Assessed Practical I: Predicting the Olympic Games

Shreyansh Parihar, Student ID: 201684933

Term 2 2023

Task1

The goal of this task is to create a linear regression model using Population and GDP as input and medal count in 2012 as output from the data given and determine modal prediction for medal count in 2016. In the below code we have stored the given data in variable called `medal_gdp_data` and stored necessary column vectors in it's own variable for easy access.

```
medal_gdp_data <- read.csv("medal_pop_gdp_data_statlearn.csv")
gdp <- medal_gdp_data$GDP
medal_2012 <- medal_gdp_data$Medal2012
medal_2016 <- medal_gdp_data$Medal2016
population <- medal_gdp_data$Population
country <- medal_gdp_data$Country

medal_2016_df <- data.frame(GDP = gdp,Population = population)
```

To perform multiple linear regression we have used `glm()` function with column name Medal2012 as input and columns in format `GDP+Population` as input.

```
model_2012 <- glm(Medal2012 ~ GDP+Population, data = medal_gdp_data, family = gaussian)
```

Once the model is created we can see the result using `summary()` to see the details of model. In summary we get many information, but here we are printing just the necessary information which are the estimated coefficients and their standard errors for GDP and Population, and AIC(Akaike Information Criterion) value which tells us about model performance. The close the AIC value to 0 the better the model is.

```
#Displaying Coefficients and AIC value of model
summary(model_2012)$coefficients[, 1:2]
```

```
##                 Estimate   Std. Error
## (Intercept) 6.076092e+00 1.499954e+00
## GDP         7.564081e-03 7.325406e-04
## Population  5.246750e-09 7.192637e-09
```

```
cat("\nAIC value: ", AIC(model_2012),"\n")
```

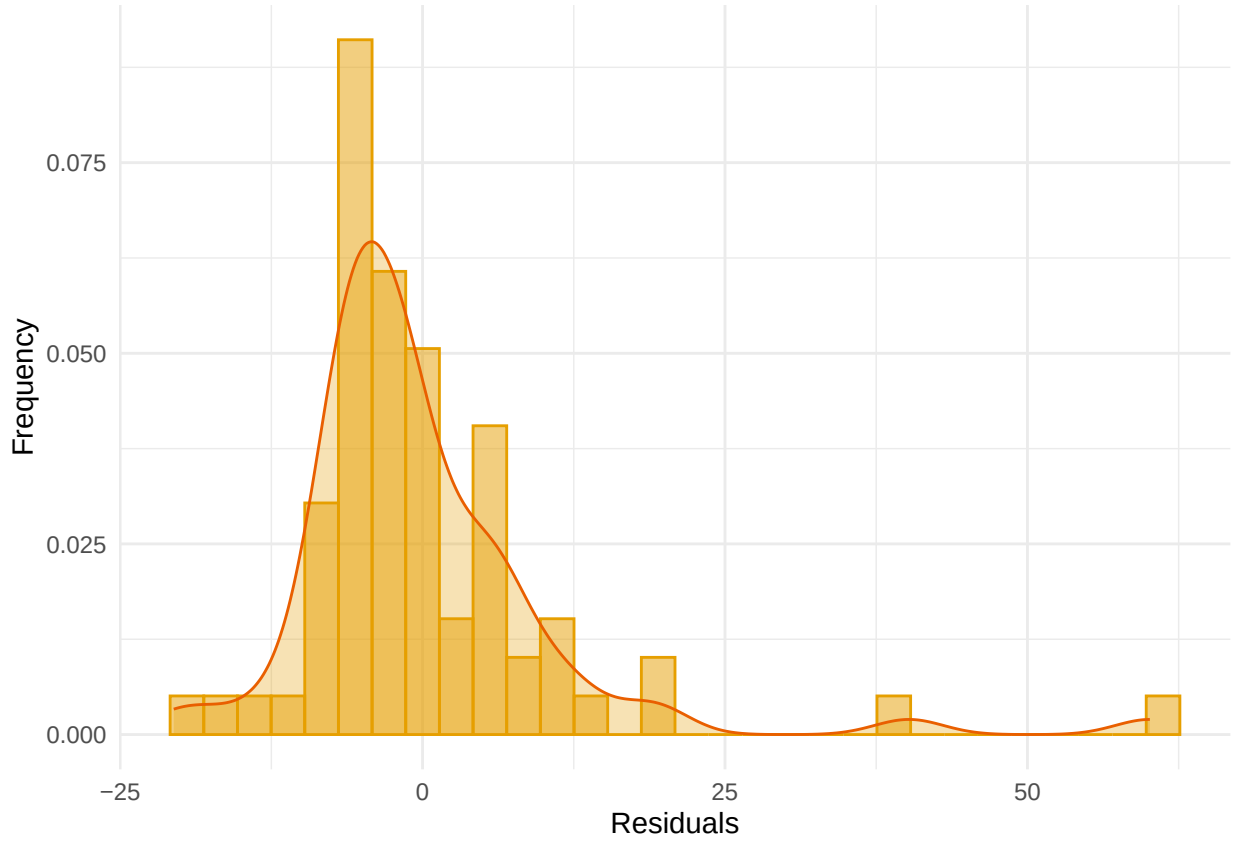
```
##
## AIC value:  553.187
```

From below code we can see histogram of the residuals and observe that our model follows normal distribution roughly and there are some outliers which if we can deal with them, then maybe our model fits more accurately.

```
#loading necessary library for graphs
library(ggplot2)
```

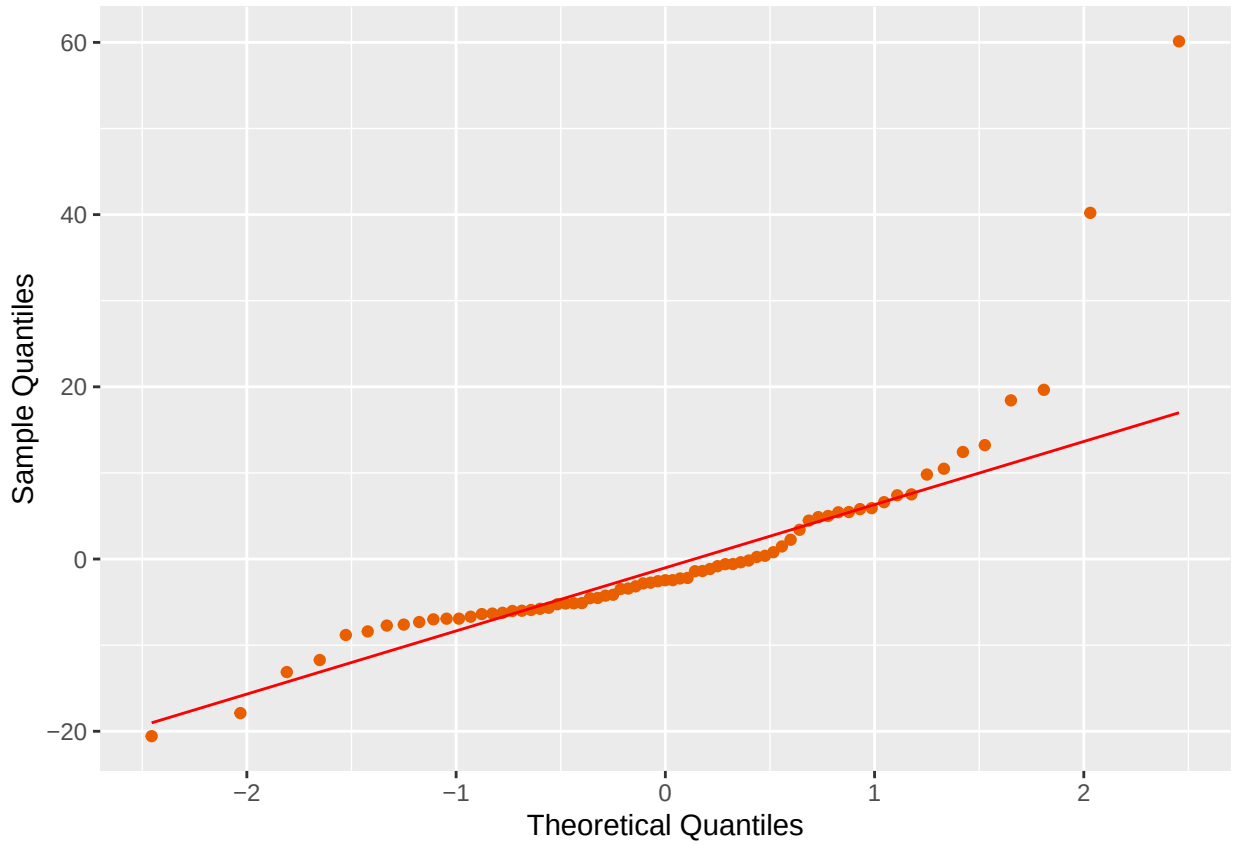
```
# Histogram of residuals
ggplot(data = medal_gdp_data, aes(x = model_2012$residuals)) +
  geom_histogram(aes(y=..density..), colour="#E69F00", fill="#E69F00", alpha=0.5) +
  geom_density(alpha=.3, fill="#E69F00", color="#E95F00") +
  xlab("Residuals") +
  ylab("Frequency") +
  scale_color_brewer(palette="Accent") +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The below QQ plot for the residuals from the model with Population and GDP as inputs also shows that the residuals are approximately normally distributed, as the points on the plot follow a roughly straight line except for some outliers.

```
# QQ plot of residuals
ggplot(data = medal_gdp_data, aes(sample = model_2012$residuals)) +
  stat_qq(color='#E95F00') +
  stat_qq_line(color='red') +
  xlab("Theoretical Quantiles") +
  ylab("Sample Quantiles")
```

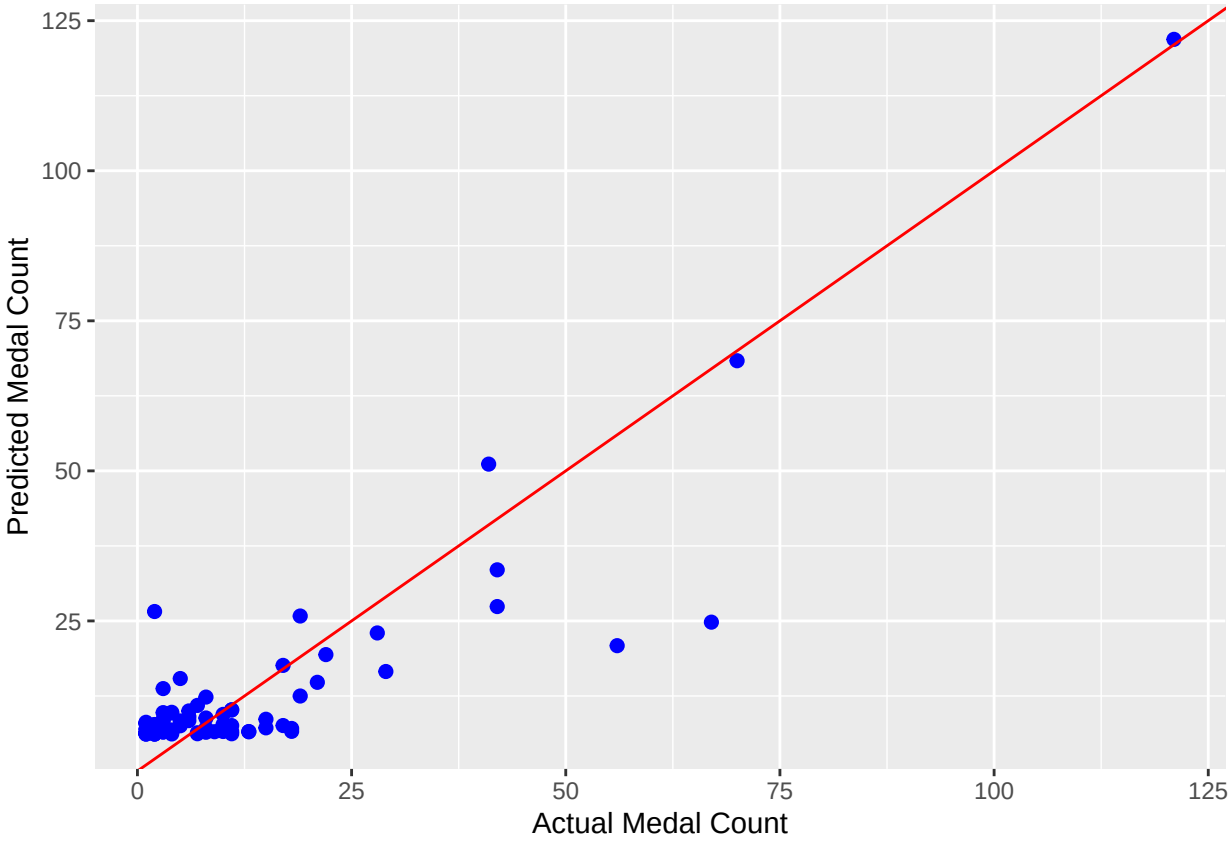


Now to assess the model prediction in comparison to actual 2016 data we first need to see the predicted value from using `predict()` to get predicted medals.

```
predict_model1 <- predict(model_2012,data.frame(GDP = gdp,Population = population))
```

Below code shows the graph between actual medal count value in 2016 vs the predicted value to see how well the prediction is done. And we can see that the there are more points which lies closer to diagonal line (y = x) and few points quite far from that line,showing model performed fairly well.

```
# Scatter plot of actual vs. predicted medal count
ggplot(data = medal_gdp_data, aes(x = Medal2016, y = predict_model1)) +
  geom_point(size = 2, color = "blue") +
  geom_abline(intercept = 0, slope = 1, linetype = "solid", color = "red") +
  xlab("Actual Medal Count") +
  ylab("Predicted Medal Count")
```



We have also used coefficient of determination also called as R squared method which gives us how well the regression line approximates the actual medal 2016 data.It is given by :

$$R^2 = 1 - SSR/SST$$

Where SSR is sum squared regression which is calculated by

$$SSR = \sum((Predicted\ Medal\ Count\ in\ 2016)_i - (Actual\ Medal\ Count\ in\ 2016)_i)^2$$

and SST is total sum of squares which is calculated by

$$SST = \sum(Actual\ Medal\ Count\ in\ 2016)_i - Mean\ of\ Actual\ Medal\ Count\ in\ 2016)_i)^2$$

The more closer the value of R^2 to 1 the better the model is. By applying above in equation in our code we get the value of R^2 to be 0.7788646 which shows that model did considerably good in predicting medal values

```
rsquare_1 <- 1-
  (sum((predict_model1 - medal_gdp_data$Medal2016) ^ 2)/
     sum((medal_gdp_data$Medal2016 - mean(medal_gdp_data$Medal2016))^2))
print(rsquare_1)
```

```
## [1] 0.7788646
```

Task2

In this task, instead of normal inputs we will take log inputs and perform linear regression and compare performance of this model with the model in task 1. Below code is for storing the log values of GDP and Population in a new dataframe.

```
log_gdp <- log(medal_gdp_data$GDP)
log_population <- log(medal_gdp_data$Population)
log_medal_DataFrame <- data.frame(log_gdp,log_population,medal_2012,country)
```

Once we have logged inputs we can use that and create our linear regression model using glm() and see the estimated coefficients and their standard errors for GDP and Population, and AIC value.

```
#Performing linear regression model on log() inputs
model_2012_log <- glm(medal_2012 ~ log_gdp+log_population, data = log_medal_DataFrame)

#Displaying Coefficients and AIC value of model
summary(model_2012_log)$coefficients[, 1:2]
```

```
##                  Estimate Std. Error
## (Intercept)    -49.363193  24.614990
## log_gdp          5.544723   1.543123
## log_population   1.982957   1.789906
```

```
cat("\nAIC value: ", AIC(model_2012_log),"\n")
```

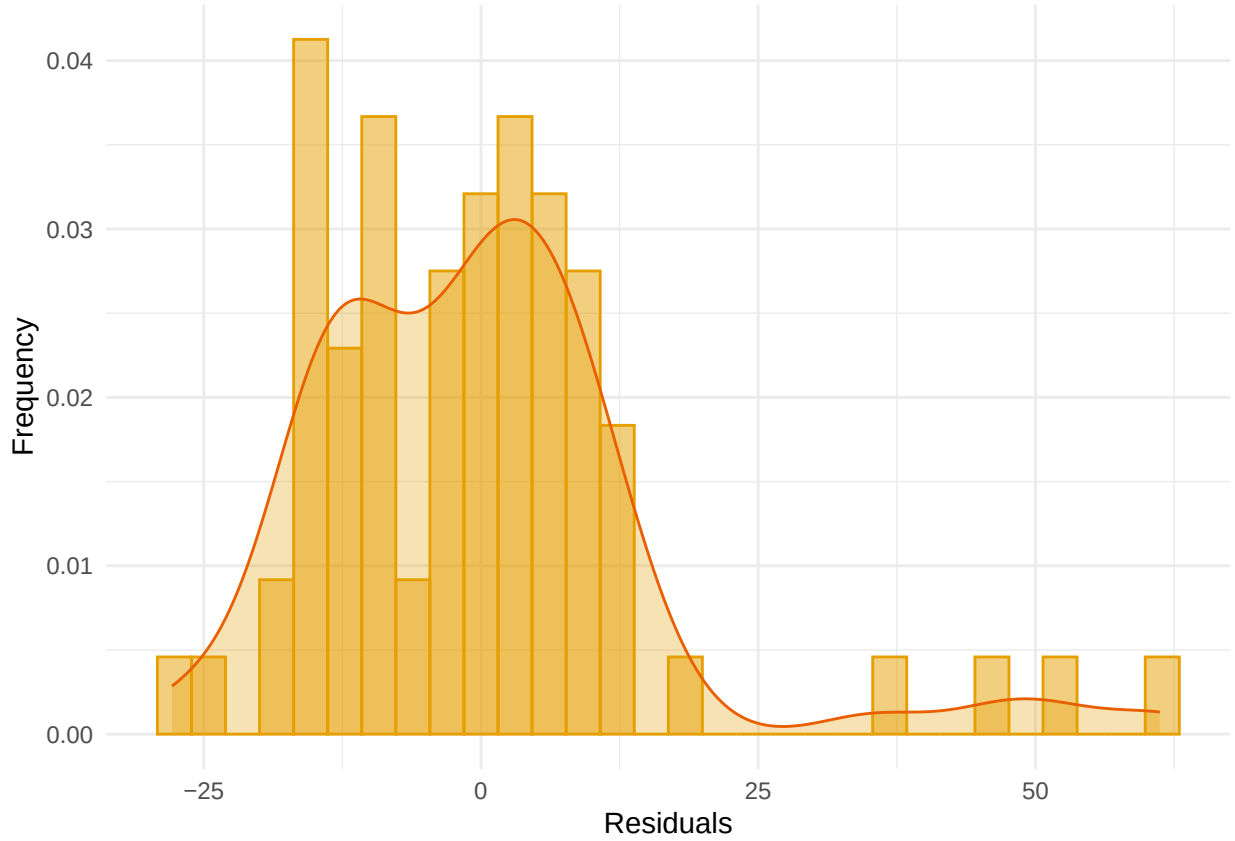
```
##
## AIC value:  599.7297
```

From the above code we get AIC value as 599.73 for this model suggesting that model in Task-1 was better than this model as it had the AIC value of 553.187 which is lower than our current model.

We can see the residual distribution of this model from below code and from this graph it is evident that this model may follow normal distribution, in comparison to model in Task-1 this model shows weaker sign of it being following normal distribution.

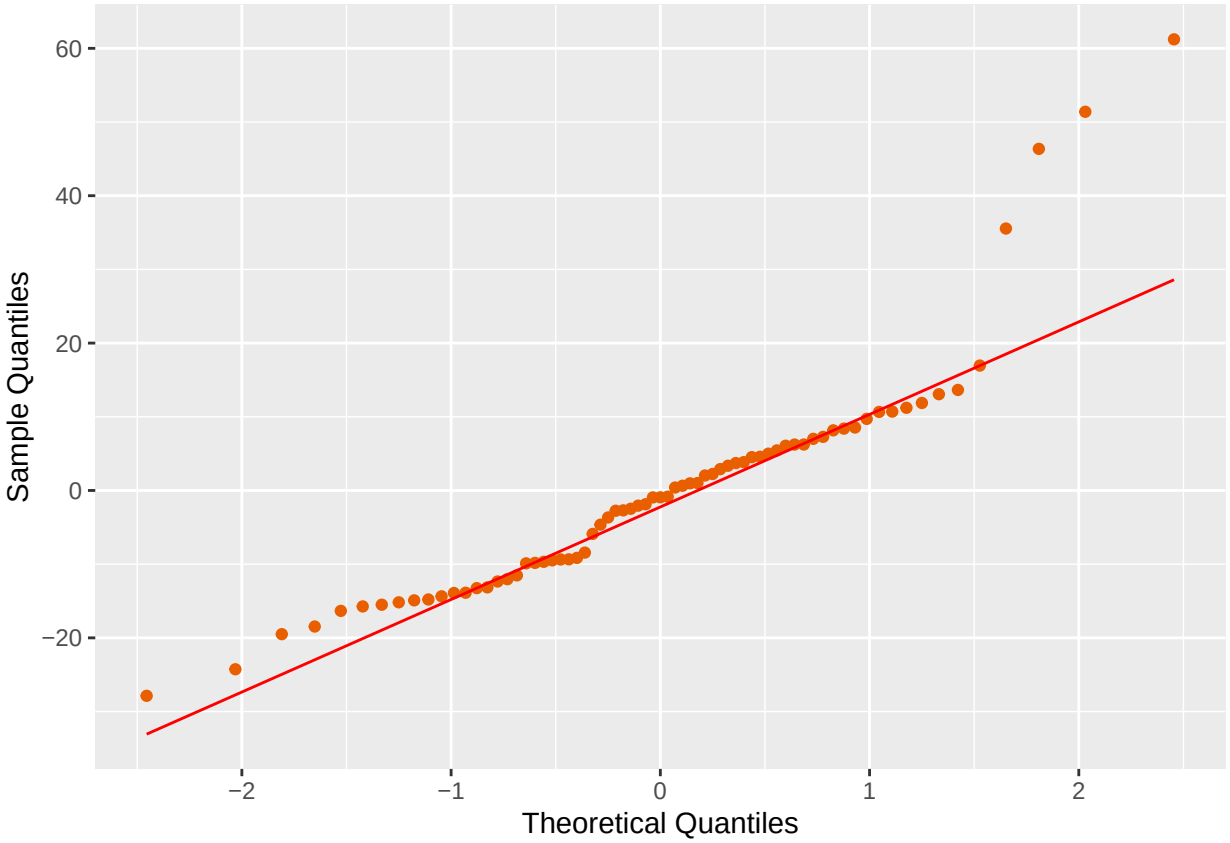
```
# Histogram of residuals
ggplot(data = medal_gdp_data, aes(x = model_2012_log$residuals)) +
  geom_histogram(aes(y=..density..), colour="#E69F00", fill="#E69F00",position="identity", alpha=0.5) +
  geom_density(alpha=.3, fill="#E69F00", color="#E95F00") +
  xlab("Residuals") +
  ylab("Frequency") +
  scale_color_brewer(palette="Accent") +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Let's look at the Q-Q plot to confirm our claim from below code.

```
# QQ plot of residuals
ggplot(data = medal_gdp_data, aes(sample = model_2012_log$residuals)) +
  stat_qq(color='#E95F00') +
  stat_qq_line(color='red') +
  xlab("Theoretical Quantiles") +
  ylab("Sample Quantiles")
```



From the above plot we observed that the residuals roughly follow straight line suggesting they are approximately normally distributed.

In the below code we can predict the values from this model using `predict()` function and then calculate the R^2 value to compare to model in Task-1

```
medal_2016_df_log <- data.frame(GDP = log_gdp,Population = log_population)
predict_model2 <- predict(model_2012_log,medal_2016_df_log)
rsquare_2 <- 1-
  (sum((predict_model2 - medal_gdp_data$Medal2016) ^ 2)/
     sum((medal_gdp_data$Medal2016 - mean(medal_gdp_data$Medal2016))^2))
print(rsquare_2)
```

```
## [1] 0.3924613
```

We observed that the R^2 value is far smaller than what we got in task-1 confirming once again that the model in Task-1 was far better than this model as we got 0.7788 which is much closer to 1 in comparison value 0.3925 form our current model.

Task3

For this task first we determine the best k value for K-means clustering on log-input data. It can be done using NbClust library in R as done in below code.

We chose Nbclust as it is used to determine the optimal number of clusters in a dataset using various clustering algorithms and clustering validity indices. The clustering validity indices are used to evaluate the quality of the clustering results and to compare different clustering solutions.By default it will compute a set of 30 clustering validity indices for each number of clusters (k) ranging from 1 to a specified maximum.These

indices include popular metrics such as silhouette width, Dunn index, and Calinski-Harabasz index, as well as many others.

Since this method uses combination of many techniques to get optimum K value we chose this method over others. Using this method we got optimum K value as 3.

Once we have optimum k value we can use k-means model for clustering the log input data using `kmeans()` function with parameter `centers = 3` for 3 clusters

```
#Fixing seed value so that k-means evaluate same clusters
set.seed(123)
kmean_model <- kmeans(medal_2016_df_log, centers = 3)
```

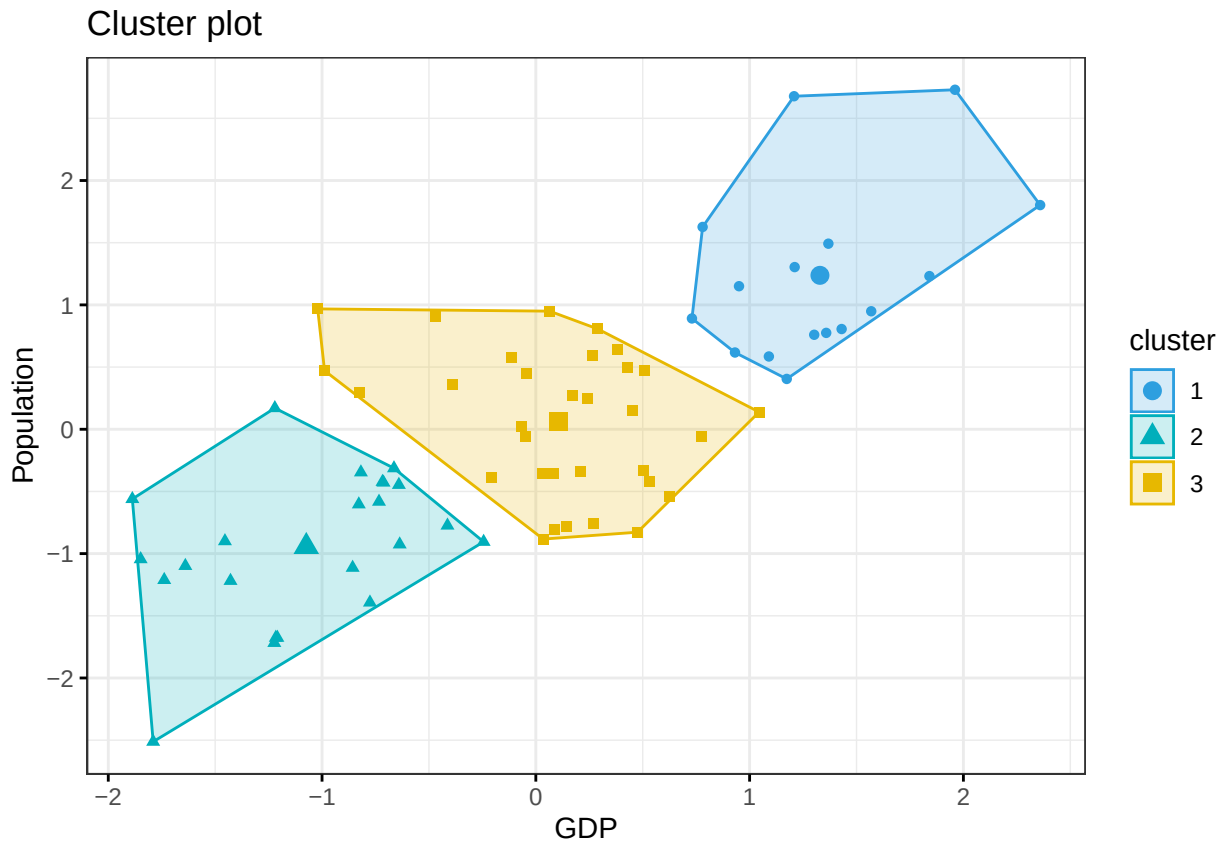
We can visualize the clusters, for this we are using R library called factoextra and using it's function `fviz_cluster()`.

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.2.3
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_cluster(kmean_model, data = medal_2016_df_log,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Now we have to perform linear regression on on every cluster for that we have created a new dataframe which

will also store predicted 2016 medal count values when we do the prediction.

```
modified_medal_2016_df <- data.frame(GDP = medal_2016_df$GDP,
                                     Population = medal_2016_df$Population,
                                     cluster = kmean_model$cluster,medal_2012,
                                     Actual_Medal2016=medal_2016,Medal2016=NA)
```

We have to divide our data as per clusters and and perform linear regression on each cluster separately. For this we have used for loop to iterate through every cluster and within each loop we divide data as per their clusters and only performing linear regression on that divided data and storing the combined result in a single dataframe which we have created.

```
#looping through every cluster from 1 to 3
for (i in 1:max(kmean_model$cluster)) {
  #dividing data as per clusters assignment
  cluster_i <- subset(modified_medal_2016_df, kmean_model$cluster == i)

  #performing linear regression
  model_i <- glm(medal_2012 ~ GDP + Population, data = cluster_i)

  #Printing linear regression model AIC value for each clusters
  cat("\n---------------------------------------")
  cat("\nAIC value in cluster ",i)
  cat(": ",AIC(model_i))
  cat("\n---------------------------------------\n")

  #Storing the predicted value in respective data rows for further model evaluation
  modified_medal_2016_df$Medal2016[modified_medal_2016_df$cluster == i] <- predict(model_i, cluster_i)
}
```

```
##
## ---------------------------------------
## AIC value in cluster  1:  149.9607
## ---------------------------------------
##
## ---------------------------------------
## AIC value in cluster  2:  130.3958
## ---------------------------------------
##
## ---------------------------------------
## AIC value in cluster  3:  215.0628
## ---------------------------------------
```

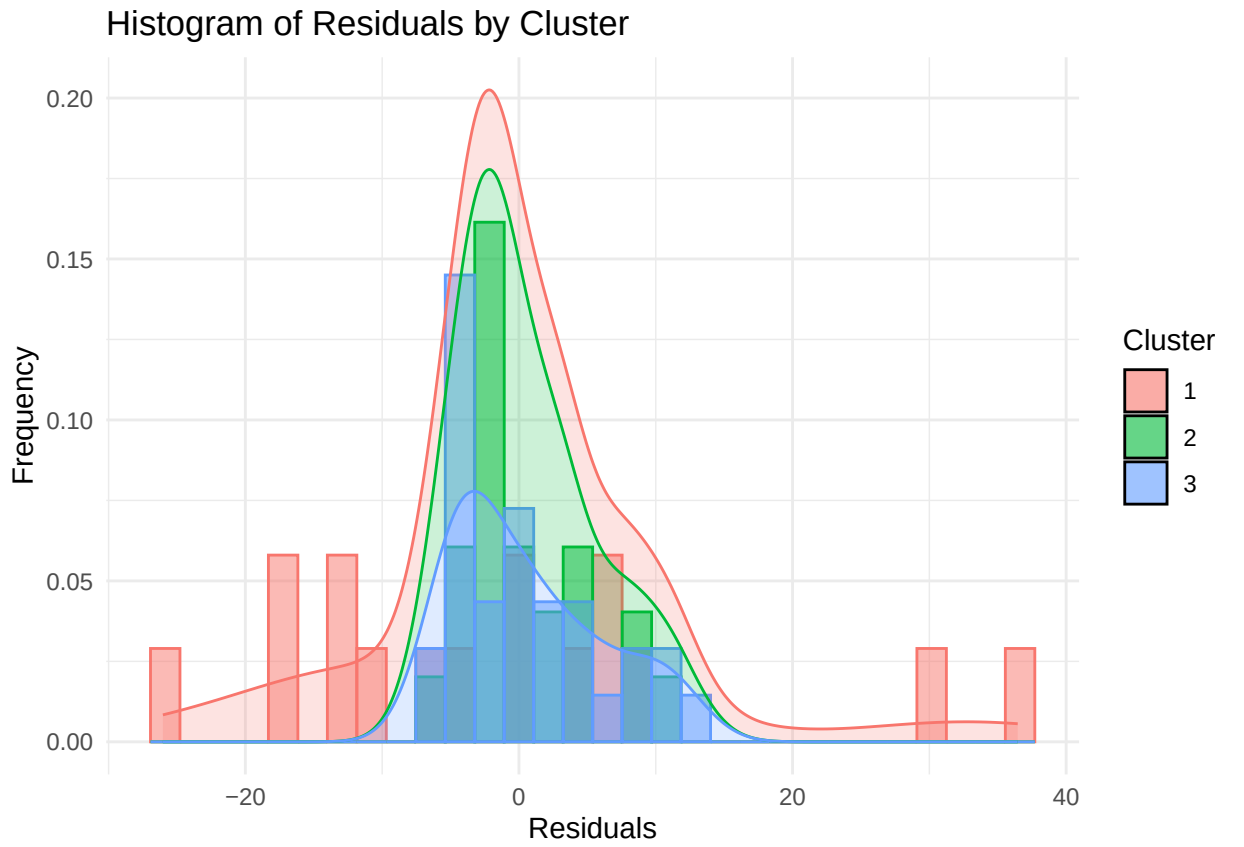
Let's visualize histogram for residual for all 3 clusters using below code.Before visualizing we need to calculate residual manually as difference of actual 2016 medal count and predicted count, that is done in `resid` variable.

```
#Calculating residual
resid <- modified_medal_2016_df$Actual_Medal2016 - modified_medal_2016_df$Medal2016

# Histogram of residuals from each linear regression model for each cluster
ggplot(modified_medal_2016_df, aes(x = resid, fill = as.factor(cluster), colour = as.factor(cluster))) +
  geom_histogram(aes(y=..density..),bins = 30, alpha = 0.5, position = "identity") +
  geom_density(alpha=0.2, position = "stack") +
  xlab("Residuals") +
  ylab("Frequency") +
  ggtitle("Histogram of Residuals by Cluster") +
  labs(fill = "Cluster")+
```

```
  guides(col = FALSE) +
  theme_minimal()
```

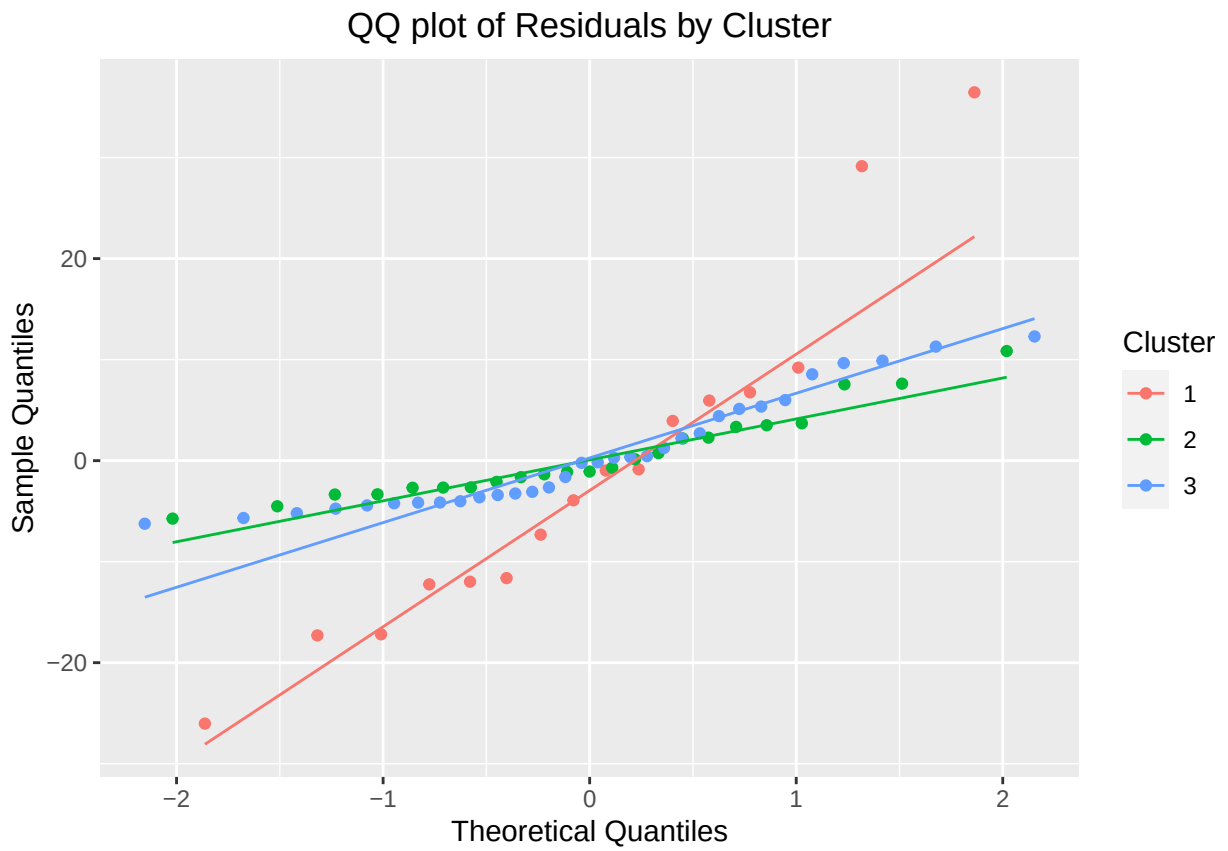
```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
```



Form above graph we might say that residuals from all 3 models from 3 clusters follow normal distribution.

We can also Q-Q plot for the same (see below code), and we can say that approximately, the 3 models do follow normal distribution as the residual follows staight line.

```
# QQ plot of residuals from each linear regression model for each cluster
Cluster <- as.factor(modified_medal_2016_df$cluster)
ggplot(modified_medal_2016_df, aes(sample = resid, color = Cluster)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Sample Quantiles") +
  ggtitle("QQ plot of Residuals by Cluster") +
  theme(plot.title = element_text(hjust = 0.5))
```



Once we have all the predicted medal count values from each cluster's linear regression model we can calculate R^2 to get the overall model performance as shown in below code.

```
rsquare_3 <- 1-
  (sum((modified_medal_2016_df$Medal2016 - medal_gdp_data$Medal2016) ^ 2)
   /sum((medal_gdp_data$Medal2016 - mean(medal_gdp_data$Medal2016))^2))
print(rsquare_3)
```

```
## [1] 0.7954947
```

The benefit of clustering the data and fitting separate models for each cluster is that we can capture different relationships between the input variables and the output variable within each cluster. For example, countries with high population and high GDP may have a different relationship with medal count than countries with low population and low GDP. By fitting separate models for each cluster, we can better capture these different relationship in the data.

Task4

To calculate the probability of UK winning at least 1 medal using linear regression model from task-2, we will

1. Calculate the predicted medal count for the UK using the linear regression model from Task-2

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Calculate the residuals for the UK
gb_preds <- predict(model_2012_log,
                    newdata = data.frame(Population = log(62262000), GDP = log(2431.59)),
                    se.fit = TRUE)$fit
```

2. Calculate the residuals for Great Britain by subtracting the predicted values from the actual values in the dataset.

```
gb_resid <- medal_gdp_data$Medal2016[medal_gdp_data$Country == "Great Britain"] - gb_preds
```

3. Calculate the mean and standard deviation of the residuals.

```
resid_mean <- mean(gb_resid)
resid_sd <- sd(gb_resid)
```

4. As we saw in task-2 that the residual follows normal distribution we will use the cumulative distribution function (CDF) of the normal distribution to calculate the probability of Great Britain winning at least one medal.

```
prob <- pnorm(0, mean = resid_mean, sd = resid_sd, lower.tail = FALSE)
prob
```

```
## [1] 0.9999901
```

We get the probability of 0.99 which is almost equal to 1, thus we can conclude that UK has 100% chance of getting at least 1 medal.

Task5

Based on the evaluation metrics, it can be observed that Task-3 model (K-means clustering and linear regression for each cluster) outperforms the other two models in terms of R-squared as well as in terms of AIC score.

```
## [1] "------------------------------------------------------------"
## Task-1 model Rsquare:  0.7788646
## Task-1 model AIC Score:  553.187
## [1] "------------------------------------------------------------"
## Task-2 model Rsquare:  0.3924613
## Task-2 model AIC Score:  599.7297
## [1] "------------------------------------------------------------"
## Task-3 model Rsquare:  0.7954947
## Task-3 model overall AIC Score: 499.4193
## (AIC_cluster1 + AIC_cluster2 + AIC_cluster3 + 2k, where k=2)
## [1] "------------------------------------------------------------"
```

Below code shows clear visualization for AIC values and R^2 value for all 3 models.

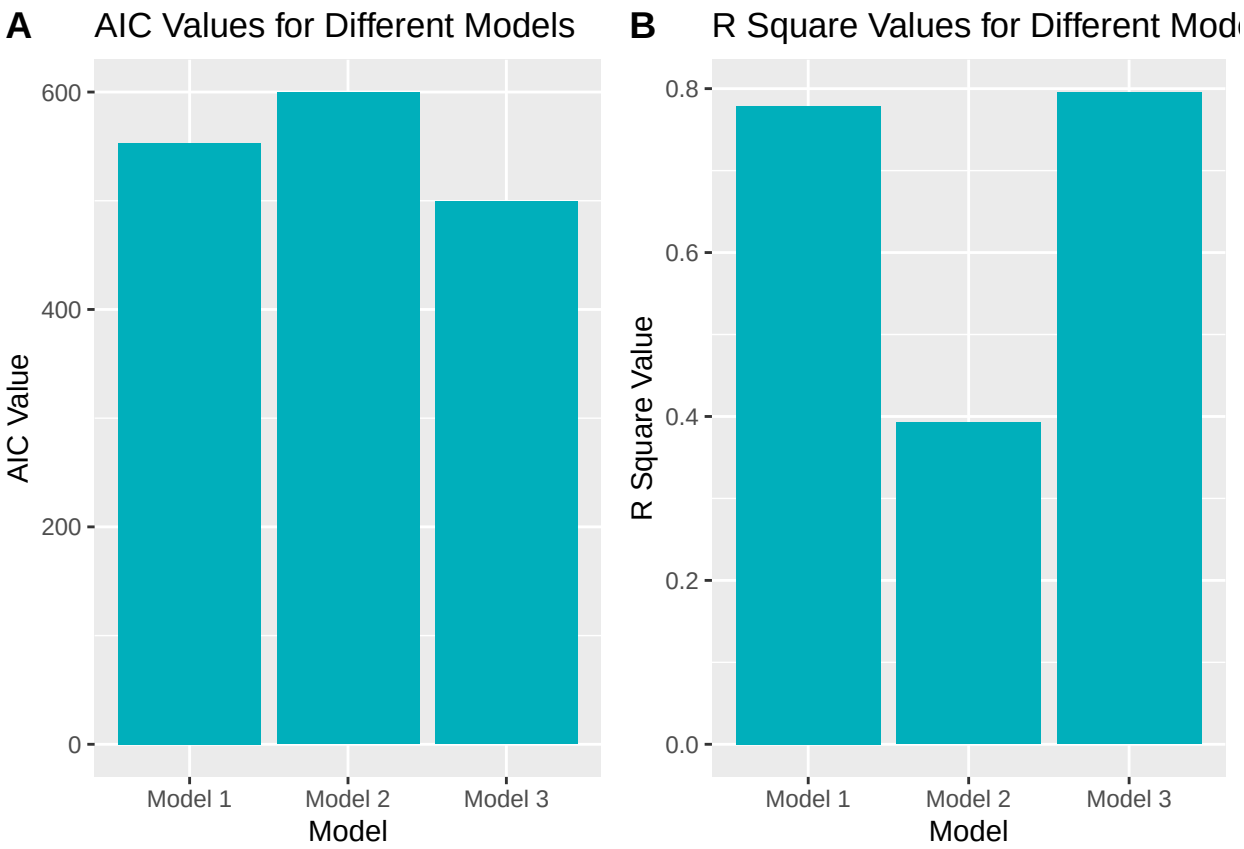
```r
#Library for arranging plot in single area
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.2.3
```

```r
# Bar chart of AIC values
aic_values <- c(AIC(model_2012), AIC(model_2012_log), 499.4193)
models <- c("Model 1", "Model 2", "Model 3")
data_aic <- data.frame(models, aic_values)
aic_plot <- ggplot(data_aic, aes(x = models, y = aic_values)) +
  geom_bar(stat = "identity", fill = "#00AFBB") +
  labs(title = "AIC Values for Different Models",
       x = "Model",
       y = "AIC Value")

# Bar chart of AIC values
rsquare_values <- c(rsquare_1, rsquare_2, rsquare_3)
models <- c("Model 1", "Model 2", "Model 3")
data_rsquare <- data.frame(models, rsquare_values)
rsquare_plot <- ggplot(data_rsquare, aes(x = models, y = rsquare_values)) +
  geom_bar(stat = "identity", fill = "#00AFBB") +
  labs(title = "R Square Values for Different Models",
       x = "Model",
       y = "R Square Value")

ggarrange(aic_plot, rsquare_plot,
          labels = c("A", "B"),
          ncol = 2, nrow = 1)
```



The Task-3 model is preferred over the other models as it takes into account the potential non-linear relationship between the inputs and the output, which is not captured by the linear models in Task-1 and Task-2. The K-means algorithm allows us to partition the data into clusters and create linear regression models for each cluster, which can capture the unique relationships within each cluster. This approach can be particularly beneficial when dealing with complex data that may have non-linear relationships and/or interactions between the variables.

In conclusion, based on the evaluation metrics and reasoning discussed above, the Task-3 model (K-means clustering and linear regression for each cluster) is the preferred model to accurately predict the medal count.