2016 Olympics: Medal Analysis

Jessica Murphy, Alex Rotondo, and Alex Schwartz

September 23, 2019

Read Data

We chose two datasets from the 2016 Olympics that are located here on Kaggle.

```
athletes = read.csv("athletes.csv", header=T, as.is=T)
countries = read.csv("countries.csv", header=T, as.is=T)
```

Based on these datasets, there were 11,538 athletes and 201 countries that partcipated in the Olympics.

Format Data

We then merged information from the two datasets together into a new dataset.

```
# convert olympic nationality codes to country codes
# IOA - Kuwait (KUW), ROU - Romania (ROM), SRB -Serbia (SCG), TTO - Trinidad
& Tobago (TRI)
# https://en.wikipedia.org/wiki/List_of_IOC_country_codes
athletes$nationality[athletes$nationality == "IOA"] = "KUW"
athletes$nationality[athletes$nationality == "ROU"] = "ROM"
athletes$nationality[athletes$nationality == "SRB"] = "SCG"
athletes$nationality[athletes$nationality == "TTO"] = "TRI"

# convert athletes' dob to age at start of olympics
athletes$dob = as.Date(athletes$dob, "%m/%d/%Y")
athletes = athletes[!is.na(athletes$dob),]
athletes$age = floor(age_calc(athletes$dob, enddate=as.Date("2016-08-05"),
units="years"))

# determine number of medals per country
medals = athletes %>% group_by(nationality) %>%
summarize_at(vars(gold:bronze), sum) %>%
  mutate(total_medals = gold + silver + bronze)

# determine number of athletes per country and their median age
n_athletes = athletes %>% group_by(nationality) %>% summarize(total_athletes
= n(), med_age = median(age))

# determine number of males and females per country
athletes$sex = as.factor(athletes$sex)
females = athletes %>% filter(sex == "female") %>% count(nationality,
name="females")
```

```r
males = athletes %>% filter(sex == "male") %>% count(nationality,
name="males")

# merge data into one data frame
olympics = full_join(medals, n_athletes, by="nationality")
olympics = full_join(olympics, females, by="nationality")
olympics = full_join(olympics, males, by="nationality")

colnames(olympics)[1] = "code"
olympics = full_join(countries, olympics, by="code")

# look at data summary
summary(olympics)
```

```
##    country              code             population
##  Length:208         Length:208         Min.   :1.022e+04
##  Class :character   Class :character   1st Qu.:1.638e+06
##  Mode  :character   Mode  :character   Median :7.450e+06
##                                        Mean   :3.723e+07
##                                        3rd Qu.:2.557e+07
##                                        Max.   :1.371e+09
##                                        NA's   :12
##  gdp_per_capita          gold              silver            bronze
##  Min.   :   277.1   Min.   :  0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:  1781.1   1st Qu.:  0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median :  5233.6   Median :  0.000   Median : 0.000   Median : 0.000
##  Mean   : 12882.6   Mean   :  3.217   Mean   : 3.164   Mean   : 3.401
##  3rd Qu.: 15494.7   3rd Qu.:  1.000   3rd Qu.: 1.000   3rd Qu.: 2.000
##  Max.   :101450.0   Max.   :139.000   Max.   :55.000   Max.   :71.000
##  NA's   :32         NA's   :1         NA's   :1        NA's   :1
##   total_medals     total_athletes      med_age         females
##  Min.   :  0.000   Min.   :  1.00   Min.   :18.00   Min.   :  1.00
##  1st Qu.:  0.000   1st Qu.:  6.00   1st Qu.:23.00   1st Qu.:  2.00
##  Median :  0.000   Median : 11.00   Median :25.00   Median :  5.00
##  Mean   :  9.783   Mean   : 55.73   Mean   :24.72   Mean   : 25.77
##  3rd Qu.:  5.000   3rd Qu.: 56.00   3rd Qu.:26.50   3rd Qu.: 23.75
##  Max.   :264.000   Max.   :567.00   Max.   :33.00   Max.   :303.00
##  NA's   :1         NA's   :1        NA's   :1       NA's   :6
##      males
##  Min.   :  1.00
##  1st Qu.:  3.00
##  Median :  7.00
##  Mean   : 30.74
##  3rd Qu.: 33.50
##  Max.   :269.00
##  NA's   :2
```

After looking at this new data, a few of the variables have NA values that need to be addressed.

```
# convert male and female NAs to 0
olympics$females[is.na(olympics$females)] = 0
olympics$males[is.na(olympics$males)] = 0

# fill in country NAs
olympics[olympics$code == "KIR", 1] = "Kiribati"
olympics[olympics$code == "KOS", 1] = "Kosovo"
olympics[olympics$code == "MHL", 1] = "Marshall Islands"
olympics[olympics$code == "MNE", 1] = "Montenegro"
olympics[olympics$code == "ROT", 1] = "Refugee Olympic Team"
olympics[olympics$code == "SSD", 1] = "South Sudan"
olympics[olympics$code == "TUV", 1] = "Tuvalu"

# alphabetize by country
olympics = olympics %>% arrange(country)

# remove countries with no medal information (just 1)
olympics = olympics[!is.na(olympics$total_medals),]

# remove countries with missing gdp values
olympics = olympics[!is.na(olympics$gdp_per_capita),]
```

We also scaled a few of the variables that we were interested in so we could accurately compare them across different countries.

```
# add percentages
olympics = olympics %>% mutate(perc_athletes =
(total_athletes/population)*100000,
                              perc_females = (females/total_athletes)*100,
                              perc_males = (males/total_athletes)*100,
                              gdp_per_capita = gdp_per_capita/1000)
```

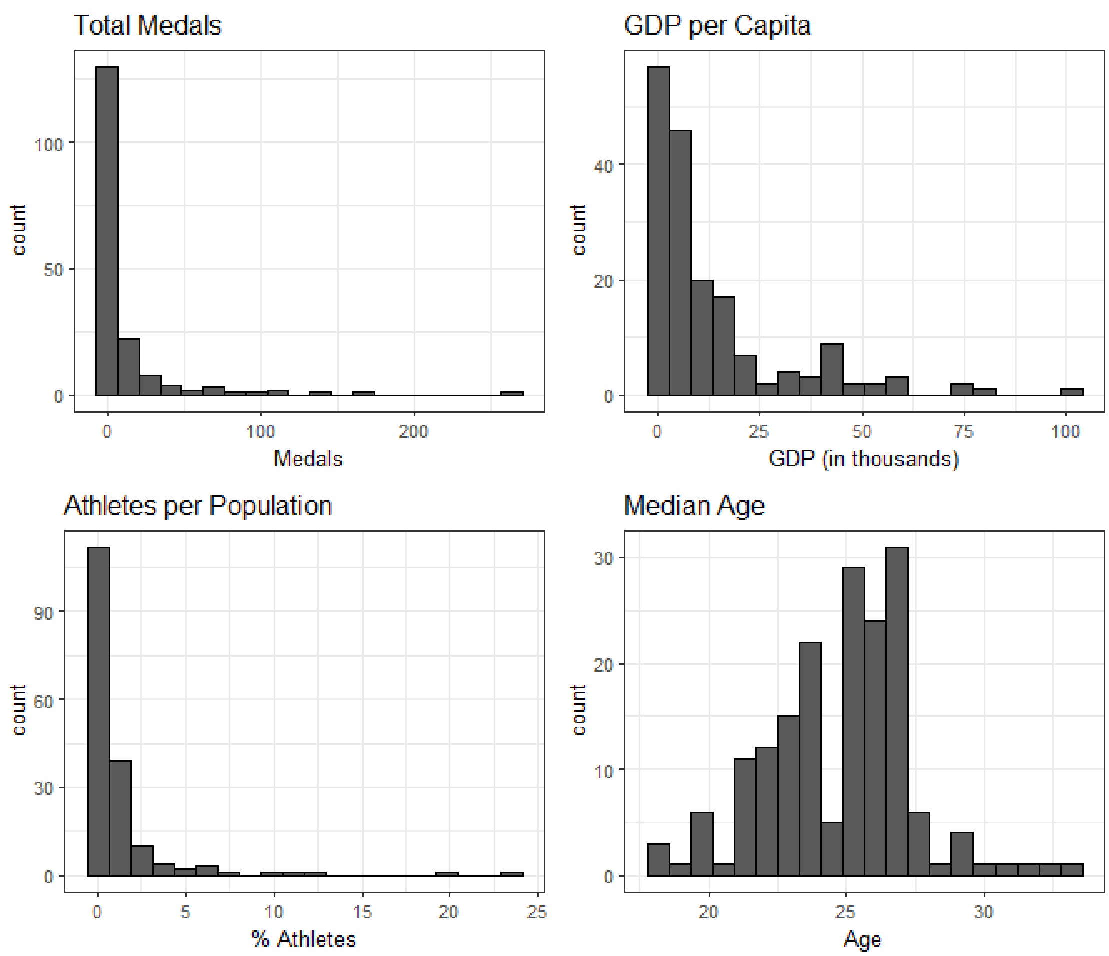
Exploratory Plots

Histograms

Here we look at some histograms of the variables to see their distributions.

```
# histogram of total medals
plot1 = ggplot(olympics, aes(total_medals)) +
  geom_histogram(col="black", bins=20) +
  labs(x="Medals", title="Total Medals") +
  theme_bw(base_size=10)

# histogram of gdp per capita
plot2 = ggplot(olympics, aes(gdp_per_capita)) +
  geom_histogram(col="black", bins=20) +
  labs(x="GDP (in thousands)", title="GDP per Capita") +
  theme_bw(base_size=10)
```

```r
# histogram of percent athlets
plot3 = ggplot(olympics, aes(perc_athletes)) +
  geom_histogram(col="black", bins=20) +
  labs(x="% Athletes", title="Athletes per Population") +
  theme_bw(base_size=10)

# histogram of median age
plot4 = ggplot(olympics, aes(med_age)) +
  geom_histogram(col="black", bins=20) +
  labs(x="Age", title="Median Age") +
  theme_bw(base_size=10)

# arrange plots in a 2 by 2 grid
grid.arrange(plot1, plot2, plot3, plot4, ncol=2, nrow=2)
```

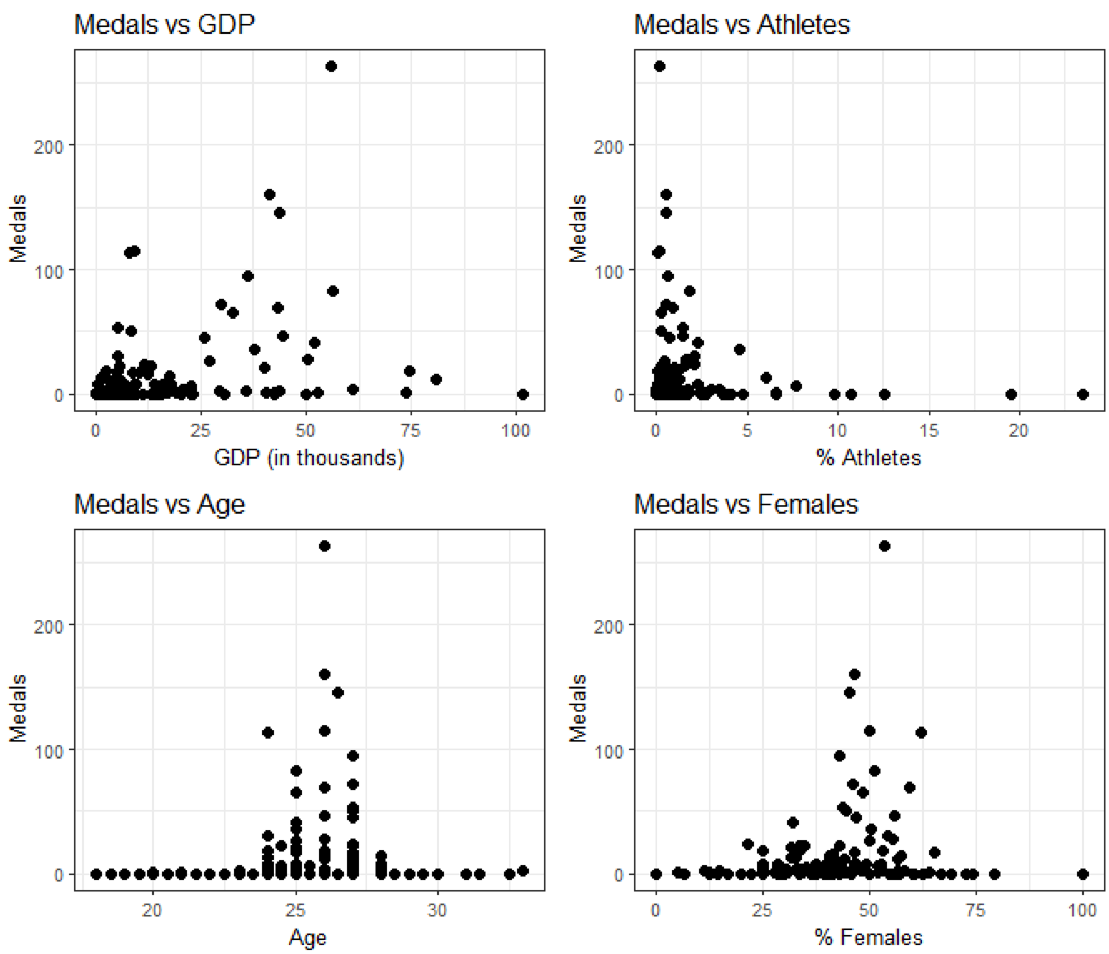


The distributions of total medals, gdp, and percent athletes are all very skewed to the right whereas the the distribution of age is fairly normal, approximately centered around 25.

Scatterplots

We also look at some scatterplots of our data to see if any relationships are visible.

```r
# scatterplot of medals vs gdp
plot5 = ggplot(olympics, aes(x=gdp_per_capita, y=total_medals)) +
  geom_point(size=2) +
  labs(x="GDP (in thousands)", y="Medals", title="Medals vs GDP") +
  theme_bw(base_size=10)

# scatterplot of medals vs percent athletes
plot6 = ggplot(olympics, aes(x=perc_athletes, y=total_medals)) +
  geom_point(size=2) +
  labs(x="% Athletes", y="Medals", title="Medals vs Athletes") +
  theme_bw(base_size=10)

# scatterplot of medals vs median age
plot7 = ggplot(olympics, aes(x=med_age, y=total_medals)) +
  geom_point(size=2) +
  labs(x="Age", y="Medals", title="Medals vs Age") +
  theme_bw(base_size=10)

# scatterplot of medals vs percent females
plot8 = ggplot(olympics, aes(x=perc_females, y=total_medals)) +
  geom_point(size=2) +
  labs(x="% Females", y="Medals", title="Medals vs Females") +
  theme_bw(base_size=10)

# arrange plots in a 2 by 2 grid
grid.arrange(plot5, plot6, plot7, plot8, ncol=2, nrow=2)
```



The plots show there could be a slight positive relationship between medals and gdp as well as a slight negative relationship between medals and percent athletes.

Linear Regression

Then we fit a linear model to our data based on our research question, with total medals as the response and gdp per capita, percent athletes, median age, and percent females as our predictors.

Model

```
# fit linear model
lmod = lm(total_medals ~ gdp_per_capita + perc_athletes + med_age +
perc_females, data = olympics)
sumary(lmod)

##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -16.81673   22.26699 -0.7552   0.4511
## gdp_per_capita    0.64262    0.12956  4.9600 1.69e-06
## perc_athletes    -0.74400    0.79418 -0.9368   0.3502
## med_age           0.51262    0.87565  0.5854   0.5590
## perc_females      0.19142    0.14746  1.2981   0.1960
```

```
##
## n = 176, p = 5, Residual SE = 29.01646, R-Squared = 0.16
```

The summary shows that the only significant predictor based on a 0.05 significance level is gdp per capita. However, it also shows that our model fit is pretty low, with an R-squared value of only 0.16.

Collinearity check

Here we check for collinearity to ensure none of our predictors are linearly depenent on one another.

```
# extract X matrix from model
x = model.matrix(lmod)
x = x[,-1] # remove intercept

# correlation matrix
round(cor(x), 2)

##                gdp_per_capita perc_athletes med_age perc_females
## gdp_per_capita           1.00          0.10    0.27         0.04
## perc_athletes            0.10          1.00   -0.09        -0.22
## med_age                  0.27         -0.09    1.00         0.07
## perc_females             0.04         -0.22    0.07         1.00

# variance inflation factors
vif(lmod) #looks good (<5)

## gdp_per_capita  perc_athletes        med_age   perc_females
##       1.098880       1.080952       1.092270       1.059294
```

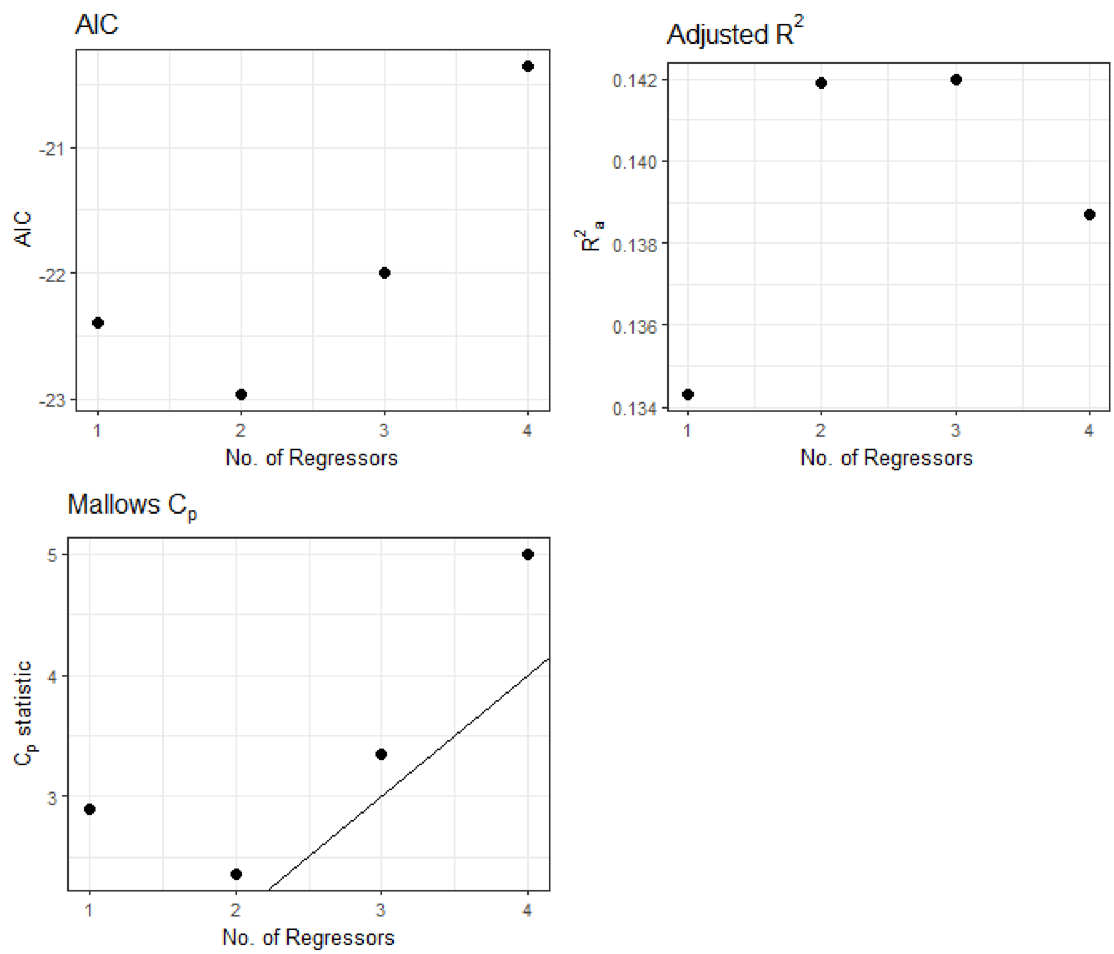
All of the correlations are less than 0.5 and all of the vifs are less than 5, so there is no evidence of collinearity.

Variable selection

Next, we determine which variables should stay in our model, based on the AIC, Adjusted R-squared, and Mallow's Cp statistic. We then fit a new model with only the selected variables.

```
# extract model information
model_info = regsubsets(total_medals ~ gdp_per_capita + perc_athletes +
med_age + perc_females, data = olympics)
b = summary(model_info)

# get variable decision matrix
b$which

##   (Intercept) gdp_per_capita perc_athletes med_age perc_females
## 1        TRUE           TRUE         FALSE   FALSE        FALSE
## 2        TRUE           TRUE         FALSE   FALSE         TRUE
```

```
## 3        TRUE           TRUE          TRUE    FALSE          TRUE
## 4        TRUE           TRUE          TRUE     TRUE          TRUE

# AIC: p=3
p = 2:5
AIC = b$bic + p * (2 - log(176))
plot9 = ggplot() + aes(x=c(1:4), y=AIC) +
  geom_point(size=2) +
  labs(x="No. of Regressors", title="AIC") +
  theme_bw(base_size=10)

# Adjusted R-squared: p=4
plot10 = ggplot() + aes(x=c(1:4), y=b$adjr2) +
  geom_point(size=2) +
  labs(x="No. of Regressors", y=expression({R^2}[a]),
       title=expression(paste("Adjusted R"^"2"))) +
  theme_bw(base_size=10)

# Mallows Cp: p=3
plot11 = ggplot() + aes(x=c(1:4), y=b$cp) +
  geom_point(size=2) +
  geom_abline(intercept=0, slope=1) +
  labs(x="No. of Regressors", y=expression(paste(C[p], " statistic")),
       title=expression(paste("Mallows ", C[p]))) +
  theme_bw(base_size=10)

# arrange plots in a 2 by 2 grid
grid.arrange(plot9, plot10, plot11, ncol=2, nrow=2)
```



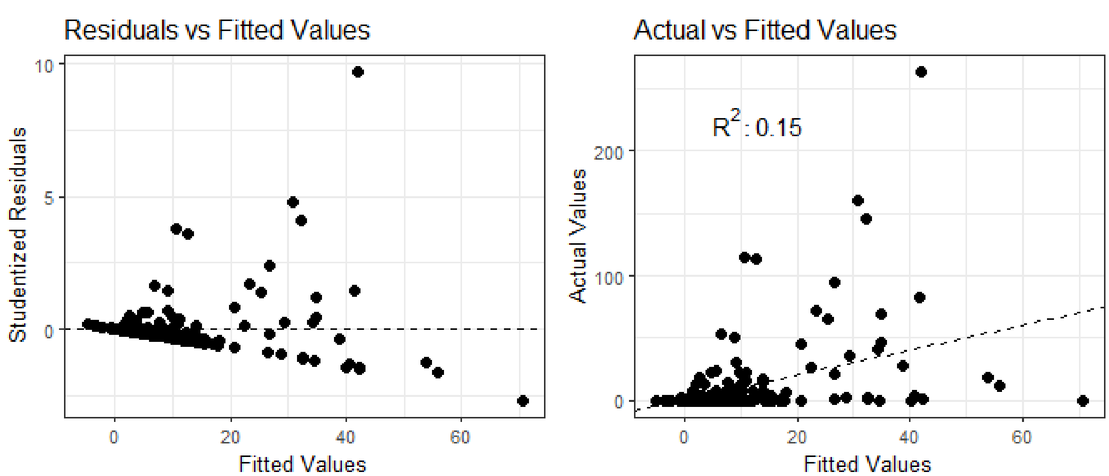

```
# Remove variables
lmod2 = update(lmod, . ~ . - med_age - perc_athletes)
sumary(lmod2)

##                 Estimate Std. Error t value  Pr(>|t|)
## (Intercept)     -6.66975    6.54294 -1.0194    0.3094
## gdp_per_capita   0.64873    0.12348  5.2536 4.345e-07
## perc_females     0.22822    0.14315  1.5943    0.1127
##
## n = 176, p = 3, Residual SE = 28.96248, R-Squared = 0.15
```

The plots show that the optimal number of regressors is 2, which includes gdp per capita and percent females. The summary of the updated model shows that R-squared value is only 0.01 less than the model with all four variables. Also, the percent females variable is slightly more significant, but not based on a 0.05 significance level.

Check assumptions

Here we check the model assumptions: linearity, independence, homoscedascity, and normality.

```
# studentized residuals
studentized2 = rstudent(lmod2)
fitted2 = fitted(lmod2)

# check linearity / homoscedasticity
plot12 = ggplot(lmod2, aes(x=fitted2, y=studentized2)) +
  geom_point(col="black", size=2) +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw(base_size=10) + labs(x="Fitted Values", y="Studentized Residuals",
title="Residuals vs Fitted Values")

# visualize R-squared
r2 = format(summary(lmod2)$r.squared, digits=2)
plot13 = ggplot(lmod2, aes(x=fitted2, y=olympics$total_medals)) +
  geom_point(col="black", size=2) +
  geom_abline(aes(intercept=0, slope=1), lty=2) +
  theme_bw(base_size=10) + labs(x="Fitted Values", y="Actual Values",
title="Actual vs Fitted Values") + annotate("text", label=paste("R^2: ", r2,
sep=""), x=13, y=225, parse=T, size=4)

# arrange plots in a 1 by 2 grid
grid.arrange(plot12, plot13, ncol=2)
```

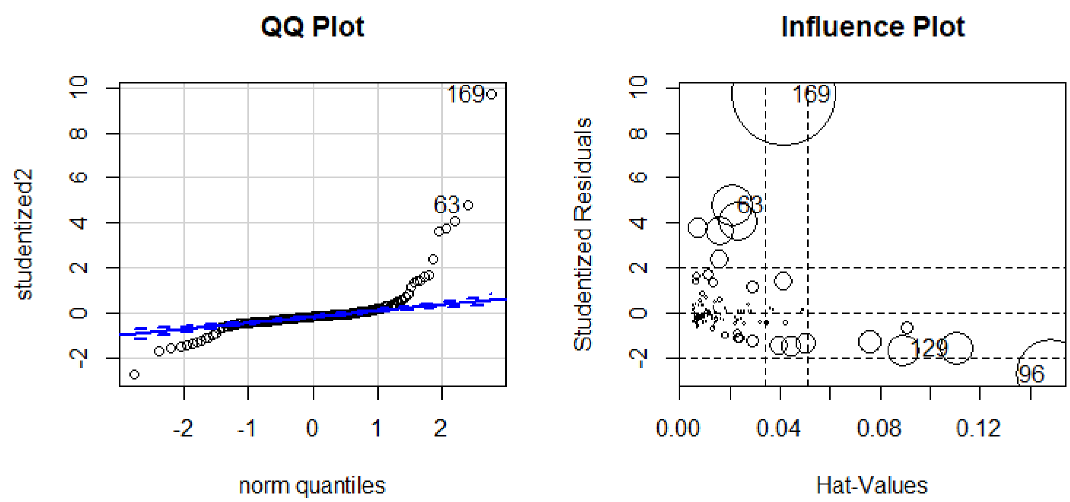


Though we do see that the residuals are normally distributed and most dense around zero, the constant variance assumption appears to be violated when the fitted values are plotted against the residuals. At lower fitted values, the residuals tend closer to zero, but as the fitted values increase, the spread of the residuals from zero increases.

```
# check normality
par(mfrow=c(1,2))
qqPlot(studentized2, main="QQ Plot")

## [1] 169  63

shapiro.test(studentized2)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  studentized2
## W = 0.54783, p-value < 2.2e-16
```

```r
# check leverage points / outliers
influencePlot(lmod2, main="Influence Plot") #198
```




```
##        StudRes        Hat      CookD
## 63   4.788896 0.02057438 0.14251626
## 96  -2.687015 0.14849336 0.40513278
## 129 -1.516923 0.11043051 0.09450632
## 169  9.722938 0.04187666 0.89395452
```

Regarding the influence plot, observations with high leverage and potentially outlying values are identified on the plot. Due to the large discrepencies in population size of countries who won medals versus countries who did not, removing such observations would lessen the model's predictive value.

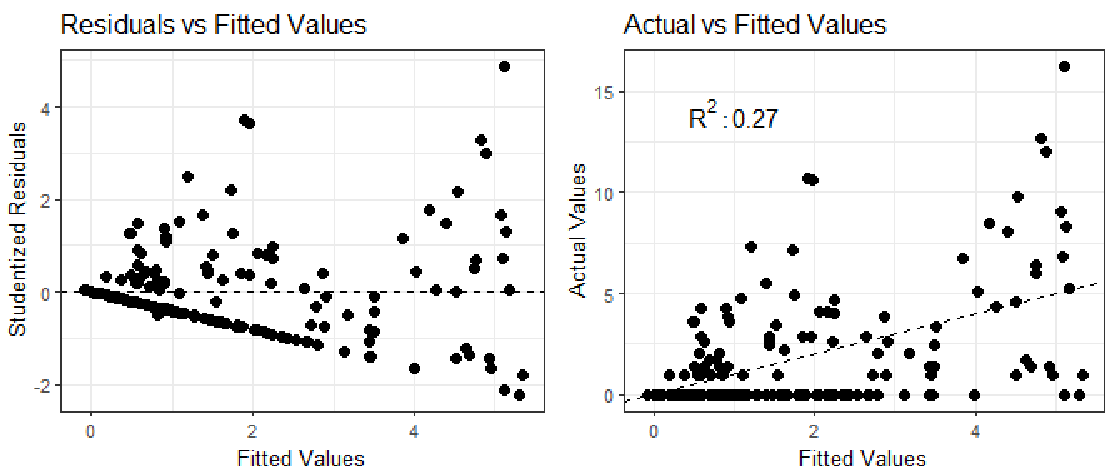
Transformations

Now, we fit a new model with a square root transformation of the response and a quadratic transformation of gdp. We also re-evaluate our previous assumptions.

```r
# fit new linear model with transformations
lmod3 = lm(sqrt(total_medals) ~ poly(gdp_per_capita,2) + perc_females, data =
olympics)
sumary(lmod3)

##                            Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                0.993364   0.549403  1.8081   0.07234
## poly(gdp_per_capita, 2)1  15.841465   2.482432  6.3814 1.570e-09
## poly(gdp_per_capita, 2)2 -11.023226   2.481410 -4.4423 1.589e-05
```

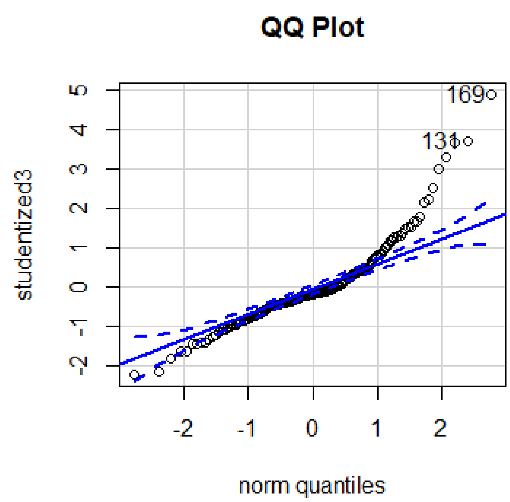
```
## perc_females                0.018108   0.012264  1.4765   0.14164
##
## n = 176, p = 4, Residual SE = 2.48001, R-Squared = 0.27
```

```
# studentized residuals
studentized3 = rstudent(lmod3)
fitted3 = fitted(lmod3)

# check linearity / homoscedasticity
plot14 = ggplot(lmod3, aes(x=fitted3, y=studentized3)) +
  geom_point(col="black", size=2) +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw(base_size=10) + labs(x="Fitted Values", y="Studentized Residuals",
title="Residuals vs Fitted Values")

# visualize R-squared
r2.3 = format(summary(lmod3)$r.squared, digits=2)
plot15 = ggplot(lmod3, aes(x=fitted3, y=sqrt(olympics$total_medals))) +
  geom_point(col="black", size=2) +
  geom_abline(aes(intercept=0, slope=1), lty=2) +
  theme_bw(base_size=10) + labs(x="Fitted Values", y="Actual Values",
title="Actual vs Fitted Values") + annotate("text", label=paste("R^2: ",
r2.3, sep=""), x=1, y=14, parse=T, size=4)

# arrange plots in a 1 by 2 grid
grid.arrange(plot14, plot15, ncol=2)
```




```
# check normality
par(mfrow=c(1,2))
qqPlot(studentized3, main="QQ Plot")

## [1] 169 131

shapiro.test(studentized3)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  studentized3
## W = 0.88776, p-value = 3.039e-10
```



Applying a square root transformation to the response and squaring gdp_per_capita shows the studentized residuals deviating slightly to the left of zero, with significantly less deivation from homoscedasticity than the previous non-transformed model.

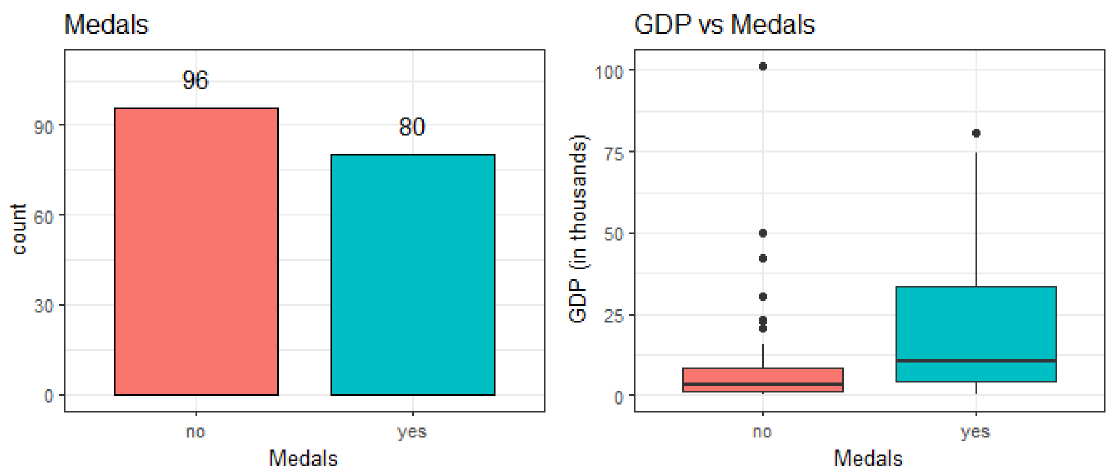
Logistic Regression

Thinking of each country as its own district and total medals as the count of Olympic medals obtained by each country, a scenario better described by a Poisson distribution emerges. As a consequence, a linear model may not best predict the number of Olympic medals won by each country.

An approach using logistic regression is described below to determine the probability of a country medaling in the Olympics.

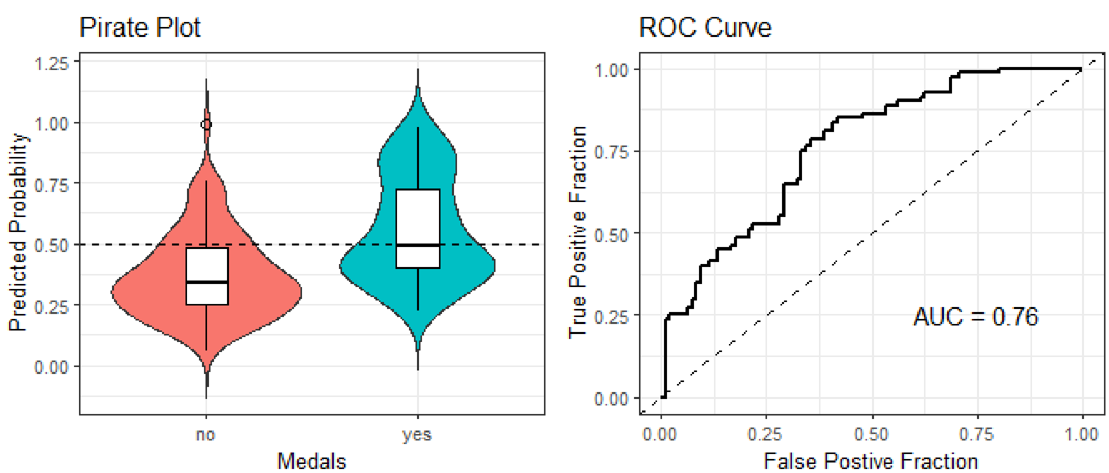
```
# create new categorical variable for medals
olympics$medals = cut(olympics$total_medals, breaks = c(-Inf, 0, Inf), labels
= c("no", "yes"))
summary(olympics$medals)

##  no yes
##  96  80

# look at distribution of new response variable
plot16 = ggplot(olympics, aes(x=medals, fill=medals)) +
  geom_bar(color="black", width=0.75) +
  scale_y_continuous(limits = c(0, 110)) +
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
```

```
  labs(x="Medals", title="Medals") +
  theme_bw(base_size=10) +
  theme(legend.position="none")

# compare new response to gdp
plot17 = ggplot(olympics, aes(x=medals, y=gdp_per_capita, fill=medals)) +
  geom_boxplot() +
  labs(x="Medals", y="GDP (in thousands)", title="GDP vs Medals") +
  theme_bw(base_size=10) +
  theme(legend.position="none")

# arrange plots in a 1 by 2 grid
grid.arrange(plot16, plot17, ncol=2)
```




```
# fit logistic regression model
glm.out = glm(medals ~ gdp_per_capita + perc_athletes + med_age +
perc_females, data = olympics, family = binomial)
sumary(glm.out)

##                Estimate Std. Error z value  Pr(>|z|)
## (Intercept)    -3.791511   1.753830 -2.1618 0.0306300
## gdp_per_capita  0.051873   0.013767  3.7678 0.0001647
## perc_athletes  -0.103828   0.073256 -1.4173 0.1563839
## med_age         0.158320   0.070311  2.2517 0.0243397
## perc_females   -0.019408   0.011829 -1.6407 0.1008607
##
## n = 176 p = 5
## Deviance = 207.12834 Null Deviance = 242.53125 (Difference = 35.40291)

# convert medals to binary variable (0-no, 1-yes)
olympics$medals2 = ifelse(olympics$medals == "no", 0, 1)

# make data frame of predicted and actual values for medals
df = data.frame(predictor = predict(glm.out, olympics, type="response"),
                known.truth = olympics$medals2)
```

```
# convert predicted probabilities to 0's and 1's
threshold=0.5
df$predicted.medals = ifelse(df$predictor<threshold,0,1)

# pirate plot of predicted probabilites
plot18 = ggplot(olympics, aes(y = df$predictor, x = medals, fill=medals)) +
  geom_violin(trim=FALSE, show.legend=F) +
  geom_boxplot(width=0.2, fill="white", outlier.color="black",
outlier.shape=1,
               color="black", outlier.size=2) +
  labs(y="Predicted Probability", x="Medals", title="Pirate Plot") +
  geom_hline(yintercept=0.5, linetype="dashed") +
  theme_bw(base_size=10)

# plot roc curve
roc.plot = ggplot(df, aes(d = known.truth, m = predictor)) +
  geom_roc(n.cuts=0) +
  geom_abline(intercept = 0, slope = 1, linetype="dashed") +
  labs(x="False Postive Fraction", y="True Positive Fraction", title="ROC
Curve") +
  theme_bw(base_size=10)

# add auc value to plot
plot19 = roc.plot + annotate("text", x = .75, y = .25, size = 4, label =
paste("AUC =", round(calc_auc(roc.plot)$AUC, 2)))

# arrange plots in a 1 by 2 grid
grid.arrange(plot18, plot19, ncol=2)
```



Based on the AUC value, the model has approxmiately a 76% chance of distinguishing between countries who won medals versus countries who did not win medals. However, since this was evaluated on the data used to train our model, we would expect the classification to not do as well on a test dataset.