Statistical Thinking

Week 9 Tutorial

Introduction

This week we are looking at simple regression and model selection/evaluation using Olympic Medal data from Rio and London, using medal counts from the London 2012 Summer Olympic games to predict medal counts from the Rio 2016 Summer Olympic games.

Data Before you begin, you'll need to download the two data files
- **rio_olympics2016.csv**, and
- **london_olympics2012.csv**
which are both available on Moodle.[1]

In 2016, 205 teams[2] competed for 973 medals, but only 86 teams scored at least one medal. In 2012, 204 teams participated for 962 medals, with 85 countries receiving at least one medal.

Our analysis will focus on teams that win medals, though there are many teams that compete in the Olympics that do not win any medals. It is important to keep this fact in mind as you interpret your results.[3]

Once you have completed the Lab questions, you can attempt the Lab submission.

Part A: Modelling the population assuming i.i.d. data

Before we we start you'll need to read in the datafiles, and create the initial tibble for your analysis,as per the code chunk shown below.

```r
df16 <- read_csv("data/rio_olympics2016.csv")
df12 <- read_csv("data/london_olympics2012.csv")
```

Visual and Numerical descriptive analysis Before attempting a formal statistical analysis, it is important to take a look at the data. Normally we would produce both visual and numerical summaries to illustrate the main features that are apparent to explore any aspects of the data and its related information that you feel you understand what the data represents.

[1]For more information, see https://en.wikipedia.org/wiki/2016_Summer_Olympics, https://en.wikipedia.org/wiki/2012_Summer_Olympics and https://www.olympic.org.

[2]Including, for the first time, the Refugee Olympic Team (ROT) comprised of athletes displaced by their home countries. Other countries that participated for the first time are Kosovo and South Sudan. In addition, due to problems of governmental interference by the Kuwait Olympic Committee, athletes from Kuwait participated under a team referred to as the Independent Olympic Athletes (IOA).

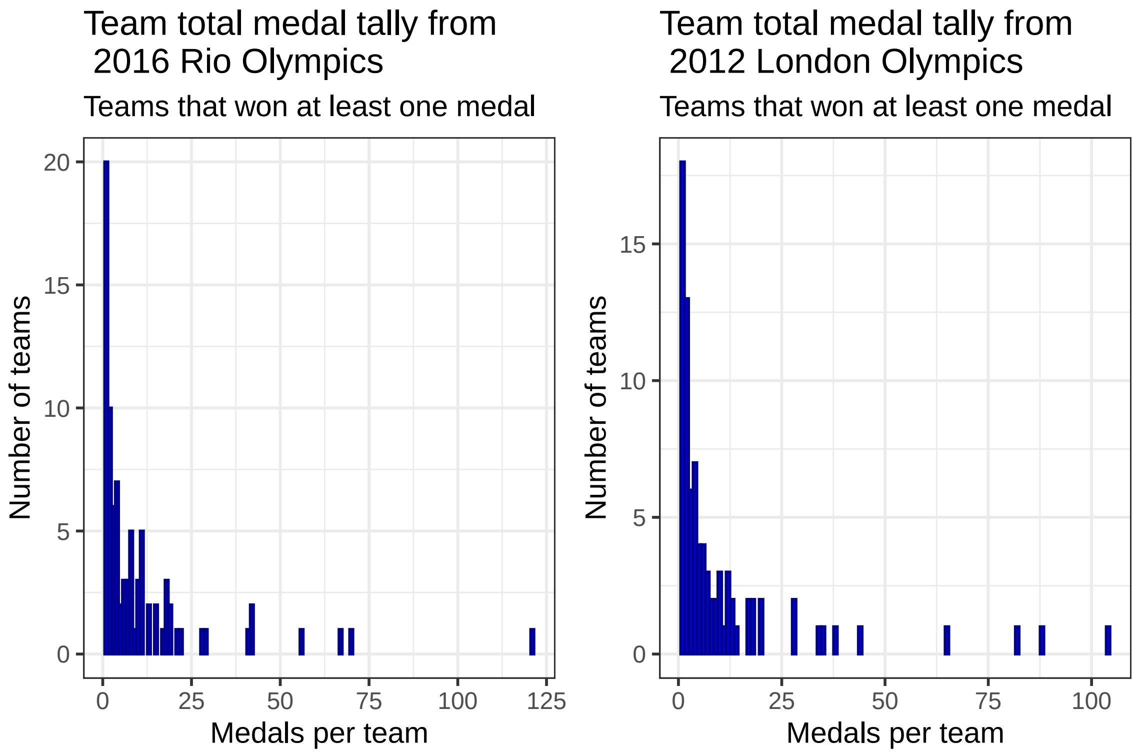
[3]Philosophically, one might prefer to analyse data that has the additional zero medal counts included in the dataset. However, statistically the incorporation of the large number of zero values requires more complex models than what we have covered in this unit.

Question 1: Visual summary Let's look at the bar plots of the 2012 and 2016 total medal count for all teams that won at least one medal.

```r
p1 <- df16 %>%
    ggplot(aes(x = Total)) + geom_bar(fill = "blue",
    colour = "darkblue") + theme_bw() + ggtitle("Team total medal tally from \n 2016 Rio Olympics",
    "Teams that won at least one medal") + ylab("Number of teams") +
    xlab("Medals per team")

p1b <- df12 %>%
    ggplot(aes(x = Total)) + geom_bar(fill = "blue",
    colour = "darkblue") + theme_bw() + ggtitle("Team total medal tally from \n 2012 London Olympics",
    "Teams that won at least one medal") + ylab("Number of teams") +
    xlab("Medals per team")

grid.arrange(p1, p1b, ncol = 2)
```



We can see that both are heavily skewed. So far we have looked at fitting these distributions (using MLE or Bayesian).

Reminder

If we were to fit a distribution using MLE, we might first choose a Poisson distribution for example since the data are counts. There are issues with this since that the Poisson assumes that the mean = variance, which is not likely, and there is a finite number of medals. The Poisson assumes there is no upper bound to the counts. It turns out that the Poisson is not a good fit.

A better fit is the log-normal distribution. We can use the log-normal directly on the raw data (medal counts), or take the log of the data (log(medal counts)) and fit a Normal distribution to it. Both methods are identical.

However with regression, we care about the correlation between the two variables. So we may not need to transform the data. We also care about the shape of the residuals. It is still important to look at the data first though.

Simple linear regression

Let's use the 2012 Olympic medal totals for each country as a **predictor** (or explanatory variable) for the 2016 Olympic medal totals in a simple linear regression (SLR) model.

Notice that the two datasets are contained in two different .csv files, and have already been read into **R** as two separate tibbles, so we need to merge them. To fit the SLR model, we need to use the *full_join()* function in the **dplyr** package to merge the *Country* and *Total* columns from the two tibbles we have (df12 and df16) to create a single tibble, named "oly".

```r
oly <- full_join(df16[, c("Country", "Total")], df12[,
    c("Country", "Total")], by = "Country")
# df12[,c('Country', 'Total')], by='Country',
# all.x=TRUE)
oly
```

In particular, the two tibbles are matched according to the **by="Country"** argument, as shown above.

Now take a look at the "oly" tibble. Did the *full_join*() function work as expected?

We need to do a little data wrangling now… which may be done by running the code chunk below. It will
1. Rename some of the column titles in **oly** so that we keep track of the Olympic years, and
2. Replace the missing values with zeros. Be sure to check that the replacement works as expected.

Note the sample size

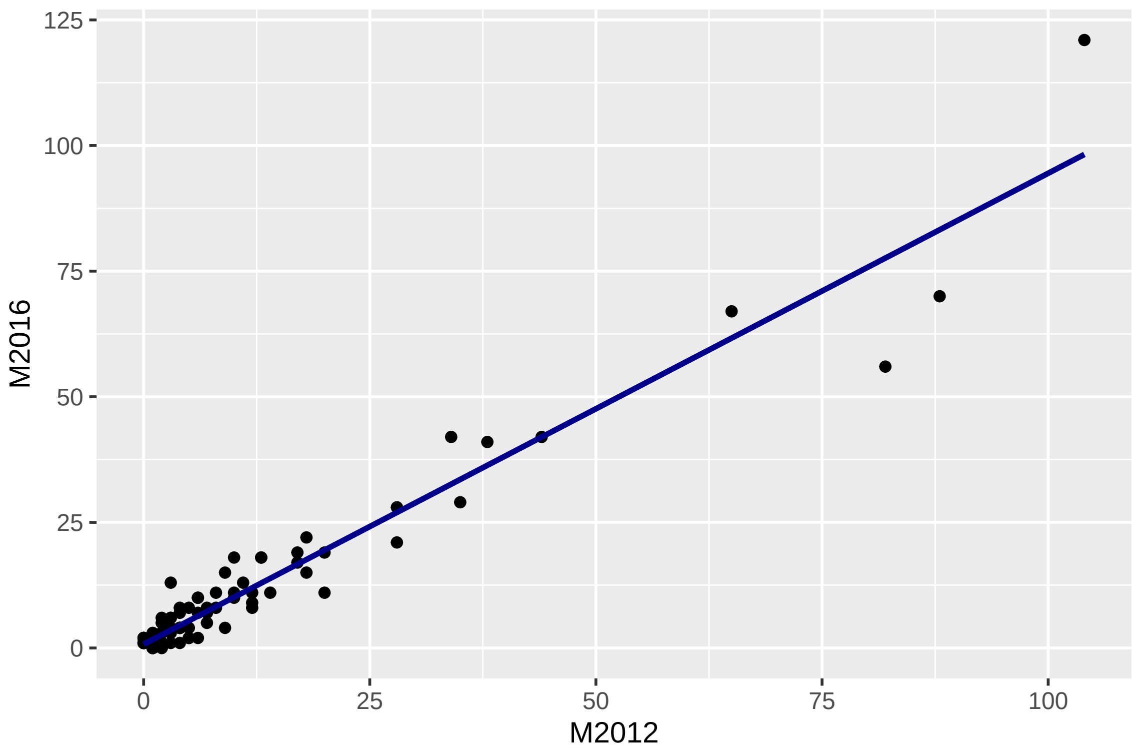
```r
colnames(oly)[2] <- "M2016"
colnames(oly)[3] <- "M2012"
oly <- oly %>%
    replace_na(list(M2016 = 0, M2012 = 0))
```

We want to try to *explain* some of the variability (shape, spread) in the marginal distribution of the *M2016* (the Total medal counts from the 2016 Olympics) using the observed variables of *M2012* (the Total medal counts from the 2012 Olympics). We'll do this using *simple linear regression* (SLR).

Produce a scatterplot of *M2016* on *M2012*. Since this is a simple regression, the scatterplot will give us a good idea of the relationship between 2012 and 2016.

Note: you can add the ordinary least squares (OLS) regression line to your scatterplot by adding **geom_smooth(method="lm", se=FALSE)** to your plot.

```r
p6 <- oly %>%
    ggplot(aes(x = M2012, y = M2016)) + geom_point() +
    geom_smooth(formula = y ~ x, method = "lm", se = FALSE,
        colour = "darkblue")
p6
```



So it looks as though there is a positive and strong linear relationship.

Let's check the correlation. What do you think it will be?

```r
# NOT GIVEN TO STUDENTS
cor(oly$M2012, oly$M2016)
# [1] 0.9662
```

Now we want to get estimates for the line above.
The (population) model we estimate is

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i.$$

- The regression model explains how response variable, y, changes in relation to a change in the explanatory variable, x, **on average**.
- This means not all points will fall exactly on the regression line, but should appear to be randomly scattered (in the vertical direction) around the line.
- The regression line is the **explained** part, the difference between the line and the observations (the dots) ins the **unexplained** part

Fitting process

How do we obtain the coefficients of the OLS regression line? We can fit the model using the *lm*() function. Note the **formula** input in the *lm*() function call below. This is what determines the relationship between the outcome variable and the regressor(s). It takes the form **y~x** when we want to fit a model to the outcome **y** using just one varying regressor **x**. The *lm*() function automatically includes an intercept term.

```r
oly_lm <- oly %>%
    lm(formula = M2016 ~ M2012)
coefs <- tidy(oly_lm)
kable(coefs) %>%
    kable_styling(latex_options = "HOLD_position")
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.7255	0.5236	1.385	0.1691
M2012	0.9375	0.0255	36.696	0.0000

We report the fitted model as:

$\widehat{M_{2016}} = 0.7255 + 0.9375\ M_{2012}$

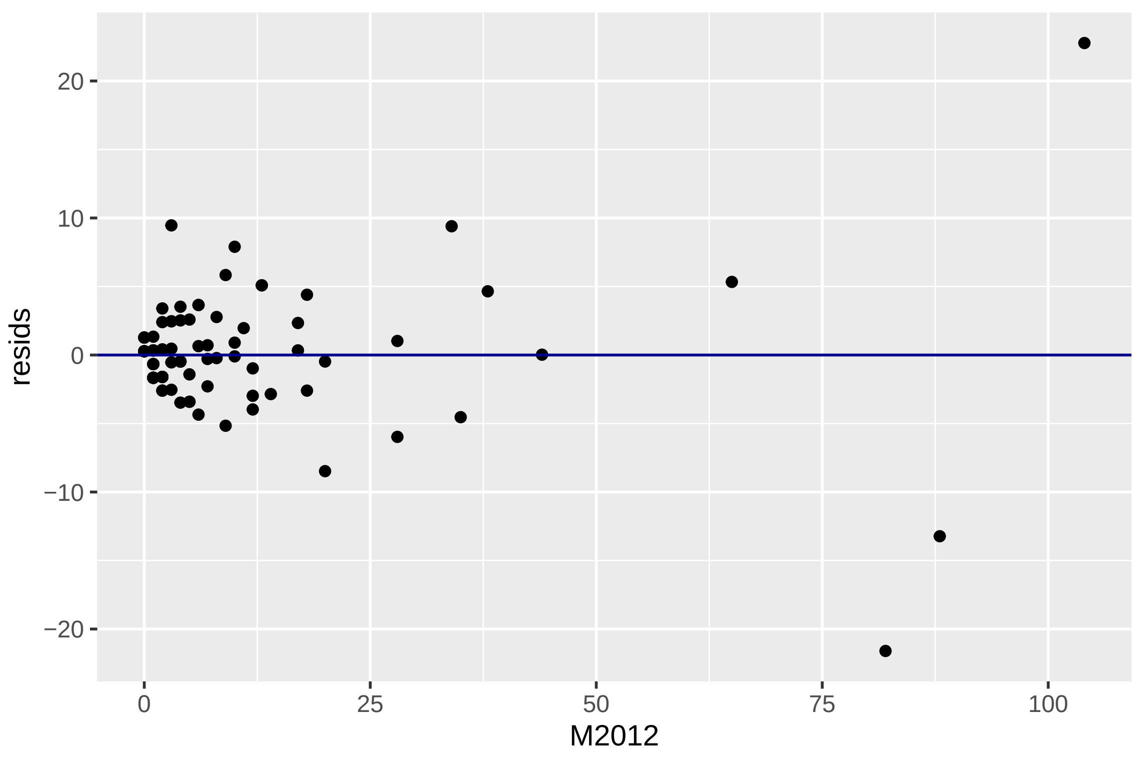
Parameter interpretation

- *Line of best fit: $\hat{y} = b_0 + b_1 x$, for any value of x. Note that there is no error or residual term. This equation represents the regression LINE*
- *b_0 is the intercept of the OLS line with y-axis. It represents the estimated number of medals that we expect for a team 2016 that had not won any medals in 2012.*
- *b_1 is the slope of the OLS line. It represents the average or increase in the total medal count in 2016 for a given unit increase in the total medal count in 2012.*

Residuals In regression, we want to explain any relationship between the explanatory variables and the dependent variable. If we have done a good job, then what is left over (the residuals), should be random.

The **oly_lm** object contains more than just the model summaries we printed above. It also contains the **residuals** from the model fit, which we can plot, as shown below:

```r
oly <- oly %>%
    mutate(resids = oly_lm$residuals)
p7 <- oly %>%
    ggplot(aes(x = M2012, y = resids)) + geom_point() +
    geom_hline(yintercept = 0, colour = "darkblue")
p7
```



Do you notice any pattern in the residuals? **Solution**
It looks like there is increased variance (width of dots from the horizontal line) as M2012 increases. This may suggest heteroscedasticity.

How much of the variation in y does the regression model output suggest it *explains*? To answer this question, we need to find the **R-squared**, or goodness of fit statistic. The "R-squared" (R^2) represents the **proportion of variation** in the observed y_i's explained by the regression line. It is computed as

$$R^2 = 1 - \frac{\sum_{i=1}^n e_i^2}{\sum_{j=1}^n (y_j - \bar{y})^2},$$

but we can extract its value from the **oly_lm** object without having to calculate it explicitly, as shown in the code chunk below.

```r
summary(oly_lm)$r.squared
# [1] 0.9335
```

Solution

A good R-squared value does not guarantee a good model - it just tells us that the regression captures most of the variation in y. We need to be sure that the model doesn't have any major problems with it first. We can visually check the model using residual plots, and we can check whether the fitted model is unduly influenced by at most just a few observation.

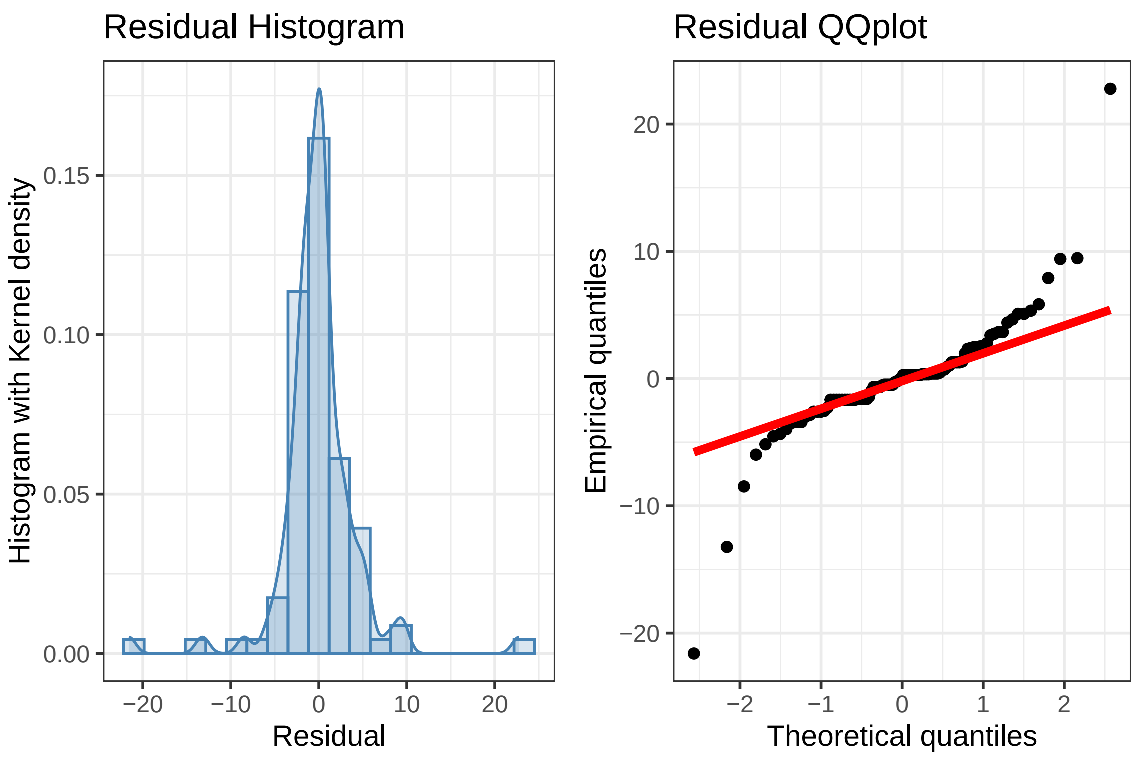
Residual plots - what are we looking for? If we plot the residuals e_i against the corresponding x_i, we should not detect any real patterns (i.e. for a "good fit" we should just see random scatter! With no discernible patterns in the residuals).

We can also plot the **histogram of residuals**. For a good fit the shape should be approximately symmetric and bell-shaped. We could also look at a QQ-plot against the corresponding theoretical normal quantiles.

```r
rhis <- oly %>%
    ggplot(aes(x = resids, y = after_stat(density))) +
    geom_histogram(colour = "steelblue", fill = "steelblue",
        alpha = 0.2, bins = 20) + geom_density(colour = "steelblue",
    fill = "steelblue", alpha = 0.2) + ggtitle("Residual Histogram") +
    xlab("Residual") + ylab("Histogram with Kernel density") +
    theme_bw()

rqq <- ggplot(oly, aes(sample = resids)) + stat_qq(distribution = qnorm) +
    stat_qq_line(distribution = qnorm, color = "red",
        size = 1.5) + theme(aspect.ratio = 1) + ggtitle("Residual QQplot") +
    xlab("Theoretical quantiles") + ylab("Empirical quantiles") +
    theme_bw()


grid.arrange(rhis, rqq, ncol = 2)
```



Solution

Potential influential observations

We are also concerned about potentially overly-influential observations. In this case, we know that the USA has won largest number of medals in 2016. And looking at the residual plots produced above, it seems that this country is associated with one of the largest residuals. So let's first just take a look at the OLS regression line produced **without** including the USA data point.

The table below will display the estimated coefficients for both regressions

```r
oly_noUSA <- oly %>%
    filter(Country != "UnitedStates")
oly_noUSA_lm <- oly_noUSA %>%
    lm(formula = M2016 ~ M2012)
coefs_noUSA <- tidy(oly_noUSA_lm)
comp_estimates <- tibble(all = coefs$estimate, noUSA = coefs_noUSA$estimate)
comp_estimates$estimate <- c("intercept", "slope")
kable(comp_estimates) %>%
    kable_styling(latex_options = "HOLD_position")
```

all	noUSA	estimate
0.7255	1.3328	intercept
0.9375	0.8423	slope

Use the code chunk below to plot both models and see how similar they are (or not!).

"'{r. eval = FALSE} comp_estimates_m <- comp_estimates %>% pivot_longer(-estimate, names_to="model", values_to = "coefs") %>% pivot_wider(names_from="estimate", values_from="coefs")

p8 <- oly %>% ggplot(aes(x=M2012, y=M2016)) + geom_point() + geom_abline(data=comp_estimates_m, aes(intercept=intercept, slope=slope, colour=model))
p8

```
**Solution**



What about $R^2$ and the residuals?

'''r
summary(oly_noUSA_lm)$r.squared
```

```r
oly_noUSA <- oly_noUSA %>%
    mutate(resids = oly_noUSA_lm$residuals)

rhis <- oly_noUSA %>%
    ggplot(aes(x = resids, y = after_stat(density))) +
    geom_histogram(colour = "steelblue", fill = "steelblue",
        alpha = 0.2, bins = 20) + geom_density(colour = "steelblue",
    fill = "steelblue", alpha = 0.2) + ggtitle("Residual Histogram") +
    xlab("Residual") + ylab("Histogram with Kernel density") +
    theme_bw()

rqq <- ggplot(oly, aes(sample = resids)) + stat_qq(distribution = qnorm) +
    stat_qq_line(distribution = qnorm, color = "red",
        size = 1.5) + theme(aspect.ratio = 1) + ggtitle("Residual QQplot") +
    xlab("Theoretical quantiles") + ylab("Empirical quantiles") +
    theme_bw()
```

```
grid.arrange(rhis, rqq, ncol = 2)
```

Solution

Leverage Leverage h_{ii} is defined for each observation, $1, ..., n$, and is the i^{th} diagonal element of the hat matrix:

$$H = X(X^TX)^{-1}X^T$$

where X is "design" matrix containing all of the regressors. For example, for SLR, since we have only one regressor "x", for which we have n values $(x_1, x_2, \ldots, x_n)$, the design matrix is:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}.$$

Intuitively, observations which are far from the mean of the explanatory variables will have higher **leverage**, i.e. they have greater influence on the fitted regression function. Changing the y value of a high leverage point even a little can really effect the fitted line.

Note, we can calculate the **Leverage** values *without* fitting all n models (i.e. n different regressions where we leave out one observation).

The rule of thumb for high leverage values is $2p/n$, where p is the number of regressors **including** the constant.

Highest leverage for medal tally model

To extract the *leverage* values from the *lm*() function output, we can use the *augment*() function from the **broom** package. See the code chunk below.

*[Students may want to print the** ***augment(oly_lm)*** **output to see what else it contains.]*

```
aug_oly <- augment(oly_lm)
aug_oly <- aug_oly %>%
    mutate(Country = oly$Country)

aug_oly %>%
    arrange(desc(.hat)) %>%
    dplyr::select(Country, .hat) %>%
    head(15)
# # A tibble: 15 x 2
#    Country                     .hat
#    <chr>                      <dbl>
#  1 UnitedStates              0.290
#  2 China                     0.203
#  3 RussianFed                0.174
#  4 GreatBritain              0.106
#  5 Germany                   0.0470
#  6 Japan                     0.0352
#  7 Australia                 0.0302
#  8 France                      0.0286
#  9 SouthKorea                  0.0206
# 10 Italy                       0.0206
# 11 Netherlands                 0.0135
# 12 Ukraine                     0.0135
# 13 Vietnam                     0.0132
# 14 IvoryCoast                  0.0132
# 15 IndependentOlympicAthletes 0.0132
```

Solution

Note: this is an informal way to identify points of high influence. Any .hat values that are relatively close to this cut-off, even if slightly over, may not be a big problem. We are looking for variables that substantially exceed this threshold.

```
aug_oly %>%
    arrange(desc(.hat)) %>%
    dplyr::select(Country, .hat) %>%
    filter(.hat > threshold)
```

Run the code chunk below to find the countries whose leverage values exceed the threshold. Which country has a leverage that is just over the threshold value? **Solution**

It may be easier to visualise this and use a histogram plot of the leverage values, with a vertical red line at the threshold value.

```
p9 <- aug_oly %>%
    ggplot(aes(x = .hat)) + geom_bar(colour = "blue",
    fill = "blue") + theme_bw()
p9 + geom_vline(xintercept = threshold, colour = "red",
    size = 1)
```

Cook's D Another influence measure that uses the response variable is **Cook's D**:

$$D_i = \frac{e_i^2}{pMSE}\frac{h_{ii}}{(1-h_{ii})^2}$$

Here
- e_i is the i^{th} residual
- p = number of explanatory variables (regressors, including the intercept)
- MSE is the mean squared error of the linear model ($MSE = SSE/(n-p)$)

Again, as a "rule of thumb", we check any point with Cook's D value greater than the same $2p/n$ threshold value. This incorporates leverage and the residual, so looks for influential observations horizontally and vertically.

```
aug_oly %>%
    arrange(desc(.cooksd)) %>%
    dplyr::select(Country, .cooksd) %>%
    filter(.cooksd >= threshold)
```

```r
p10 <- aug_oly %>%
    ggplot(aes(x = .cooksd)) + geom_histogram(colour = "blue",
    fill = "blue", bins = 30) + theme_bw()

p10 + geom_vline(xintercept = threshold, colour = "red",
    size = 1)
```

Which countries show concerning Cook's D values? Produce a table and a plot similar to those produced for leverage above. Solution

What does this mean? Ultimately, it is up to you to think of solutions. Given that the USA, Russian Federation and China had both high leverage and Cook's D, we need to investigate why. Typically, these countries always win a lot of medals at the Olympics. We could try removing them. This would change our analysis, or we could try adding variables to account for their influence.
It all depends upon our research problem and what makes sense.
We do not automatically remove influential observations though.