Predicting Life Expectancy of various countries: A Predictive Model Building using Linear Regression

Agniva Konar Archita Biswas Shrestha De Tannishtha Sen

02/11/2021

Including necessary packages

```
library(readxl)
library(qpcR)
library(car)
library(carData)
library(nlme)
library(lmtest)
library(BSDA)
library(MASS)
library(ROCR)
library(rmarkdown)
library(pls)
library(psych)
library(Metrics)
```

Importing the dataset and dropping unnecessary columns.

```
dataset <- read_excel(file.choose())
df=dataset[,-c(1,2,3,4,5,6,9,12,15,18,20,22,25,27,29,31,33,35)]
dim(df)
```

```
## [1] 2938   19
```

Now renaming all the covariates and the response. Then creating a dataframe “df1” using the 18 variables and the response.

```
y=df$`Life expectancy(new)` #response
#covariates
x1=df$`Adult Mortality(new)`
x2=df$`infant deaths`
x3=df$`Alcohol(new)`
x4=df$`percentage expenditure`
x5=df$`Hepatitis B(new)`
x6=df$Measles
x7=df$`BMI(new)`
x8=df$`under-five deaths`
x9=df$`Polio(new)`
x10=df$`Total expenditure(new)`
x11=df$`Diptheria(new)`
x12=df$`HIV/AIDS`
x13=df$`GDP(new)`
x14=df$`population(new)`
x15=df$`thinness 1-19 years(new)`
x16=df$`thinness 5-9 years(new)`
x17=df$`Income composition of resources(new)`
x18=df$`Schooling(new)`

df1 <- data.frame(y,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18)
```

Next, splitting the dataset into two datasets “train” and “test”. Train contains 75% of the original dataset and test contains rest of it. All kinds of data manipulation and model building is done on the “train” dataset.

```
set.seed(123)
sample <- floor(0.75 * nrow(df1))
train_ind <- sample(seq_len(nrow(df1)), size = sample)
train <- df1[train_ind, ]
test <- df1[-train_ind, ]
dim(train)
```

```
## [1] 2203   19
```

```
dim(test)
```

```
## [1] 735  19
```

Now a primary model is created , named “Life_model” using “train” dataset.

```
Life_model=lm(y~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18, data=train)
summary(Life_model)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.1459  -2.2697  -0.0453   2.3430  16.3095
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.423e+01  6.717e-01  80.735  < 2e-16 ***
## x1          -1.991e-02  9.207e-04 -21.629  < 2e-16 ***
## x2           1.011e-01  1.015e-02   9.964  < 2e-16 ***
## x3           1.347e-01  2.796e-02   4.819 1.54e-06 ***
## x4           1.109e-04  1.032e-04   1.074  0.28287
## x5          -1.484e-02  4.635e-03  -3.201  0.00139 **
## x6          -1.931e-05  9.524e-06  -2.028  0.04272 *
## x7           3.893e-02  5.718e-03   6.809 1.27e-11 ***
## x8          -7.638e-02  7.436e-03 -10.271  < 2e-16 ***
## x9           3.000e-02  5.312e-03   5.647 1.84e-08 ***
## x10          1.161e-01  3.871e-02   2.999  0.00274 **
## x11          4.044e-02  5.601e-03   7.221 7.12e-13 ***
## x12         -4.597e-01  2.012e-02 -22.845  < 2e-16 ***
## x13          4.367e-05  1.592e-05   2.742  0.00615 **
## x14          2.170e-09  2.276e-09   0.953  0.34048
## x15         -7.105e-02  5.877e-02  -1.209  0.22680
## x16         -5.236e-03  5.825e-02  -0.090  0.92838
## x17          6.288e+00  7.471e-01   8.417  < 2e-16 ***
## x18          6.603e-01  4.906e-02  13.460  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.127 on 2184 degrees of freedom
## Multiple R-squared:  0.815,  Adjusted R-squared:  0.8135
## F-statistic: 534.5 on 18 and 2184 DF,  p-value: < 2.2e-16
```

Adjusted r square of the created model is 0.8135 which suggests that our primary model is quite efficient. But this efficiency can be improved if we use a cleansed dataset.This can be done by removing the influential observations, outliers and high leverage values from the dataset.

Now checking for influential observation, high leverage points and outliers in the dataset and we have to remove them tactically from the dataset.

```
### cooks distance ###

cook = cooks.distance(Life_model)
c = cook[cook>(4/2203)]
length(c)
```

```
## [1] 145
```

```
# Influential observations

### Studentized residual###

student = studres(Life_model)
s = student[abs(student)>3]
length(s)
```

```
## [1] 22
```

```
### high leverage ###

hat = hatvalues(Life_model)
h = hat[hat>(54/2203)]
length(h)
```

```
## [1] 95
```

```
influential=as.numeric(names(c)) #storing the indexes of the values that are IOs
outliers=as.numeric(names(s)) #storing the indexes of the values that are outliers
highleverage=as.numeric(names(h)) #storing the indexes of the values that are HLVs

a=intersect(influential,outliers) #common values b/w IO and outliers
b=intersect(outliers,highleverage) #common values b/w outliers and HLV
c=intersect(highleverage,influential) #common values b/w HLV and IO
d=intersect(influential,b) #common values in all three of them
e=intersect(a,c) #common values

newdata=train[-c(a,c),]
dim(newdata)
```

```
## [1] 2160   19
```

We must keep in mind that data is very precious and losing a huge amount of data for the sake of data cleansing can prove to be disadvantageous. Hence, we remove some of the data impurities.

Now, a new model based on the cleansed dataset is created.

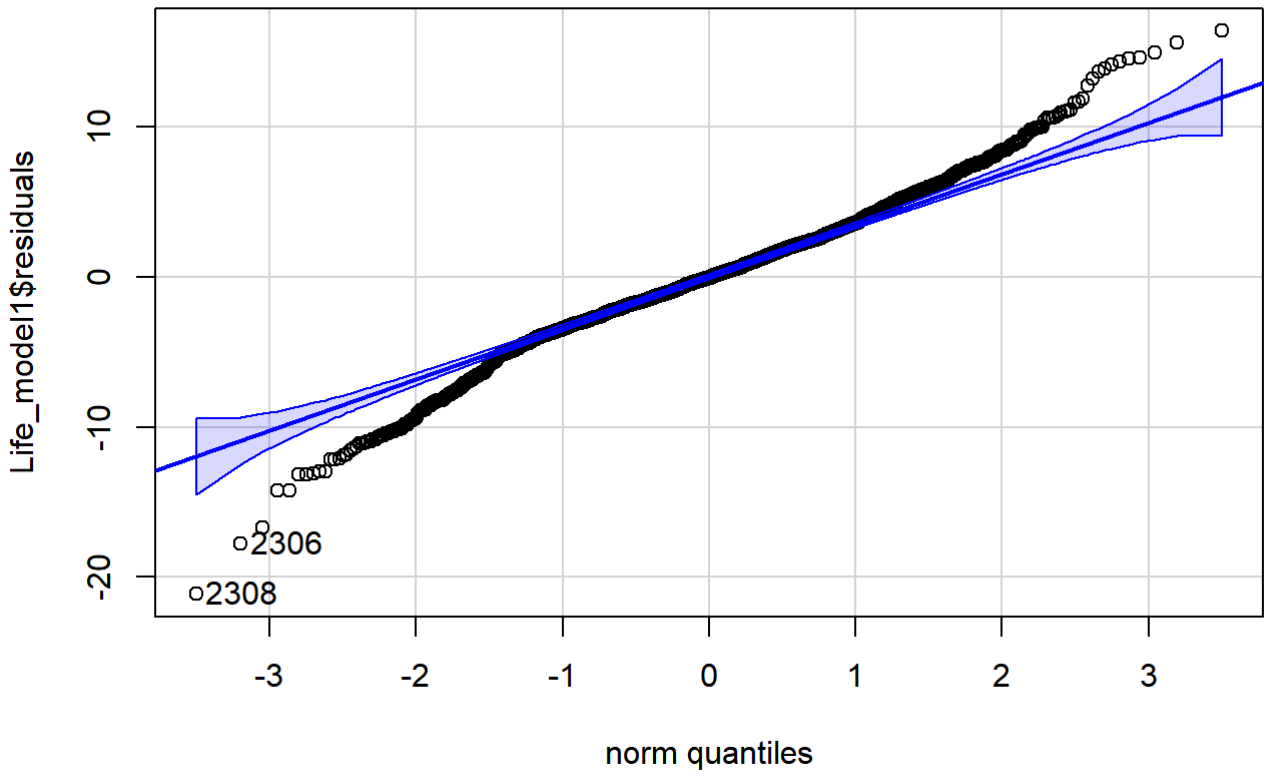
```
Life_model1=lm(y~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18, data=newdat
a)
summary(Life_model1)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18, data = newdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.1363  -2.2761  -0.0437   2.3394  16.4112
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.417e+01  6.776e-01  79.945  < 2e-16 ***
## x1          -1.989e-02  9.309e-04 -21.361  < 2e-16 ***
## x2           1.001e-01  1.029e-02   9.726  < 2e-16 ***
## x3           1.321e-01  2.812e-02   4.697 2.81e-06 ***
## x4           1.104e-04  1.039e-04   1.062  0.28822
## x5          -1.451e-02  4.660e-03  -3.114  0.00187 **
## x6          -1.772e-05  9.601e-06  -1.846  0.06504 .
## x7           3.744e-02  5.759e-03   6.501 9.87e-11 ***
## x8          -7.604e-02  7.497e-03 -10.143  < 2e-16 ***
## x9           2.957e-02  5.336e-03   5.541 3.37e-08 ***
## x10          1.180e-01  3.913e-02   3.015  0.00260 **
## x11          4.014e-02  5.611e-03   7.154 1.15e-12 ***
## x12         -4.552e-01  2.019e-02 -22.543  < 2e-16 ***
## x13          4.285e-05  1.599e-05   2.679  0.00743 **
## x14          3.801e-09  2.596e-09   1.464  0.14333
## x15         -7.491e-02  6.101e-02  -1.228  0.21960
## x16         -2.841e-03  6.028e-02  -0.047  0.96241
## x17          6.335e+00  7.483e-01   8.467  < 2e-16 ***
## x18          6.711e-01  4.937e-02  13.595  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.122 on 2141 degrees of freedom
## Multiple R-squared:  0.816,  Adjusted R-squared:  0.8144
## F-statistic: 527.4 on 18 and 2141 DF,  p-value: < 2.2e-16
```

Adjusted R square of this model is 0.8144 which is a bit more than the last model, which indicates that using a cleansed dataset resulted in a more efficient model.

Now, let's check for normality for this model using QQPlot.

```
qqPlot(Life_model1$residuals)
```




```
## 2308 2306
## 2042 1309
```

We can observe that, the quantiles of this model, fits moderately with those of any normal sample. Hence, we can conclude that our model somewhat satisfies the condition of normality.

Since, there isn't any time component present in our dataset, hence, we don't have to check for presence of autocorrelation using Durbin Watson test.

Hence, we are moving onwards to check for homoscedasticity using Breusch Pagan test.

```
bptest(Life_model1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Life_model1
## BP = 287.99, df = 18, p-value < 2.2e-16
```

Breusch Pagan test fails, that means the data is not homoscadastic. We will have to opt for GLS technique in our final predictive model to solve this issue of heteroscedasticity.

Let's check for the multicollinearity of the data using Variance Inflation Factor (VIF).

```
vif(Life_model1)
```

```
##         x1         x2         x3         x4         x5         x6         x7
##   1.751867 135.636949   1.573369   5.097566   1.393686   1.371874   1.697805
##         x8         x9        x10        x11        x12        x13        x14
## 136.371314   2.003802   1.165024   2.272051   1.458114   5.283758   1.455168
##        x15        x16        x17        x18
##   9.147116   9.199946   2.929374   3.245281
```

So from the above code it is clear that the covariates x2 and x8 have high multicollinearity.

To get rid of the multicollinearity problem, PCA is performed on the train dataset which was last renamed as “newdata” after data cleansing.

```
pc.fit <- prcomp(~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18,data=newdat
a, scale=TRUE)
summary(pc.fit)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.3419 1.5822 1.32159 1.1758 1.10846 0.93407 0.91437
## Proportion of Variance 0.3047 0.1391 0.09703 0.0768 0.06826 0.04847 0.04645
## Cumulative Proportion  0.3047 0.4438 0.54080 0.6176 0.68586 0.73433 0.78078
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.88619 0.78262 0.75628 0.69300 0.65755 0.63083 0.54871
## Proportion of Variance 0.04363 0.03403 0.03178 0.02668 0.02402 0.02211 0.01673
## Cumulative Proportion  0.82441 0.85844 0.89021 0.91689 0.94091 0.96302 0.97975
##                           PC15    PC16    PC17    PC18
## Standard deviation     0.44834 0.31948 0.24033 0.06074
## Proportion of Variance 0.01117 0.00567 0.00321 0.00020
## Cumulative Proportion  0.99092 0.99659 0.99980 1.00000
```

From the output, we can conclude that 11 principal components explain the 90% variance of the data (evident from “Cumulative Proportion” of the attached output) which is quite satisfactory. Hence, we are going to reduce the dimensions of our model and we will continue working with just 11 principal components rather than using the 18 covariates.

We need to create a new training and testing dataset by using the linear transformation of PCA on our previous training and testing datasets.

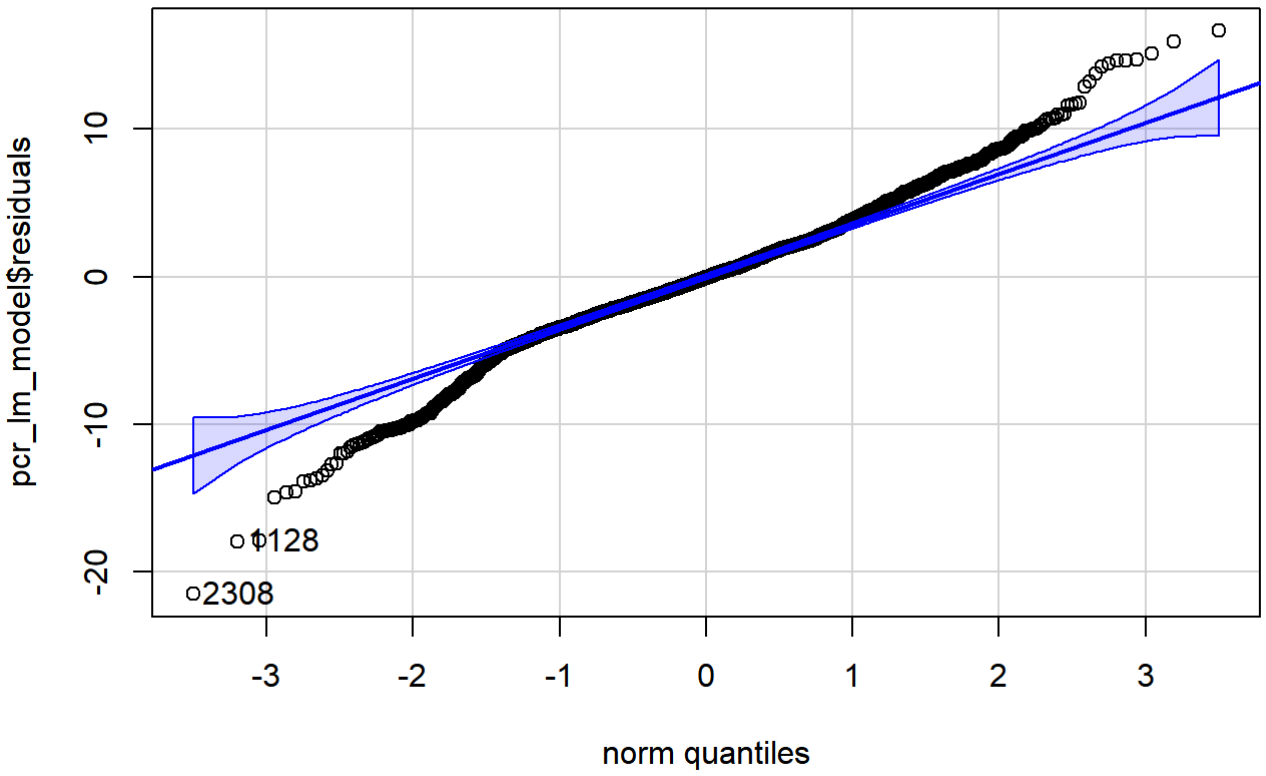
```
trans_test <- as.data.frame(predict(pc.fit, test)[,1:11])
new_train <- as.data.frame(cbind(newdata$y, pc.fit$x[,1:11]))
colnames(new_train)[1]<- "Life.expectancy"

pcr_lm_model <- lm(new_train$Life.expectancy~., data=new_train)
summary(pcr_lm_model)
```

```
##
## Call:
## lm(formula = new_train$Life.expectancy ~ ., data = new_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.4894  -2.3332  -0.0763   2.3386  16.6319
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 69.26587    0.09094 761.657  < 2e-16 ***
## PC1          3.16884    0.03884  81.584  < 2e-16 ***
## PC2         -1.78513    0.05749 -31.051  < 2e-16 ***
## PC3          1.07189    0.06883  15.574  < 2e-16 ***
## PC4         -1.18788    0.07736 -15.354  < 2e-16 ***
## PC5          1.86202    0.08206  22.690  < 2e-16 ***
## PC6         -0.55323    0.09738  -5.681 1.52e-08 ***
## PC7         -0.59047    0.09948  -5.936 3.41e-09 ***
## PC8          1.49775    0.10264  14.592  < 2e-16 ***
## PC9         -0.21678    0.11623  -1.865   0.0623 .
## PC10        -0.47102    0.12028  -3.916 9.28e-05 ***
## PC11         0.22775    0.13126   1.735   0.0829 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.227 on 2148 degrees of freedom
## Multiple R-squared:  0.8058, Adjusted R-squared:  0.8049
## F-statistic: 810.5 on 11 and 2148 DF,  p-value: < 2.2e-16
```

“trans_test” is the transformed test dataset with 11 principal components as covariates and “new_train” is the new dataset. This model has Adjusted R square value of 0.8049 which is quite satisfactory.

Now, we have to check for normality assumption and homoscedasticity once again and take measures to fix those if necessary.

```
qqPlot(pcr_lm_model$residuals) # normality assumption is satisfied
```




```
## 2308 1128
## 2042 1170
```

```
bptest(pcr_lm_model) # bptest fails. Need to opt for gls
```

```
##
##  studentized Breusch-Pagan test
##
## data:  pcr_lm_model
## BP = 189.56, df = 11, p-value < 2.2e-16
```

```
vif(pcr_lm_model) #issue of multicollinearity solved
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10 PC11
##    1    1    1    1    1    1    1    1    1    1    1
```

From the QQPlot, we can say that our model fulfills the condition of normality but once again the test for homoscedasticity fails. We need to opt for GLS to solve this. The issue of multicollinearity is solved using PCA.

```
Life_model_gls = gls(Life.expectancy~., correlation = corAR1(), data=new_train)
Life_model_gls
```

```
## Generalized least squares fit by REML
##   Model: Life.expectancy ~ .
##   Data: new_train
##   Log-restricted-likelihood: -6190.707
##
## Coefficients:
## (Intercept)         PC1         PC2         PC3         PC4         PC5
##  69.2658630   3.1685844  -1.7855303   1.0709555  -1.1894460   1.8619931
##         PC6         PC7         PC8         PC9        PC10        PC11
##  -0.5531998  -0.5907971   1.4986661  -0.2163998  -0.4707751   0.2262706
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##          Phi
## -0.004442001
## Degrees of freedom: 2160 total; 2148 residual
## Residual standard error: 4.226547
```

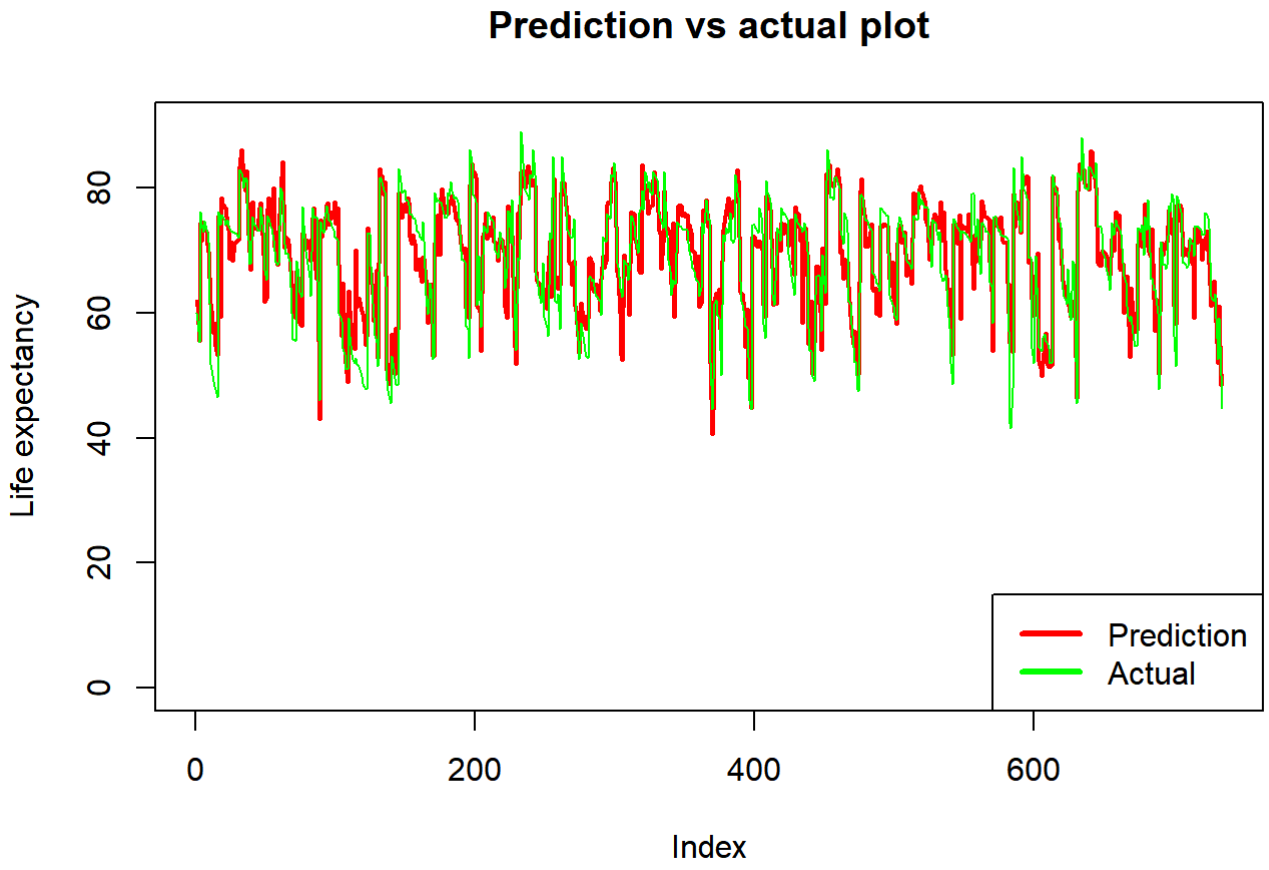
```
Rsq.ad(Life_model_gls)
```

```
## [1] 0.8048457
```

GLS is applied to solve the problems of heteroscadasticity. After applying GLS the Adjusted r square becomes 0.8049 which says that our predictive model is quite efficient.

“Life_model_gls” is our final predcitive model. Let’s move on to predictions.

```
# Predicting the fitted model on test dataset
pred1 <- predict(Life_model_gls, trans_test)
plot(pred1, col="red", type="l", lwd=2, ylab="Life expectancy", main= "Prediction vs actual p
lot", ylim=c(0,90))
lines(test$y, col="green", type="l", lwd="1")
legend(x="bottomright", legend = c("Prediction","Actual"), col=c("red","green"), cex=1, lty =
1, lwd=3)
```



So from the graph it is clear that the prediction is quite satisfactory and our model fitting is moderate.