AST Effect RMSE	5.756907
Home Game & FG & FG3 & FT & AST & REB Effect RMSE	5.715233
Home Game & FG & FG3 & FT & AST & REB & Player > 40pts Effect RMSE	5.417136
Regularization of all Effect RMSE	5.408194
Linear Model RMSE	15.563903
GLM RMSE	15.563903
SVM Linear RMSE	6.581140
kNN RMSE	6.634824
gamLoess RMSE	6.384514
Random Forest RMSE	6.850299
Final Test on Validation Set RMSE	5.829128

3 Results

```
## [1] "Final RMSE score is:  5.82912757249492"
```

The final result measured in this analysis considered only the team-based statistics, not individual contribution, as these were the most accurate predictors for the Point Spread of a game. This result was under half the original RMSE for naive analysis of the dataset, which I consider a very good outcome.

4 Conclusion

This analysis and report stepped through generating models that included various effects, and outputted a consolidated model that was run on a validation set. The effects that were assessed include:

- Home Team advantage.
- Field Goal % , 3 Point % , Free-throw %, Assists and Rebounds.
- Individual Performance Effect.
- Regularized Home Team and Field Goal Effect.

I also experimented with built-in models from the Caret R package, including LM/GLM, SVMLinear, KNN, gamLoess and Random Forest. Surprisingly these models didn't perform as well as I had expected compared to the the basic Home+FG model, and as such weren't used for final model evaluation.

The analysis showed that by factoring in the various effects our models were consistently better than the Naive RMSE, however the most impact to RMSE was made in the second and third models, indicating playing at Home and shooting a good percentage from the field resulted in positive outcomes.

The final model produced an RMSE on the Validation made a good improvement from the original Naive analysis. This value could potentially be lowered by considering other forms of effects such as:

- Arena capacity vs attendance - do larger crowds influence outcomes?
- Team Steaks - do teams on winning streaks perform better as the win more? Is there a point were this performance drops off?
- Away trips - do teams playing on the road perform better/worse?

Some of the data to assess these effects are available in the dataset and could be considered in a subsequent analysis.

Also, interestingly the data suggests the spread of multiple statistics is lowering over time, which indicates teams are on average becoming more competitive with each other. This could be due to various reasons, such as increasing professionalization of the sport, increased funding to teams/players based on higher popularity of the sport, or expanding revenue streams from advertising and media streaming service deals. These potential reasons are not able to be validated based on the data we have here, however they could be addressed in subsequent more thorough analysis of the NBA.