Coursework for Statistical Methods and Data Analysis

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")
```

1. Datesets

Q1a. 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 information, along with the variable finalGame; ii. replacing birthCountry is by bornUSA, a logical variable indicating if the player was born in the USA.

```
Peopledata<-People %>%
  select("playerID", "birthYear", "nameFirst", "nameLast", "weight", "height", "bats", "throws", "debut", "birthC
ountry") %>%
  mutate(bornUSA = birthCountry %in% "USA") %>%
  select(!birthCountry)

Peopledata<-as.tibble(Peopledata)
```

Q1b. 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" and "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.

```
#Creation of Batting data
Battingdata <- Batting %>%
  filter(yearID == 1985 | yearID == 2015) %>%
  select(!c(G,teamID, lgID)) %>%
  mutate(batav = case_when(H == 0 ~ 0, H > 0 ~  H / AB))

Battingdata %>% sapply(function(x) sum(is.na(x)) ) #No missing values in Battingdata, code used after filter() fu
nction
Battingdata <- as.tibble(Battingdata)
```

```
playerID   yearID    stint       AB        R        H      X2B      X3B
       0        0        0        0        0        0        0        0
      HR      RBI       SB       CS       BB       SO      IBB      HBP
       0        0        0        0        0        0        0        0
      SH       SF     GIDP    batav
       0        0        0        0
```

```
#Creation of Fielding data
Fieldingdata <- Fielding %>%
  filter(yearID == 1985 | yearID == 2015) %>%
  select(!c(PB, WP, SB, CS, ZR)) %>%
  select(!c( teamID, lgID))

Fieldingdata  %>% sapply(function(x) sum(is.na(x)) ) #Found columns (PB, WP, SB, CS, ZR) >25 missing cases, code
used after filter() function
Fieldingdata<- as.tibble(Fieldingdata)
```

```
playerID   yearID    stint      POS        G       GS  InnOuts       PO
       0        0        0        0        0        0        0        0
       A        E       DP
       0        0        0
```

Q1c. 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 AllstarFull dataset, iv. creating a new variable ‘age’ equal to each player’s age in the relevant year, v. dropping unused levels of any categorical variable.

```
Playerdata <- Salaries %>%
  filter(yearID == 1985 | yearID == 2015) %>%
  inner_join(Fieldingdata,
             by = c('yearID', 'playerID'),
             keep = FALSE) %>%

  mutate(allstar = playerID %in% AllstarFull$playerID) %>%
  inner_join(Battingdata,
             by = c('yearID', 'playerID', 'stint'),
             keep = FALSE) %>%
  inner_join(Peopledata,
             by = c("playerID" = "playerID"),
             keep = FALSE) %>%
  mutate(age = yearID - birthYear) %>%
  drop_na %>%
  droplevels()

levels(Playerdata$bats) #Checked there were 3, maintained after dropping
levels(Playerdata$throws) #Checked there were 3 levels and now 2
Playerdata <- as.tibble(Playerdata)
```

```
[1] "B" "L" "R"
[1] "L" "R"
```

Q1d. Create 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.

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

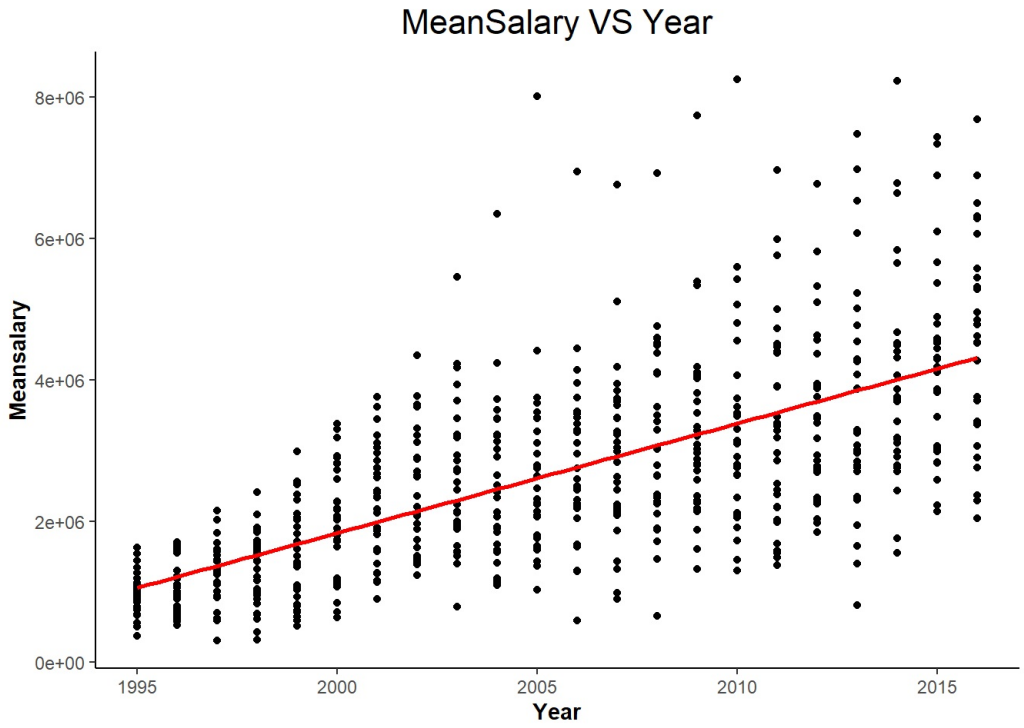
e. Create a dataset ‘Teamdata’ by taking the data from the Teams dataset for the years 1984 to 2016, inclusive and adding to that data the variables in TeamSalaries. Drop any incomplete cases from the dataset.

```
Teamsdata <- Teams %>%
  filter(yearID >= 1984, yearID <= 2016) %>%
  inner_join(TeamSalaries, by = c('yearID', 'teamID'),
             keep = FALSE) %>%
  drop_na()
```

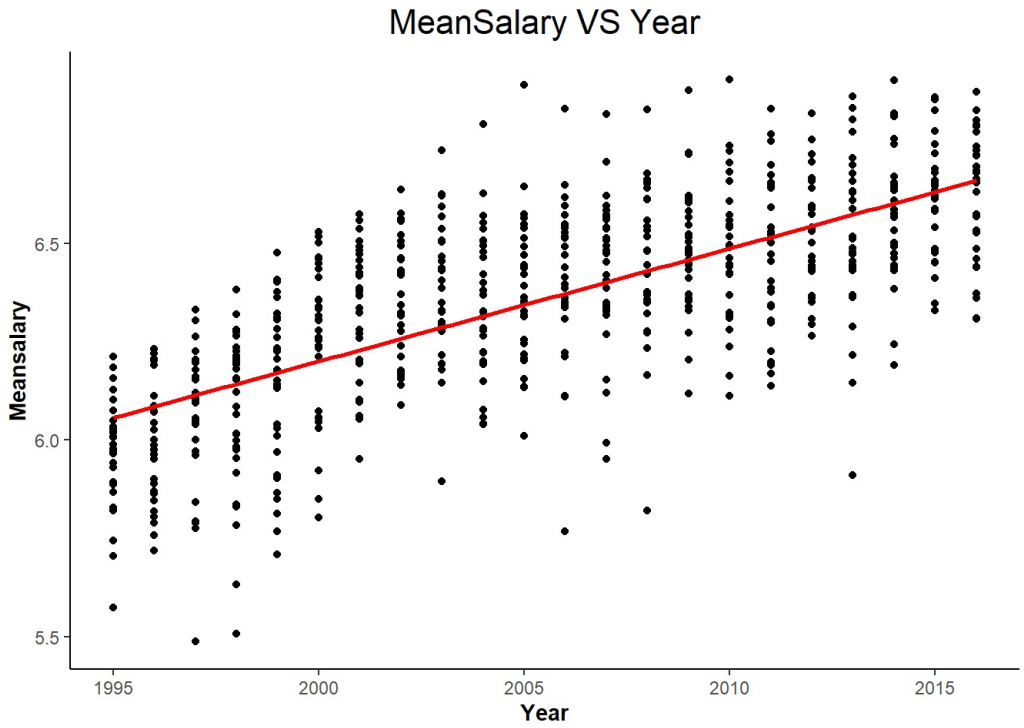
2. Simple Linear Regression

Q2a. [2 + 2 points] 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. Give two reasons why a linear model is more appropriate for log base 10 mean salaries than for raw mean salaries.

```
Teamsdata %>%
  ggplot(mapping = aes(x = yearID, y = meansalary)) +
  geom_point()+
  labs(x = "Year", y = "Meansalary") +
  ggtitle("MeanSalary VS Year") +
  geom_smooth(method = "lm", se = FALSE,colour="red") +
  theme_classic() +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```




```
Team_new <- Teamsdata %>%
  mutate(log10_meansalary = log10(meansalary))

Team_new %>%
  mutate(log10_meansalary = log10(meansalary)) %>%
  ggplot(mapping = aes(x = yearID, y = log10_meansalary)) +
  geom_point()+
  labs(x = "Year", y = "Meansalary") +
  ggtitle("MeanSalary VS Year") +
  geom_smooth(method = "lm", se = FALSE,colour="red") +
  theme_classic() +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```



Reasons why second graph is better:

1. After comparing the 2 graphs (MeanSalary VS Year & LogMeanSalary VS Year), we can see that the spread above and below the regression line on the LogMeanSalary graph indicating a more homoscedascedity graph.

2. there is also less dispersion display when compared. Noted that we can see increase in dispersion on the MeanSalary graph when Year increases.

Q2b. [1 + 3 points] Fit a model of log_{10}(meansalary) as a function of yearID. Write the form of the model and explain what the Multiple R-Squared tells us.

```
lm_teamsal <- lm(log10_meansalary  ~ yearID ,data = Team_new )
summary(lm_teamsal)
```

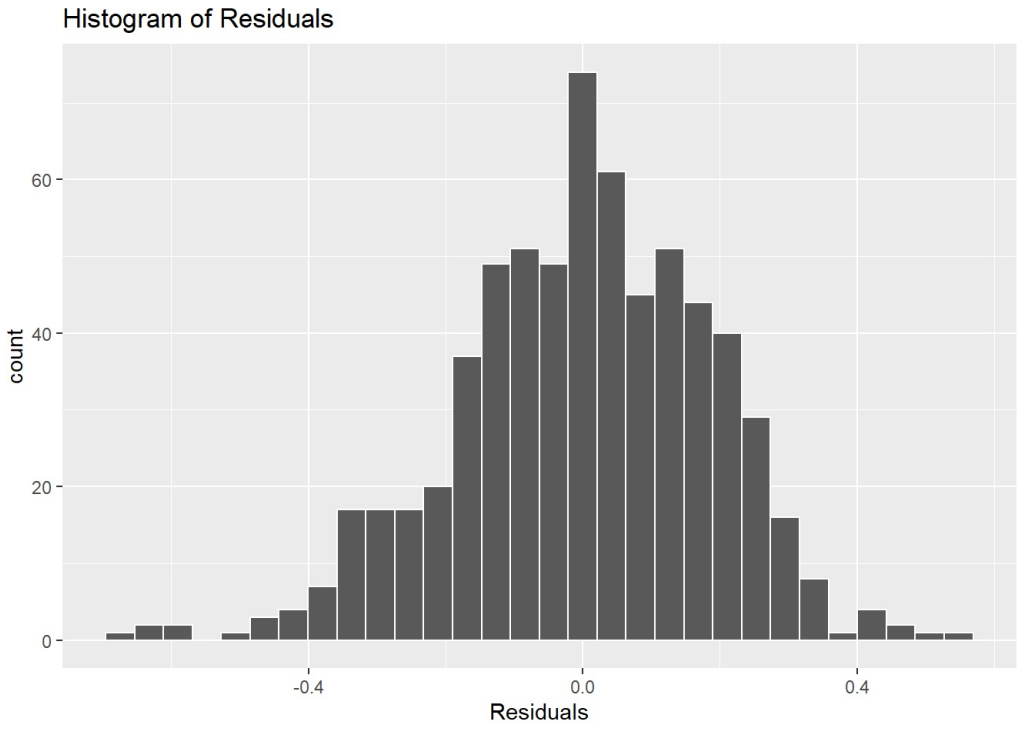
```
Call:
lm(formula = log10_meansalary ~ yearID, data = Team_new)

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
```

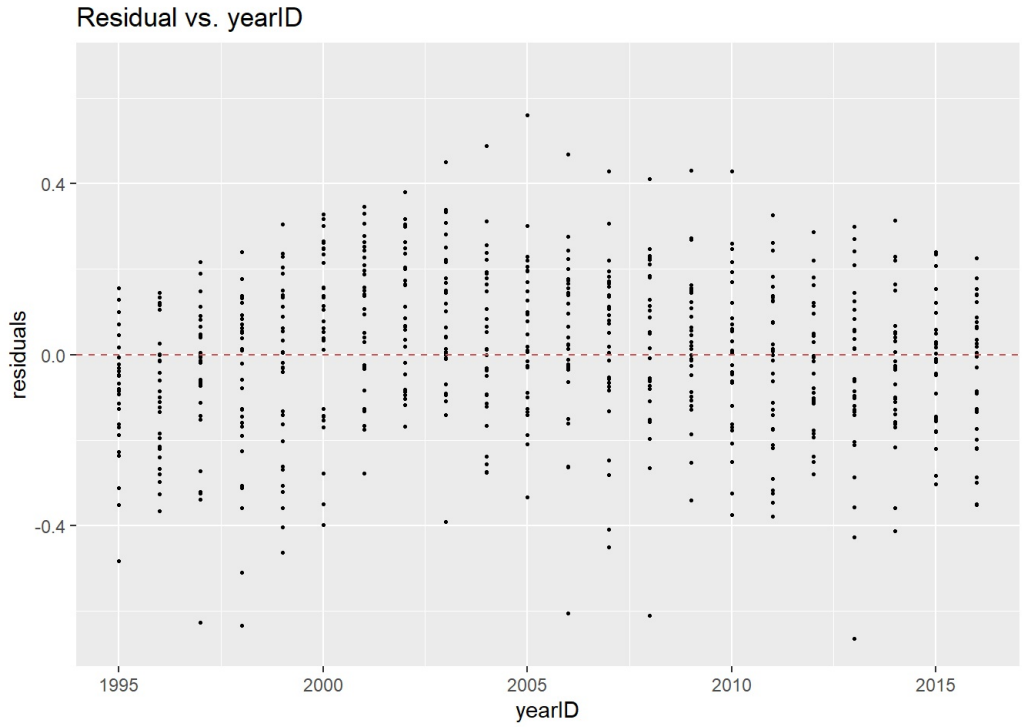
$$\log10(\text{MeanSalary}) \sim N(-51.222416 + 0.028711 \times \text{Year}, 0.1858)$$

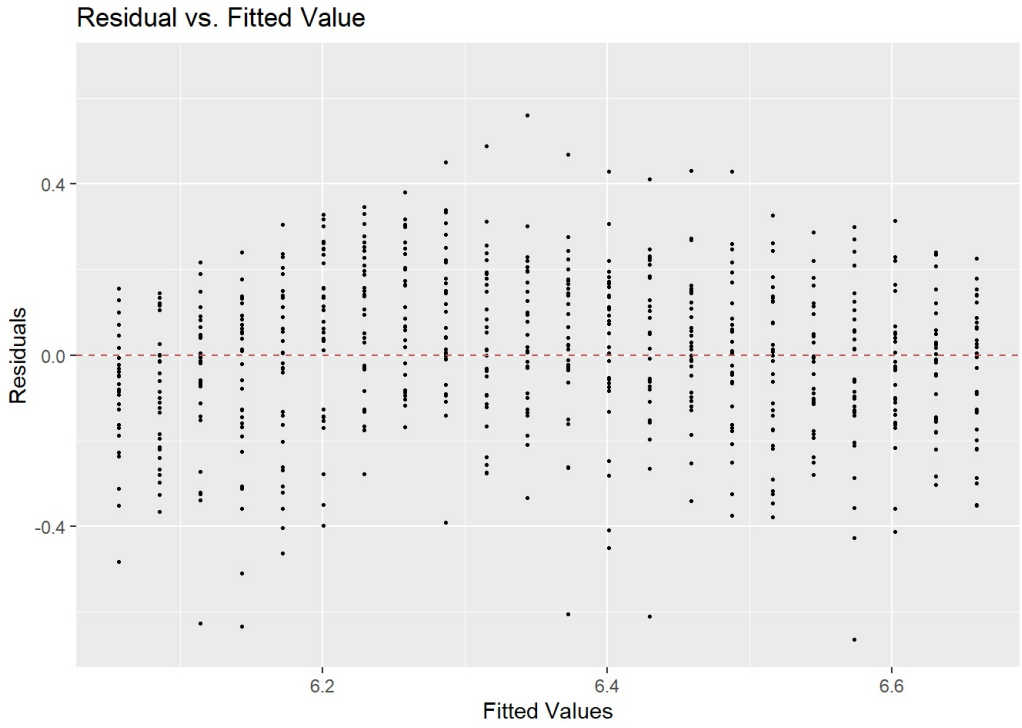
Multiple R-squared = 0.4878 This represents that estimate of 49% of the variances of log MeanSalary are explained by the yearID.

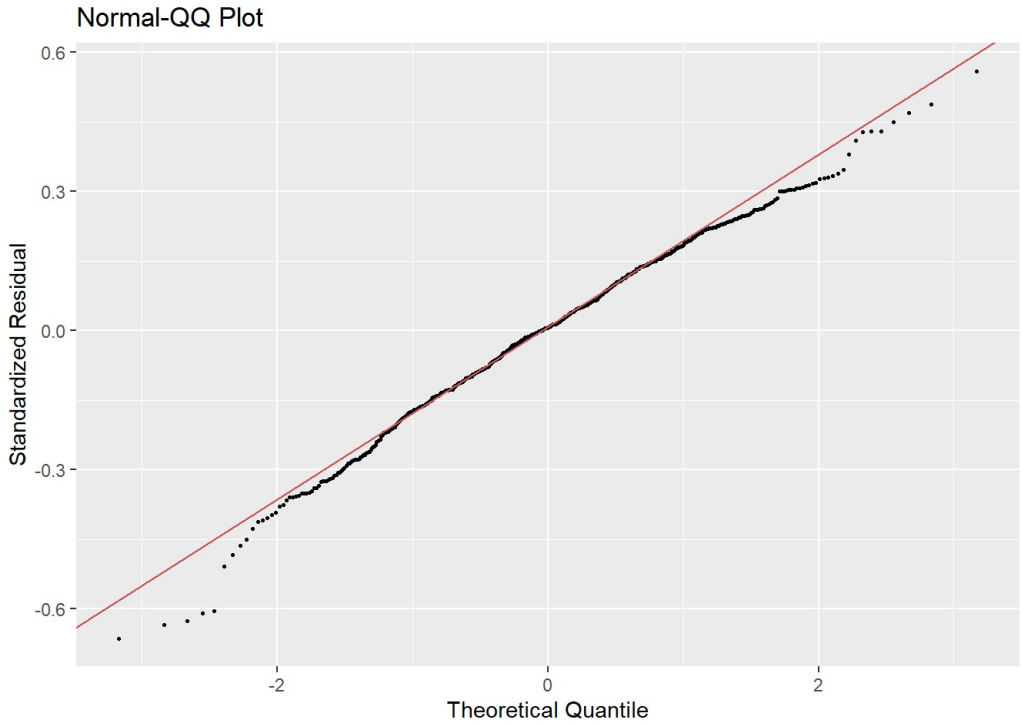
Q2c. [1 + 8 points] State and evaluate the four assumptions of linear models for this data.

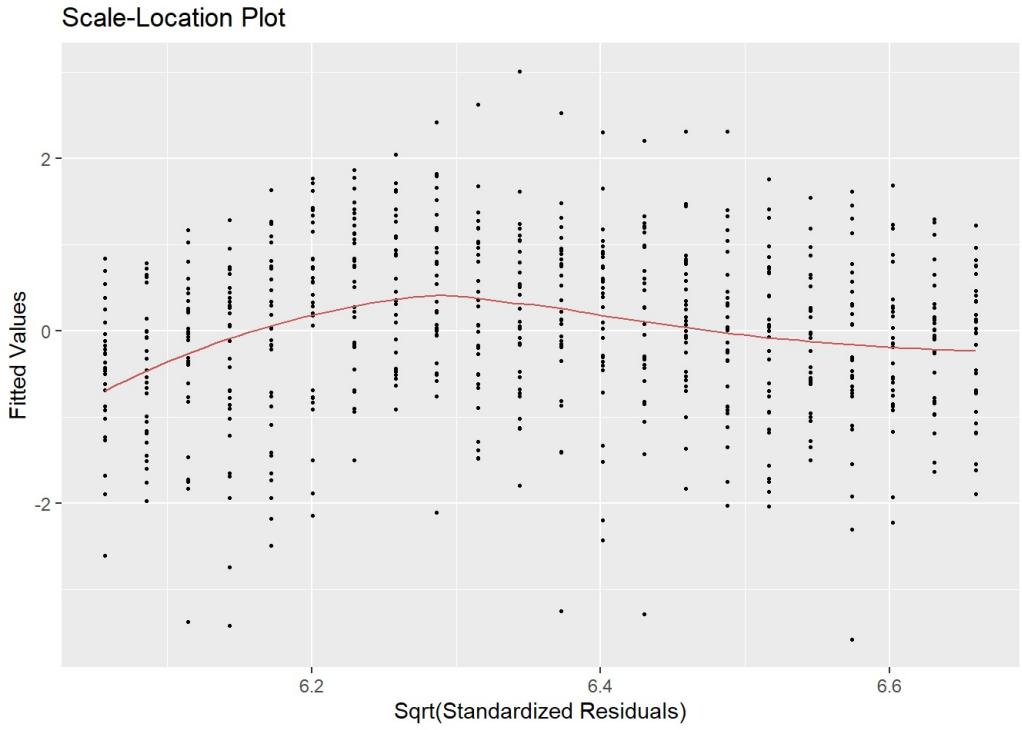
```
lm_teamsal %>%
 gg_diagnose(max.per.page = 1)
```

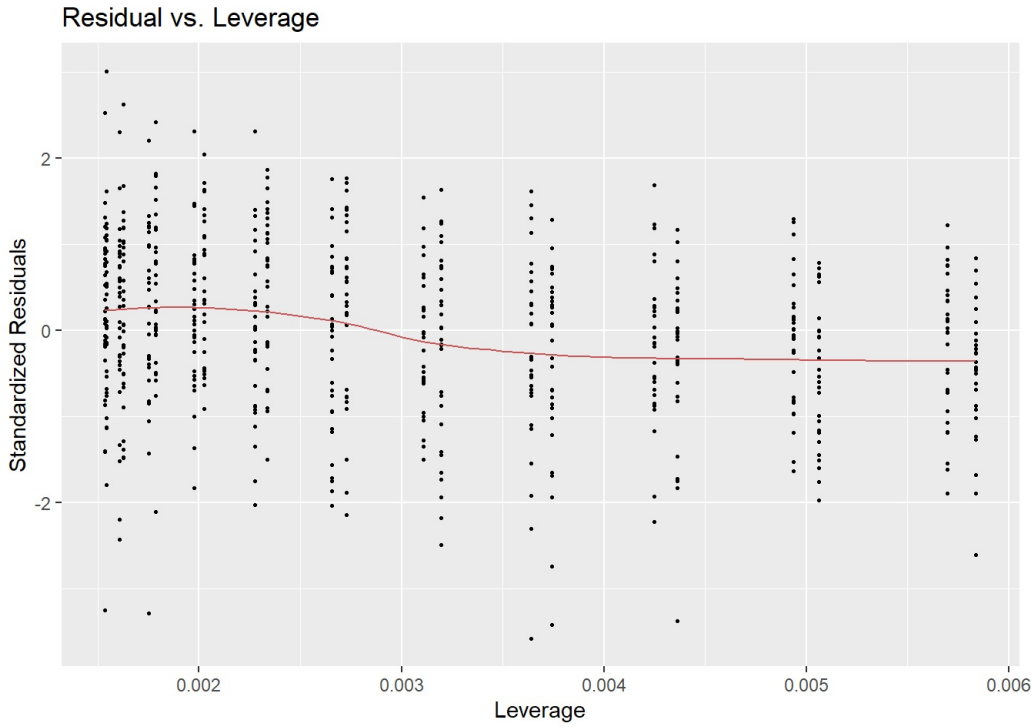


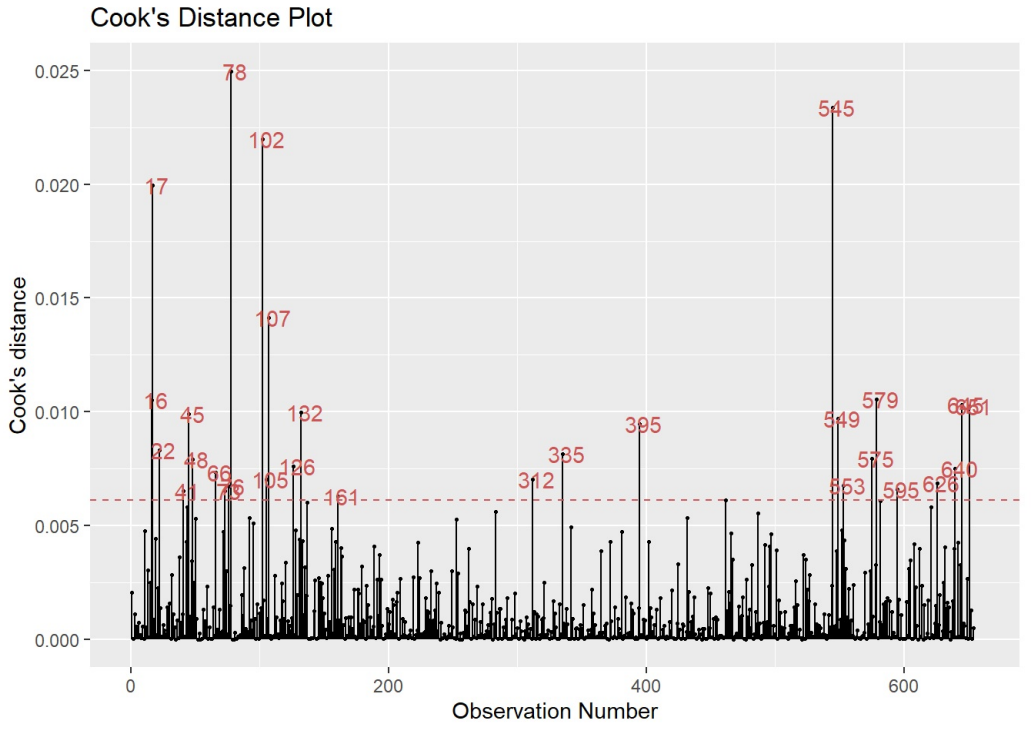

Residual vs. yearID
residuals
0.4
0.0
-0.4
1995
2000
2005
2010
2015
yearID


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
2
0
-2
6.2
6.4
6.6
Sqrt(Standardized Residuals)





Four assumptions for linear regression model:

-Normality Lets first consider the histogram and QQ plot for the residuals. We can see the histogram is displaying a bell-shape where there are more counts at the center and less counts on both sides evenly, it looks roughly Gaussian distributed. QQ plot also reinforces this finding, the scatter values lie along the line of theoretical distribution which again, supports the normality assumption where the data is normally distributed.

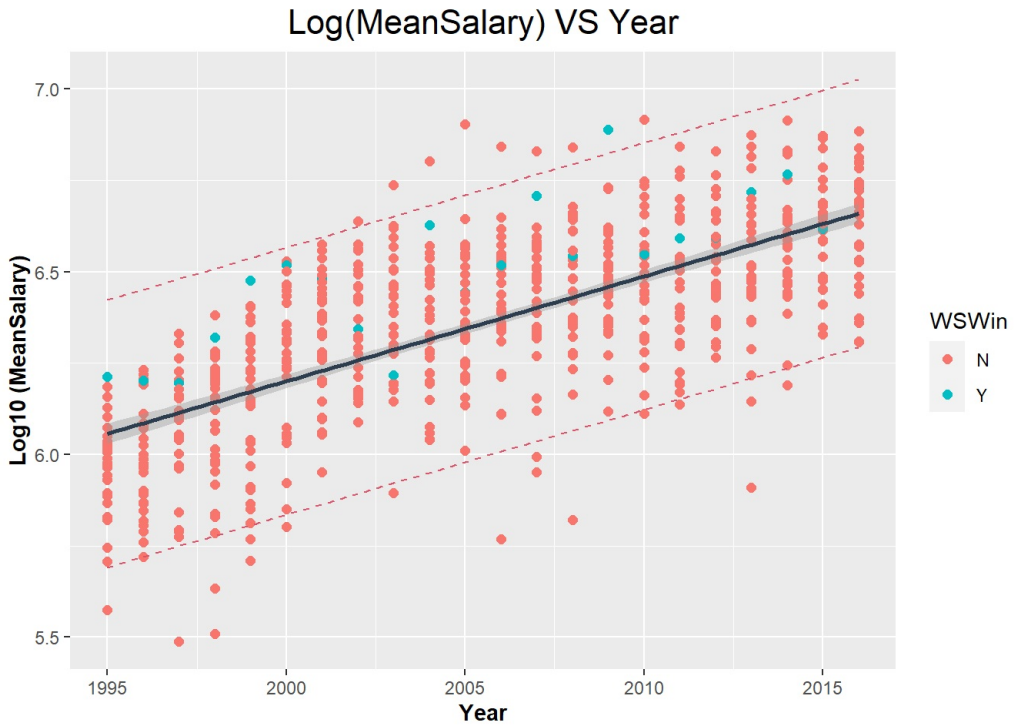
-Linearity The scatter plot for log10(MeanSalary) versus year was roughly linear. The residuals versus fitted value and residuals versus year graphs also showed a linear pattern where the points lay around the 0.0 residuals value.

-Homoscedascitiy The scatter of residuals versus year and residuals versus fitted, are roughly (although there could be a slight trend where the spread increase from left to right) the same width across the plot and which doesn't show any indication of trend.

-Independence Points does appears to show some indications of under autocorrelation when we look at the graphs of residuals versus year. This means that standard errors in the output we cannot reply on but further tests such as Durbin-Wastson test and Breusch-Godfrey test can be used to confirm this.

Q2d. [3 + 1 points] Plot confidence and prediction bands for this model. Colour the points according to who won the World Series each year. Comment on what you find.

```
#Confidence and prediction band
predsal <- predict(lm_teamsal, interval="prediction") %>%
  as.data.frame()
myTeams_new <- cbind(Team_new, predsal)

ggplot(myTeams_new ,aes(yearID,log10_meansalary, colour = 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 = "Log(MeanSalary) VS Year",
    x = "Year",
    y = "Log10 (MeanSalary)"
  ) +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```



World Series winner teams usually have a higher meansalry as they lay above the confident band.

Q2e. [1 + 1 points] Investigate the points that appear above the top prediction band. What team or teams do they relate to?

```
myTeams_new %>%
  filter(log10_meansalary > upr) %>%
  count(teamID)
```

teamID <fct>
NYA

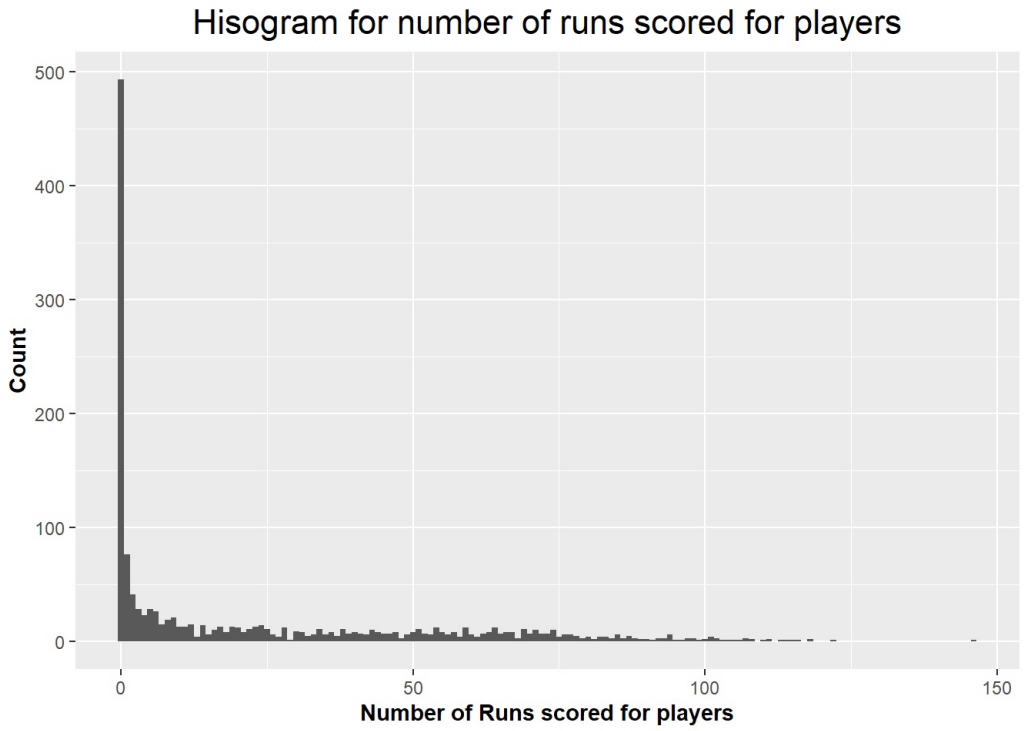
1 row | 1-1 of 2 columns

Team NYA (9 counts), appears to have a log10(meansalary) higher than the prediction band in the year between 1984 to 2016.

3. Multiple regression for Count Data

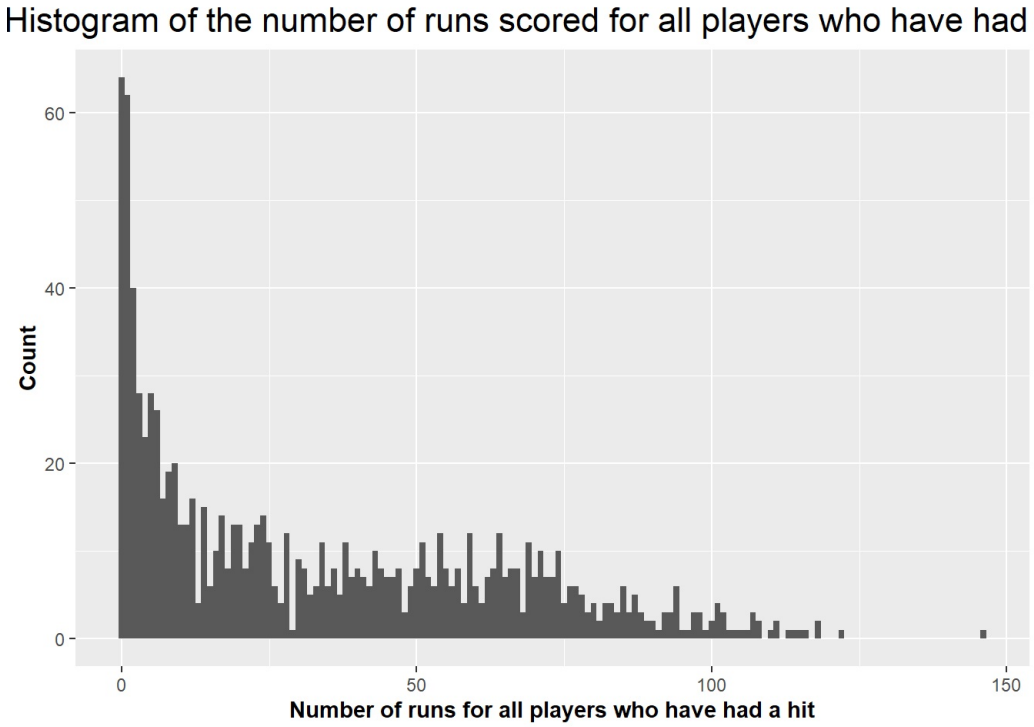
Q3a. [2 + 2 points] Create a histogram of the number of runs scored for players in the Playerdata dataset so each bar is a single value (0,1,2 runs, etc). Next create a histogram of the number of runs for all players who have had a hit. Give a domain-based and a data-based reason why it is more reasonable to create a Poisson data for the second set than the first.

```
#Histogram of the number of runs scored for players
Playerdata %>%
  select(yearID, playerID, R) %>%
  distinct()%>%
   ggplot(aes(R)) +
  geom_histogram(binwidth = 1) +
  labs(
     title = "Hisogram for number of runs scored for players",
    x = "Number of Runs scored for players",
    y = "Count"
  ) +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```




```
#Noted that players will be playing different roles in the a game but end score of a match or matches will be the same. This creates extra counts and hence is worth removing the duplication as to answer the question 3a
```

```
#Histogram of the number of runs for all players who have had a hit
Playerdata %>%
  select(yearID, playerID, R, H) %>%
  distinct()%>%
  filter(H >= 1)%>%
   ggplot(aes(R)) +
  geom_histogram(binwidth = 1)  +
  labs(
     title = "Histogram of the number of runs scored for all players who have had a hit",
    x = "Number of runs for all players who have had a hit",
    y = "Count"
  ) +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```




```
#Noted that players will be playing different roles in the a game but end score of a match or matches will be the same. This creates extra counts and hence is worth removing the duplication as to answer the question 3a
```

Players will only run after they had hit a ball. Therefore, there is no point including players who do not hit the ball (i.e. H = 0) in a histogram where we are count number of run scored of the players when some of them will never score. Including players with zero hit could result in is zero inflation in the data, hence using poisson model for the first histogram will not be appropriate.

Q3b. [3 + 0 points] Create a new dataset, OnBase of all players who have had at least one hit. Transform yearID to a factor. 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.

```
OnBase <- Playerdata %>%
  filter(H >= 1) %>%
  mutate(yearID = as.factor(yearID))

#As we are trying to produce a model seeing if position could acts as a predictor for run score, there is no need the distinct process as mentioned in 3a.

glm1 <- glm(R ~ H + yearID + POS + height + age ,data = OnBase,family="poisson") #If a POS (only categorical variable in here) is accumulating in zero, it would badly impact model

glm1
```

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

Coefficients:
(Intercept)            H   yearID2015        POS2B        POS3B         POSC
   2.494066     0.012854     0.022306    -0.011519     0.005319    -0.062970
      POSOF         POSP         POSSS       height          age
   0.063224    -1.170625    -0.011226    -0.003584     0.004693

Degrees of Freedom: 1481 Total (i.e. Null);  1471 Residual
Null Deviance:      38810
Residual Deviance: 5696     AIC: 12620
```

Q3c. [2 + 4 points] Find the p-value for each of the predictor variables in this model using a Likelihood Ratio Test. What hypothesis does each p-value test, and what mathematically does a p-value tell you about a variable? Use this definition to say what is meant by the p-value associated to POS and to the p-value associated to height.

```
Anova(glm1)
```

H
yearID

POS
height
age

5 rows | 1-1 of 4 columns

P values for: Number of hit scored in game (H) = 2.2e-16 year (yearID) = 0.01625 Position (POS) = 2.2e-16 Height (height) = 0.10994 Age (age) = 9.705e-05

In statistic, P values is used in null hypothesis significance testing. P-value below 0.05 indicates the parameter is statistically significant and in here, will act as an important predictor of the response variable.

From the above Deviance Table from glm1, we can notice that 4 variables, H, yearID, POS and age are below 0.05 or in fact, 3 of them, i.e. H, POS, age, are even below 0.001. This indicates the results were highly significant and are important predictors.

P-value of POS suggests that it is an important predictor when comparing the model with variables POS + height + age + yearID to another model with variables height + age + yearID, . Similar comment can be made for p-value associated to height but in opposite. un-statistically significant p-value of height suggests that it is not an important predictor when comparing the model with variables POS + height + age + yearID to another model with variables POS + age + yearID.

3d. [1 + 8 points] State the assumptions of Poisson models and check these where possible.

-The response variable has to be a count. The response variable of number of runs for player who have had a hit is a count where only whole number is counted in (i.e. 1,2,3 etc) and not a fraction. We used “Histogram of the number of runs for all players who have had a hit” to display that each column represent a whole number.

```
class(OnBase$H) #Check if it is integer
```

```
[1] "integer"
```

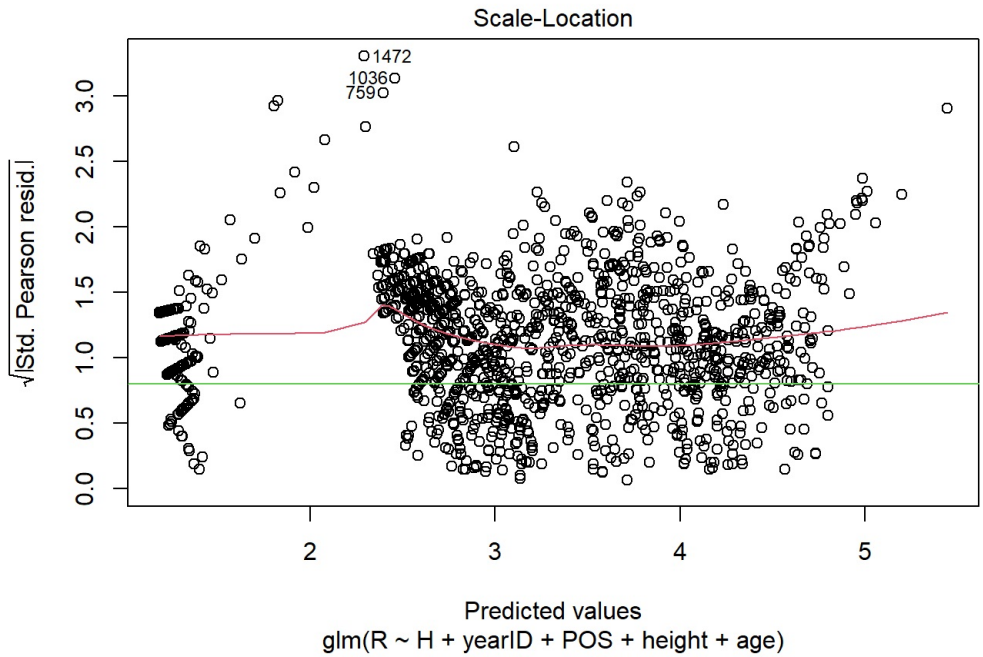
-The expected value of response variable x has to be greater or equal to zero, so is E(x) of course. Number of runs for player who have had a hit will not have be negative, as showed in the “Histogram of the number of runs for all players who have had a hit” shows that no column is counting negative number of 1 of below zero.

```
min(OnBase$H) #Check if greater than 0, i.e. positive number
```

```
[1] 1
```

-Assumption of variance = mean is reasonable for this dataset. We can use the plot of the absolute value of residuals versus predicted means, which should look flat, and hover around 0.8(green line).

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




```
integer(0)
```

The data seems to be clustering into 2 groups right and left. The red line is not flat, and it is above the 0.8. This suggests that a slight overdispersion in the data that increases linearly as the prediction increases. This situation is very common when we have not accounted for all of the important predictors in the model and in this case, it is not so great.

We can double check the dispersion of the model:

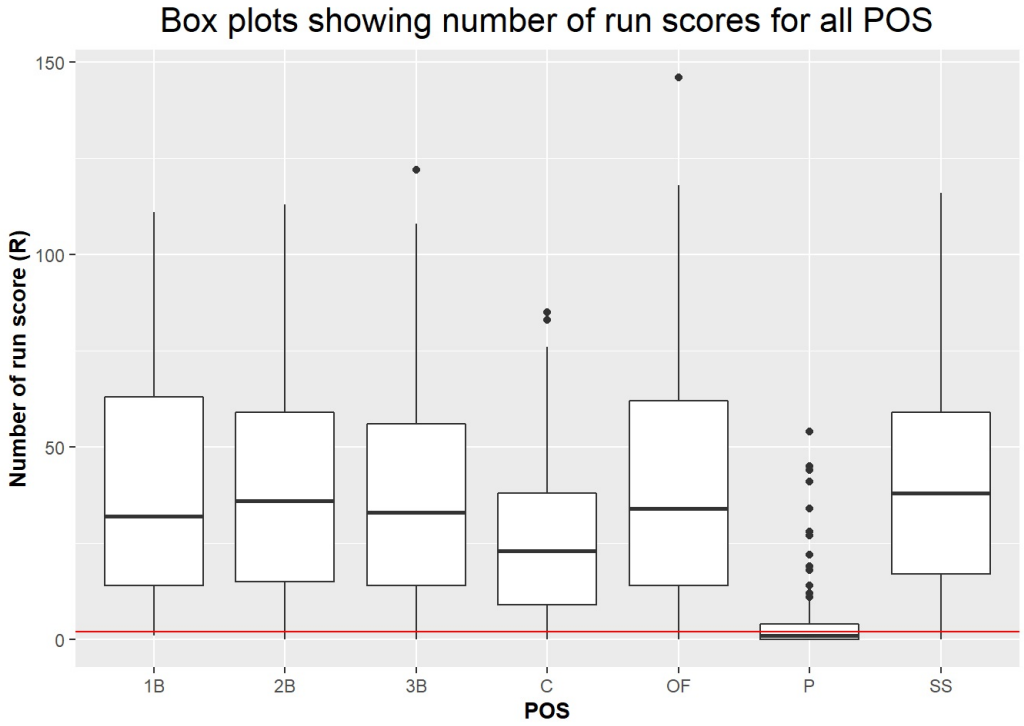
```
overdisp_fun <- function(model) {
rdf <- df.residual(model)
rp <- residuals(model,type="pearson")
Pearson.chisq <- sum(rp^2)
prat <- Pearson.chisq/rdf
pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=FALSE)
c(chisq=Pearson.chisq,ratio=prat,rdf=rdf,p=pval)
}

overdisp_fun(glm1)
```

```
     chisq      ratio        rdf          p
5393.43622    3.66651 1471.00000    0.00000
```

we can see that p is <0.05, there is evidence of overdispersion

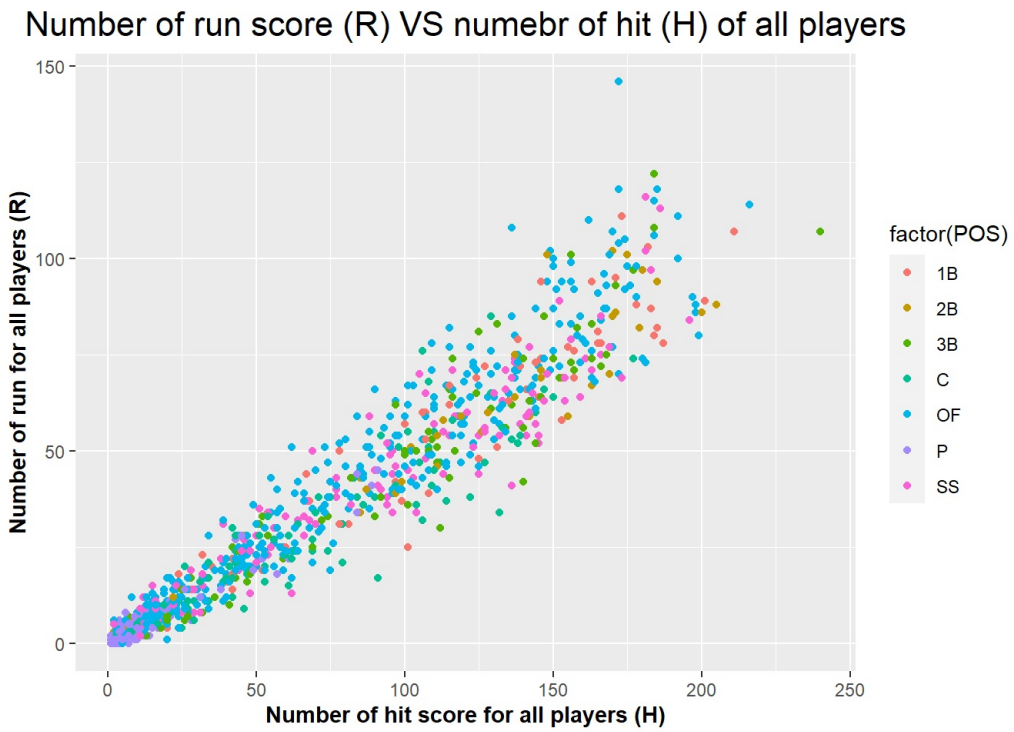
Lets investigate the source of 2 cluster of data points in our graph of absolute value of residuals versus predicted means.

```
glm1 %>%
ggplot(aes(POS, R))+
  geom_boxplot()+
  geom_hline(yintercept=2,col="red") +
  labs(
     title = "Box plots showing number of run scores for all POS",
    x = "POS",
    y = "Number of run score (R)"
  ) +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```



Seems like majority of POS = “P” do not really has any run score.

```
#Investigating the Run score and Hit relationship
glm1 %>% ggplot(aes(x=H,y=R,colour = factor(POS)))+
  geom_point() +
  labs(
     title = "Number of run score (R) VS numebr of hit (H) of all players",
    x = "Number of hit score for all players (H)",
    y = "Number of run for all players (R)"
  ) +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```

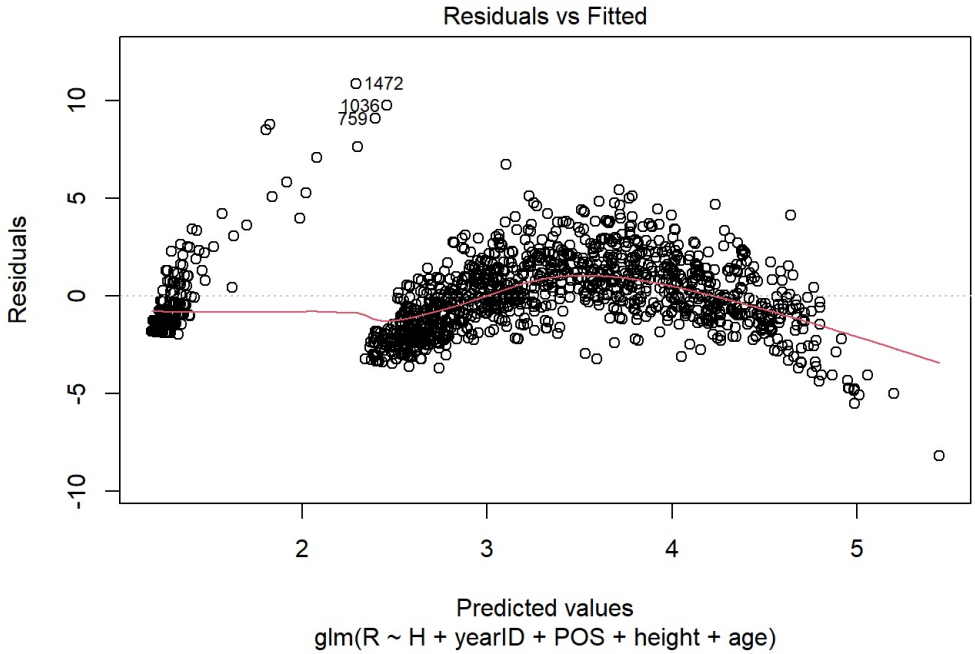


We can see position P clusters at zero hit and zero run. Lots of “P” are accumulating in H = 0, R = 0, they are not really not relevant to our Running score activities and should be involve in our prediction model. Therefore, we should really eliminate POS = P into our OnBase data before modeling!

```
#For demonstration
glm1_withoutP <- glm(R ~ H + yearID + POS + height + age ,data = OnBase[!OnBase$POS == "P",],family="poisson")
```

-Linearity: Evaluate using (deviance) residuals vs fitted and see if its fairly flat.

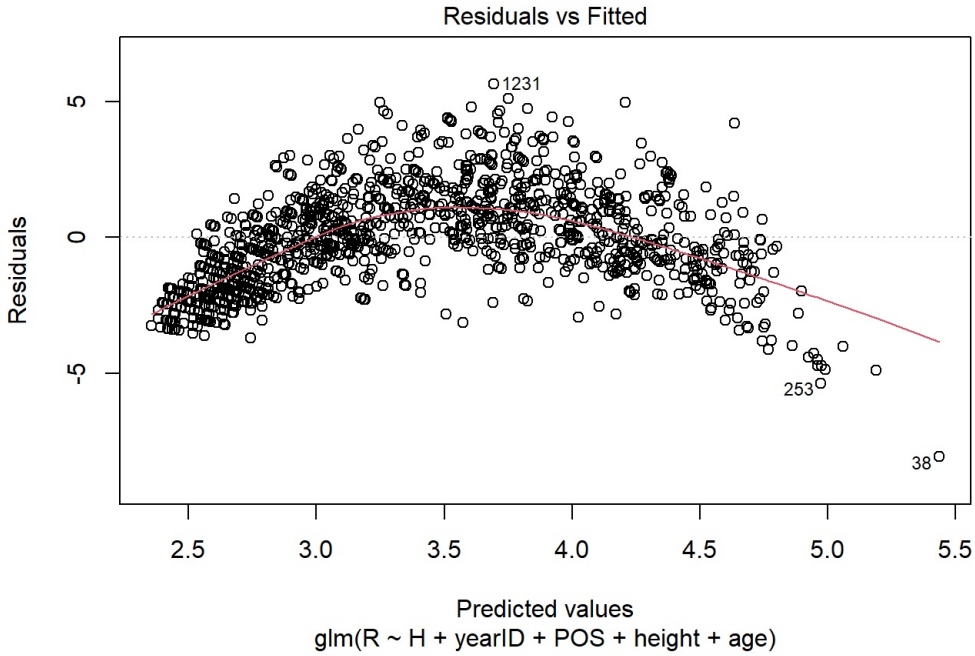
```
plot(glm1,which=1)
```



when comparing the red line to the black line, we can see its not straight and again 2 cluster displayed due to POS = “S”.

Lets review a Residuals vs Fitted when the POS = “P” is out.

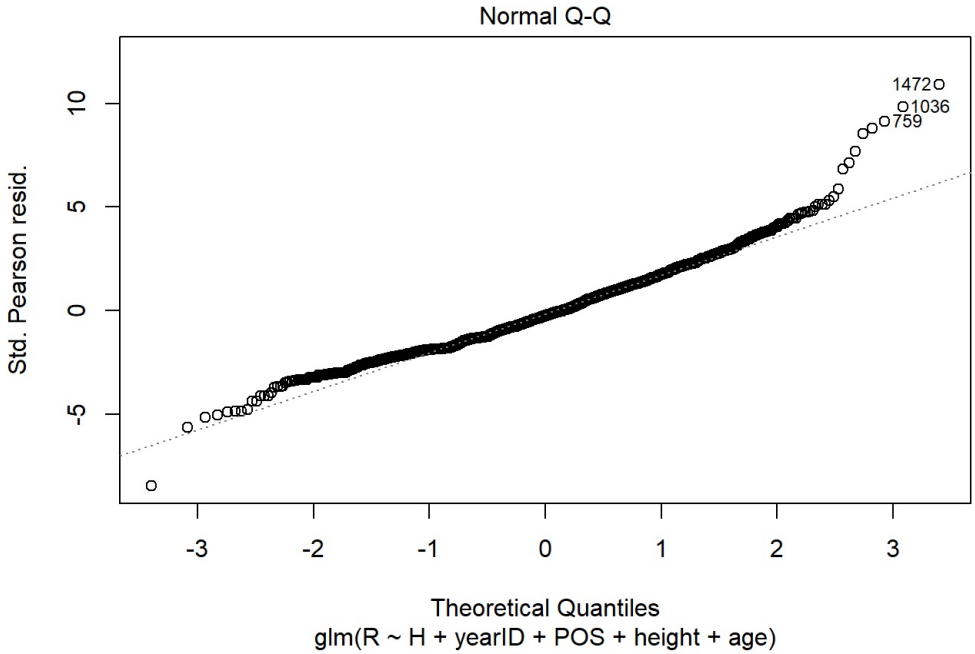
```
plot(glm1_withoutP,which=1)
```



We can see the data points are not flat in the graph and showing a inverted parabola shape. We have compromised linearity in here. Excluding it again would be a good idea but applying a different model could also potentially produce a better outcome.

-Distribution: For deviance residuals, we again investigate the qqplot.

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



This is showing a nice straight line but still showing a tail. In this case, we don't need to use robust confidence intervals later.

-Independence: We need to investigate (deviance) residuals as a function of order of datapoints and look for evidence of "snaking". We don't have a natural order in this dataset, so we can't investigate in here.

Q3e. [2 + 4 points] Now create a new model that includes teamID as a random effect. Ensure there are no fit warnings. What does the result tell us about the importance of team on number of runs that players score? Is this a relatively large or small effect? How could we check the statistical significance of this effect in R?

```
glmer1 <-glmer(R ~ H + yearID + POS + height + age + (1|teamID),data = OnBase,family="poisson", nAGQ=0)
summary(glmer1)
#Noted that I did not exclude POS=P in here as been told
```

```
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
 12286.5  12350.1  -6131.2  12262.5     1470

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.7776 -1.4380 -0.2174  0.9938 10.9249

Random effects:
 Groups Name        Variance Std.Dev.
 teamID (Intercept) 0.009313 0.0965
Number of obs: 1482, groups:  teamID, 33

Fixed effects:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.617e+00  1.748e-01  14.969  < 2e-16 ***
H            1.300e-02  9.188e-05 141.458  < 2e-16 ***
yearID2015   3.197e-02  1.010e-02   3.165 0.001552 **
POS2B       -9.808e-03  1.698e-02  -0.578 0.563562
POS3B        9.049e-03  1.584e-02   0.571 0.567888
POSC        -6.275e-02  2.085e-02  -3.009 0.002617 **
POSOF        6.560e-02  1.330e-02   4.933 8.09e-07 ***
POSP        -1.138e+00  3.730e-02 -30.523  < 2e-16 ***
POSSS       -1.099e-02  1.767e-02  -0.622 0.534108
height      -5.710e-03  2.306e-03  -2.476 0.013269 *
age          4.783e-03  1.256e-03   3.807 0.000141 ***
---
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          -0.037
yearID2015 -0.028  0.153
POS2B      -0.321 -0.002 -0.029
POS3B      -0.197  0.024  0.004  0.462
POSC       -0.133  0.115  0.025  0.340  0.352
POSOF      -0.163  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      -0.256  0.005 -0.012  0.444  0.433  0.320  0.503  0.164
height     -0.962 -0.082 -0.036  0.265  0.146  0.088  0.095 -0.066  0.193
age        -0.231  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        -0.005
```

Here is our model $$

$$
\begin{aligned}
log(\text{mean Number of runs that players score, R}) \sim \text{pois}(&2.617 + 0.013 \times H + 0.03197 \times yearID2015 \\
&- 0.009808 \times POS2B + 0.009049 \times POS3B - 0.06275 \times POSC + 0.0656 \times POSOF \\
&- 1.138 \times POSP - 0.01099 \times POSSS - 0.00571 \times height + 0.004783 \times age + \text{u}) \\
&\text{u} \sim N(0, 0.0965)
\end{aligned}
$$

$$ We can see that Tau τ, standard deviation of random effect, i.e. teamID, is 0.0965.

calculation the multiplicate effect of top team in τ of 95% confident intervals:

```
exp(1.96*0.0965)
```

```
[1] 1.20821
```

Intecept tells us that the mean number of runs that players score is 2.617 and 95% confident intervals of τ = ±1.204498, with range of 2.4.

Just being in the top team in opposed to the average team, top team scores 20% more run scores and this is huge if the context we are talking about is in a elite athlete competition. Just by looking at the coefficients, the difference between the best team against the worst team has a much greater impact than any variables included in the equation, in particularly for all categorical variables. So clearly, team is an important factor in determining the run scores.

```
glmer2 <-glmer(R ~ H + yearID + POS + height + age + (1|teamID),data = OnBase,family="poisson") #noted that nAGQ
is removed in-order for this operation to work.
confint(glmer2)
```

```
                  2.5 %       97.5 %
.sig01       0.075227043  0.127737586
(Intercept)  2.274611454  2.958568581
H            0.012816823  0.013177096
yearID2015   0.012164641  0.051771514
POS2B       -0.043097274  0.023449248
POS3B       -0.022017698  0.040082793
POSC        -0.103745449 -0.022014292
POSOF        0.039567788  0.091687732
POSP        -1.212189337 -1.065960922
POSSS       -0.045659573  0.023597764
height      -0.009959629 -0.001320943
age          0.002319087  0.007244012
```

As the .sig01 represents the tau 95% confident intervals and it doesn't crosses zero, this tells us that the random effect is significant.

Another way to test the significance of random effects is by comparing a nested model. Lets perform a likelihood ratio tests:

```
anova(glmer1,glm1)
```

glm1
glmer1

2 rows | 1-1 of 9 columns

we can see p < 0.05 and hence the random effect is significant.

Q3f. [2 + 0 points] What is the mean number of runs could you expect 30-year old, 72 inch tall outfielders playing for the Baltimore Orioles in 2015 with 20 hits to have scored?

```
predict(glmer1,newdata=data.frame(age = 30, height = 72, POS = "OF", teamID = "BAL", yearID = "2015", H = 20),typ
e = "response")
```

```
      1
17.5339
```

So we would predict this player will have 17.5339.

4. Lasso Regression for Logistic Regression

Q4a. [4 + 0 points] 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'. Split the resulting into a training and a testing set so that the variable 'DivWin' is balanced between the two datasets. Use the seed 123.

```
#Creation of Divwinners
Divwinners <- Teamsdata  %>%
  select( !starts_with('team')) %>%
  select(!starts_with('park')) %>%
  select(!c('lgID', 'Rank','franchID','divID', 'WCWin','LgWin','WSWin', 'name')) %>%
  drop_na()
```

Set.seed(123) & creating training and testing data in a 80:20 split.

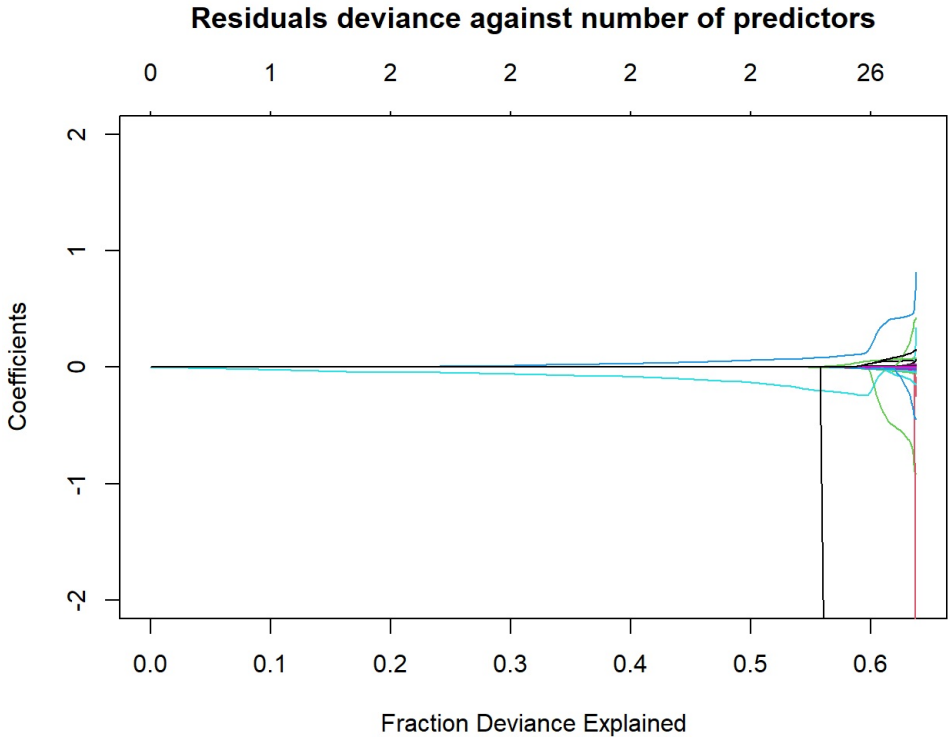
```
set.seed(123)

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

Q4b. [4 + 0 points] Use the training data to fit a logistic regression model using the 'glmnet' command. Plot residual deviance against number of predictors.

```
DivWintrain <- as.vector(train.data$DivWin)

Divwinnerspredict <-model.matrix(~.-1,train.data[,c(-6)]) #Removing response variable column "DivWin"

Divwinnersfit <- glmnet(Divwinnerspredict, DivWintrain, family = "binomial")
```

Lets plot a graph for Residual deviance against number of predictors

```
par(mar = c(4,4,4,3))
plot(Divwinnersfit,xvar="dev", ylim=c(-2,2))
title("Residuals deviance against number of predictors", line = 3)
```



Q4c. [2 + 2 points] How many nonzero model coefficients are needed to explain 50% of the deviance? 60%? Which coefficients are these in each case?

```
#To figure out which coefficients are involved, print out the values of lambda compared to the % Deviance and Df
Divwinnersfit
```

```
Call:  glmnet(x = Divwinnerspredict, y = DivWintrain, 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 variances: In Divwinnersfit, row number = 21, df = 2, %Dev = 50.02 and lambda = 0.038030

50% of the variance was explained with just 2 coefficient and we need at least 26 coefficient to explain 60% of the variance.

```
#To explain 50% of deviance in the result we select lambda = 0.038030
Divwinners50 <- coef(Divwinnersfit, s=0.038030)
Divwinners50@Dimnames[[1]][1+Divwinners50@i]
```

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

These are the variable required to explain 50% of variance. After checking with the data set Divwinners, these are all quantitative variables.

To explain 60% of variances: In Divwinnersfit, row number = 53, df = 26, %Dev = 60.13 and lambda = 0.001937

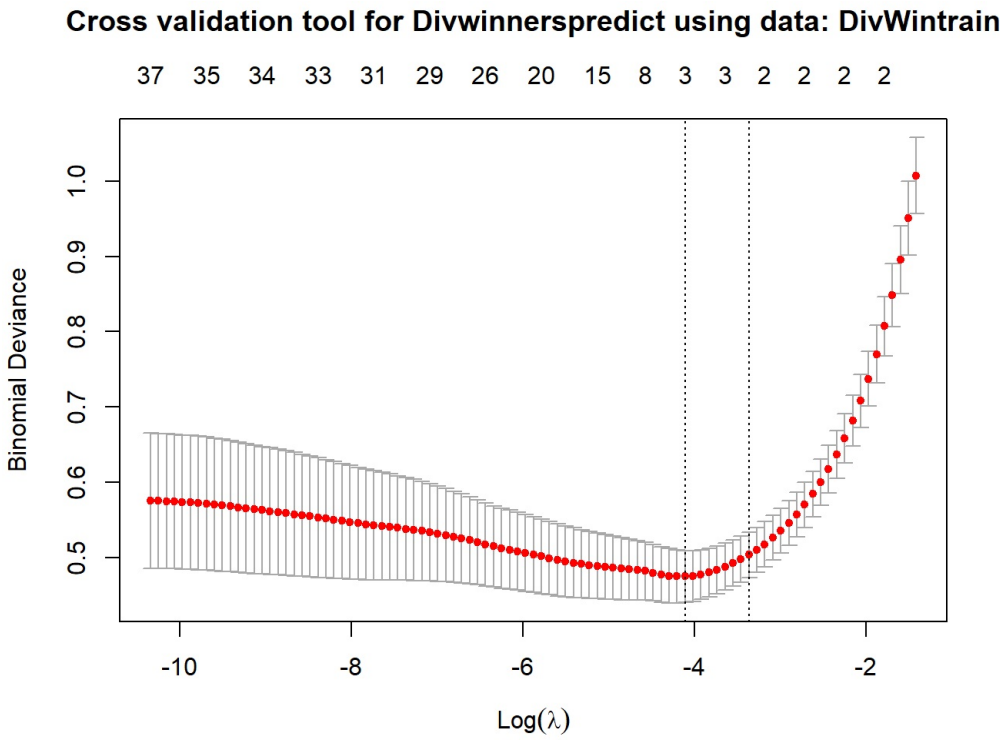
```
#To explain 60% of deviance in the result we select lambda = 0.001937
Divwinners60 <- coef(Divwinnersfit, s=0.001937)
Divwinners60@Dimnames[[1]][1+Divwinners60@i]
```

```
 [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"
```

These are the variable required to explain 60% of variance. After checking with the data set Divwinners, these are all quantitative variables.

Q4d. [2 + 1 points] Now use cross-validation to choose a moderately conservative model. State the variables you will include.

```
set.seed(321)
cv <- cv.glmnet(Divwinnerspredict,DivWintrain, family = "binomial")

par(mar = c(4,4,4,3))
plot(cv)
title("Cross validation tool for Divwinnerspredict using data: DivWintrain", line = 3)
```



We should include 2 to 4 variables as the range of Binomial Deviance is smallest in this range.

```
Divwinners1se<-coef(Divwinnersfit,s=cv$lambda.1se)
setdiff(Divwinners1se@Dimnames[[1]][1+Divwinners1se@i],Divwinners60@Dimnames[[1]][1+Divwinners60@
i])
```

```
character(0)
```

There is no set diff between using the s = 0.001937 which explain 60% variances in the data then using lambda = 1se, with no surprise nothing new added.

```
setdiff(Divwinners1se@Dimnames[[1]][1+Divwinners1se@i], Divwinners50@Dimnames[[1]][1+Divwinners50@i])
```

```
character(0)
```

Also checked on s for explaining 50% variance, also no set diff of course.

Lets look at lambda.min where binomial deviance is lowest:

```
Divwinnersmax<-coef(Divwinnersfit,s=cv$lambda.min)
Divwinnersmax@Dimnames[[1]][1+Divwinnersmax@i]
cv$lambda.min #s value where least binomial deviance
```

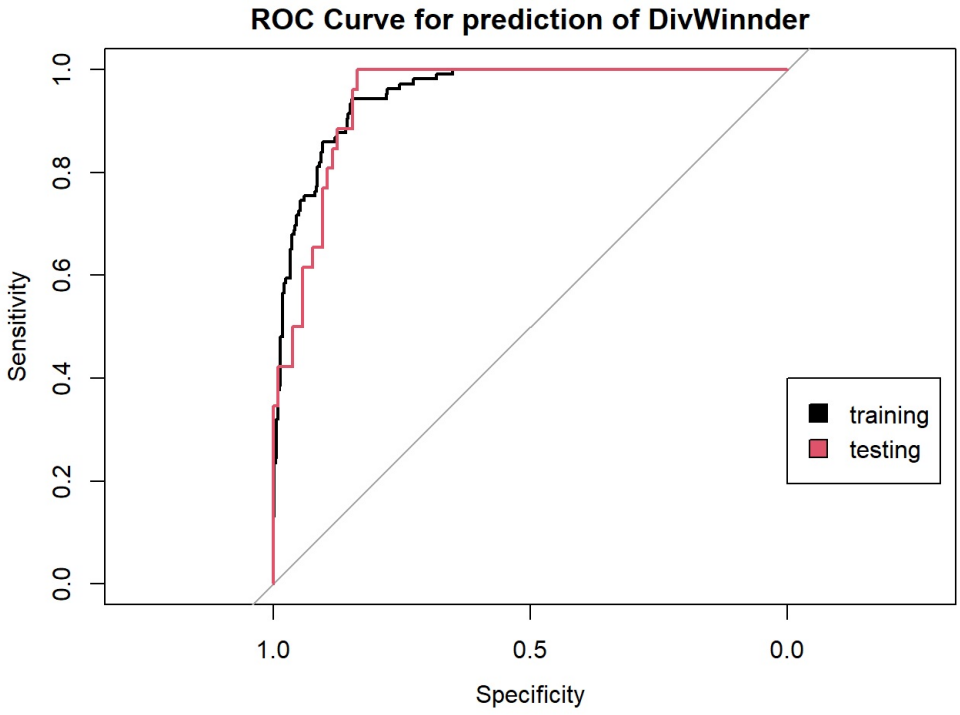
```
[1] "(Intercept)" "W"           "L"           "attendance"
[1] 0.01646383
```

Now we have: 3 quantitative variable:“W” , “L” , “attendance” and their total levels is only 3.

The value of s = 0.01646383 where there is the least binomial deviance. If we look at the graph, this explains around 54% of the variances of the result. This is pretty good, and requires a lot more variables in the equation to push the variances to 60%. Hence, I am selecting these 3 variables: “W” , “L” , “attendance” for my equation as can have 54% explained and its worth adding just an additional variable when you can have the the least binomial deviance.

Q4e. [4 + 2 points] Fit the model on the training data, then predict on the testing data. Plot comparative ROC curves and summarise your findings.

```
set.seed(123)
train.model <- glm(as.factor(DivWin) ~ W + L + attendance, data = train.data, family = "binomial")

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

# Plotting the ROC curve
roctrain<-roc(response=train.data$DivWin,predictor=predtrain,plot=TRUE,main="ROC Curve for prediction of DivWinnd
er",auc=TRUE)
roc(response=test.data$DivWin,predictor=predtest,plot=TRUE,auc=TRUE,add=TRUE,col=2)
legend(0,0.4,legend=c("training","testing"),fill=1:2)
```




```

Call:
roc.default(response = test.data$DivWin, predictor = predtest,     auc = TRUE, plot = TRUE, add = TRUE, col = 2)

Data: predtest in 104 controls (test.data$DivWin N) < 26 cases (test.data$DivWin Y).
Area under the curve: 0.9442
```

These two curves are very similar in shape. This is a good sign that the model is not overfitted to the training data.

Q4f. [4 + 2 points] Find Youden’s index for the training data and calculate confusion matrices at this cutoff for both training and testing data. Comment on the quality of the model for prediction in terms of false negative and false positive rates for the testing data.

```
# Youden's Index for the training data
youdenN<-coords(roctrain,"b",best.method="youden",transpose=TRUE)
youdenN
youdenN[2]+youdenN[3] #Sum of sensitivity + specificity for Youden's Index
```

```
  threshold specificity sensitivity
  0.1836071   0.8468900   0.9433962
specificity
   1.790286
```

Youden's Index is 0.1836071 for the training data.

```
# Calculate Confusion matrix for training data
train.data$predwins <- ifelse(predict(train.model,newdata=train.data, type="response")>= 0.1836071,"Y","N")
T1 <- table(train.data$predwins,as.factor(train.data$DivWin))
names(dimnames(T1))<- list("Predicted", "Actual")
T1
```

```
         Actual
Predicted   N   Y
        N 354   6
        Y  64 100
```

```
sensitivity(T1)
specificity(T1)
```

```
[1] 0.84689
[1] 0.9433962
```

Note that this model is testing for "N". Sensitivity for training data = 354/(354+64) = 0.84689 Specificity for training data = 100/(100+6) = 0.9433962

False positive rate for training data = 1-0.9433962 = 0.15311 False Negative rate for training data = 1-0.84689 = 0.0566038

```
# Calculate Confusion matrix for testing data
test.data$predwins <- ifelse(predict(train.model,newdata=test.data, type="response")>= 0.1836071,"Y","N")
T2 <- table(test.data$predwins,as.factor(test.data$DivWin))
names(dimnames(T2))<- list("Predicted", "Actual")
T2
```

```
         Actual
Predicted  N  Y
        N 87  1
        Y 17 25
```

```
sensitivity(T2)
specificity(T2)
```

```
[1] 0.8365385
[1] 0.9615385
```

Sensitivity for testing data = 87/(87+17) = 0.8365385 Specificity for testing data = 25/(25+1) = 0.9615385

False positive rate for testing data = 1-0.8365385 = 0.1634615 False Negative rate for testing data = 1-0.9615385 = 0.0384615

From the findings above, we can see the sensitivity and specificity are both very similar as well as high on the 2 data sets. This implies that the false positive and false negative rate are low in the data sets using the model we built from training data.

Q4g. [5 + 1 points] Calculate the sensitivity+specificity on the testing data as a function of divID and plot as a barchart. Is the prediction equally good for all divisions?

```
train.model <- glm(as.factor(DivWin) ~ W + L + attendance, data = train.data, family = "binomial") #recall

#Merging data with Teamsdata to regain divID column into the test.data
divID <- merge(x = test.data, y = Teamsdata[ , c("yearID","W","L","R","H","divID")], by = c("yearID","W","L","R",
"H"), all.x=TRUE)

#There are 3 divID: "E" "W" "C"

#Begin with creating the sensitivity+specificity value for div_E
divID_E <- divID %>%
  filter(divID == "E")

Table_E <- table(divID_E$predwins,as.factor(divID_E$DivWin))
names(dimnames(Table_E))<- list("Predicted", "Actual")
Table_E #noted that confusion matrix is not necessary to display in here but included for better visualisation.

E_SS <- sensitivity(Table_E) + specificity(Table_E)
E_SS
```

```
         Actual
Predicted  N  Y
        N 25  0
        Y  8  8
[1] 1.757576
```

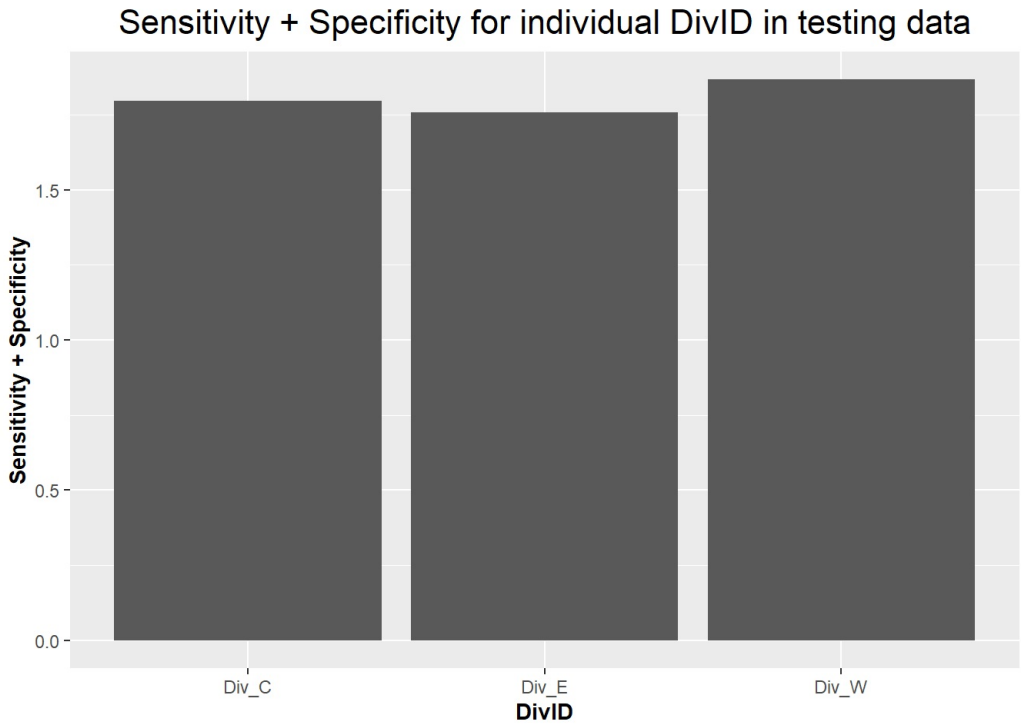
```
#Creating the sensitivity+specificity value for div_W
divID_W <- divID %>%
  filter(divID == "W")

Table_W <- table(divID_W$predwins,as.factor(divID_W $DivWin))
names(dimnames(Table_W))<- list("Predicted", "Actual")
Table_W

W_SS <- sensitivity(Table_W) + specificity(Table_W)
W_SS
```

```
         Actual
Predicted  N  Y
        N 26  0
        Y  4  6
[1] 1.866667
```

```
#Creating the sensitivity+specificity value for div_C
divID_C <- divID %>%
  filter(divID == "C")

Table_C <- table(divID_C$predwins,as.factor(divID_C$DivWin))
names(dimnames(Table_C))<- list("Predicted", "Actual")
Table_C

C_SS <- sensitivity(Table_C) + specificity(Table_C)
C_SS
```

```
         Actual
Predicted  N  Y
        N 36  1
        Y  5 11
[1] 1.794715
```

```
#Creating a dataframe for the sensitivity+specificity with individual divID labelled

SnS <- c(E_SS, W_SS, C_SS)
DivID <- c("Div_E", "Div_W", "Div_C")

Sum_SnS <- data.frame(DivID, SnS)

ggplot() +
geom_col(mapping = aes(x = DivID,y = SnS))+
scale_y_continuous() +
  labs(
  x = "DivID",
  y = "Sensitivity + Specificity",
  title = "Sensitivity + Specificity for individual DivID in testing data")  +
  theme(axis.title.x = element_text(face = "bold"),
axis.title.y = element_text(face = "bold")) +
  theme(plot.title = element_text(
                       hjust = 0.5,
                       size  = rel(1.5)
                     )
  )
```



Although we can see that Div_W has the highest sensitivity+specificity value, followed by Div_C and Div E, they are actually very similar when we observe them from the bar chart. The difference between each DivID is really minimal.

We can conclude that the sensitivity + specificity are fairly similar when we are applying the model into individual divID which means the quality of the model is equally good for each divID.