Exercise 2

PLSCI 7201

9/4/2020

Summary

The aims of this exercise is to introduce students to correlation and simple linear regression using ordinary least squares (OLS), and to learn how to perform some basic model diagnostics and identify outlier samples. The goal of ordinary least squares is to explore the relationship between two continuous variables and make inferences on the strength or direction of this relationship.

Example

I will first walk you through a simple example of how to perform OLS "by hand" as well as using some wrapper functions in R. The exercise can be broken down into four main parts: basic exploratory analyses, fitting a linear model, model diagnostics and outlier detection, and data preprocessing (i.e. outlier removal).

Exploratory analysis

You can learn a lot about the expected relationship between two variables by generating a basic scatter plot and performing some simple correlation analyses. The two most popular methods to estimate the correlation between variables is Pearson's correlation and Spearman's correlation. We covered Pearson's correlation (r) in the lecture and showed how it can be expressed from a geometric perspective. Pearson's correlation is used to explore the linear relationship between two or more variables. Spearman's correlation explores the relationship between the ranks of two or more variables. Although it wasn't covered in class it is worthwhile reading up on Spearman's correlation. You will most likely come across it again in the future. The (Wikipedia page)[https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient] may give you some insight.

First load the data, check out the first few lines and create a scatter plot.

```
exData <- readRDS("exampleData.RDS")

# head() will print the first six lines (or elements) of an object to the
# console.
head(exData)
```

```
         X         Y
1 22.07591 11.417032
2 23.16354  9.061673
3 19.39786 11.676959
4 21.60261 10.319963
5 20.43702 11.944056
6 21.84358  9.161665
```

```
# tail() does the same for the last six lines.
tail(exData)
```

```
          X         Y
```

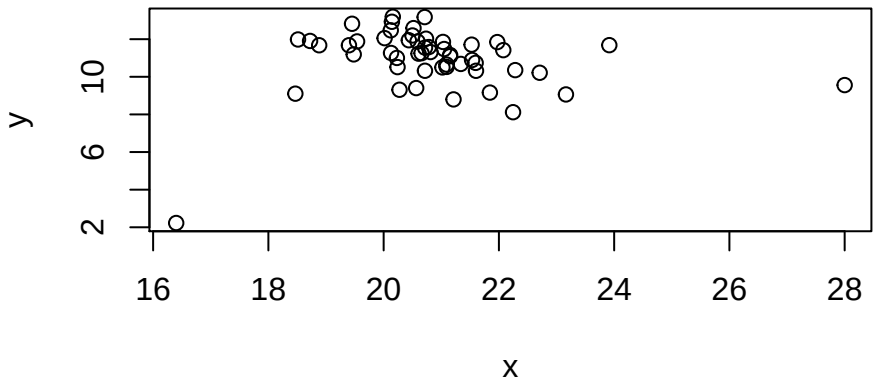
```
47 20.15915 13.191362
48 19.53922 11.880763
49 21.21248  8.798378
50 20.22992 11.003177
51 28.00000  9.560000
52 16.40000  2.228000
```

```r
# Create some basic statistics with summary()

summary(exData)
```

```
       X               Y         
 Min.   :16.40   Min.   : 2.228  
 1st Qu.:20.16   1st Qu.:10.463  
 Median :20.72   Median :11.261  
 Mean   :20.83   Mean   :10.951  
 3rd Qu.:21.39   3rd Qu.:11.876  
 Max.   :28.00   Max.   :13.191  
```

```r
plot(exData$X, exData$Y, xlab = "x", ylab = "y")
```



It looks like there may be a weak relationship between X and Y, and there are a few data points that may be problematic.

Next, we'll calculate the correlation between X and Y using the two approaches mentioned above.

```r
# Pearson correlation
cor(exData$X, exData$Y, method = "pearson")
```

```
[1] 0.05006338
```

```r
# Spearman correlation
cor(exData$X, exData$Y, method = "spearman")
```

```
[1] -0.3991292
```

Least squares regression

Now we'll fit a linear model to the data. First we'll do this "by hand" using the methods discussed in class. Recall, we can estimate the coefficients for OLS using

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

We will regress our variable Y onto X. This means that we will treat Y as the response variable and X as the dependent/predictor variable. Matrix algebra should be covered in the course notes, but here is a very quick dirty summary of what the equation above means and what we are doing. We are transposing (flip a matrix

on its side) our matrix of predictors (intercept and X) and multiplying this by our matrix of predictors. This multiplication works out because the number of columns of the transposed matrix matches the number of rows of our predictor matrix. This will return a $p \times p$ matrix where p is the number of predictors. We are taking the inverse ($^{-1}$) of this $p \times p$ matrix. This is then multiplied by the product of the transpose of the predictor matrix and $\mathbf{y}$. The dimensions of y is $n \times 1$ and $\mathbf{X}$ is $n \times p$.

In R, the `%*%` operator does matrix multiplication, `t()` transposes a matrix or vector, and `solve` will take the inverse of a matrix. The easiest way to set up your predictor matrix is using the `model.matrix` function. This function takes a formula and a data frame and will construct the corresponding predictor matrix. The way formulas are specified in R (at least in the base functions) is by separating the response and predictor variables with a `~`. Suppose we have three predictors (a,b, and c) and a single response (z) contained in a data frame called `tmp`. We want a to create a predictor matrix with the three predictors above and an intercept term. We would use `model.matrix(~ 1 + a + b + c, data = tmp)`. We don't need to specify the response variable because it doesn't tell us anything about the predictors. If we don't want an intercept then the code would be `model.matrix(~ 0 + a + b + c, data = tmp)`. We'll use this in the next few lectures, so you'll have more time to get used to the function.

```
Xincd <- model.matrix(~1 + X, exData)

solve(t(Xincd) %*% Xincd) %*% t(Xincd) %*% exData$Y
```

```
                  [,1]
(Intercept) 9.85629153
X           0.05257968
```

We can also do OLS in R using the linear model wrapper function `lm()`. It accepts a formula like the one used for `model.matrix` and a data frame (or response and predictor vectors or matrices). `lm()` should include the intercept by default, so the `1 +` below is not really necessary.

```
mod1 <- lm(Y ~ 1 + X, data = exData)
mod1  # prints the coefficents
```

```

Call:
lm(formula = Y ~ 1 + X, data = exData)

Coefficients:
(Intercept)            X
    9.85629      0.05258
```

```
summary(mod1)  # returns the SE and p-values
```

```

Call:
lm(formula = Y ~ 1 + X, data = exData)

Residuals:
    Min      1Q  Median      3Q     Max
-8.4906 -0.5034  0.3329  0.9513  2.2751

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.85629    3.09889   3.181  0.00252 **
X            0.05258    0.14834   0.354  0.72450
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

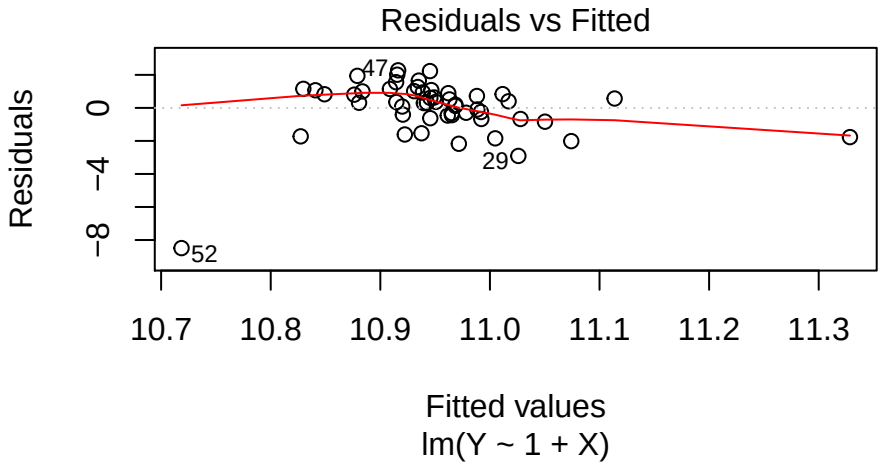
```
Residual standard error: 1.7 on 50 degrees of freedom
Multiple R-squared:  0.002506,  Adjusted R-squared:  -0.01744
F-statistic: 0.1256 on 1 and 50 DF,  p-value: 0.7245
```

Model diagnostics

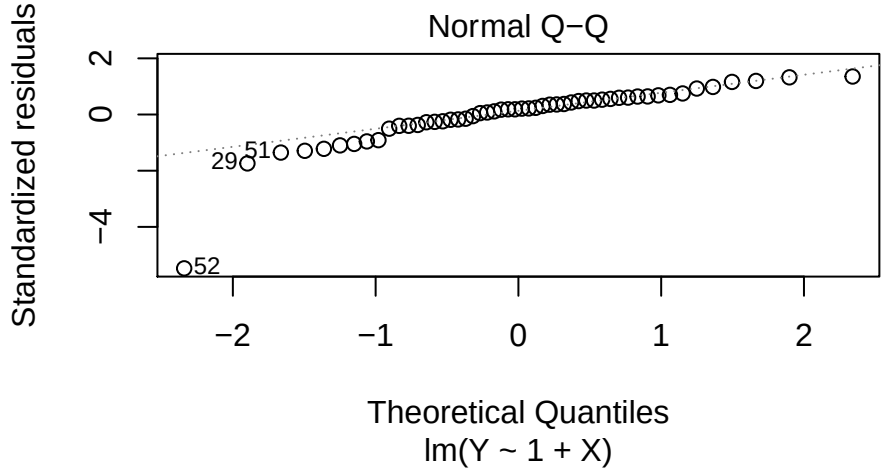
Now that we have our linear model stored in an object, we can run some diagnostics to see if we're violating any assumptions of OLS. This is really easy because we can just call `plot()` on our object. By default R will plot five plots and you will have to press enter to switch to the next. Alternatively, we can plot only the interesting ones using the `which` argument in`plot()`. `which` accepts a number from 1 - 5.

We'll generate a residual vs fitted plot (`which = 1`), a scale-location plot (`which = 3`), and a QQ plot (`which = 2`). Some of these points are numbered. The numbers refer to the corresponding rows in the data frame. With a lot of these plots we are not looking for perfect patterns. Most of the time we are looking for very bad points. Our decisions are based on the size of our data set, how bad the point(s) looks, and whether a suitable solution exists.

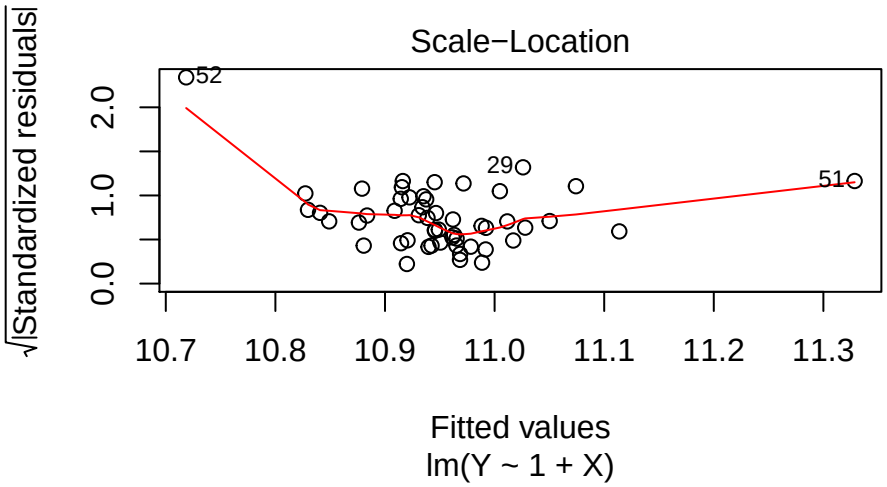
```
plot(mod1, which = 1)
```




```
plot(mod1, which = 2)
```




```
plot(mod1, which = 3)
```



Now we'll look for leverage points and use the general rule that influential points are those with a value that is more than twice the mean value over all points. We can use the `which()` function (different from above) to return the elements that meet this requirement. `which()` will return an index, so we need to wrap `which` in brackets to pull out the element of the hat vector (vector of leverage statistics).

```
levPts <- hatvalues(mod1)  # hatvalues() computes the leverage and accepts a model
meanLev <- mean(levPts)

levPts[which(levPts > 2 * meanLev)]  # find the points that have a hat value greater than twice the mea
```

```
        16         51         52
0.09174227 0.41055071 0.16854829
```

The last two points (51 and 52) look pretty lousy, so we'll drop them from the data. **Make sure you have good reason to remove a data point. I am being very lazy here and just dropping these points. However, not all points that are influential necessary need to be removed. Keep this in mind when you are fitting you own data.**

```
newData <- exData[-c(51, 52), ]
```

Your turn...

Two of the most important traits in soybean is seed oil and seed protein content. We are interested in the relationship between these two traits. Seed from 52 soybean varieties were randomly collected and seed oil and protein content were measured. The measurements are in g/100g of seed. Use protein as the predictor variable and oil as the response variable. The file is called `SoybeanSeedQual.RDS`

Section 1 - Exporatory data analysis

- Create a scatter plot. Does there seem to be a linear trend between the variables?
- Calculate the correlation between seed oil and protein using Pearson's and Spearman's method. Does Spearman's and Pearson's correlation return the same (or very similar) values? Report the coefficients for each. We will refence them later in the exercise.

Section 2 - Least squares regression

- Estimate the coefficients of the linear model "by hand" using OLS. Compare these estimates with those from `lm()`. Do the estimates of the model cofficients agree? Interpret the coefficients. What do they tell us about the relationship between oil and protein content?

Section 3 - Model diagnostics

- Generate a residual vs fitted plot (`which = 1`), a scale-location plot (`which = 3`), and a QQ plot (`which = 2`). Do you believe we are violating any assumptions of OLS (linearity, constant variance, etc.)? Are there any problematic points? If so, which?
- Calculate leverage statistics and identify any points that may be leverage points. Do any data points meet the "2 $\times$ average rule"? If so, which points?

Section 4 - Preprocessing

- Using the information from the model diagnostics (residual plots, QQ, leverage, etc.) do you believe there are any points that should be removed? Explain you reasoning for removal if applicable.

Section 5 - Drawing conclusions

- If you decided to drop any data points, then refit the model and go through the same steps above for model diagnostics, outlier checking, etc.
- Did the estimates of the model coefficients change much compared to the first model? If so, which coefficients changed? What does this model tell us about the relationship between oil and protein content?
- Compute Pearson's and Spearman's correlation of the new, cleaned-up data set. Compare these values with those obtained from the analyses on the original data. Which correlation approach is more robust? Give some reasoning to why one may be more robust.