Baseball Project

Description

In this project, we will be using several datasets about baseball from the package 'Lahman'. You can access the list of datasets and all of the variables contained in each one by examining this package in the Packages tab in RStudio. Most of the code in this project is in tidyverse syntax.

Preamble

```
library("car")
```

```
library("tidyverse")
library("magrittr")
library("here")
library("janitor")
library("lubridate")
library("gridExtra")
library("readxl")
library("glmnet")
library("Lahman")
library("viridis")
library("lindia")
library("lme4")
library("caret")
library("pROC")
# install.packages("dbplyr", type="binary")
# install.packages("hardhat", type="binary")
```

1. Datasets

The first step is to create a new dataset called 'Peopledata' that contains all of the variables in the 'People' dataset by

```
i. removing all birth information except birthYear and birthCountry and all death informatio
n, along with the variable finalGame;


ii. replacing birthCountry is by bornUSA, a logical variable indicating if the player was bor
n in the USA;
```

```
People %>%
  select(!c("birthMonth", "birthDay", "birthState", "birthCity", "deathYear", "deathMonth",
"deathDay", "deathCountry", "deathState", "deathCity", "deathDate", "finalGame", "birthDate"
)) %>%
  rename(bornUSA = birthCountry) %>%
  mutate(bornUSA = bornUSA == "USA") -> Peopledata
```

Next we will create new datasets called Battingdata and Fieldingdata by

```
i. choosing data from the years 1985 and 2015,

ii. selecting only those variables that for those years have fewer than 25 missing cases,

iii. removing the variable 'G' from the batting dataset and removing the variables "teamID" a
nd "lgID" from both datasets,

iv. creating a variable in 'Battingdata' called batav which is equal to the number of hits
(H) over the number of at bats (AB) if the number of hits >0, and =0 if H=0.
```

```
Batting %>%
  filter(yearID %in% c(1985, 2015)) -> Batting_year_filtered

Batting_year_filtered %>%
sapply(function(x) sum(is.na(x))) < 25 -> na_checker

tibble(na_checker) %>%
  mutate(variables = names(na_checker)) %>%
  filter(na_checker == TRUE) %>%
  pull(variables) -> variables

Batting_year_filtered %>%
  select(variables) %>%
  select(!c("G", "teamID", "lgID")) -> Battingdata

Battingdata %>%
  mutate(
    batav = case_when(
      H > 0 ~ H/AB,
      H == 0 ~ 0
    )
  ) -> Battingdata

Fielding %>%
  filter(yearID %in% c(1985, 2015)) -> Fielding_year_filtered
Fielding_year_filtered %>%
sapply(function(x) sum(is.na(x))) < 25 -> na_checker2

tibble(na_checker2) %>%
  mutate(variables = names(na_checker2)) %>%
  filter(na_checker2 == TRUE) %>%
  pull(variables) -> variables2

Fielding_year_filtered %>%
  select(variables2) %>%
  select(!c("teamID", "lgID")) -> Fieldingdata
```

Next we will create a dataset ‘Playerdata’ from the dataset ‘Salaries’ by

```
i. selecting data from the years 1985 and 2015,

ii. adding all distinct variables from the Fieldingdata, Battingdata and Peopledata datasets,

iii. creating a new variable 'allstar' indicating if the player appears anywhere in the Allst
arFull dataset,

iv. creating a new variable 'age' equal to each player's age in the relevant year,

iv. dropping incomplete cases from the dataset,

v. dropping unused levels of any categorical variable.
```

```
Salaries %>%
  filter(yearID %in% c(1985, 2015)) %>%
  full_join(Fieldingdata) %>%
  full_join(Battingdata) %>%
  full_join(Peopledata) %>%
  mutate(allstar = playerID %in% AllstarFull$playerID) %>%
  mutate(age = yearID - birthYear) %>%
  drop_na() %>%
  droplevels() -> Playerdata
```

I then created a dataset called ‘TeamSalaries’ in which there is a row for each team and each year and the variables are:

```
i. 'Rostercost' = the sum of all player salaries for the given team in the given year

ii. 'meansalary' = the mean salary for that team that year

iii. 'rostersize' = the number of players listed that year for that team.
```

```
Salaries %>%
  group_by(teamID, yearID) %>%
  summarise(
  Rostercost = sum(salary),
  meansalary = mean(salary),
  rostersize = n_distinct(playerID)
  ) -> TeamSalaries
```

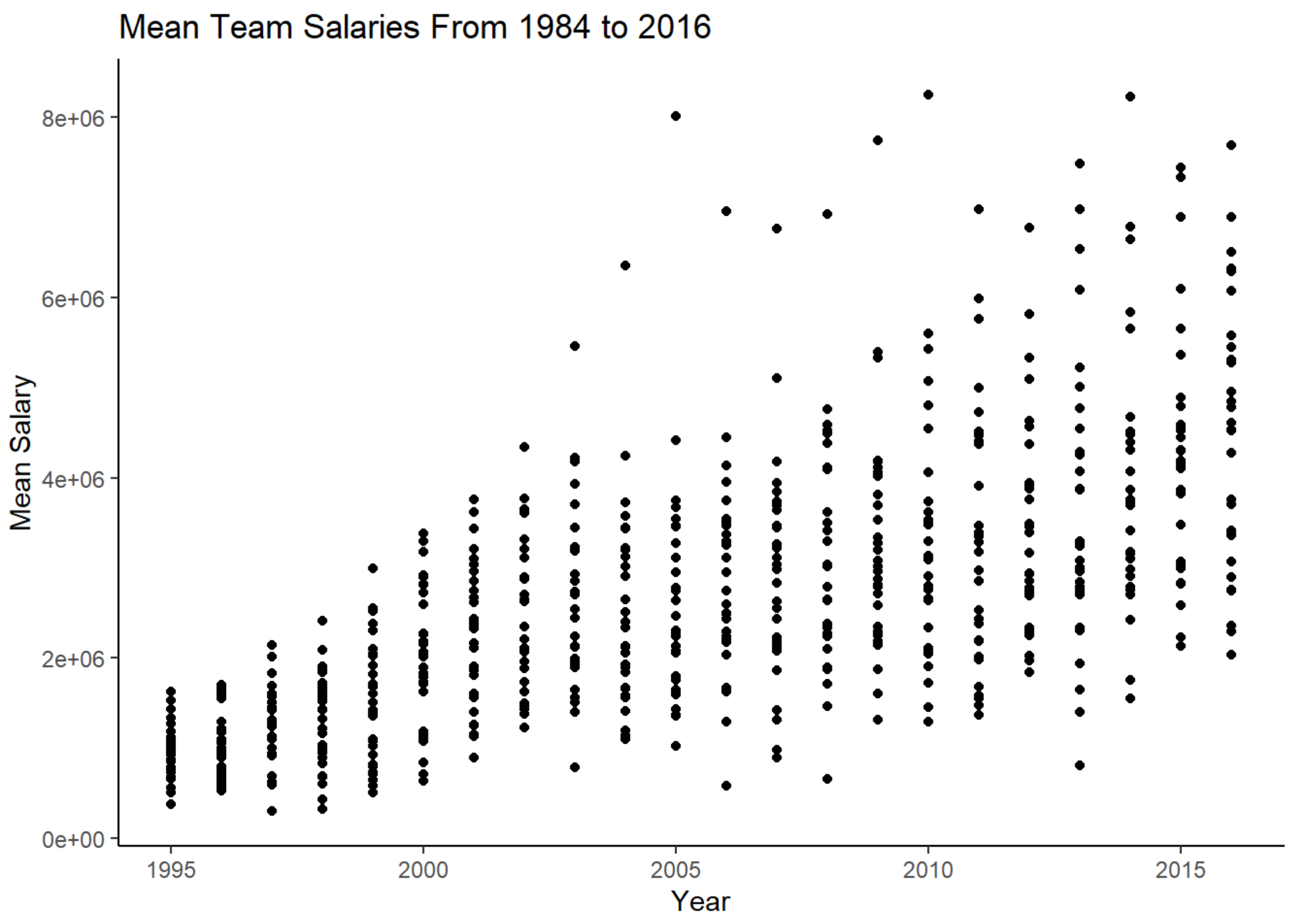
Finally, in this section I created a dataset ‘Teamdata’ by taking the data from the Teams dataset for the years 1984 to 2016, inclusive and adding to that data the variables in TeamSalaries. Drop any incomplete cases from the dataset.

```
Teams %>%
  filter(yearID %in% 1984:2016) %>%
  full_join(TeamSalaries) %>%
  drop_na() -> Teamdata
```

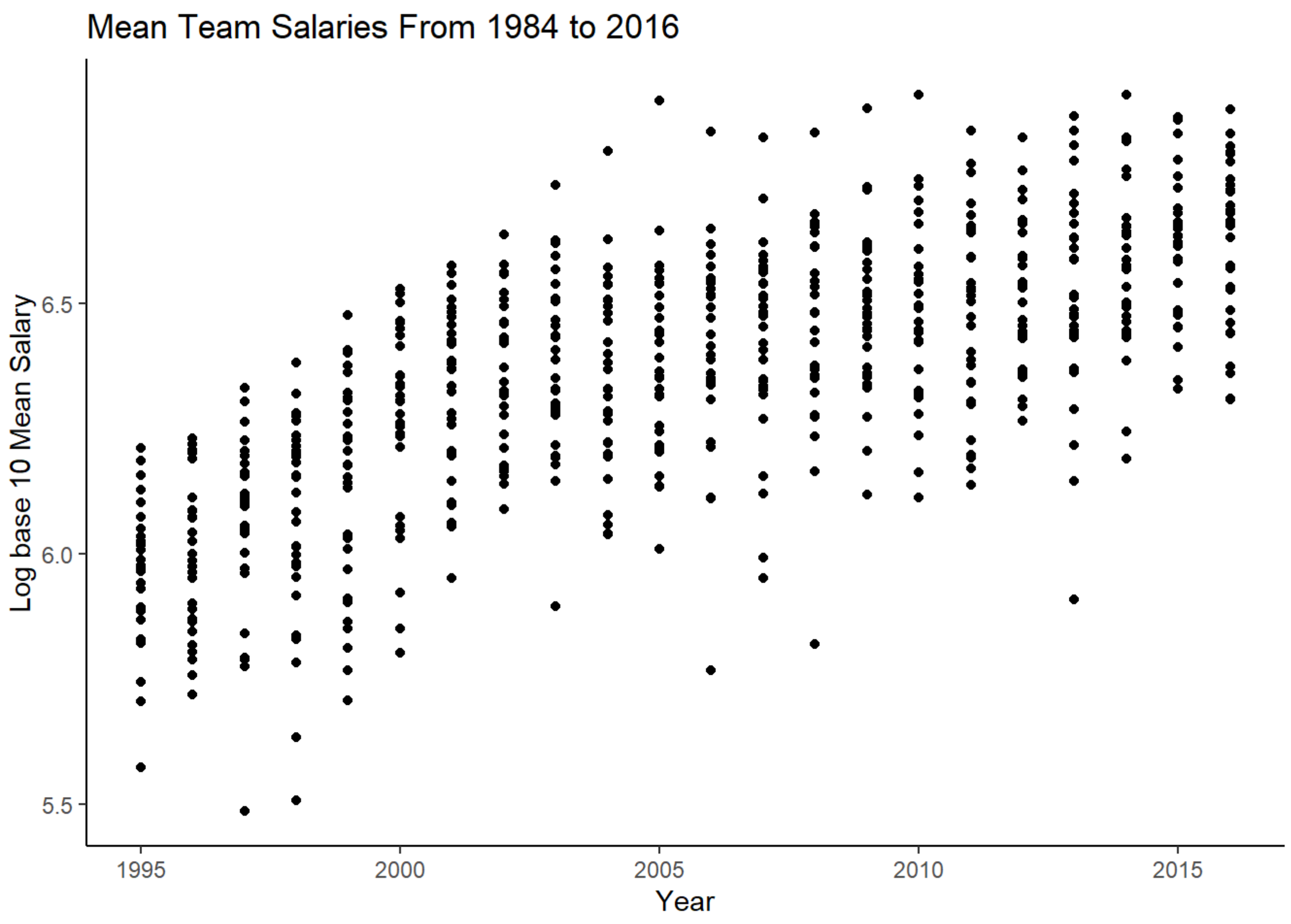
2. Simple Linear Regression

Here, we will first create one plot of mean team salaries over time from 1984 to 2016, and another of the log base 10 of team mean salaries over time from 1984 to 2016.

```
Teamdata %>%
  ggplot(mapping = aes(x = yearID, y = meansalary)) +
  geom_point() +
  labs(
    title = "Mean Team Salaries From 1984 to 2016",
    x = "Year",
    y = "Mean Salary"
  ) +
  theme_classic()
```



```
Teamdata %>%
  ggplot(mapping = aes(x = yearID, y = log10(meansalary))) +
  geom_point() +
  labs(
    title = "Mean Team Salaries From 1984 to 2016",
    x = "Year",
    y = "Log base 10 Mean Salary"
  ) +
  theme_classic()
```



- Upon visual inspection of the two plots, it is evident that the log base 10 plot has a more distinct linear trend as compared to the raw mean salaries plot. The log base 10 plot indicates that as the year increases, the log base 10 mean salaries increase as well. The raw mean salaries plot also shows a slight upward trend as well, but the spread of mean salaries for each year is very large, and thus linearity is harder to see. A linear model is therefore more suitable for the log base 10 mean salaries plot.
- A linear model is also appropriate for the log base 10 plot because the log transformations help make skewed data more normal.

Next, we will fit a model of log_{10}(meansalary) as a function of yearID.

```
linmod<-lm(log10(meansalary)~yearID,data=Teamdata)
linmod
summary(linmod)
```

```

Call:
lm(formula = log10(meansalary) ~ yearID, data = Teamdata)

Coefficients:
(Intercept)       yearID  
  -51.22242      0.02871  



Call:
lm(formula = log10(meansalary) ~ yearID, data = Teamdata)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.66345 -0.11692  0.00644  0.13394  0.55976 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -51.222416   2.310867  -22.17   <2e-16 ***
yearID        0.028711   0.001152   24.92   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1858 on 652 degrees of freedom
Multiple R-squared:  0.4878,	Adjusted R-squared:  0.487 
F-statistic: 620.9 on 1 and 652 DF,  p-value: < 2.2e-16
```

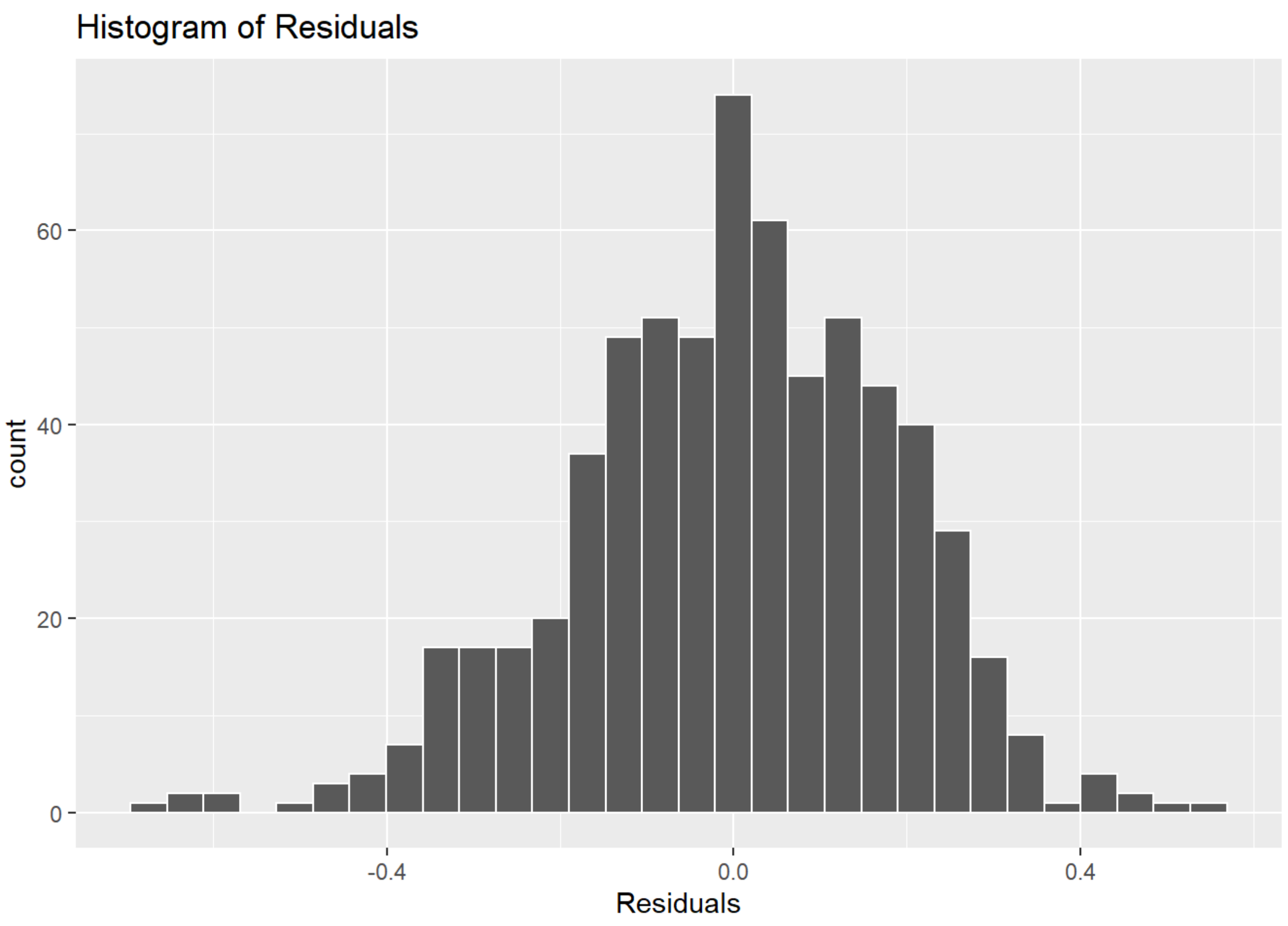
- The form of the model to 4 significant figures is:

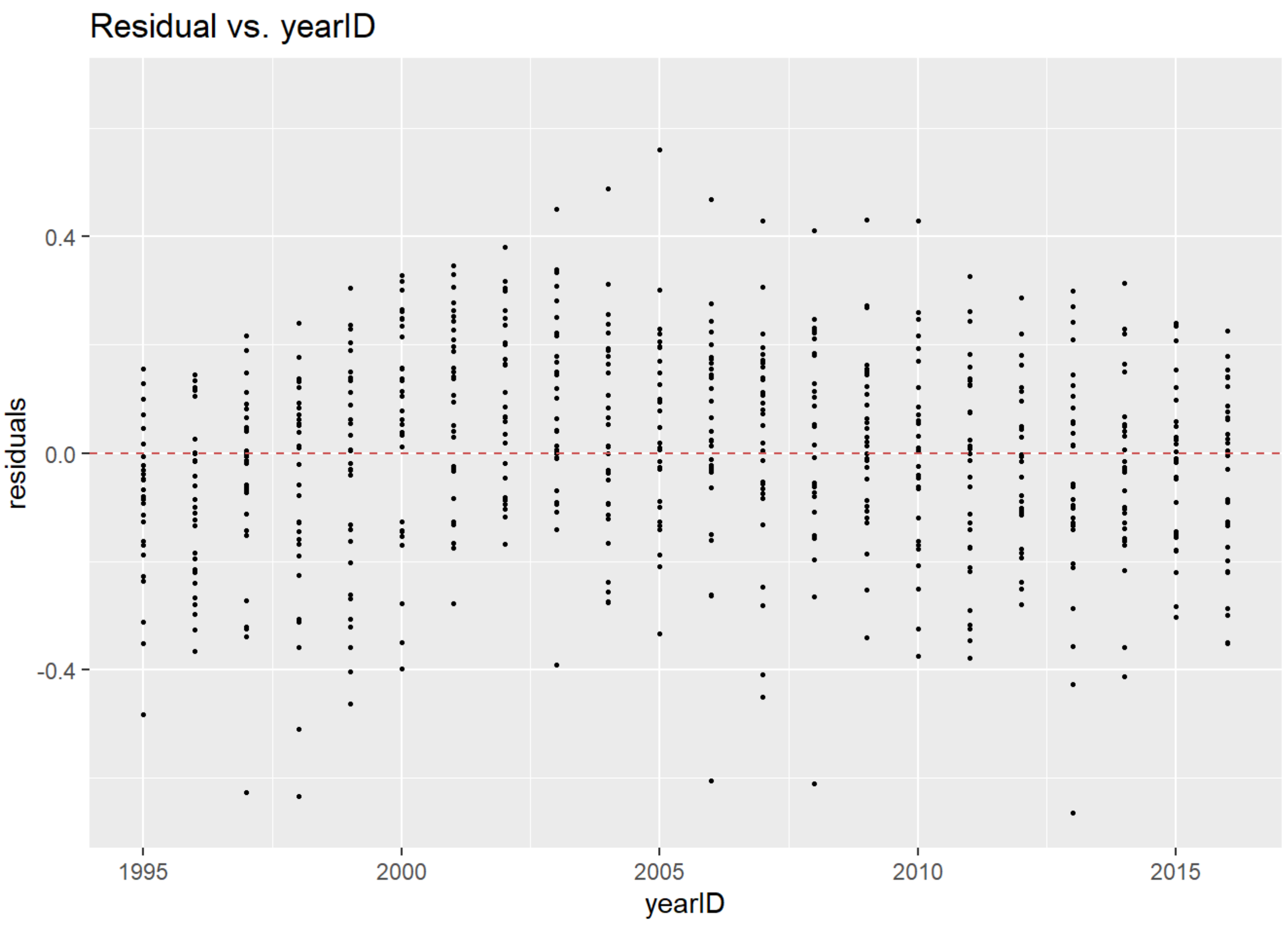
$$log_{10}(meansalary) \sim N(-51.22 + 0.0287 \times yearID, 0.1858)$$

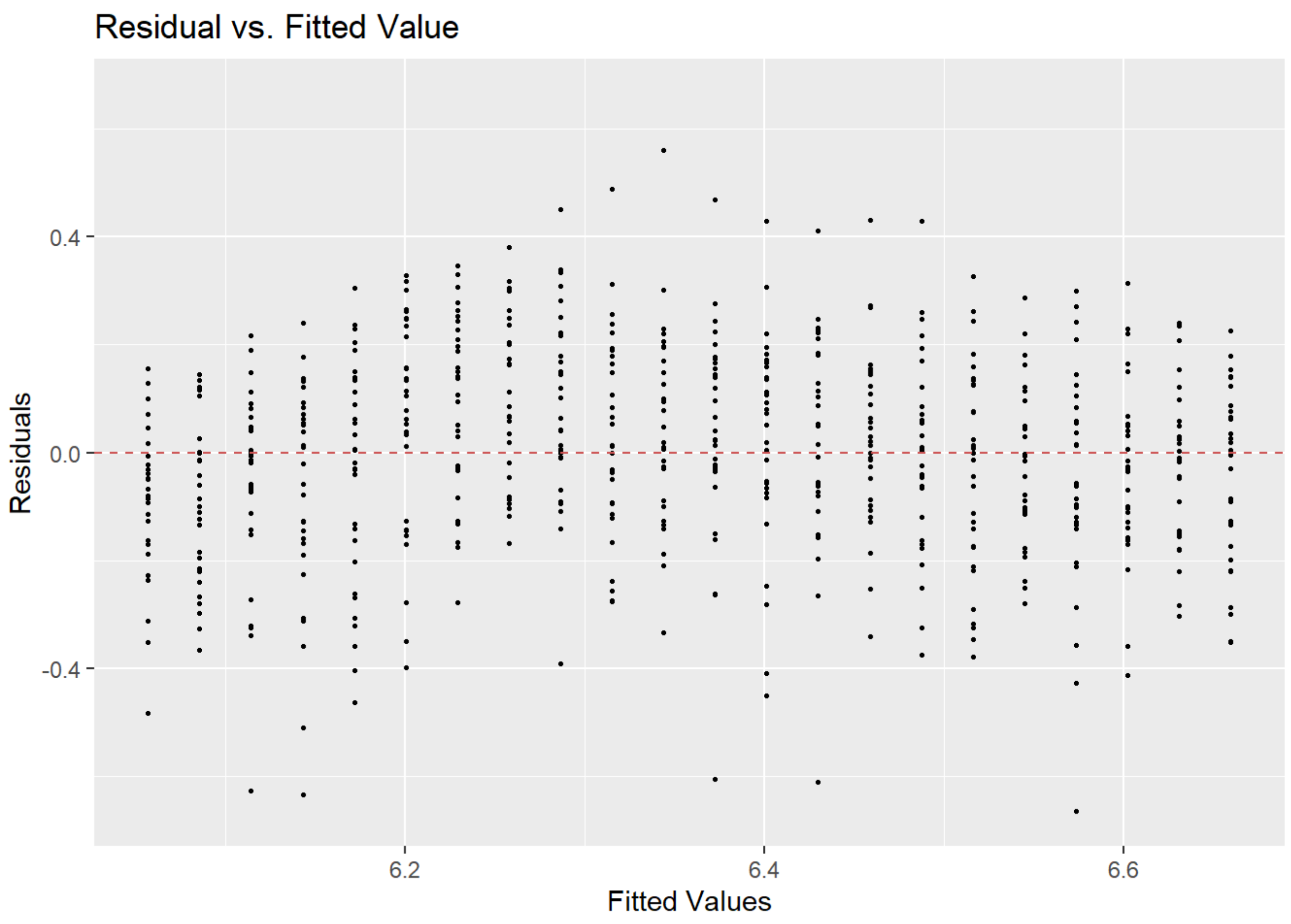
- The multiple R-Squared, which is 0.4878, tells us that 48.78% of the variance in log10(meansalary) is accounted for by the model.

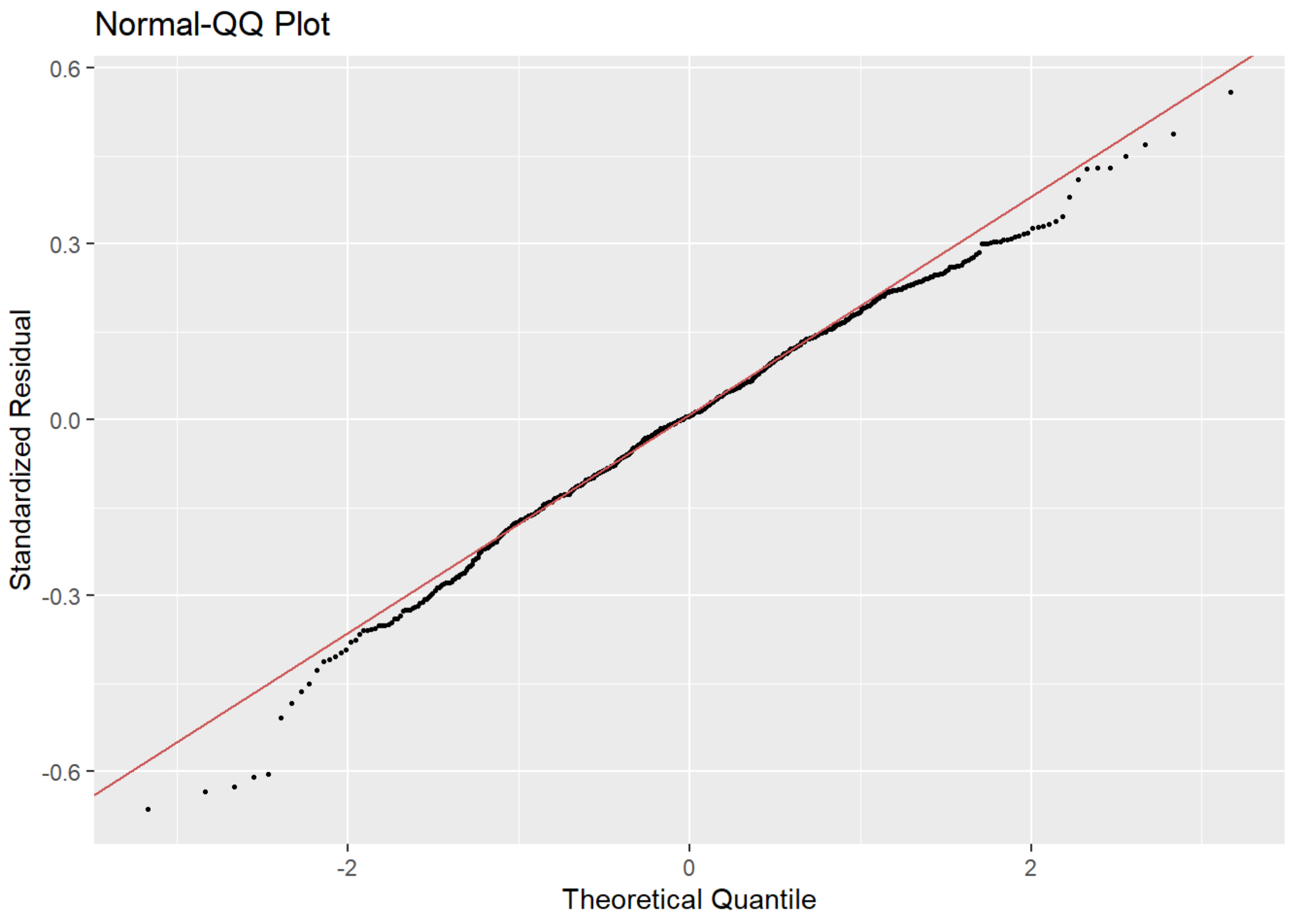
I will then evaluate four assumptions of linear models for this data.

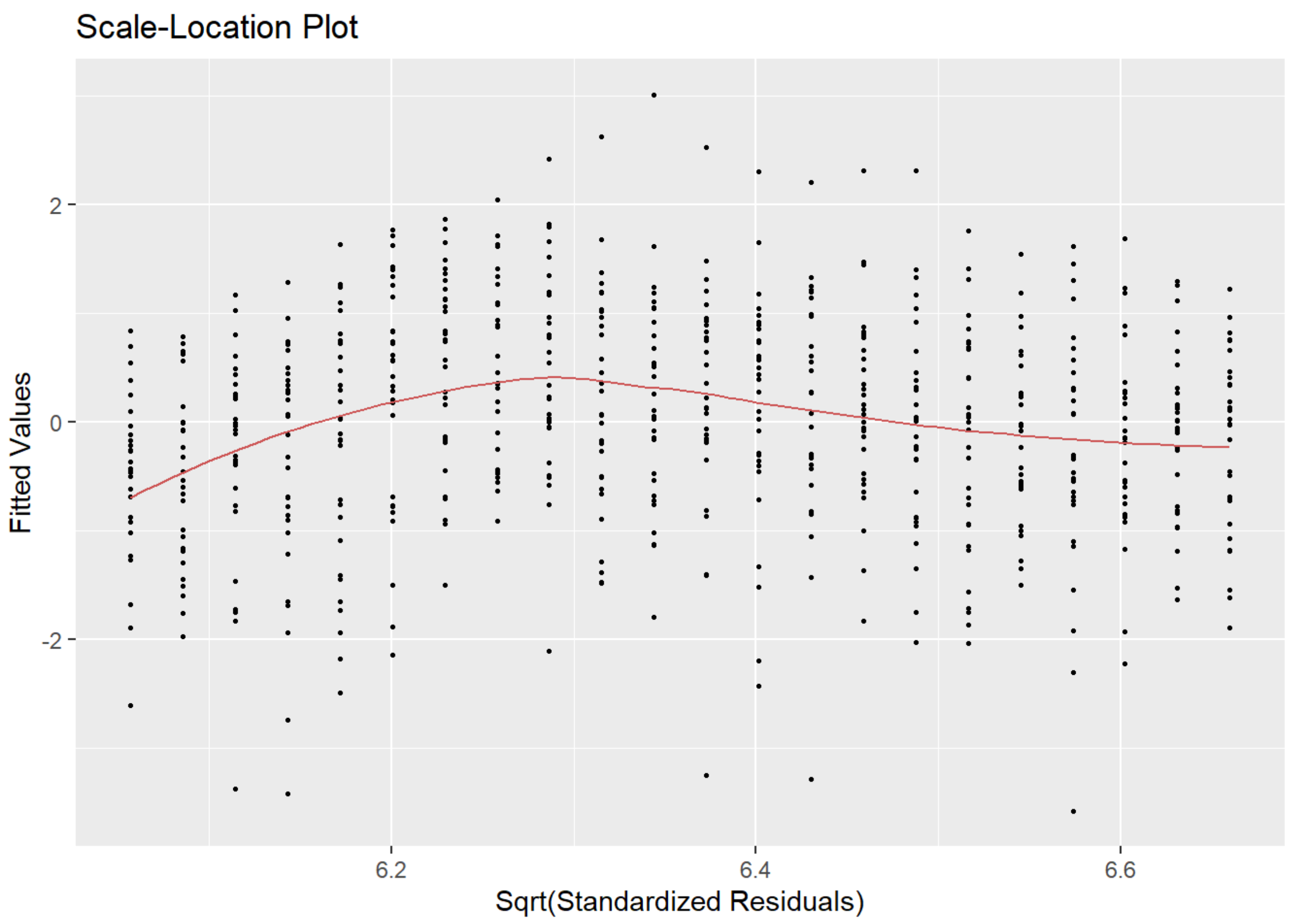
```
linmod %>%
  gg_diagnose(max.per.page = 1)
```

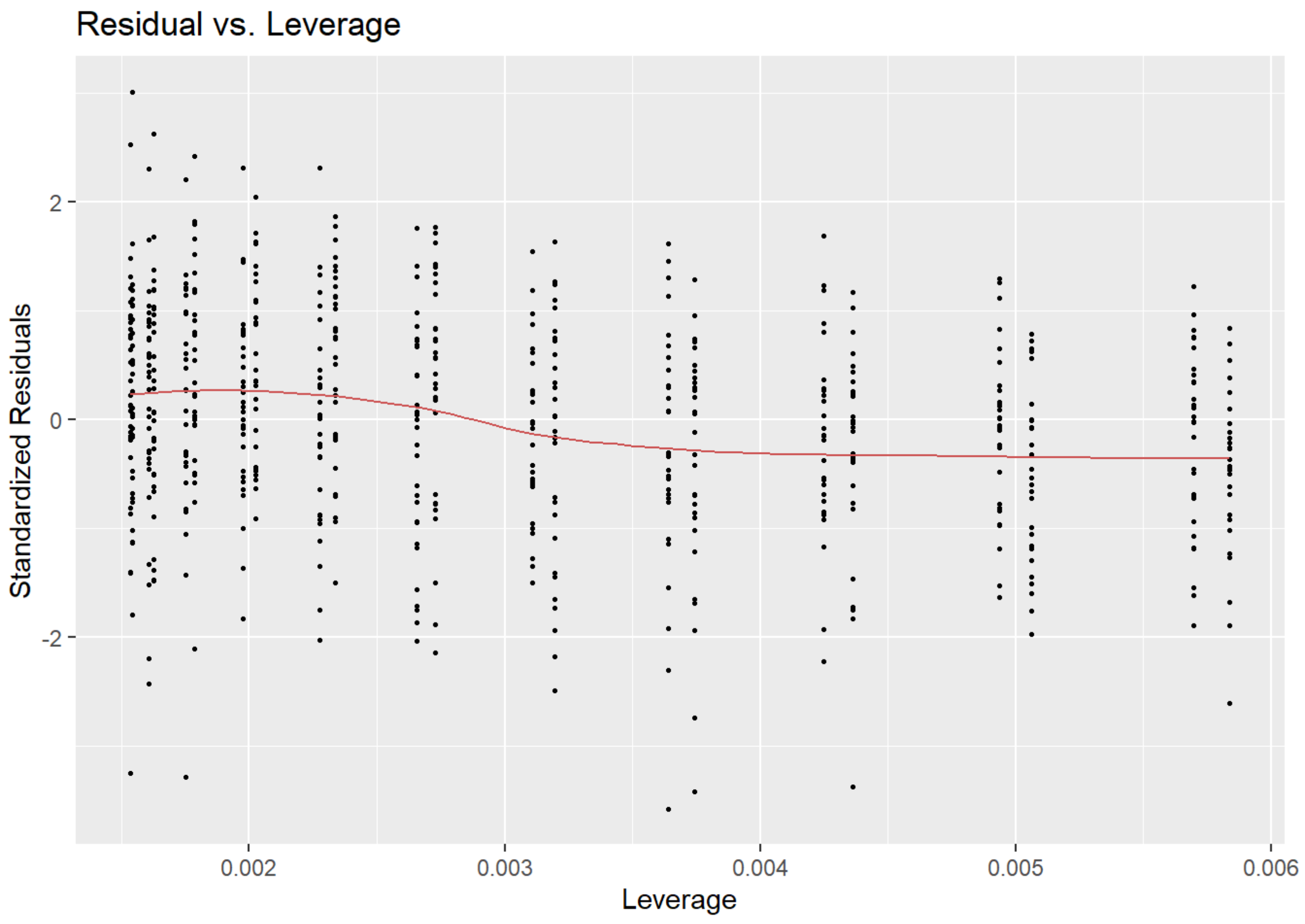

Histogram of Residuals
count
60
40
20
0
-0.4
0.0
0.4
Residuals

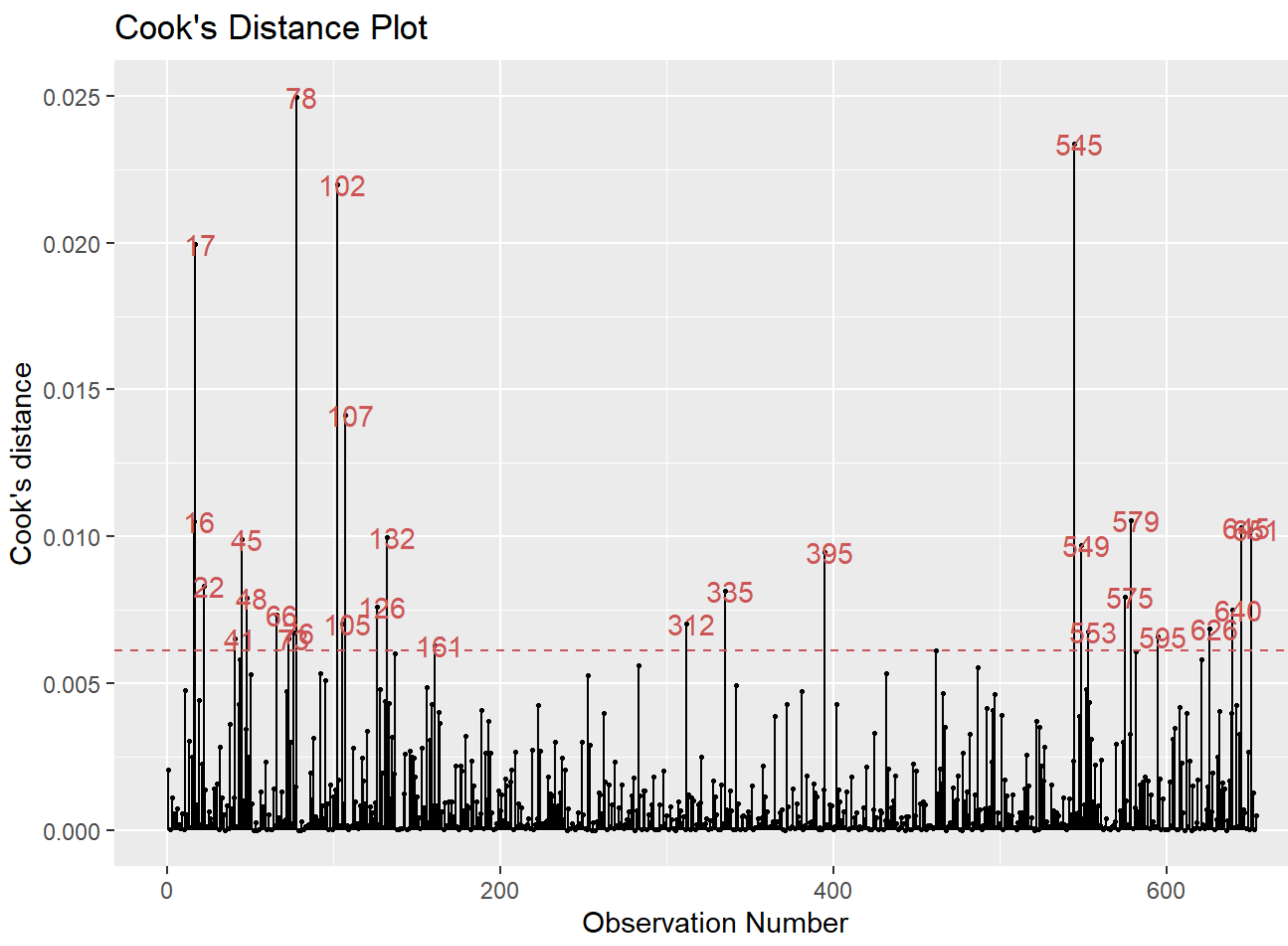

Residual vs. yearID
residuals
0.4
0.0
-0.4
1995
2000
2005
2010
2015
yearID


Residual vs. Fitted Value
Residuals
0.4
0.0
-0.4
6.2
6.4
6.6
Fitted Values


Normal-QQ Plot
Standardized Residual
0.6
0.3
0.0
-0.3
-0.6
-2
0
2
Theoretical Quantile


Scale-Location Plot
Fitted Values
Sqrt(Standardized Residuals)
2
0
-2
6.2
6.4
6.6


Residual vs. Leverage
Standardized Residuals
Leverage
2
0
-2
0.002
0.003
0.004
0.005
0.006



i. The first assumption to evaluate is linearity. This is checked using the residual vs fitted values plot. Linearity is confirmed when there is a horizontal line with no observable trend. The Residual vs Fitted Value plot for this model shows no observable trend, albeit the spread of values at each fitted value point slightly varies. This confirms that linearity is appropriate.

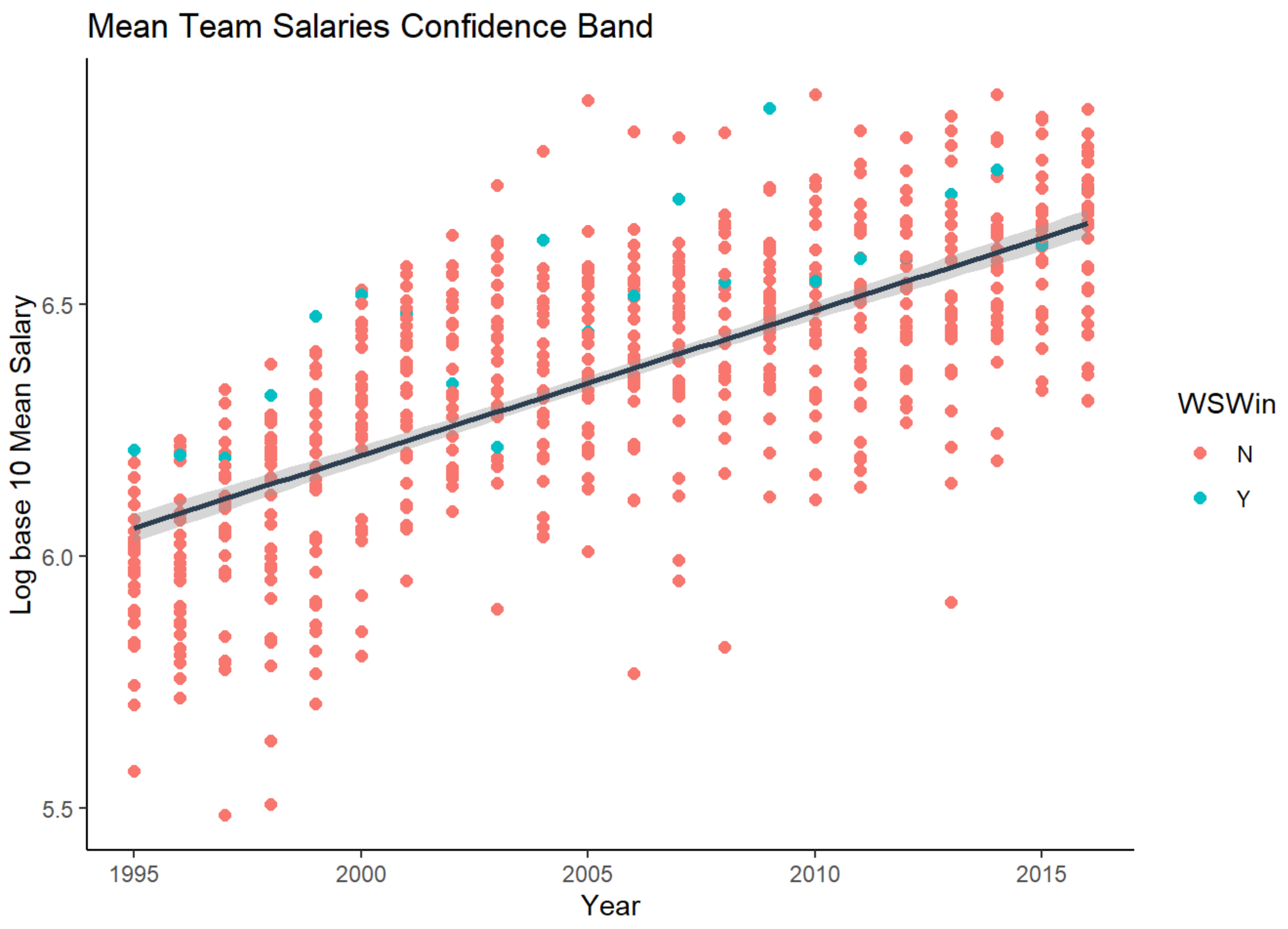
ii. The second assumption to evaluate is normality of residuals. Evidence of normality would be a straight line in a QQ plot. In the QQ plot for this model, the values have a very close to a straight line, thus confirming the assumption of normality.

iii. The third assumption to evaluate is homoscedasticity of residuals. A horizontal line with an even spread on the Scale-Location plot would confirm homoscedasticity. In this model's Scale-Location plot, there is a slight upward then downward trend, showing some evidence of heteroscedasticity. However, the trend line is not too far from the zero line, thus we can conclude that homoscedasticity is most dominant.

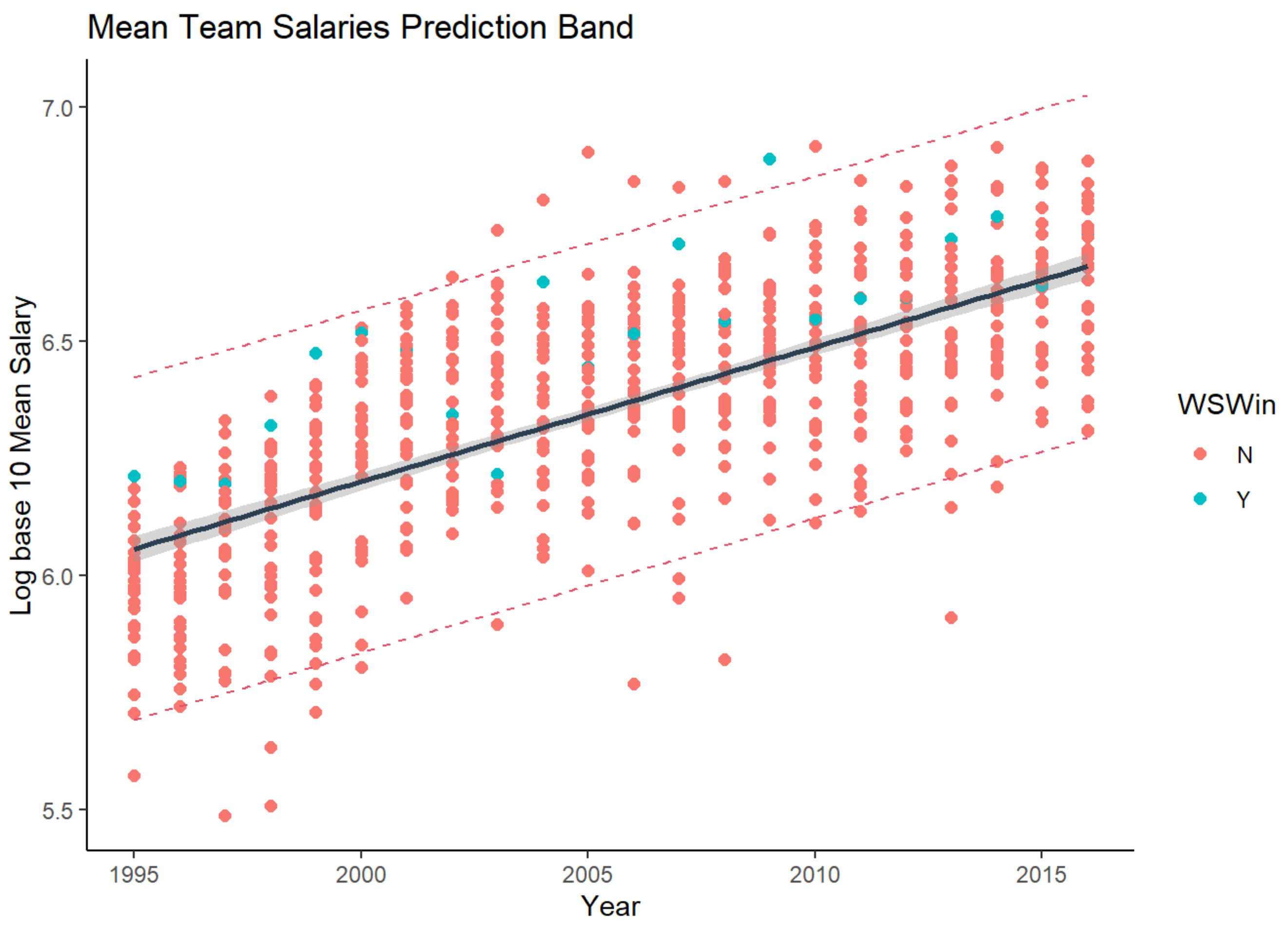
iv. The last assumption to evaluate is independence of residuals. The plot to investigate is residuals against predictors. For this model, the plot of residuals vs yearID is showing some slight wavy line trend, but it is close to the zero mark. We can therefore conclude that the residuals are mostly independent, but not fully independent.

Next, we will plot confidence and prediction bands for this model. The points will be coloured according to who won the World Series each year.

```
Teamdata %>%
  ggplot(mapping = aes(x = yearID, y = log10(meansalary), color = WSWin))+
  geom_point(size=2)+
  geom_smooth(method=lm, color='#2C3E50') +
  labs(
    title = "Mean Team Salaries Confidence Band",
    x = "Year",
    y = "Log base 10 Mean Salary"
  ) +
  theme_classic()
```



```
pred<-predict(linmod,interval="prediction")
Teamdata %>%
  cbind(pred) %>%
  ggplot(mapping = aes(yearID, log10(meansalary), color = WSWin)) +
  geom_point(size=2)+
  geom_smooth(method=lm, color='#2C3E50') +
  geom_line(aes(y=lwr), color=2,lty=2) +
  geom_line(aes(y=upr), color=2,lty=2) +
  labs(
    title = "Mean Team Salaries Prediction Band",
    x = "Year",
    y = "Log base 10 Mean Salary"
  ) +
  theme_classic()
```



- It is evident that there is not too much of a variation in the width of the confidence band. It is only slightly wider at the ends, and this indicates that there is a good amount of data collected for each year. It is also evident that most of the teams which won the world series each year have higher mean salaries as compared to the predicted average mean salary. There is only one winning team which had a mean salary below average, and that occurred in 2003.

We will then investigate the points that appear above the top prediction band.

```
Teamdata %>%
  cbind(pred) %>%
  mutate(log10salary = log10(meansalary)) %>%
  filter(log10salary > upr) %>%
  pull(name)
```

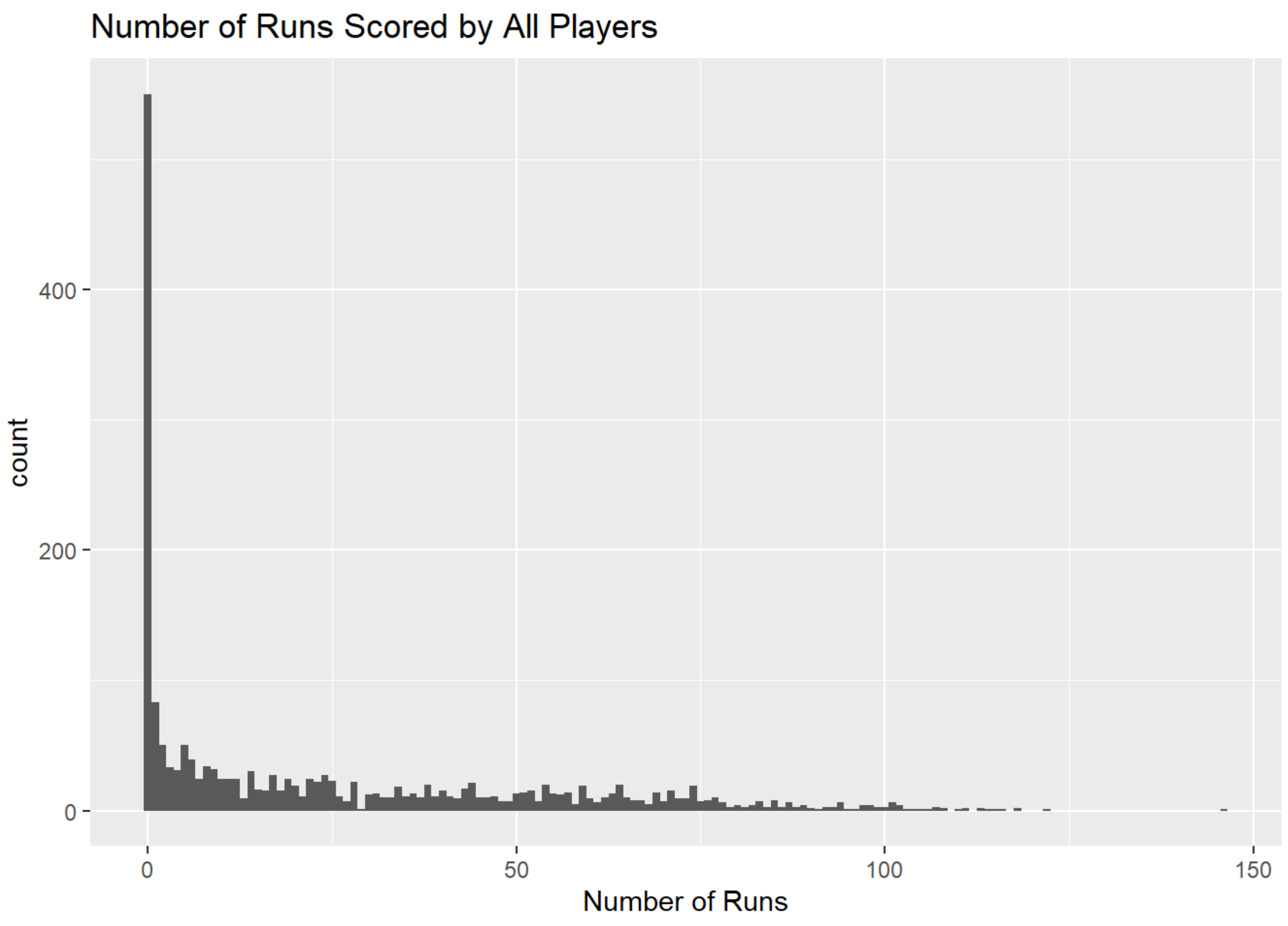
```
[1] "New York Yankees" "New York Yankees" "New York Yankees" "New York Yankees"
[5] "New York Yankees" "New York Yankees" "New York Yankees" "New York Yankees"
[9] "New York Yankees"
```

- The points that appear above the top prediction band relate to the New York Yankees team.

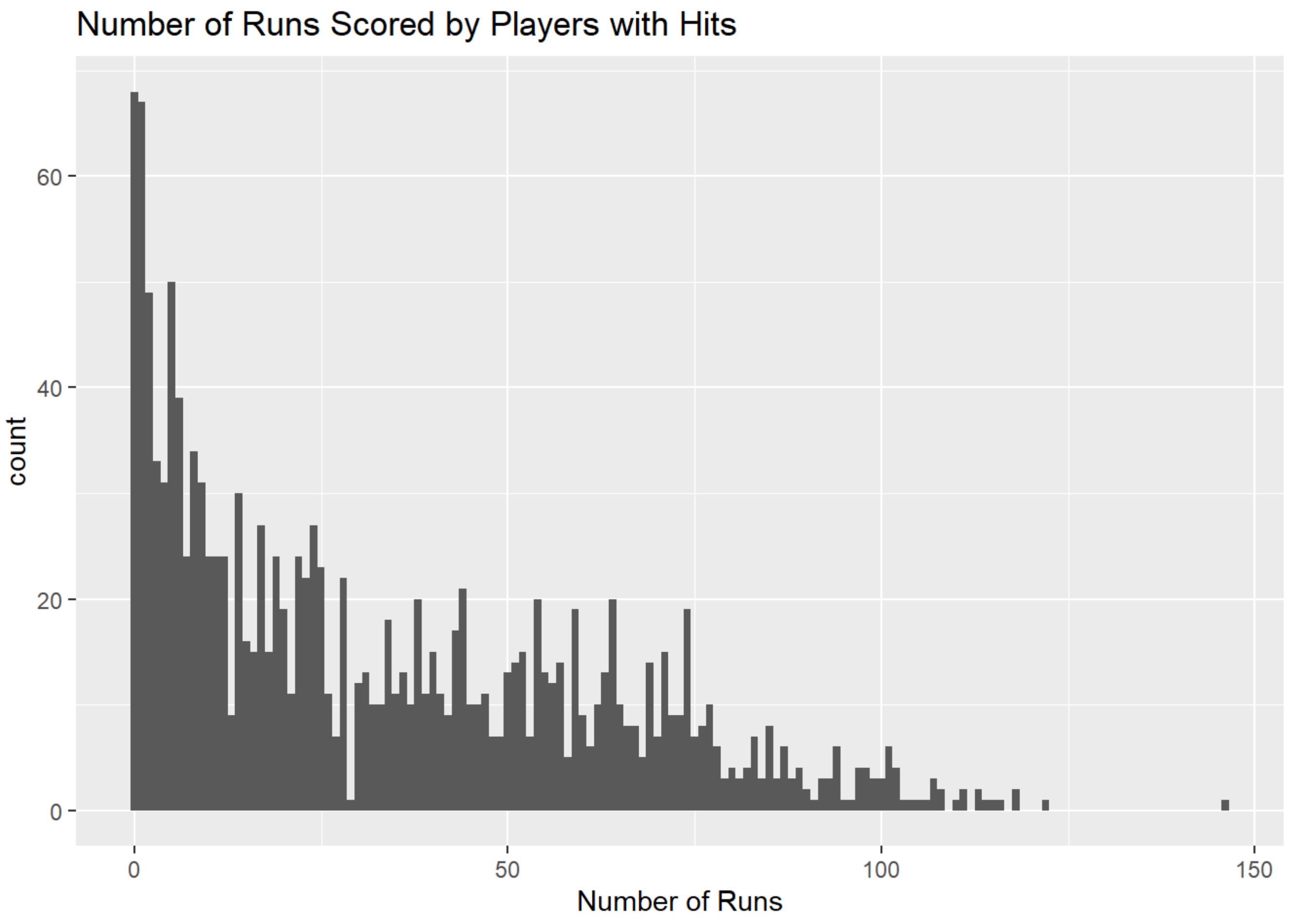
3. Multiple regression for Count Data

In this section, first we will create a histogram of the number of runs scored for players in the Playerdata dataset so each bar is a single value (0,1,2 runs, etc). Next we will create a histogram of the number of runs for all players who have had a hit.

```
Playerdata %>%
  ggplot(mapping = aes(x = R)) +
  geom_histogram(binwidth = 1) +
  labs(
    title = "Number of Runs Scored by All Players",
    x = "Number of Runs"
  )
```



```
Playerdata %>%
  filter(H > 0) %>%
  ggplot(mapping = aes(x = R)) +
  geom_histogram(binwidth = 1) +
  labs(
    title = "Number of Runs Scored by Players with Hits",
    x = "Number of Runs"
  )
```



- Players who have had a hit are more likely to score a run than players who have had 0 hits. One needs a hit to move on to first base, then second base, then third base and finally home for them to score a run. Players who have 0 hits have not even moved to first base, so the runs will automatically be 0, hence they have been removed and the remaining players who have had hits form a decaying exponential plot, with players who have had 0 and 1 runs being the most, and better players having 100+ runs being the least. This confirms that a Poisson distribution will be more suitable for the second plot, since most players are situated around zero, and a decreasing exponential curve is seen with less players having more runs. In addition, number of runs can't be negative, and will always be a positive count variable, hence confirming that a Poisson distribution is suitable. According to the data, the second set should have a Poisson distribution because the shape of the data forms a decaying exponential.

Next, we will create a new dataset, OnBase of all players who have had at least one hit, transform yearID to a factor and construct a Poisson model, glm1, of the number of runs as a function of the number of hits, the year as a factor, position played and player height and age.

```
Playerdata %>%
  filter(H > 0) %>%
  mutate(yearID = as.factor(yearID)) -> OnBase

glm1 <- glm(R ~ H + yearID + POS + height + age , data = OnBase,family="poisson")
summary(glm1)
```

```

Call:
glm(formula = R ~ H + yearID + POS + height + age, family = "poisson",
    data = OnBase)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-9.1815  -1.5838  -0.2644   1.0703   7.9465

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.464e+00  1.690e-01  14.579  < 2e-16 ***
H            1.285e-02  8.954e-05 143.540  < 2e-16 ***
yearID2015   2.221e-02  9.286e-03   2.392   0.0168 *
POS2B       -1.071e-02  1.671e-02  -0.641   0.5214
POS3B        5.674e-03  1.574e-02   0.360   0.7185
POSC        -6.271e-02  2.074e-02  -3.024   0.0025 **
POSOF        6.341e-02  1.319e-02   4.806 1.54e-06 ***
POSP        -1.171e+00  3.710e-02 -31.568  < 2e-16 ***
POSSS       -1.068e-02  1.754e-02  -0.609   0.5428
height      -3.176e-03  2.239e-03  -1.419   0.1560
age          4.692e-03  1.202e-03   3.903 9.51e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 38805.9  on 1481  degrees of freedom
Residual deviance:  5696.4  on 1471  degrees of freedom
AIC: 12616

Number of Fisher Scoring iterations: 5
```

We will then find the p-value for each of the predictor variables in this model using a Likelihood Ratio Test.

```
Anova(glm1)
```

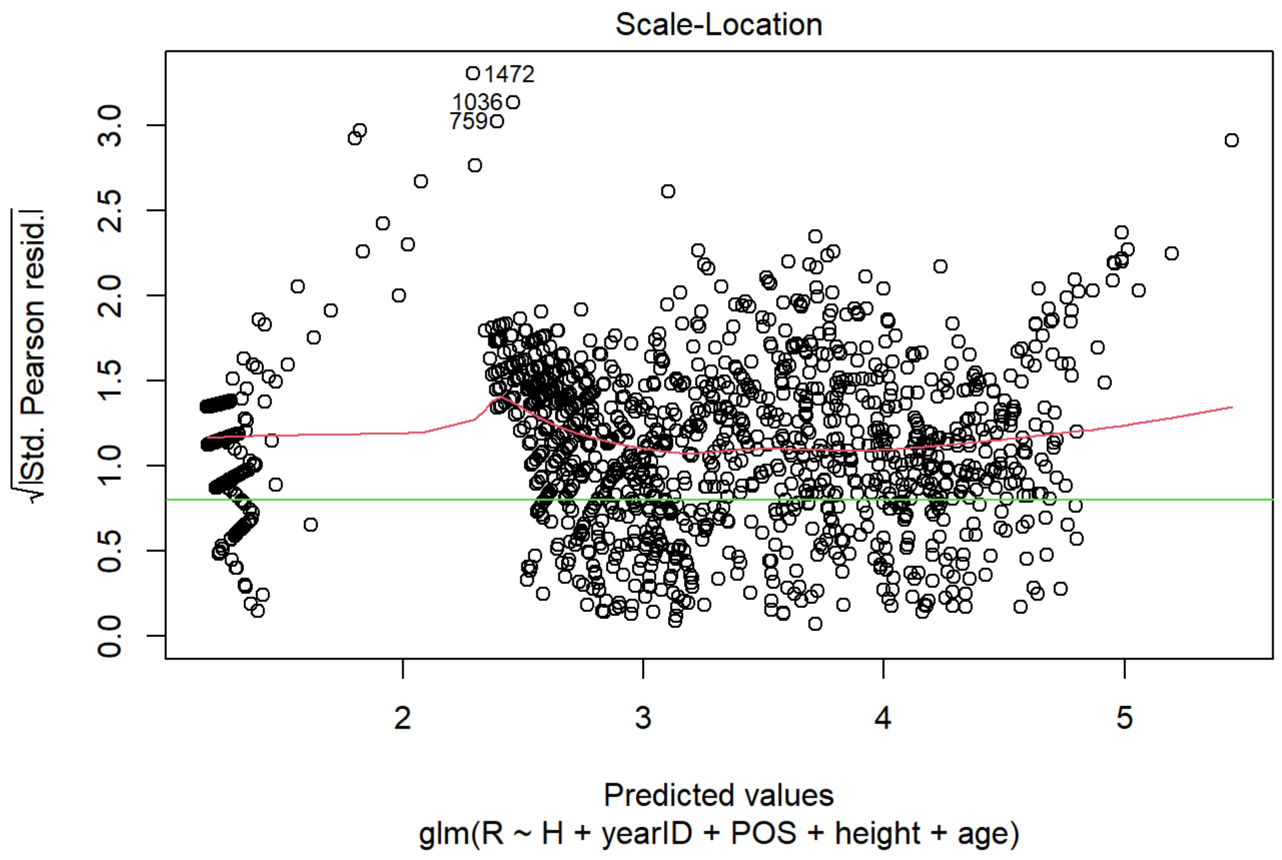
```
Analysis of Deviance Table (Type II tests)

Response: R
       LR Chisq Df Pr(>Chisq)
H       21529.6  1  < 2.2e-16 ***
yearID      5.7  1    0.01671 *
POS      1586.1  6  < 2.2e-16 ***
height      2.0  1    0.15610
age        15.2  1   9.75e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- A p-value tests the hypothesis whether a variable is statistically significant. A p-value is between 0 and 1. For a variable to be statistically significant, it’s p value must be less than or equal to 0.05. The p-value threshold of 0.05 tells us that there is a probability of 0.05 that an observed difference occurred randomly. The p-value for POS is very small at a value of 2.2e-16, meaning there is a probability of 2.2e-16 that variations caused by POS occurred randomly. This shows that POS is statistically significant, since the p-value is less than the threshold of 0.05. However, the p-value for height is 0.1099, and this is larger than the threshold of 0.05. This means that height is not a statistically significant variable.

We will then state the assumptions of Poisson models and check these where possible.

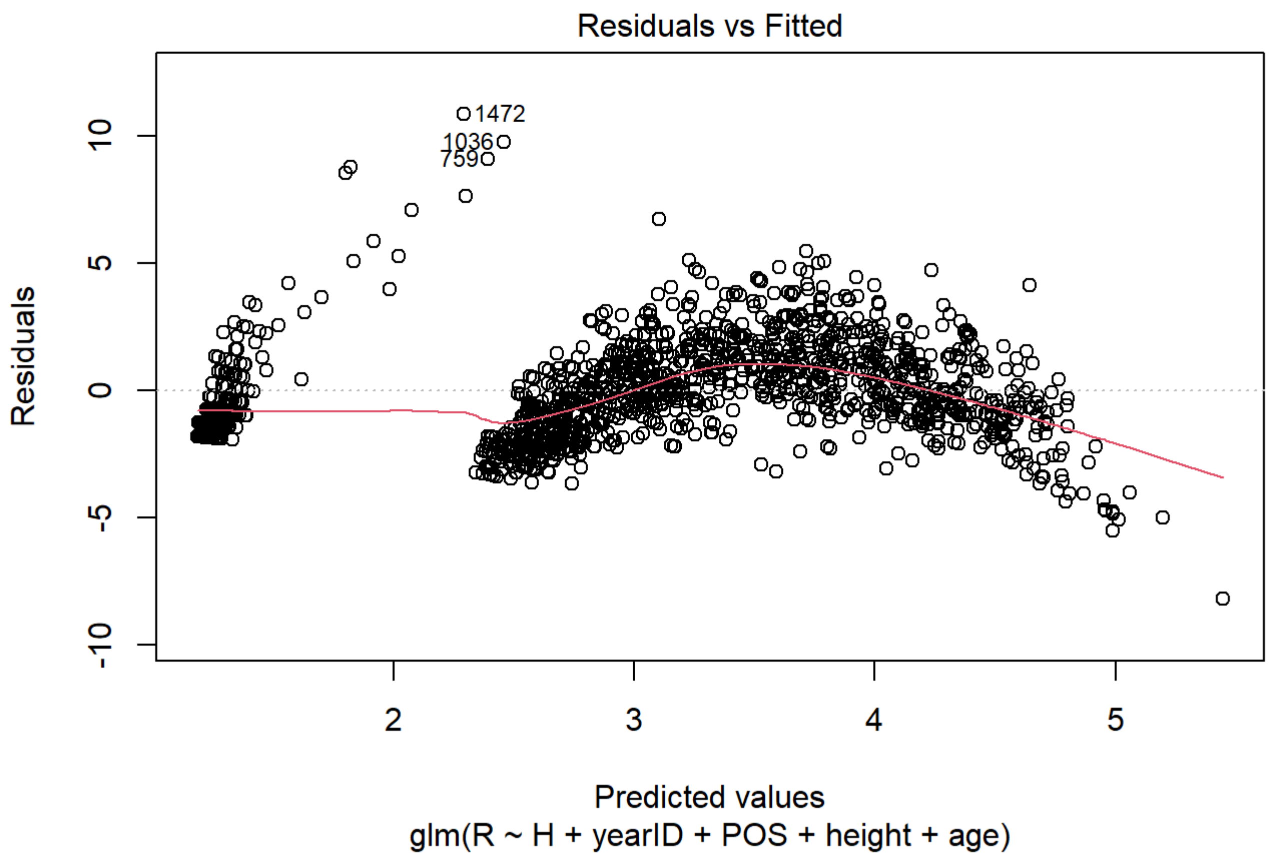
i. The first assumption of a Poisson model is that of dispersion. There should be no under-dispersion and overdispersion. For this to happen, the variance should be equal to mean. This is checked by plotting absolute value of residuals versus predicted means. The trend line should look flat and be around 0.8. In this model, the data suggests overdispersion as the red line is above the 0.8 mark, and is between 1.0 and 1.5. This means that the choice of predictor variables should be improved.

```
plot(glm1,which=3)
abline(h=0.8,col=3)
```



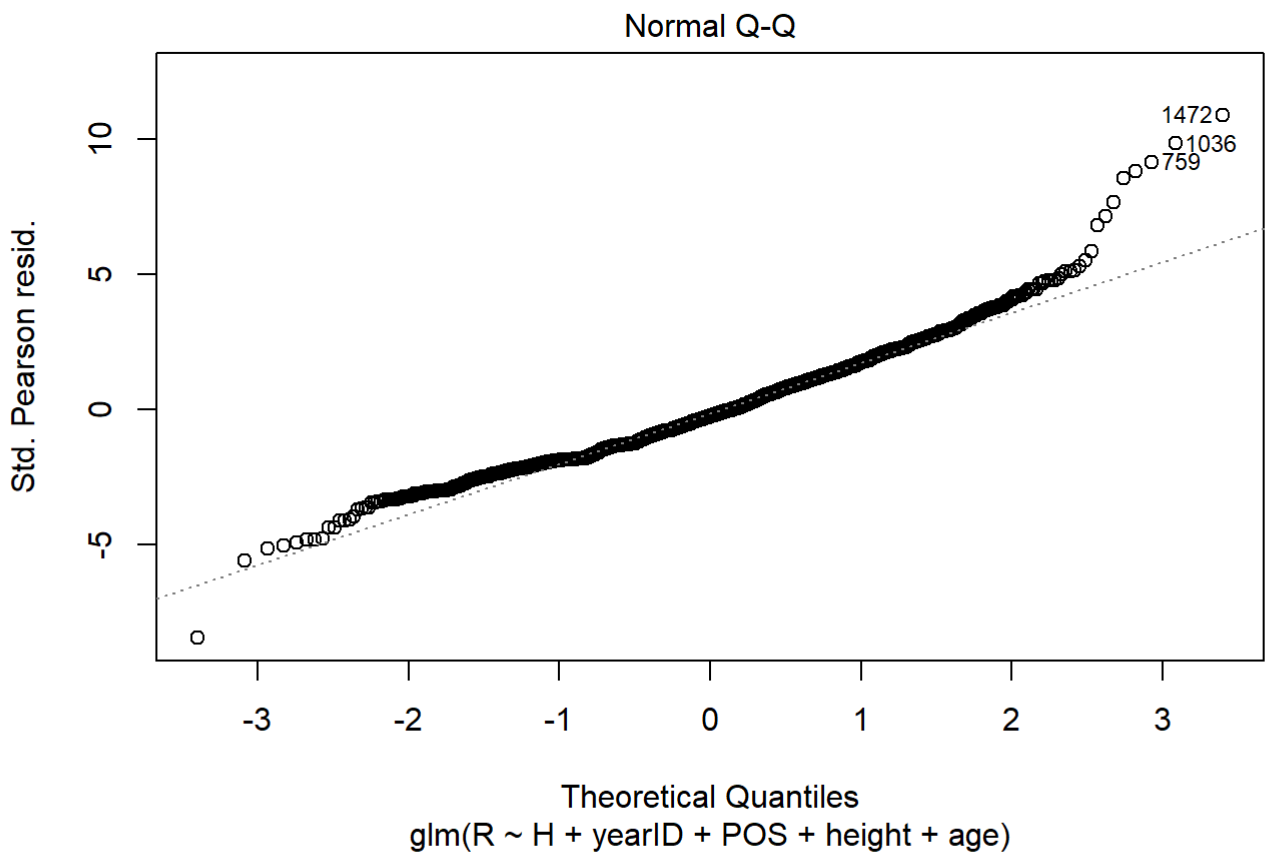
ii. The second assumption to check is linearity. This is evaluated by looking at the Residuals vs Fitted plot and a flat trend line on the zero mark would indicate linearity. The trend line in this model's Residual vs Fitted plot is not entirely flat, as it is first close to below zero, rises slightly above it and then linearly decreases below zero. The model therefore is mostly linear but shows some indication of non-linearity.

```
plot(glm1,which=1)
```



iii. The third assumption to check is residual distribution, and for this the QQ plot is investigated. The QQ plot for this model is fairly straight, thus indicating the residuals are normally distributed.

```
plot(glm1,which=2)
```



iv. The last assumption to check is independence. The residuals are investigated as a function of order of data points and evidence of snaking is looked for. There is no natural order in this data set hence can't be investigated.

Now we will create a new model that includes teamID as a random effect.

```
glm2 <- glmer(R ~ H + yearID + POS + height + age + (1 | teamID) , data = OnBase, family="poi
sson", nAGQ=0)
glm2
```

```
Generalized linear mixed model fit by maximum likelihood (Adaptive
  Gauss-Hermite Quadrature, nAGQ = 0) [glmerMod]
 Family: poisson  ( log )
Formula: R ~ H + yearID + POS + height + age + (1 | teamID)
   Data: OnBase
      AIC       BIC    logLik  deviance  df.resid
12286.986 12350.600 -6131.493 12262.986      1470
Random effects:
 Groups Name        Std.Dev.
 teamID (Intercept) 0.09652
Number of obs: 1482, groups:  teamID, 33
Fixed Effects:
(Intercept)            H   yearID2015        POS2B        POS3B         POSC
   2.598421     0.012996     0.031987    -0.009269     0.009236    -0.062606
      POSOF         POSP        POSSS       height          age
   0.065719    -1.138692    -0.010687    -0.005451     0.004779
```

- The teamsID random effect has a standard deviation of 0.0965, which tells us that a good team will score $exp(2 \times 0.0965) = 1.213$ times more runs than an average team. So, if an average team had a total of 4 runs, being in a good team would indicate that about 5 runs would be scored. This is a relatively small effect, since final baseball scores are not that high. The statistical significance of this effect can be obtained by comparing the AIC values of the model that doesn’t have the random effect (glm1) to the model with the random effect (glm2). The first model, glm1, has an AIC of 12616 and the second model has an AIC of 12286.5. Since the latter has a lower value, it means that the model with the teamsID random effect is more statistically significant since it is a more explanatory model.

```
summary(glm1)
summary(glm2)
```

```
Call:
glm(formula = R ~ H + yearID + POS + height + age, family = "poisson",
    data = OnBase)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-9.1815  -1.5838  -0.2644   1.0703   7.9465

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.464e+00  1.690e-01  14.579  < 2e-16 ***
H            1.285e-02  8.954e-05 143.540  < 2e-16 ***
yearID2015   2.221e-02  9.286e-03   2.392   0.0168 *
POS2B       -1.071e-02  1.671e-02  -0.641   0.5214
POS3B        5.674e-03  1.574e-02   0.360   0.7185
POSC        -6.271e-02  2.074e-02  -3.024   0.0025 **
POSOF        6.341e-02  1.319e-02   4.806 1.54e-06 ***
POSP        -1.171e+00  3.710e-02 -31.568  < 2e-16 ***
POSSS       -1.068e-02  1.754e-02  -0.609   0.5428
height      -3.176e-03  2.239e-03  -1.419   0.1560
age          4.692e-03  1.202e-03   3.903 9.51e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 38805.9  on 1481  degrees of freedom
Residual deviance:  5696.4  on 1471  degrees of freedom
AIC: 12616

Number of Fisher Scoring iterations: 5

Generalized linear mixed model fit by maximum likelihood (Adaptive
  Gauss-Hermite Quadrature, nAGQ = 0) [glmerMod]
 Family: poisson  ( log )
Formula: R ~ H + yearID + POS + height + age + (1 | teamID)
   Data: OnBase

     AIC      BIC   logLik deviance df.resid
 12287.0  12350.6  -6131.5  12263.0     1470

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.7835 -1.4362 -0.2170  0.9967 10.9330

Random effects:
 Groups Name        Variance Std.Dev.
 teamID (Intercept) 0.009316 0.09652
Number of obs: 1482, groups:  teamID, 33

Fixed effects:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.598e+00  1.747e-01  14.870  < 2e-16 ***
H            1.300e-02  9.189e-05 141.430  < 2e-16 ***
```

```
yearID2015   3.199e-02  1.010e-02   3.167 0.001543 **
POS2B       -9.269e-03  1.698e-02  -0.546 0.585097
POS3B        9.236e-03  1.585e-02   0.583 0.560020
POSC        -6.261e-02  2.085e-02  -3.002 0.002681 **
POSOF        6.572e-02  1.330e-02   4.942 7.74e-07 ***
POSP        -1.139e+00  3.730e-02 -30.529  < 2e-16 ***
POSSS       -1.069e-02  1.768e-02  -0.605 0.545461
height      -5.451e-03  2.304e-03  -2.366 0.017985 *
age          4.779e-03  1.256e-03   3.804 0.000143 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
           (Intr) H      yID201 POS2B  POS3B  POSC   POSOF  POSP   POSSS
H          -0.035
yearID2015 -0.026  0.153
POS2B      -0.320 -0.003 -0.030
POS3B      -0.199  0.024  0.004  0.462
POSC       -0.134  0.115  0.024  0.341  0.352
POSOF      -0.164  0.026  0.022  0.531  0.544  0.410
POSP        0.016  0.247  0.021  0.171  0.188  0.167  0.228
POSSS      -0.258  0.004 -0.013  0.444  0.433  0.321  0.503  0.164
height     -0.962 -0.084 -0.038  0.264  0.148  0.089  0.095 -0.067  0.195
age        -0.233  0.226  0.093  0.124  0.065  0.044  0.096  0.068  0.153
           height
H
yearID2015
POS2B
POS3B
POSC
POSOF
POSP
POSSS
height
age        -0.004
```

We will then check What is the mean number of runs we expect 30-year old, 72 inch tall outfielders playing for the Baltimore Orioles in 2015 with 20 hits to have scored.

```
predict(glm2,newdata=data.frame(age = 30, height = 72 , teamID = "BAL", yearID = "2015", H =
20, POS = "OF"))
```

```
       1
2.864109
```

- The mean number of runs predicted is 2.864

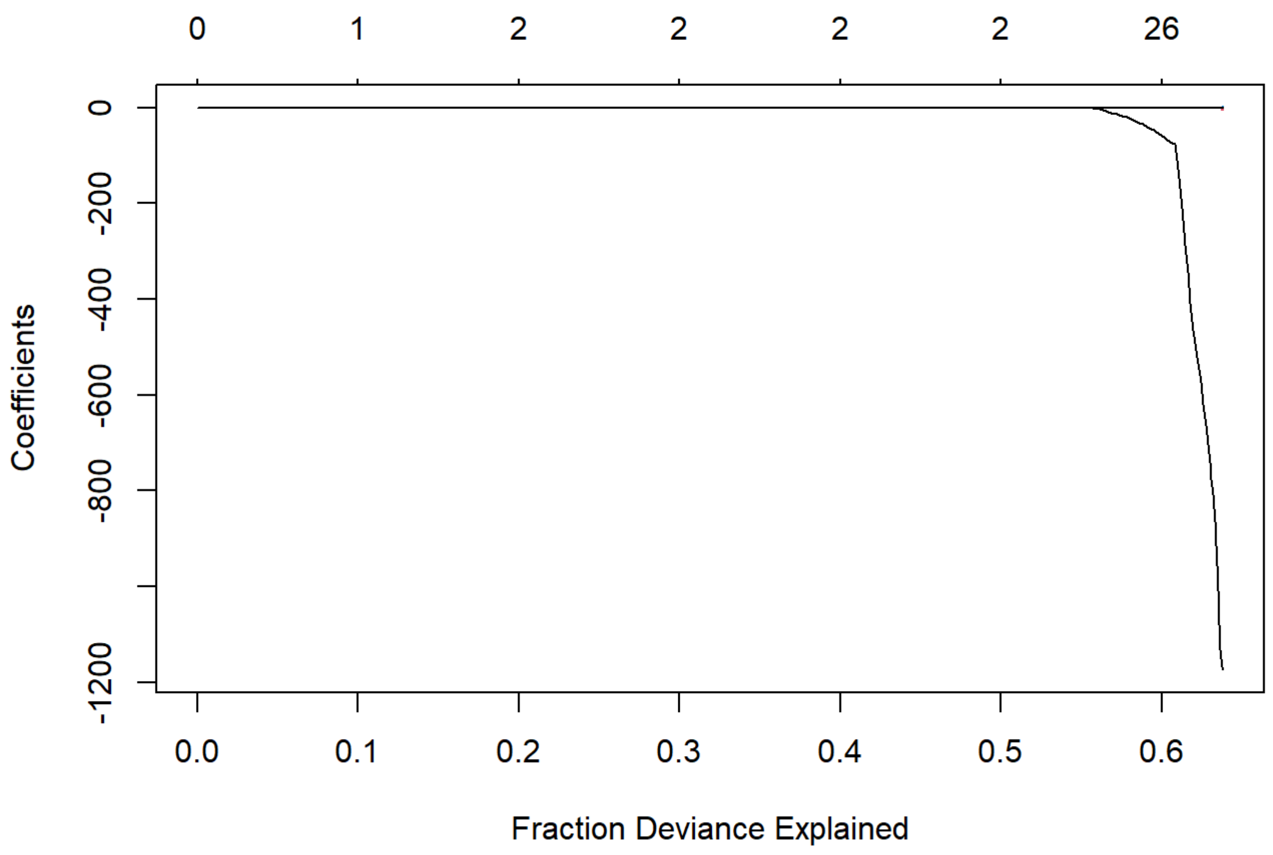
4. Lasso Regression for Logistic Regression

In this section we will create a new dataset DivWinners by removing all of the variables that are team or park identifiers in the dataset, as well as ‘lgID’, ‘Rank’,‘franchID’,‘divID’, ‘WCWin’,‘LgWin’, and ‘WSwin’. We will then split the resulting into a training and a testing set so that the variable ‘DivWin’ is balanced between the two datasets. The seed is 123.

```
Teamdata %>%
  select(! c("lgID", "Rank", "franchID", "divID", "WCWin", "LgWin", "WSWin", "teamID", "name"
, "park", "teamIDBR", "teamIDlahman45", "teamIDretro")) -> DivWinners

set.seed(123)
training.samples <- DivWinners$DivWin %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- DivWinners[training.samples, ]
test.data <- DivWinners[-training.samples, ]
```

Next the training data will be used to fit a logistic regression model using the ‘glmnet’ command. Residual deviance against number of predictors will then be plotted.

```
divwin <- as.vector(train.data$DivWin)
Divpredict<-model.matrix(~.-1,train.data[,-c(6)])
divwinfit<-glmnet(Divpredict, divwin, family = "binomial")
plot(divwinfit,xvar="dev")
```



We will then investigate How many nonzero model coefficients are needed to explain 50% and 60% of the deviance.

```
divwinfit
```

```
Call:  glmnet(x = Divpredict, y = divwin, family = "binomial")

   Df  %Dev   Lambda
1   0  0.00 0.244500
2   1  6.30 0.222800
3   1 11.67 0.203000
4   1 16.31 0.184900
5   2 20.45 0.168500
6   2 24.13 0.153500
7   2 27.40 0.139900
8   2 30.31 0.127500
9   2 32.92 0.116100
10  2 35.27 0.105800
11  2 37.38 0.096430
12  2 39.29 0.087860
13  2 41.02 0.080060
14  2 42.58 0.072940
15  2 44.00 0.066460
16  2 45.27 0.060560
17  2 46.43 0.055180
18  2 47.47 0.050280
19  2 48.41 0.045810
20  2 49.26 0.041740
21  2 50.02 0.038030
22  2 50.70 0.034650
23  3 51.31 0.031580
24  3 51.91 0.028770
25  3 52.45 0.026220
26  3 52.92 0.023890
27  3 53.34 0.021760
28  3 53.71 0.019830
29  3 54.04 0.018070
30  3 54.33 0.016460
31  3 54.58 0.015000
32  4 54.80 0.013670
33  6 55.04 0.012450
34  7 55.29 0.011350
35  8 55.74 0.010340
36  9 56.15 0.009421
37  9 56.51 0.008584
38 12 56.84 0.007822
39 14 57.18 0.007127
40 14 57.52 0.006494
41 15 57.81 0.005917
42 15 58.10 0.005391
43 16 58.36 0.004912
44 18 58.61 0.004476
45 19 58.84 0.004078
46 19 59.04 0.003716
47 19 59.21 0.003386
48 20 59.36 0.003085
49 21 59.49 0.002811
50 22 59.63 0.002561
51 23 59.76 0.002334
```

```
52 25 59.93 0.002126
53 26 60.13 0.001937
54 27 60.29 0.001765
55 26 60.44 0.001609
56 27 60.61 0.001466
57 27 60.80 0.001335
58 28 61.04 0.001217
59 28 61.28 0.001109
60 28 61.48 0.001010
61 27 61.62 0.000920
62 29 61.79 0.000839
63 29 61.98 0.000764
64 30 62.21 0.000696
65 31 62.42 0.000634
66 31 62.60 0.000578
67 31 62.76 0.000527
68 31 62.89 0.000480
69 31 63.00 0.000437
70 32 63.10 0.000398
71 32 63.18 0.000363
72 32 63.25 0.000331
73 32 63.30 0.000301
74 32 63.35 0.000275
75 33 63.40 0.000250
76 33 63.43 0.000228
77 34 63.46 0.000208
78 34 63.49 0.000189
79 34 63.51 0.000172
80 34 63.53 0.000157
81 34 63.55 0.000143
82 34 63.56 0.000130
83 34 63.57 0.000119
84 35 63.58 0.000108
85 35 63.59 0.000099
86 35 63.60 0.000090
87 35 63.61 0.000082
88 36 63.62 0.000075
89 36 63.65 0.000068
90 35 63.66 0.000062
91 35 63.67 0.000056
92 35 63.68 0.000051
93 36 63.69 0.000047
94 37 63.73 0.000043
95 37 63.78 0.000039
96 37 63.81 0.000035
97 37 63.81 0.000032
```

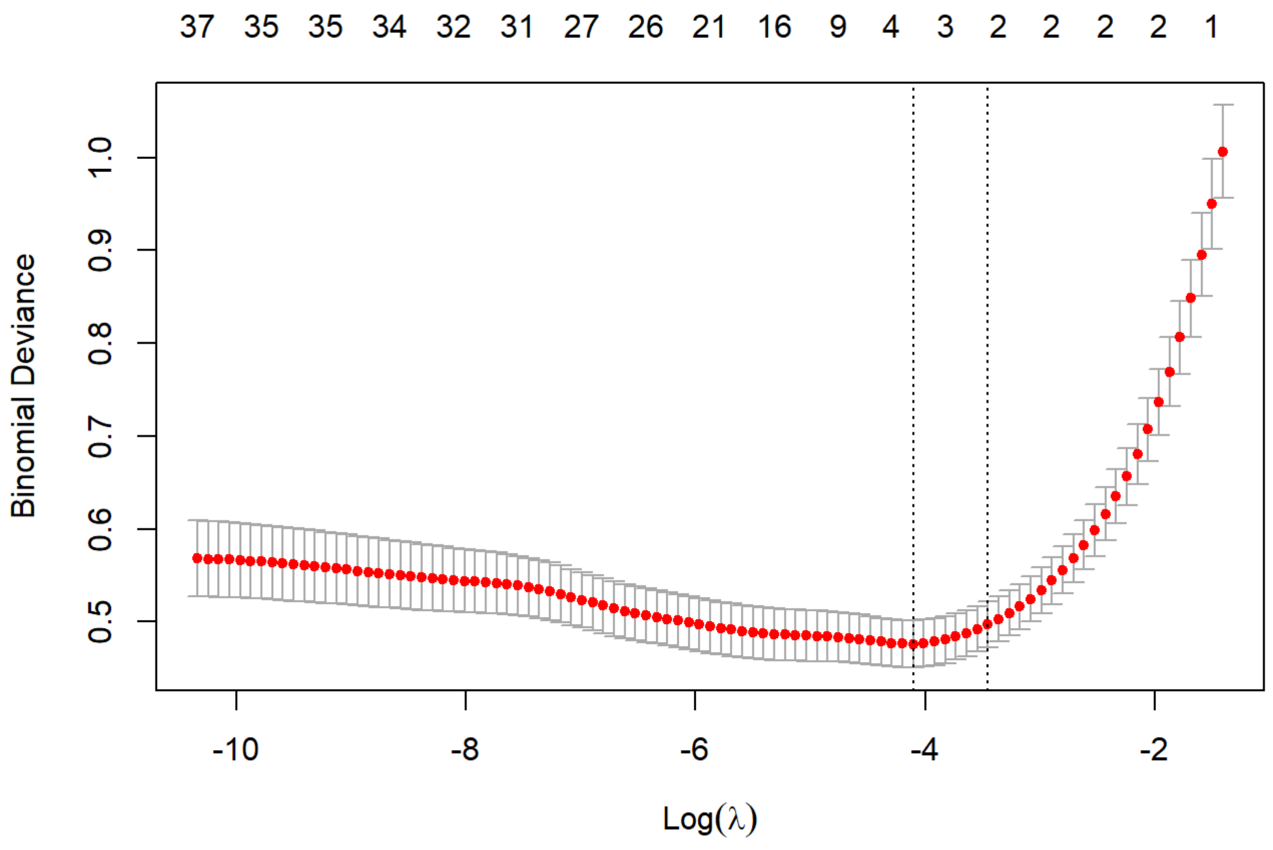
- To explain 50% of the deviance, only 2 non-zero model coefficients are needed. To explain 60% of the deviance, 26 non-zero model coefficients are needed. The coefficients needed to explain 50% of the data are W and L. The coefficients necessary to explain 60% of the data are yearID, Ghome, W, L, AB, H, X2B, X3B, HR, BB, SO, SB, CS, HBP, SF, RA, CG, SV, HA, HRA, BBA, SOA, DP, FP, attendance, PPF and rostersize.

```
div50<-coef(divwinfit, s=0.038030)
div50@Dimnames[[1]][1+div50@i]

div60<-coef(divwinfit, s=0.001937)
div60@Dimnames[[1]][1+div60@i]
```

```
[1] "(Intercept)" "W"           "L"
 [1] "(Intercept)" "yearID"      "Ghome"       "W"           "L"
 [6] "AB"          "H"           "X2B"         "X3B"         "HR"
[11] "BB"          "SO"          "SB"          "CS"          "HBP"
[16] "SF"          "RA"          "CG"          "SV"          "HA"
[21] "HRA"         "BBA"         "SOA"         "DP"          "FP"
[26] "attendance"  "PPF"         "rostersize"
```

Cross-validation will then be used to choose a moderately conservative model.

```
set.seed(123)
divcv<-cv.glmnet(Divpredict,divwin, family = "binomial")
plot(divcv)
```



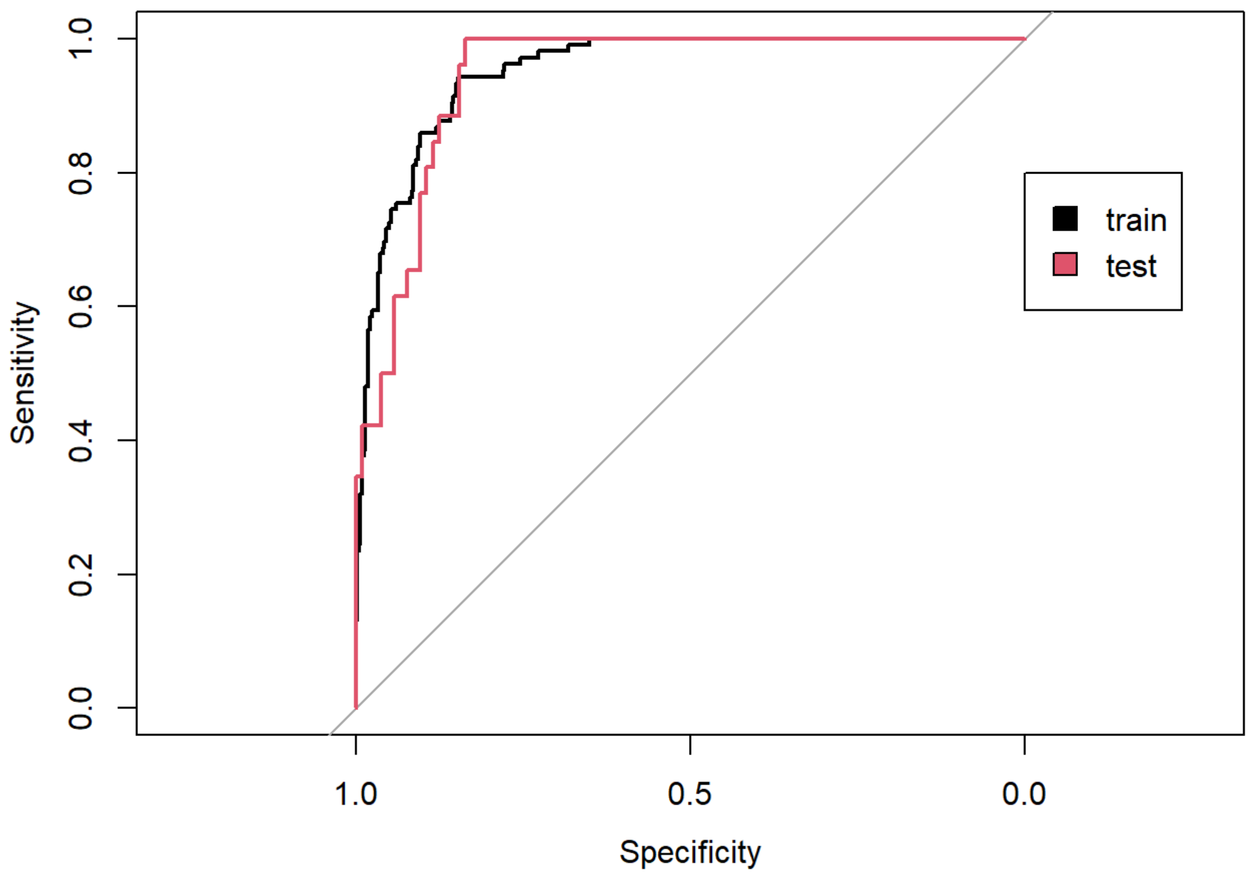
```
div_sd<-coef(divwinfit,s=divcv$lambda.1se)
div_sd@Dimnames[[1]][1+div_sd@i]
```

```
[1] "(Intercept)" "W"           "L"           "attendance"
```

- The variables that will be chosen are W (number of wins), L (number of losses) and attendance.

The model will then be fit on the training data, then predict on the testing data.

```
divmodel <- glm(as.factor(DivWin) ~ W + L + attendance, data = train.data, family = "binomia
l")

predtrain<-predict(divmodel,type="response")
predtest<-predict(divmodel,newdata=test.data,type="response")

roctrain<-roc(response=train.data$DivWin,predictor=predtrain,plot=TRUE,auc=TRUE)
roctest<-roc(response=test.data$DivWin,predictor=predtest,plot=TRUE,auc=TRUE,add=TRUE,col=2)
legend(0,0.8,legend=c("train","test"),fill=1:2)
```



- There is not too much space between the ROC curve of the training data and the testing data, indicating there’s not much overfitting. This is also a good model because an estimated good compromise of sensitivity and specificity is about 0.9 and 0.8 respectively, and both ROC curves reflect this.

Youden’s index will be found for the training data and we will calculate confusion matrices at this cutoff for both training and testing data.

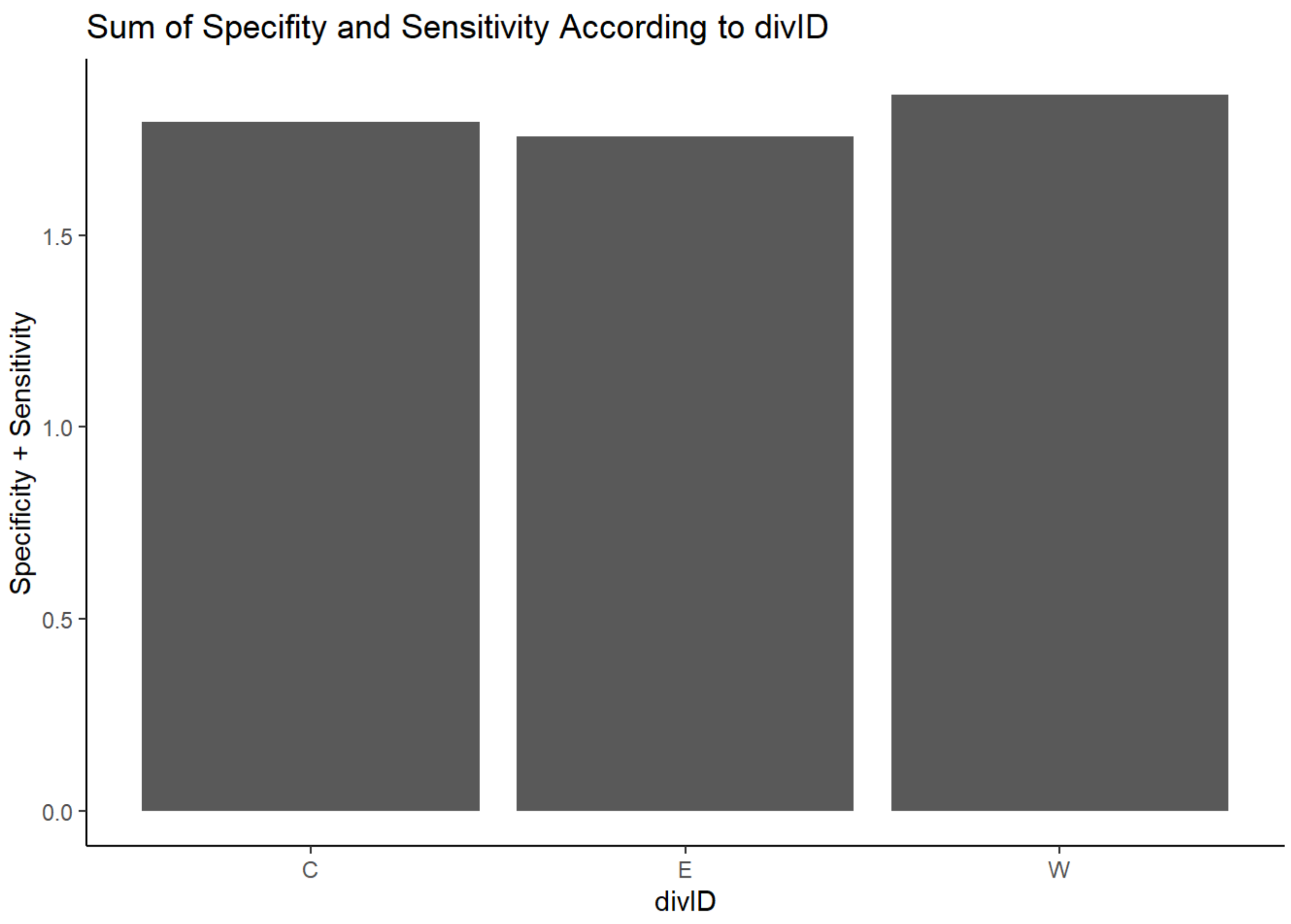
```
youdendiv<-coords(roctrain,"b",best.method="youden",transpose=TRUE)
youdendiv

youdendiv[2]+youdendiv[3]

#youdendivtest<-coords(roctest,x=0.1836071,transpose=TRUE)
#youdendivtest
#youdendivtest[2]+youdendivtest[3]

train.data$preddiv<-ifelse(predict(divmodel,newdata=train.data, type="response")>= 0.1836071,
"Y","N")
table(train.data$preddiv,as.factor(train.data$DivWin))

test.data$preddiv<-ifelse(predict(divmodel,newdata=test.data, type="response")>= 0.1836071,
"Y","N")
table(test.data$preddiv,as.factor(test.data$DivWin))
```

```
  threshold specificity sensitivity
  0.1836071   0.8468900   0.9433962
specificity
   1.790286

      N   Y
  N 354   6
  Y  64 100

     N  Y
  N 87  1
  Y 17 25
```

- Youden’s index for the training data is 0.1836. For the prediction on the test data, the model quality can be summarised using false negative and false positive rates. A false positive rate is given as False positive rate = FP/(TN+FP) where FP means false positive and TN means true negative. The false positive rate would be therefore 1/(87+1) = 0.01136. A false negative rate is given as False negative rate = FN/(FN+TP) where FN means false negative and TP means true positive. The false negative rate would be therefore 17/(17+25) = 0.4048. Smaller false positive and false negative rates imply that the model is of good quality. In this case, the small false positive rate means that the model is good at predicting teams which lose, but the higher false negative rate means that the model is not good at predicting teams which will win. The model needs to be improved to reduce the false positive rate, as knowing which teams will win the division is more beneficial.

Next we will calculate the sensitivity+specificity on the testing data as a function of divID and plot as a barchart.

```
Teamdata %>%
  select(! c("lgID", "Rank", "franchID", "WCWin", "LgWin", "WSWin", "teamID", "name", "park",
"teamIDBR", "teamIDlahman45", "teamIDretro")) -> DivWinners2

set.seed(123)

training.samples2 <- DivWinners2$DivWin %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data2  <- DivWinners2[training.samples2, ]
test.data2 <- DivWinners2[-training.samples2, ]

test.data2 %>%
  filter(divID == "E") -> test.data2E

test.data2 %>%
  filter(divID == "C") -> test.data2C

test.data2 %>%
  filter(divID == "W") -> test.data2W



predtest2E<-predict(divmodel,newdata=test.data2E,type="response")
roctest2E<-roc(response=test.data2E$DivWin,predictor=predtest2E,plot=FALSE)

predtest2C<-predict(divmodel,newdata=test.data2C,type="response")
roctest2C<-roc(response=test.data2C$DivWin,predictor=predtest2C,plot=FALSE)

predtest2W<-predict(divmodel,newdata=test.data2W,type="response")
roctest2W<-roc(response=test.data2W$DivWin,predictor=predtest2W,plot=FALSE)

youdendivtest2E<-coords(roctest2E,x=0.1836071,transpose=TRUE)
youdendivtest2C<-coords(roctest2C,x=0.1836071,transpose=TRUE)
youdendivtest2W<-coords(roctest2W,x=0.1836071,transpose=TRUE)

youdendivtest2E[2]+youdendivtest2E[3] -> sumE
youdendivtest2C[2]+youdendivtest2C[3] -> sumC
youdendivtest2W[2]+youdendivtest2W[3] -> sumW

ggplot(mapping = aes(x = c("C", "E", "W"), y = c(sumC, sumE, sumW))) +
  geom_col() +
  labs(
    title = "Sum of Specifity and Sensitivity According to divID",
    x = "divID",
    y = "Specificity + Sensitivity"
  ) +
  theme_classic()
```



- The prediction is almost equally good for all divisions, with division C, E and W having sensitivity and specificity sums of 1.79, 1.76 and 1.87. Division W had the best prediction since it had the highest sum of 1.87.