MATH5743M: Statistical Learning: Assessed Practical 1 - Predicting the Olympic Games

Anupam Bose, 201570198, School of Mathematics

Semester 2 2022

We will investigate the medals won by 71 countries in year 2008 in Beijing, 2012 in London and 2016 in in Rio. The dataset contains countries who have won at least one gold medal in each of the last three games. It contains data of their population, GDP (in billions of US dollars) and number of medals won in those years.Here we will use multivariate regression method to predict number of medals won by training the model using various methods.

Task 1:

The libraries to import to do further analysis on data

```
library(Metrics)
library(MASS)
library(tidyverse)
library(sf)
library(tinytex)
```

The data has been imported into a dataframe (df) by using the **read.csv** function.

```
df = read.csv("medal_pop_gdp_data_statlearn.csv")
```

The first 6 rows of our data will give us some idea about it.

```
head(df)
```

```
##      Country     GDP Population Medal2008 Medal2012 Medal2016
## 1    Algeria  188.68   37100000         2         1         2
## 2  Argentina  445.99   40117096         6         4         4
## 3    Armenia   10.25    3268500         6         3         4
## 4  Australia 1371.76   22880619        46        35        29
## 5 Azerbaijan   63.40    9111100         7        10        18
## 6    Bahamas    7.79     353658         2         1         2
```

Now, let's look into summary statistics of our data:

```
summary(df)
```

```
##    Country               GDP              Population          Medal2008
##  Length:71          Min.   :    6.52   Min.   :3.537e+05   Min.   :  1.00
##  Class :character   1st Qu.:   51.52   1st Qu.:5.513e+06   1st Qu.:  2.00
##  Mode  :character   Median :  229.53   Median :1.673e+07   Median :  6.00
##                     Mean   :  903.25   Mean   :7.384e+07   Mean   : 13.11
##                     3rd Qu.:  704.37   3rd Qu.:4.958e+07   3rd Qu.: 13.50
##                     Max.   :15094.00   Max.   :1.347e+09   Max.   :110.00
##    Medal2012       Medal2016
##  Min.   :  1.0   Min.   :  1.00
##  1st Qu.:  3.0   1st Qu.:  3.00
##  Median :  6.0   Median :  7.00
##  Mean   : 13.3   Mean   : 13.44
##  3rd Qu.: 13.0   3rd Qu.: 15.00
##  Max.   :104.0   Max.   :121.00
```

Linear models are a type of model that describes a response variable as a linear combination of predictor variables.Multiple regression is an extension of simple linear regression. It is used when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable). The variables we are using to predict the value of the dependent variable are called the independent variables (or sometimes, the predictor, explanatory or regressor variables).

The multiple linear regression for Y as a function of X is given by the following equation:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon,\ \epsilon \sim N(0, \sigma^2)$$

where,

- x_1 and x_2 are input variables
- A scalar constant – β_0
- β_1, β_2 are regression coefficients
- A residual ϵ is unknown, but assumed to be normally distributed with zero mean and unknown variance:

$$\epsilon \sim (0, \sigma^2)$$

- Y is the target variable

In R, we will use glm() function to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution. Generalized linear model (GLM) is a generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution like Gaussian distribution.

```
model_train = glm(Medal2012 ~ Population + GDP , data= df)
summary(model_train)
```

```
##
## Call:
## glm(formula = Medal2012 ~ Population + GDP, data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -20.568   -5.961   -2.462    3.932   60.121
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.076e+00  1.500e+00   4.051 0.000133 ***
## Population  5.247e-09  7.193e-09   0.729 0.468225
## GDP         7.564e-03  7.325e-04  10.326 1.45e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 132.1562)
##
##     Null deviance: 28402.8  on 70  degrees of freedom
## Residual deviance:  8986.6  on 68  degrees of freedom
## AIC: 553.19
##
## Number of Fisher Scoring iterations: 2
```

Here we have considered two predictor variables which is population and GDP. Medal2012 is our target variable. The key points to note from the summary are:

- The regression coefficient of the predictor variable population is 5.247×10^{-9} which is very small with the standard error of 7.193×10^{-9}. The P-value is 0.468 which is quite large($0.4682 \gg 0.05$). This shows that the population of the country is statistically insignificant with regards to country's medal count.
- On the other hand, the regression coefficient of the predictor variable GDP(7.564e-03) which is relatively large with the standard error of 7.325×10^{-4}. The P-value is of it is 1.45×10^{-15} which is quite small and is less than $0.05(1.45 \times 10^{-15} \ll 0.05)$. It tells that the GDP of a country is statistically significant and does impact country's medal count.
- The intercept is 6.076.

To confirm the significance of the predictor variables we calculate confidence interval of their coefficient which is given by:

$$\text{C.I.: Estimate} \pm t_c \times \text{Standard Error}$$

The t-statistic value(t_c) can be calculated by using *qt* function. We have 71 data points with one intercept and two regression coefficients. Therefore, the number of degrees of freedom will be 71-3 = 68. With 95% confidence interval for which $P(t > t_c) = 0.975$, t_c is calculated.

```r
#t-statistic Value
tc = qt(p=0.975, df=68)

#Confidence Interval of population
pop_ci = summary(model_train)$coefficients[2, 1] +
  c(-1,1)*tc*summary(model_train)$coefficients[2, 2]
print(pop_ci)
```

```
## [1] -9.105934e-09  1.959943e-08
```

```r
#Confidence Interval of gdp
gdp_ci = summary(model_train)$coefficients[3, 1] +
  c(-1,1)*tc*summary(model_train)$coefficients[3, 2]
print(gdp_ci)
```

```
## [1] 0.006102319 0.009025843
```

The following observation is made from the confidence intervals:

- The confidence interval of the coefficient of the variable population(β_1) shows that 0 lies in its range. This confirms the insignificance of the variable with regards to medal count. when $\beta_1 = 0$, it has no impact on the target variable.
- The confidence interval of the coefficient of the variable gdp(β_2) is positive and 0 does not lie in its range. The values are quite small but it shows some significance. Hence, it does impact our target variable relatively.

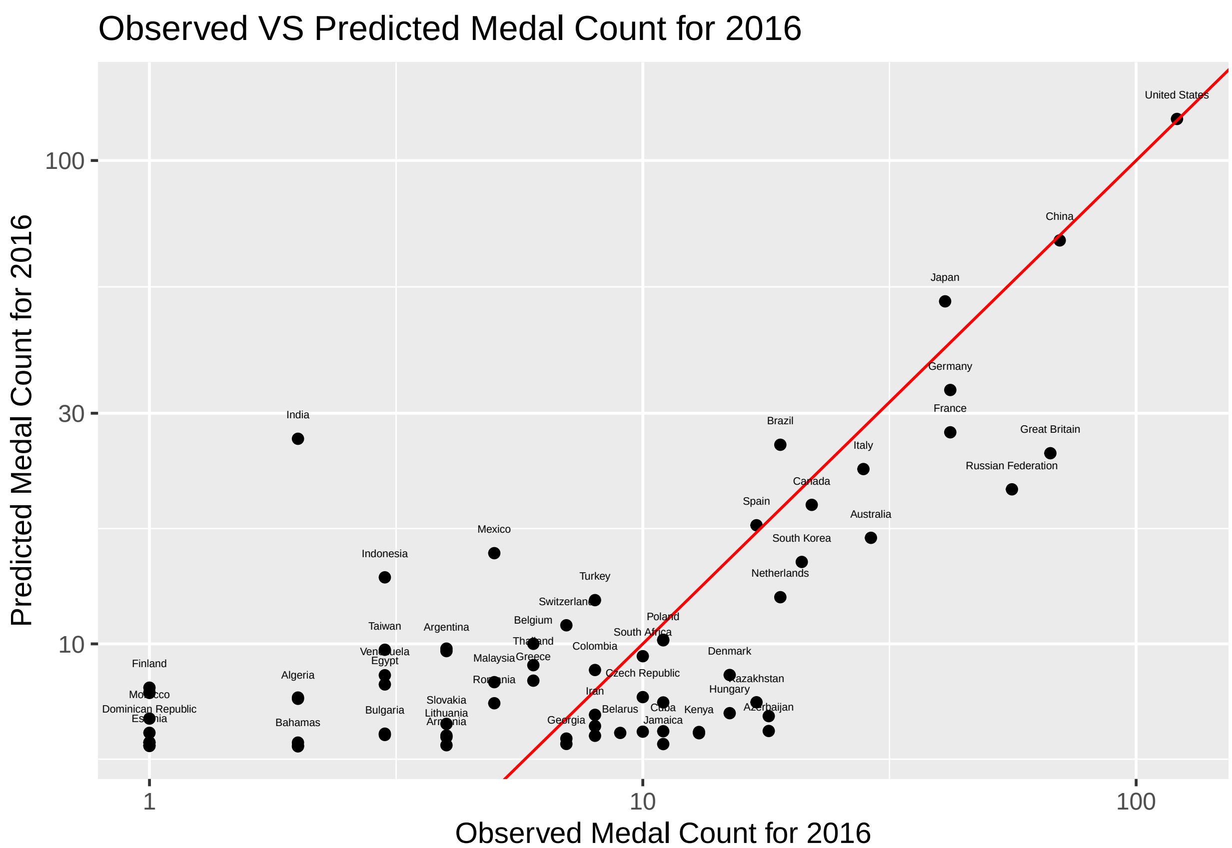
Considering no change in country's GDP and population, the trained model can now be used to predict medal count for the year 2016(Medal2016) using predict function. The first 10 predicted vs observed values of medal count has been shown below.

```r
#Testing model with prediction
model_test=df[,c(2,3)]
pred= predict(model_train, newdata = model_test)
df %>%
  mutate(Pred_Medal2016=round(pred)) %>%
  select(Medal2016,Pred_Medal2016)%>%
  head(10)
```

```
##    Medal2016 Pred_Medal2016
## 1          2              8
## 2          4             10
## 3          4              6
## 4         29             17
## 5         18              7
## 6          2              6
## 7          2              6
## 8          9              7
## 9          6             10
## 10        19             26
```

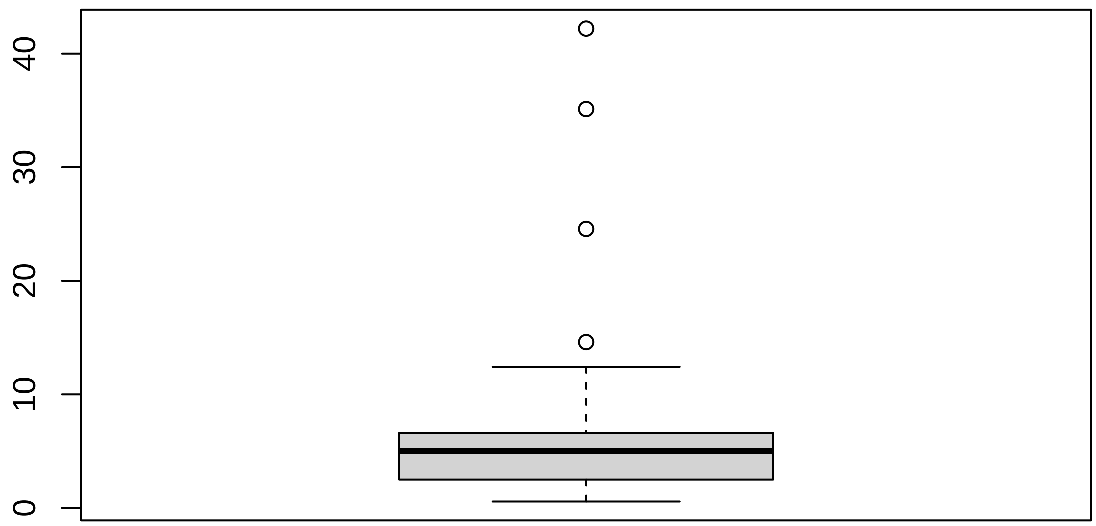
The plot of predicted vs observed value of Medal2016 gives better insights. The regression line is also fitted in the plot. To see the data points better the axes are log transformed. The plot shows that the data points are quite away from the regression line. The model is performing decently.

```r
#Plot with log transformed axes
ggplot(data=df, aes(y=pred, x=Medal2016,label = Country)) +
  geom_point() +
  ggtitle("Observed VS Predicted Medal Count for 2016")+
  scale_y_continuous(trans='log10')+
  scale_x_continuous(trans='log10')+
  ylab("Predicted Medal Count for 2016")+
  xlab("Observed Medal Count for 2016")+
  geom_abline(slope = 1, intercept = 0, col = 'red')+
  geom_text(size=1.5,nudge_y = 0.05,  check_overlap = TRUE)
```



Boxplot of the absolute difference between the predicted and observed values shows 4 outliers. It can be known by finding the 75th quantile and Interquantile range of it using function quantile()

```
#Boxplot of Absolute error
boxplot(abs(pred - df$Medal2016))
```




```r
#Quantile
quantile(abs(pred - df$Medal2016))
```

```
##         0%        25%        50%        75%       100%
##  0.5705314  2.4987819  5.0037617  6.6167195 42.2044909
```

```r
#Outliers of Absolute errors:
#Values Beyond [75th Quantile +(1.5*IQR)]

6.61 + 1.5*IQR(abs(pred - df$Medal2016))
```

```
## [1] 12.78691
```

Hence, following 4 countries are outliers with Great Britain having the highest absolute error:

```r
#Countries with Absolute Error Outliers
df %>%
  select(Country,Medal2016) %>%
  mutate(Absolute_Error = abs(pred - Medal2016)) %>%
  mutate(pred=round(pred)) %>%
  select(Country,Medal2016,pred,Absolute_Error) %>%
  mutate(Absolute_Error=round(Absolute_Error)) %>%
  filter(Absolute_Error>12.786)%>%
  arrange(desc(Absolute_Error))
```

```
##               Country Medal2016 pred Absolute_Error
## 1      Great Britain        67   25             42
## 2 Russian Federation        56   21             35
## 3              India         2   27             25
## 4             France        42   27             15
```

The model's accuracy can be determined by finding mean absolute error using function *mae*() and Root mean squared error using function *rmse*().

```
#Mean Absolute Error and Root Mean Squared Error
mae(df$Medal2016,pred)
```

```
## [1] 6.104344
```

```
rmse(df$Medal2016,pred)
```

```
## [1] 9.112593
```

The value of mean absolute error and RMSE is 6.1043 and 9.11 respectively which shows the model is performing decently.

Task 2:

We repeat the task 1 but with log-transformed medal count in 2012 to check and compare the performance of the model with the previous one.

The benefits of the logarithmic transformation are:

- Reduces overfitting of data.
- Less computational power is required.
- Helps if the distribution is skewed by transforming it.
- Improves linearity between predictor and target variables.

```
log_of_Medal2012=log(df$Medal2012)
log_model_train = glm(log_of_Medal2012 ~ Population + GDP , data= df)
summary(log_model_train)
```

```
##
## Call:
## glm(formula = log_of_Medal2012 ~ Population + GDP, data = df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73090  -0.75630   0.02616   0.77789   2.22198
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.569e+00  1.263e-01  12.422  < 2e-16 ***
## Population  1.105e-10  6.058e-10   0.182    0.856
```

```
## GDP         3.161e-04  6.170e-05   5.123 2.68e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9376449)
##
##     Null deviance: 96.505  on 70  degrees of freedom
## Residual deviance: 63.760  on 68  degrees of freedom
## AIC: 201.85
##
## Number of Fisher Scoring iterations: 2
```

The following observations could be made from the model:

- The regression coefficient of the predictor variable population is 1.105×10^{-10} which is very small with the standard error of 6.058×10^{-10}. The P-value is 0.856 which is quite large ($0.856 \gg 0.05$). This shows that the population of the country is statistically insignificant with regards to country's medal count.
- On the other hand, the regression coefficient of the predictor variable GDP (3.161e-04) is relatively large with the standard error of 6.170×10^{-5}. The P-value is of it is 2.68×10^{-6} which is quite small and is less than $0.05 (2.68 \times 10^{-6} \ll 0.05)$. It tells that the GDP of a country is statistically significant and does impact country's medal count.
- The intercept is 1.569.

```r
tc <- qt(p=0.975, df=68)
#Confidence Interval of population
pop_ci_log <- summary(log_model_train)$coefficients[2, 1] +
  c(-1,1)*tc*summary(log_model_train)$coefficients[2, 2]
print(pop_ci_log)
```

```
## [1] -1.098446e-09  1.319455e-09
```

```r
#Confidence Interval of gdp
gdp_ci_log = summary(log_model_train)$coefficients[3, 1] +
  c(-1,1)*tc*summary(log_model_train)$coefficients[3, 2]
print(gdp_ci_log)
```

```
## [1] 0.0001929751 0.0004392284
```

The following observation is made from the confidence intervals:

- The confidence interval of the coefficient of the variable population(β_1) shows that 0 lies in its range. This confirms the insignificance of the variable with regards to medal count. when $\beta_1 = 0$, it has no impact on the target variable.
- The confidence interval of the coefficient of the variable gdp(β_2) is positive and 0 does not lie in its range. The values are quite small but it shows some significance. Hence, it does impact our target variable relatively.

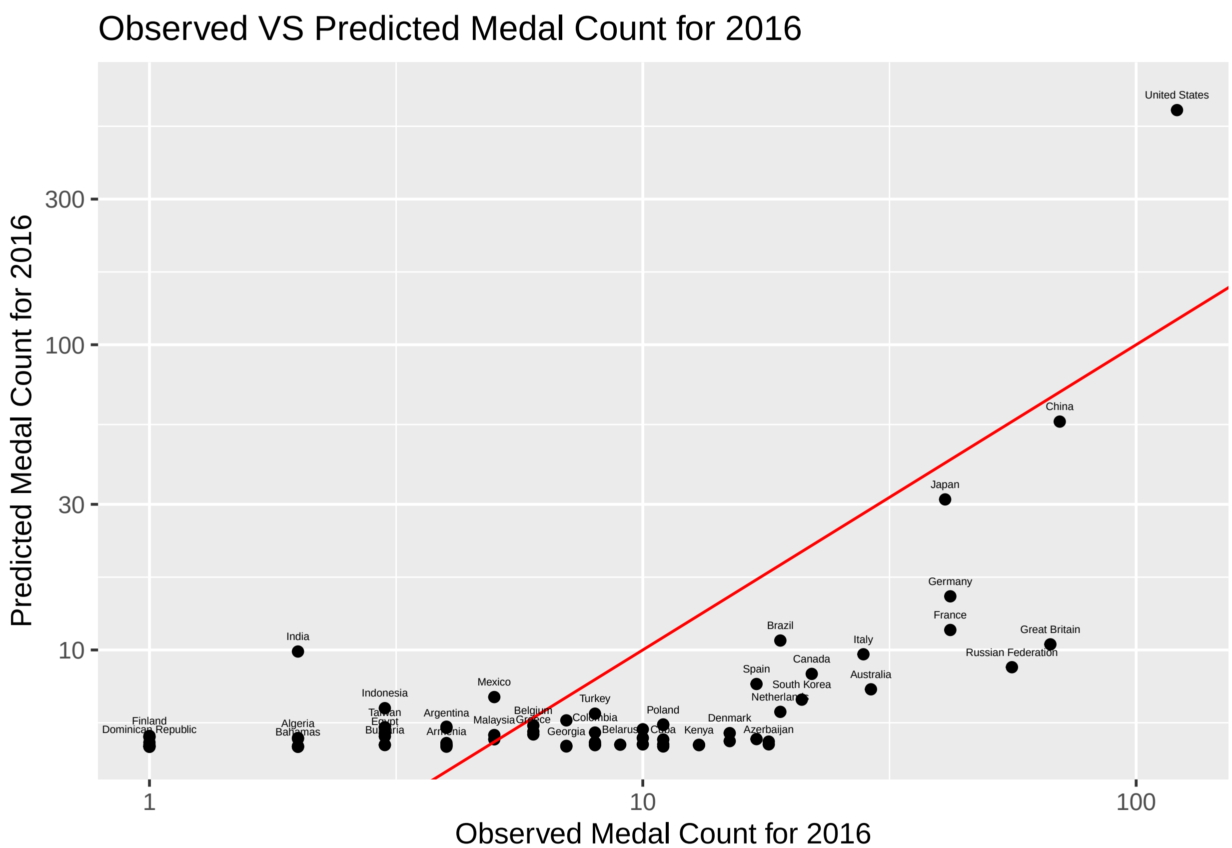
Considering no change in country's GDP and population, the trained model can now be used to predict medal count for the year 2016(Medal2016) using predict function. The first 10 predicted vs observed values of medal count has been shown below.

```
log_df <- df[,c(2,3)]
log_predictions <- exp(predict(log_model_train, newdata = log_df))
df %>%
  mutate(Pred_Medal2016=round(log_predictions)) %>%
  select(Medal2016,Pred_Medal2016)%>%
  head(10)
```

```
##    Medal2016 Pred_Medal2016
## 1          2              5
## 2          4              6
## 3          4              5
## 4         29              7
## 5         18              5
## 6          2              5
## 7          2              5
## 8          9              5
## 9          6              6
## 10        19             11
```

The plot of predicted vs observed value of Medal2016 gives better insights. The regression line is also fitted in the plot. To see the data points better the axes are log transformed. The plot shows that the data points are again away from the regression line. The performance of the model has not improved compared to the previous one. Hence, the log transformation of the target variable does not improve the model.

```
#Plot with log transformed axes
ggplot(data=df, aes(y=log_predictions, x=Medal2016,label = Country)) +
  geom_point() +
  ggtitle("Observed VS Predicted Medal Count for 2016")+
  scale_y_continuous(trans='log10')+
  scale_x_continuous(trans='log10')+
  ylab("Predicted Medal Count for 2016")+
  xlab("Observed Medal Count for 2016")+
  geom_abline(slope = 1, intercept = 0, col = 'red')+
  geom_text(size=1.5,nudge_y = 0.05,  check_overlap = TRUE)
```



We again find Outliers by calculating 75th quantile and interquantile range. This time we get 6 outliers with United States having the highest absolute error:

```
#Quantile
quantile(abs(log_predictions - df$Medal2016))
```

```
##           0%          25%          50%          75%         100%
##   0.09549282   2.19211028   3.89412223   8.76470437 466.15606827
```

```
#Outliers of Absolute errors:
#Values Beyond [75th Quantile +(1.5*IQR)]
8.76 + 1.5*IQR(abs(log_predictions - df$Medal2016))
```

```
## [1] 18.61889
```

```
#Countries with Absolute Error Outliers
df %>%
  select(Country,Medal2016) %>%
  mutate(Absolute_Error = abs(log_predictions - Medal2016)) %>%
  mutate(log_predictions=round(log_predictions)) %>%
  select(Country,Medal2016,log_predictions,Absolute_Error) %>%
  mutate(Absolute_Error=round(Absolute_Error)) %>%
  filter(Absolute_Error>18.619)%>%
  arrange(desc(Absolute_Error))
```

```
##                Country Medal2016 log_predictions Absolute_Error
## 1      United States       121             587            466
## 2      Great Britain        67              10             57
## 3 Russian Federation        56               9             47
## 4             France        42              12             30
## 5            Germany        42              15             27
## 6          Australia        29               7             22
```

Mean absolute error and RMSE of the model:

```
#Mean Absolute Error and Root Mean Squared Error
mae(df$Medal2016,pred)
```

```
## [1] 6.104344
```

```
rmse(df$Medal2016,pred)
```

```
## [1] 9.112593
```

The value of mean absolute error and RMSE is 13.70 and 56.62 respectively which shows the model isn't performing better than the previous one.

Task 3:

We repeat the task 1 by assuming $Medal2016$ has Poisson distribution and compare the performance of the model with the previous two models.

The reasons why Poisson Regression model can be considered :

- The model works best if the data is discrete with non-negative integer values.
- The outcome are counts of events and occuring randomly at constant rate.
- Medals obtained are discrete values with some counts.

```
Poisson_Model = glm(Medal2012 ~ Population + GDP , data= df,family = poisson(link='log'))
summary(Poisson_Model)
```

```
##
## Call:
## glm(formula = Medal2012 ~ Population + GDP, family = poisson(link = "log"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7459  -2.8253  -1.4710   0.9333  12.4841
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 2.193e+00  4.034e-02  54.360  < 2e-16 ***
## Population  6.049e-10  9.131e-11   6.625 3.48e-11 ***
```

```
## GDP         1.715e-04  6.672e-06  25.708  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1331.81  on 70  degrees of freedom
## Residual deviance:  690.27  on 68  degrees of freedom
## AIC: 962.24
##
## Number of Fisher Scoring iterations: 5
```

The following observations could be made from the model:

- The regression coefficient of the predictor variable population is 6.049×10^{-10} which is very small with the standard error of 9.131×10^{-11}. The P-value is 3.48×10^{-11} which is quite small ($3.48 \times 10^{-11} \ll 0.05$). This shows that the population of the country is statistically significant with regards to country's medal count.
- Also, the regression coefficient of the predictor variable GDP(1.715e-04) is relatively large with the standard error of 6.672×10^{-6}. The P-value is of it is less than 2×10^{-16} which is quite small and is less than $0.05(2 \times 10^{-16} \ll 0.05)$. It tells that the GDP of a country is statistically significant and does impact country's medal count.
- The intercept is 2.193.

```r
tc = qt(p=0.975, df=68)
#Confidence Interval of population
pop_ci_poi = summary(Poisson_Model)$coefficients[2, 1] +
  c(-1,1)*tc*summary(Poisson_Model)$coefficients[2, 2]
print(pop_ci_poi)
```

```
## [1] 4.226817e-10 7.870806e-10
```

```r
#Confidence Interval of gdp
gdp_ci_poi = summary(Poisson_Model)$coefficients[3, 1] +
  c(-1,1)*tc*summary(Poisson_Model)$coefficients[3, 2]
print(gdp_ci_poi)
```

```
## [1] 0.0001582207 0.0001848499
```

The following observation is made from the confidence intervals:

- The confidence interval of the coefficient of the variable population(β_1) is positive and 0 does not lie in its range.The values are extremely small but it does shows some significance. This confirms the significance of the variable with regards to medal count.
- The confidence interval of the coefficient of the variable gdp(β_2) is positive and again 0 does not lie in its range. The values are quite small but it shows some significance. Hence, it does impact our target variable. The values also confirms that GDP is more significant than population.

Considering no change in country's GDP and population, the trained model can now be used to predict medal count for the year 2016(Medal2016) using predict function. The first 10 predicted vs observed values of medal count has been shown below.

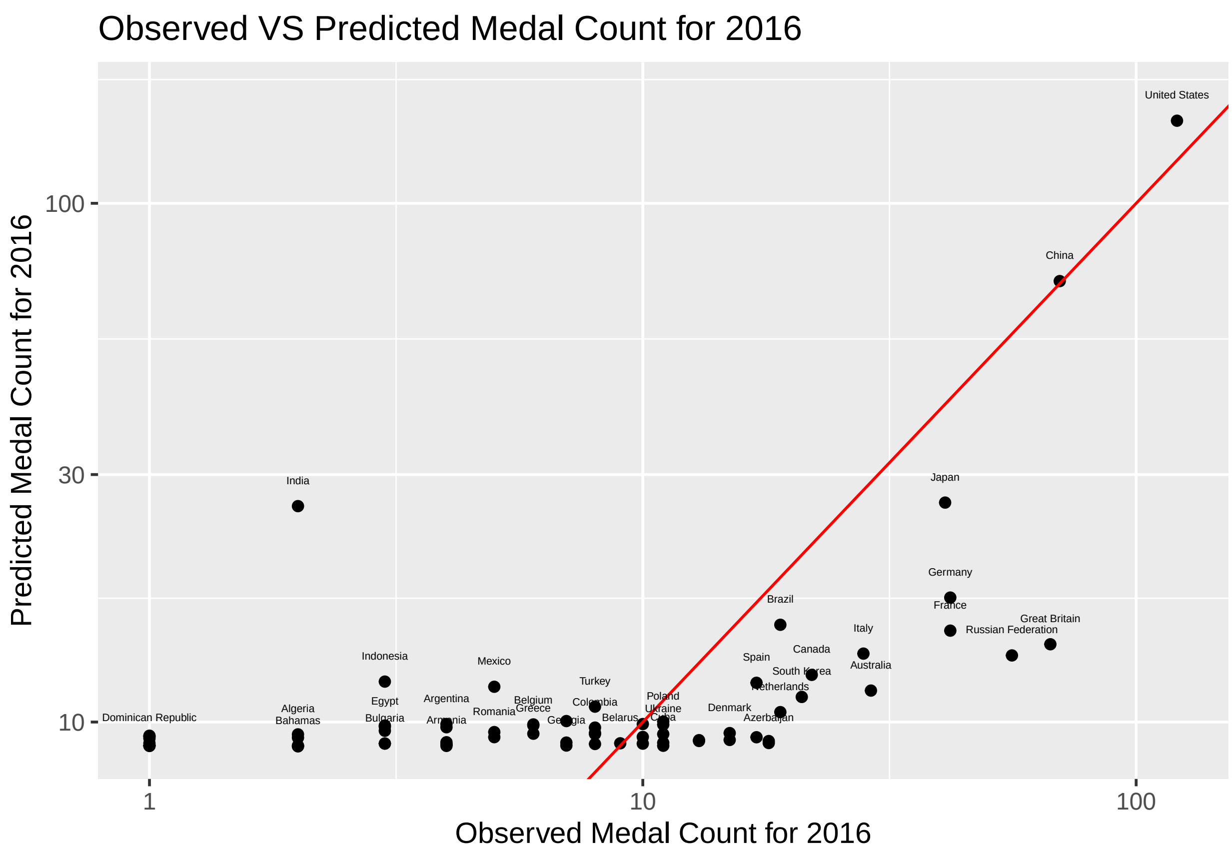
```
#Testing model with prediction
poi_df <- df[,c(2,3)]
poi_predictions= predict(Poisson_Model, newdata = poi_df, type = "response")
df %>%
  mutate(Pred_Medal2016=round(poi_predictions)) %>%
  select(Medal2016,Pred_Medal2016)%>%
  head(10)
```

```
##    Medal2016 Pred_Medal2016
## 1          2              9
## 2          4             10
## 3          4              9
## 4         29             11
## 5         18              9
## 6          2              9
## 7          2              9
## 8          9              9
## 9          6             10
## 10        19             15
```

The plot of predicted vs observed value of Medal2016 will again give better insights. The regression line is also fitted in the plot. To see the data points better the axes are log transformed.

The plot shows that the data points are again quite away from the regression line. There are very few data points which are lying on it. The model is performing quite poorly. The slope of the regression line is also more than the previous models.

```
#Plot with log transformed axes
ggplot(data=df, aes(y=poi_predictions, x=Medal2016,label = Country)) +
  geom_point() +
  ggtitle("Observed VS Predicted Medal Count for 2016")+
  scale_y_continuous(trans='log10')+
  scale_x_continuous(trans='log10')+
  ylab("Predicted Medal Count for 2016")+
  xlab("Observed Medal Count for 2016")+
  geom_abline(slope = 1, intercept = 0, col = 'red')+
  geom_text(size=1.5,nudge_y = 0.05,  check_overlap = TRUE)
```



We again find outliers by calculating 75th quantile and interquantile range.

we have 7 outliers with Great Britain having the highest absolute error:

```
#Quantile
quantile(abs(poi_predictions - df$Medal2016))
```

```
##          0%         25%         50%         75%        100%
##  0.08789888  3.26988408  6.08625971  8.33430151 52.87573293
```

```
#Outliers of Absolute errors:
#Values Beyond [75th Quantile +(1.5*IQR)]

8.33 + 1.5*IQR(abs(poi_predictions - df$Medal2016))
```

```
## [1] 15.92663
```

```
#Countries with Absolute Error Outliers
df %>%
  select(Country,Medal2016) %>%
  mutate(Absolute_Error = abs(poi_predictions - Medal2016)) %>%
  mutate(poi_predictions=round(poi_predictions)) %>%
  select(Country,Medal2016,poi_predictions,Absolute_Error) %>%
  filter(Absolute_Error>15.93)%>%
  arrange(desc(Absolute_Error))
```

```
##              Country Medal2016 poi_predictions Absolute_Error
## 1      Great Britain        67              14       52.87573
## 2 Russian Federation        56              13       42.55866
## 3             France        42              15       26.99578
## 4            Germany        42              17       24.62374
## 5              India         2              26       24.07765
## 6      United States       121             144       23.29245
## 7          Australia        29              11       17.50083
```

Mean absolute error and RMSE of the model:

```r
#Mean Absolute Error and Root Mean Squared Error
mae(df$Medal2016,poi_predictions)
```

```
## [1] 7.846745
```

```r
rmse(df$Medal2016,poi_predictions)
```

```
## [1] 11.8049
```

The value of mean absolute error and RMSE is 7.85 and 11.80 respectively. It also has more absolute error outliers than the previous model.

Task 4:

Here we will perform negative binomial regression for prediction of $Medal2016$ count. The library needed to perform it is MASS. First of all, we need to find the optimal value of theta and for that we create a function to calculate log-likelihood.

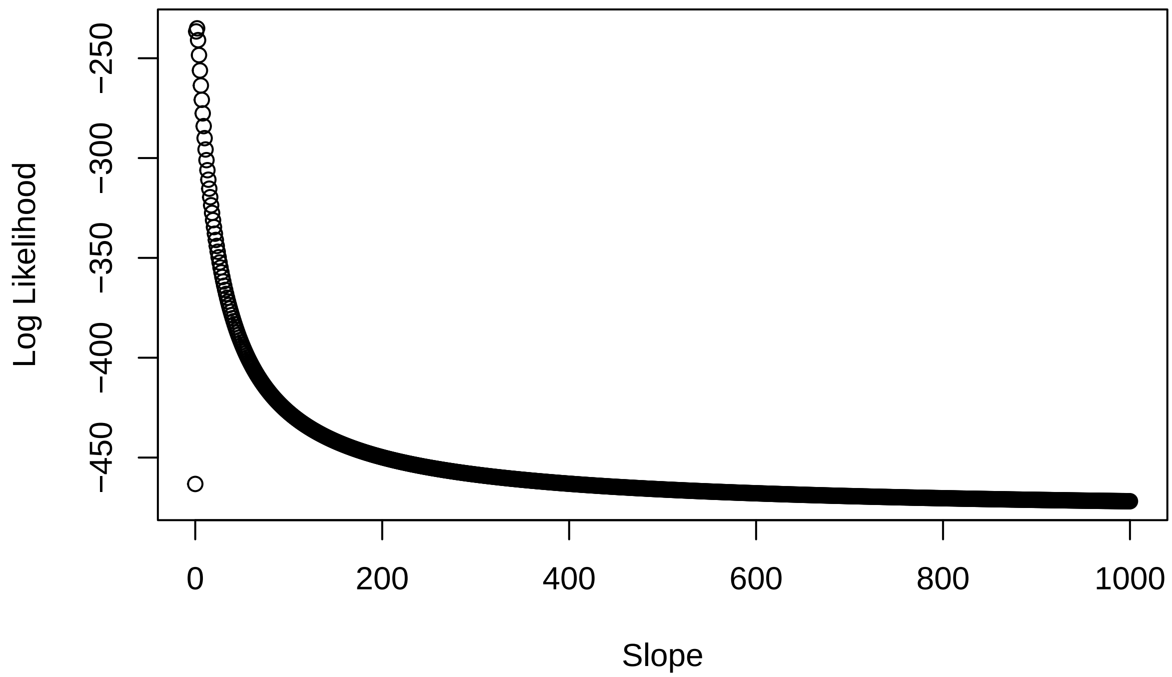
```r
logLikelihood <- function(x){
  #logLikelihood for faithful data, with slope of b1 and intercept of 33
  model_nb = glm(Medal2012 ~ Population + GDP , data= df,family = negative.binomial(theta = x))
  Lglk = logLik(model_nb)
  return(Lglk)
}
```

The function will run for the sequence of theta values ranging from 0.001 to 1000. These values are being stored in an array named Lglk.

```r
theta = seq(from = 0.01, to = 1000, length.out=1000)
Lglk = rep(NA, 1000)
for (i in 1:1000){
  Lglk[i] = logLikelihood(theta[i])
}
```

Plot of all Log Likelihoods:

```r
plot(theta, Lglk, xlab='Slope', ylab='Log Likelihood')
```



To find the optimal value of theta we use the function optim.

```
#print optimal value of theta
nLglk <- function(x){-logLikelihood(x)}
optimise_output = optim(par=1, fn = nLglk)
```

```
## Warning in optim(par = 1, fn = nLglk): one-dimensional optimization by Nelder-Mead is unreliable:
## use "Brent" or optimize() directly
```

```
print(optimise_output$par) #print the optimised slope value
```

```
## [1] 1.54375
```

The optimal value of theta is 1.54 which can now be used to train our model.

```
negbin_model = glm(Medal2012 ~ Population + GDP , data= df,family = negative.binomial(theta = 1.54))
summary(negbin_model)
```

```
##
## Call:
## glm(formula = Medal2012 ~ Population + GDP, family = negative.binomial(theta = 1.54),
##     data = df)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.9663  -1.0331  -0.3451   0.3769   2.8756
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.920e+00  1.251e-01  15.341  < 2e-16 ***
## Population  -5.259e-10  5.770e-10  -0.911    0.365
## GDP          4.460e-04  5.691e-05   7.837 4.33e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.54) family taken to be 1.189339)
##
##     Null deviance: 133.289  on 70  degrees of freedom
## Residual deviance:  72.424  on 68  degrees of freedom
## AIC: 473.98
##
## Number of Fisher Scoring iterations: 24
```

The following observations could be made from the model:

- The regression coefficient of the predictor variable population is -5.259×10^{-10} which is very small with the standard error of 1.252×10^{-10}. The P-value is 0.365 which is quite large ($0.365 \gg 0.05$). This shows that the population of the country is statistically insignificant with regards to country's medal count.
- Also, the regression coefficient of the predictor variable GDP(4.460e-04) is relatively large with the standard error of 5.691×10^{-5}. The P-value is of it is 4.33×10^{-11} which is quite small and is less than 0.05 ($2 \times 10^{-16} \ll 0.05$). It tells that the GDP of a country is statistically significant and does impact country's medal count.
- The intercept is 1.920.

```r
tc = qt(p=0.975, df=68)
#Confidence Interval of population
pop_negbin_ci = summary(negbin_model)$coefficients[2, 1] +
  c(-1,1)*tc*summary(negbin_model)$coefficients[2, 2]
print(pop_negbin_ci)
```

```
## [1] -1.677388e-09  6.255099e-10
```

```r
#Confidence Interval of gdp
gdp_negbin_ci = summary(negbin_model)$coefficients[3, 1] +
  c(-1,1)*tc*summary(negbin_model)$coefficients[3, 2]
print(gdp_negbin_ci)
```

```
## [1] 0.0003324333 0.0005595497
```

The following observation is made from the confidence intervals: The following observation is made from the confidence intervals:

- The confidence interval of the coefficient of the variable population(β_1) shows that 0 lies in its range. This confirms the insignificance of the variable with regards to medal count. when $\beta_1 = 0$, it has no impact on the target variable.

- The confidence interval of the coefficient of the variable gdp(β_2) is positive and 0 does not lie in its range. The values are quite small but it shows some significance. Hence, it does impact our target variable.

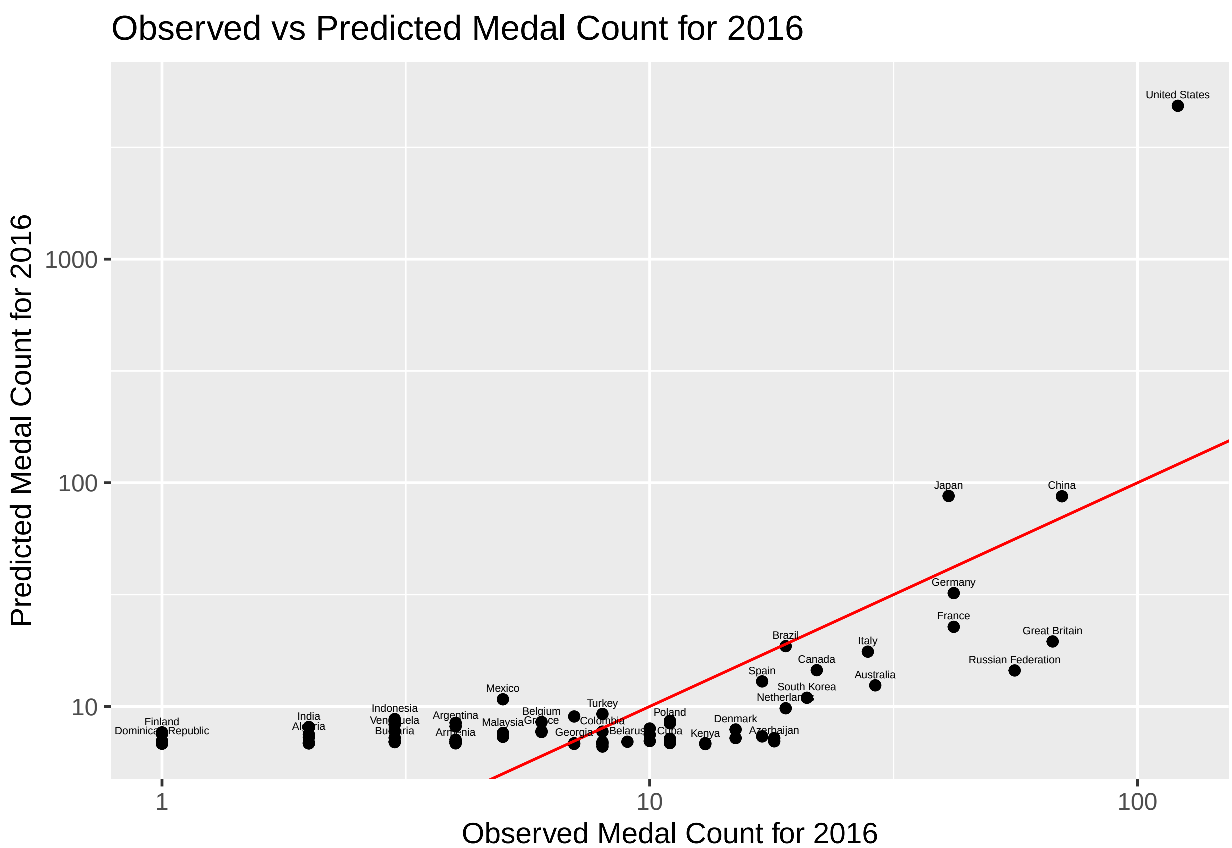
Considering no change in country's GDP and population, the trained model can now be used to predict medal count for the year 2016(Medal2016) using predict function. The first 10 predicted vs observed values of medal count has been shown below.

```
#Testing model with prediction
negbin_df <- df[,c(2,3)]
negbin_predictions= predict(negbin_model, newdata = negbin_df, type = "response")
df %>%
  mutate(Pred_Medal2016=round(negbin_predictions)) %>%
  select(Medal2016,Pred_Medal2016)%>%
  head(10)
```

```
##    Medal2016 Pred_Medal2016
## 1          2              7
## 2          4              8
## 3          4              7
## 4         29             12
## 5         18              7
## 6          2              7
## 7          2              7
## 8          9              7
## 9          6              9
## 10        19             19
```

The plot of predicted vs observed value of Medal2016 will again give better insights. The regression line is also fitted in the plot. To see the data points better the axes are log transformed.

```
#Plot with log transformed axes
ggplot(data=df, aes(y=negbin_predictions, x=Medal2016,label = Country)) +
  geom_point() +
  ggtitle("Observed vs Predicted Medal Count for 2016")+
  scale_y_continuous(trans='log10')+
  scale_x_continuous(trans='log10')+
  ylab("Predicted Medal Count for 2016")+
  xlab("Observed Medal Count for 2016")+
  geom_abline(slope = 1, intercept = 0, col = 'red')+
  geom_text(size=1.5,nudge_y = 0.05,  check_overlap = TRUE)
```



The plot shows that the data points are again quite away from the regression line. There are very few data points which are lying on it. Overall the model is performing decently. We again find outliers by calculating 75th quantile and interquantile range.

we have 7 outliers with United States having extremely large absolute error:

```
#Quantile
quantile(abs(negbin_predictions - df$Medal2016))
```

```
##         0%         25%         50%         75%        100%
##    0.1530467   2.6006691   4.8416825   6.6122735 4728.9396665
```

```
#Outliers of Absolute errors:
#Values Beyond [75th Quantile +(1.5*IQR)]
6.61 + 1.5*IQR(abs(negbin_predictions - df$Medal2016))
```

```
## [1] 12.62741
```

```
#Countries with Absolute Error Outliers
df %>%
  select(Country,Medal2016) %>%
  mutate(Absolute_Error = abs(negbin_predictions - Medal2016)) %>%
  mutate(negbin_predictions=round(negbin_predictions)) %>%
  select(Country,Medal2016,negbin_predictions,Absolute_Error) %>%
  mutate(Absolute_Error=round(Absolute_Error)) %>%
  filter(Absolute_Error>12.62)%>%
  arrange(desc(Absolute_Error))
```

```
##                Country Medal2016 negbin_predictions Absolute_Error
## 1      United States       121               4850           4729
## 2      Great Britain        67                 20             47
## 3              Japan        41                 87             46
## 4 Russian Federation        56                 14             42
## 5             France        42                 23             19
## 6          Australia        29                 12             17
## 7              China        70                 87             17
```

Mean absolute error and RMSE of the model:

```
#Mean Absolute Error and Root Mean Squared Error
mae(df$Medal2016,negbin_predictions)
```

```
## [1] 73.4231
```

```
rmse(df$Medal2016,negbin_predictions)
```

```
## [1] 561.3336
```

The value of mean absolute error and RMSE is 73.42 and 561.33 respectively.

Task 5:

The best model can be selected using several metrics like **Mean absolute error** and **Root Mean Square Error**.The library(Metrics) is used to calculate it.

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

The formula of MAE is given by: $MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t|$ where, e_t is the difference between predicted and observed values.

RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation

The formula of RMSE is given by: $RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2}$ where, e_t is the difference between predicted and observed values.

We can also see the correlation coefficient between predicted and observed values for all the models. $cor()$ function can be used to find it.

```
cor(df$Medal2016,pred)
```

```
## [1] 0.8830621
```

```
cor(df$Medal2016,log_predictions)
```

```
## [1] 0.7111247
```

```r
cor(df$Medal2016,poi_predictions)
```

```
## [1] 0.7996339
```

```r
cor(df$Medal2016,negbin_predictions)
```

```
## [1] 0.6754786
```

The MAE and RMSE of all the models have been calculated in the previous tasks. It has been summarised below:

Model 1: Linear Regression The value of mean absolute error and RMSE is 6.1043 and 9.11 respectively which shows the model is performing decently but better than rest of the models. Correlation between predicted and actual values is 0.88.

Model 2: Linear Regression for log-transformed medal counts The value of mean absolute error and RMSE is 13.70 and 56.62 respectively which shows the model isn't performing better than the previous one. Correlation between predicted and actual values is 0.71.

Model 3: Poisson Regression The value of mean absolute error and RMSE is 7.85 and 11.80 respectively. The values are smaller than than log-transformed medal counts model but the performance is still not good. Also, it has more number of absolute error outliers than Model 2 but the correlation between predicted and actual values is 0.79, which is more than Model 2.

Model 4: Negative Binomial Regression The value of mean absolute error and RMSE is 73.42 and 561.33 respectively. The values are largest compared to all the models. Also, the values of absolute error outliers are largest. Correlation between predicted and actual values is 0.67.

If metrics of all the models are compared, the **Model 1 turns out to be the best one.** It has the the smallest MAE and RMSE value. It could predict our target variable $Medal2016$ with better accuracy compared to rest of the models. Furthermore, Model 1 has least number of absolute error outliers and shows highest correlation between predicted and actual values. Model 4 was trained with optimal value of *theta* but it was the least performing model with largest MAE and RMSE values. Model 1 still needs to be improved as its MAE and RMSE values are significant. It is only best compared to rest of the models. The plot shows that the data points are quite away from the regression line and there is significant difference between the predicted and observed values. Hence, none of the models can be said to have good accuracy.