Statistical Thinking: Week 9 Lab
(with corrections in red)

Due 12noon Wednesday 21 October 2020

Introduction

The purpose of the Lab this week is to practice modelling data using both an i.i.d. population model and a simple linear regression model. The issue of transforming the data to enable the completion of an analysis will also be covered.

The Lab is structured in two main sections, Parts A and B. Part A explores using Maximum Likelihood Estimation to fit various models to the counts of Olympic medals from the 2016 Summer Olympics in Rio. Part B applies simple linear regression to the same series, using medal counts from the London 2012 Summer Olympic games.

Data for Lab 9 Before you begin, you'll need to download the two data files

rio_olympics2016.csv

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.

Lab Submission To obtain credit for this lab, you are required to complete a Lab 9 Submission Moodle quiz, which is due in Week 10 on Wednesday 21 October at 12noon. You are not required to submit the .Rmd or .pdf for this Lab.

Good luck and have fun!

[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 atheletes 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 Atheletes (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.

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

We'll first try to fit a single population model to the 2016 Olympic medal count data using Maximum Likelihood Estimation. We covered similar material in the Week 6 Lab, however here we also consider the fit of different models, and compare the MLE approach to an alternative Bayesian method.

Before we we start you'll need to set up your package libaries, read in the datafiles, and create the initial tibble for your analysis,as per the code chunk shown below. We will work with the data from the 2012 Olympics in Part C.

```
library(tidyverse)
library(MASS)
library(broom)
library(kableExtra)
options(digits = 4)

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

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

Question 1: Visual summary Produce a bar plot of the 2016 total medal count for all teams that won at least one medal. Be sure to appropriately label your axes.

Question 2: Numerical summaries Provide some useful numerical summary statistics for this series, and comment on any other interesting aspects you discover about the data. These descriptive statistics should be selected to answer help the reader be able to determine the answers to general questions such as:

- Is the distribution of the medal counts skewed to the right, skewed to the left, or symmetric?
- Which team won the most medals? And how many medals, and of what type, did they win?
- How many teams won only a single medal?
- What is the average medal *Total* for teams that won at least one medal, and its standard error?
- What is the variance (or alternatively, its standard deviation) of the medal tally?
- What are the quartiles of the total medal *Total* series?
- what range gives the number of medals won by those team whose medal *Total* is in the middle 50% of the total medal *Total* distribution?

Question 3: Which model to try? Consider the distributions we have available to use to model the medal total variable (*Total*). What are some aspects of this variable and its empirical distribution you would like to capture with a model?

Notice also that the medal totals are integer values, so a discrete distribution seems to be appropriate. The Poisson distribution has this feature, but are there any aspects of the Poisson distribution that are inconsistent with the data?

Let's move on to using Maximum Likelihood Estimation to fit the $Poisson(\lambda)$ to the medal tally (*Total*) series.

Part A.1 MLE for the $Poisson(\lambda)$ distribution

Use the **MASS::fitdistr** function, as shown below, to obtain the MLE for the $Poisson(\lambda)$ distribution, under the assumption that the data form a set of independent and identically distributed realisation of from this distribution. Store the output in an object named `Poisson_fit`, and store the "tidy" version of it in an object named **tidy_Poisson**.[4] In addition, produce a point estimate, and a 95% CLT-based confidence interval for λ.

```
Poisson_fit <- fitdistr(df16$Total, "Poisson")
tidy_Poisson <- tidy(Poisson_fit)
```

Question 4: Obtain the $Poisson(\lambda)$ point estimate and 95% confidence interval

Question 5: Interpret the resulting QQ-plot Use the code chunk below to produce a Quantile-Quantile plot (QQ-plot) for the fitted Poisson distribution. Explain what is being plotted on a) the vertical (y) axis, and b) the horizontal (x) axis? Applying the so-called "thick texta" test to the output of the code chunk below, would you conclude that the model fits the data well? Why or why not?[5]

```
params <- Poisson_fit$estimate

p <- ggplot(df16, aes(sample = Total)) + stat_qq(distribution = qpois,
    dparams = params) + theme(aspect.ratio = 1) + theme_bw()

p + stat_qq_line(distribution = qpois, dparams = params,
    color = "red", size = 1.5)
```

Where to next? Alas it seems that the Poisson distribution does not fit this data well, and we'll need to find another model to try. Although we have many distributions to choose from, of the discrete distributions available the only other one that might be potentially suitable is the Negative Binomial distribution.

Students can explore the Negative Binomial model for this data if they wish to, but there will not be any questions on the lab submission quiz about either the estimation or fit of the Negative Binomial distribution.

Instead we will move on to try using a continuous distribution.

Part A.2 MLE for the Lognormal distribution

Next, use the *fitdistr*() function to obtain the fit of a **Lognormal distribution**. Store the output in an object named *Lognormal_fit*.

Question 6: Obtain point estimates and 95% confidence intervals for LogNormal parameters [Note correction] Produce point estimates, and a 95% CLT-based confidence interval, for the parameters, μ and σ, from a $Lognormal(\mu, \sigma^2)$ distribution. (Note: the output from *fitdistr*() function refers to μ as "meanlog" and σ as "sdlog".)

[4] Note that some students may have previously learned that the MLE for the $Poisson(\lambda)$ distribution is the sample average, i.e. $\hat{\lambda}_{MLE} = \bar{x}$, however by using the *fitdistr*() package we will automatically also deliver other useful information such as the standard error of the estimator.

[5] You may want to review relevant material from the Week 7 lectures.

Question 7: Produce the corresponding QQ-plot. How does it look? Do you think the $Lognormal(\hat{\mu}_{MLE}, \hat{\sigma}^2_{MLE})$ distribution fits the data better than the $Poisson(\hat{\lambda})$ model? Why or why not?

The relationship between the LogNormal and the Normal distributions Recall that if $X \sim Lognormal(\mu, \sigma^2)$ then $log(X) \sim N(\mu, \sigma^2)$. This means that $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} LN(\mu, \sigma^2)$ will only be suitable if the observations $x_1, x_2, \ldots, x_n$ can take on only positive values. However, if we take the (natural) logarithm of these x_i values, they can take any value in $\mathbb{R}$.

Hence, rather than working with the $Lognormal(\mu, \sigma^2)$ distribution on $\mathbb{R}^+$, we can **transform the *Total* variable** and work with its logarithmic value on $\mathbb{R}$. The MLEs for μ an σ^2 should be the same no matter whether we fit the LogNormal distribution to the original data, or the Normal distribution to the log-transformed data.

Question 8: Visual summary (*logtotal*) Add the $logtotal = \log(Total)$ variable to the **df16** tibble. Then, produce an appropriate distributional plot of the 2016 total *logtotal* medal count for all teams that won at least one medal. Be sure to appropriately label your axes.

Question 9: Numerical summaries (*logtotal*) Provide some useful numerical summary statistics for the *logtotal* series, and comment on any other interesting aspects you discover about this data.

Do the MLE results match? [Note correction] Confirm that the MLE for μ and σ and their standard errors, obtained from fitting the $N(\mu, \sigma^2)$ model to the *logtotal* series are identical to those produced from fitting the $LogNormal(\mu, \sigma^2)$ model to the *Total* series.

Question 10: Produce the QQ-plot from the Normal model fit of the *logtotal* series.

Question 11: MLE-based prediction Use the fitted Lognormal distribution, obtained using the MLE approach, to estimate the following prediction probability:

$$\Pr(\ Total_{n+1} > 50 \mid \mu,\ \sigma^2).$$

Discuss the construction of a Bootstrap-based 95% confidence interval for this quantity. Once you have calculated this quantity, compare it to the corresponding *empirical proportion* of the teams from the 2016 Olympics that achieved more than 50 medals.

Note that here we are interested in the estimated probability that a thusfar unobserved team's medal count, denoted as $Total_{n+1}$, could be realised as a value of 50 or more.[6] For this you will need to use the fitted cdf from either the $Lognormal(\hat{\mu}_{MLE}, \hat{\sigma}^2_{MLE})$ or the $N(\hat{\mu}_{MLE}, \hat{\sigma}^2_{MLE})$, noting that an equivalent representation of the desired probability is

$$\Pr(\ log(Total_{n+1}) > log(50) \mid \mu,\ \sigma^2).$$

```
# estimated probability (using Lognormal fit)
muhat <- Lognormal_fit$estimate[1]
sigmahat <- Lognormal_fit$estimate[2]
prob_lognormal <- plnorm(50, meanlog = muhat, sdlog = sigmahat,
    lower.tail = FALSE)
prob_lognormal
```

[6] Of course there are no other unobserved teams for the 2016 Olympics! So this is a bit of a hypothetical question - we'll need to imagine that such an as-yet unobserved event could actually occur so that you can practice estimating such a model quantity using the MLE approach. Perhaps one could assume that the same fitted distribution will be appropriate for the 2020 (2021?) Olympics in Tokyo!

```
# estimated probability (using Normal fit)
muhat <- Normal_fit$estimate[1]
sigmahat <- Normal_fit$estimate[2]
prob_normal <- pnorm(log(50), mean = muhat, sd = sigmahat,
    lower.tail = FALSE)
prob_normal

# empirical probability
n <- df16 %>% summarise(number = n())
count <- df16 %>% filter(Total > 50) %>% summarise(number = n())
prob_empirical <- count/n
prob_empirical
```

Part B: Simple linear regression

In this section we'll 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. To fit the SLR model, we need to find a way to *join* the two separate tibbles, df16 and df12.

Fortunately the **dplyr** package provides the *full_join()* function to do this merge the *Country* and *Total* columns from the two tibbles we have (df12 and df16) to create a single tibble, named "oly".

```
oly <- full_join(df16[, c("Country", "Total")], 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?

Question 12: Why do some countries have missing values (*NA*)?[7] We need to do a little housework 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. (why?)

Be sure to check that the replacement works as expected.

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

In this Part 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).

[7] You could also fix up the issue with Kuwait and the IOA, as discussed in footnote 2, though that has not been done yet here.

Question 13: Produce a scatterplot of *M2016* on *M2012*. 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.

The model you will fit is

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

- The regression model explains how response variable, y, changes in relation to 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.

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.

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

Question 14: Write down an appropriate expression for the fitted regression line, and provide a suitable interpretation for the estimated parameters.

Residuals 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:

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

Question 15: Do you notice any pattern in the residuals?

Question 16: 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.

```
summary(oly_lm)$r.squared
```

A good R-squared value does not guarantee a good fit. 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 discernable patterns in the residuals)

We also plot the **histogram of residuals**. For a good fit the shape should be approximately symmetric and bell-shaped. (One could also formally do a QQ-plot against the corresponding theoretical normal quantiles.)

Potential influential observations

We are also concerned about potentially overly-influential observations. In this case, we know that the USA has wont largest number of medals in 2016. And looking at the residual plot 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

```
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()
```

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

```
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
```

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

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

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).

Highest leverage for medal tally model

To extract the *leverage* values from the *lm*() function output, we can use the *augment*() function[^8] from the **broom** package. See the code chunk below.

[^8] 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)
```

```
threshold <- 4/n
```

Note that this is an informal way to identify points of high influence. .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)
```

Question 17: 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? Produce the histogram plot of the leverage values, adding a vertical red line at the threshold value, as shown in the code chunk below.

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

p9
```

(Do you think Germany's data has much of a leverage problem?)

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.

Question 18: Which countries show concerning Cook's D values? Produce a table and a plot similar to those produced for leverage above.