Statistical Thinking: Week 2 Lab

Due 12noon Monday 17 August 2020

Introduction

Our focus this week is to practice some of the more commonly used **R** functions associated with the tidyverse, namely functions from the *ggplot2* and *dplyr* packages. Along the way some additional helpful functions will be covered. An outline of the Lab 2 document is provided below.

A. Instructions

B. Exercises using *ggplot2* and *dplyr* functions

C. Lab submission instructions

A. Instructions

Before you begin the exercises in part B, review each of these detailed instructions.

Establish your working environment

Learn/Review information about **RProjects** and **working directories**, by reviewing the two links below:

- article on Working Directories.
- article on RProjects.

Initiate your R Markdown file

Create, save and name a new .Rmd file, by following these steps:

- Start a new .Rmd file from the File –> New file –> R Markdown... menu and select a New R Markdown "Document" with HTML Default Output Format. This file is actually a template from **RStudio** to remind users how to use **R Markdown** files.
- In the **R Markdown YAML** section:
  - give your document a suitable title (e.g. Statistical Thinking Lab 2")
  - put your name and student-id as the author information
  - set the date "Week 2, 2020".
- Review the information provided in the .Rmd file
  - Save your file as **Lab02nnnnnnnn.Rmd**, with your Monash student ID number replacing the segment **nnnnnnnn** in the file name.

Install and load the required packages

Insert a new code chunk and load the package libraries for the *tidyverse*, *knitr*, *lubridate*, *gridExtra* and *kableExtra* packages, as shown in the code chunk below. (If any of these are not already installed on your machine, install them first before continuing.)

Review the data to be used

Load the *economics* dataset into your **Lab02nnnnnnnn.Rmd** file. (You can either insert a new code chunk, as shown below, or add it to the previous code chunk shown above.)

- What data does the *economics* data file contain? (Hint: Read the *economics* help file!)
- Confirm whether the *economics* data frame is in a tidy data format.

Review the relevant help files

Review help files for each of the following:

- *economics*: dataset from the *ggplot2* package
- *year()*, *month()*: date functions from the *lubridate* package
- *factor*: basic introduction about factors from base **R**
- *as_factor()*: function from the *forcats* package
- review the functions from *ggplot2* and *dplyr*
- review the help file for any of the loaded packages if they are not familiar to you,

Remember to also review help files as you need them, and refer to the many other resources suggested if you are unsure of how to complete any of the Lab tasks.

Submission format requirements

There are requirements for formatting the **Lab02nnnnnnnn.Rmd** file. The exercises in section **B.** are labelled with numbers, such as **Exercise 1**. In your **Lab02nnnnnnnn.Rmd** file, create a (sub-)section for each exercise by following a blank line with a new line containing two hashtag symbols (##), followed by the word **Exercise**, and then followed by the exercise number. For example, a section set aside for **Exercise 1** would be typed on an otherwise blank line at the start of the section as:

```
## Exercise 1
```

Each section in your submission must contain both answer text and any requested code chunks required to complete the exercise. Complete answers must include a statement of what the relevant code chunk produces, even if the exercise does not asked for a specific text answer. This is good practice as in a data analysis report all graphs and tables must always be described somewhere in the text of the report. Proofread your submissions, checking for correct spelling, grammar and punctuation as well as to ensure the text expresses what you intend.

Also, be sure that you keep the global option to show ('echo') all code chunks by retaining:

```
knitr::opts_chunk$set(echo = TRUE)
```

in the first code chunk. This setting is already included in the **RStudio R Markdown** template. For some Exercises below, the code chunk that results from the instructions will not produce any visible output, but the code must still be shown and evaluated at that point in the file.

Use the kable() to format tibble print output

Many of the exercises below request that some or all rows of a tibble should be printed. To improve the visual presentation of this output, use the *kable()* function from the *kableExtra* package to improve the printed output. This can be achieved by appending the *kable()* function with a *pipe* operator to the relevant line of code that prints the tibble. For example, to view the first several lines of the *economics* tibble, use:

```
head(economics) %>% kable()
```

Some modification of the resulting "kable" format can be achieved using an additional *kable_styling()* function. Other packages, such as *pander* and *xtable* are also useful for formatting tables.

B. Exercises using ggplot2 and dplyr

Exercise 1. Produce a scatterplot of the *median duration of unemployment, in weeks* (variable *uempmed*) against the corresponding *date*.

- Note that, by convention, the expression "plot y against x" refers to a plot of the points (x,y), with "x" coordinates on the horizontal axis (x-axis) and "y" coordinates on the vertical axis (y-axis).

Exercise 2. Repeat **Exercise 1** without printing the plot, rather assign the plot to an object named `p1`. Then, add (or modify) each of the following aspects of `p1`:

- Change the colour of the data points to "blue".
- Add a title of the plot, stating "Median duration of unemployment" by adding a layer that uses the `ggtitle()` function.
- Change the y-axis label to "weeks", by adding a layer using the `ylab()` function.
- Set the font size of axis labels, by adding `theme(text = element_text(size = 8))` to the plot object.
- Note, do not yet print the `p1` plot object. It will be printed later.

Exercise 3. Repeat **Exercise 2**, this time saving the result as a plot object `p2`, having eliminated all points where the median unemployment level is less than 12 weeks[1]. This may be achieved by first passing the *economics* tibble through an application of the `filter()` function, using the *pipe* operator `%>%`, and then passing this result to `ggplot()`, again using the *pipe* operator. In addition:

- Change the default range of the x- and y-axes to match those produced for `p1`, by adding the two layers:
  `+ xlim(range(economics$date)) + ylim(range(economics$uempmed))`
- Modify the title to indicate that the data have been filtered, e.g. "Median duration of unemployment (>= 12 weeks)".

Exercise 4. Use the `grid.arrange()` function to print the two plot objects produced above, named `p1` and `p2`. Place the printed plots side-by-side next to each other, using the function option `ncol=2`. (Alternatively you could specify `nrow=1`, however only one option, either `nrow=1` or `ncol=2`, is required.) Once printed, double check all titles and labels to be sure they fit the print space available.

[1] An early version of this Lab incorrectly stated *months* here. The error was corrected on Monday 10 August 2020 at 5pm.

Exercise 5. The *arrange()* function reorders the rows of a datafile according to a given logical rule. Use this function to first sort all rows of the *economics* tibble according to the personal consumption expenditures variable, *pce*. Then plot median duration unemployment, in weeks, against the reordered row number in the sorted tibble. (Hint: after sorting the original tibble, you can add a new variable to the sorted tibble that contains the row numbers `1:n`, where `n=nrow(economics)` represents the number of rows in the sorted tibble. Though not required, these steps could be achieved without actually saving the sorted tibble via the *pipe* operator.) In addition:

- Ensure your plot has a suitable title and suitable x- and y-axis labels.
- Colour the (sorted) points a colour other than black or blue.
- Save your plot object with the name `p3`.
- Print the objects `p1` and `p3` side-by-side, next to each other in a single row.

(Notice you do not need to re-create the `p1` plot object', since it has already been produced and saved.)

Exercise 6. The *select()* function extracts a subset of variables (columns) from a tibble. Use this function to extract two variables, *date* and *uempmed*, from the *economics* tibble. Print the first portion of the resulting tibble.

Exercise 7. Variables from a tibble can also be excluded from selection by specifying these variables with a minus sign (-) in the *select()* function. Produce a tibble that contains all variables in the *economics* data file EXCEPT for *pce* and *unemploy*, using *select(-pce, -unemploy)*. Then, use the *head()* function to print a portion of the resulting tibble.

As introduced in the Week 1 Lab, the *mutate()* function is used to create new variables in the tibble. Existing variables, together with the so-called *creation* functions (arithmetic operator examples listed below) and other **R** functions may be used to generate the new variables.

Arithmetic operators:

a) `+` (sum)

b) `-` (difference)

c) `*` (multiply)

d) `/` (divide)

e) `^` (raise to a power)

Exercise 8. Add four new variables to the *economics* tibble to create a new tibble named *econPlus*. The new variables must be named and created as follows:

1. *index*, comprised of the integer values from 1 to `n=nrow(economics)`
2. *pct_unemploy*, the percentage of the total population that is unemployed each month
3. *pce_sqrt*, the square root of personal consumption expenditures (in billions of dollars raised to the 0.5 power), and
4. *rx*, comprised of independently generated (pseudo-)random draws from a standard normal distribution. (These may be obtained using **R**'s *rnorm()* function.)

- Note that before generating *rx* you must set the random seed value used by **R** to generate the random sample.[2] For this exercise, include the line

```
set.seed(20201234)
```

in the **R** code chunk immediately before generating *rx*.

- Finally, use the *tail()* function to print the final rows of the resulting *econPlus* tibble.

Exercise 9. Recall from Lab 1 that the *summarise()* function reduces the full data down to a low dimensional summary. Commonly used summary functions include:

a) The mean value: **mean()**

b) The median value: **median()**

c) The variance: **var()**

d) The square root of the variance, or standard deviation: **sd()**

e) The interquartile range: **IQR()**

f) The median absolute deviation: **mad()**

g) The minimum: **min()**

h) The maximum: **max()**

i) A quantile: **quantile()**, e.g. the lower (empirical) quartile of *rx* would be obtained as *quantile(economics$rx,0.25)*.

Exercise 9 involves finding the minimum, median and maximum values associated with the *pct_unemploy* variable created in **Exercise 8**. Put the summary results of this exercise in a tibble named *econSummary*, and print the entire tibble.

Exercise 10. We now want to consider the values of summary statistics considered in **Exercise 9**, but for each decade (period of 10 years) that the data are available, rather than over the entire sample. This can be done using the *summarise()* function together with the *group_by()* function.

Before we can use this function, several intermediate steps are required. First, a new variable (call it *decade*) that identifies the group membership must be created. This *decade* variable will have values equal the *date* year rounded to the tens digit corresponding to the first year in the relevant decade. For example, we want the *decade* value corresponding to a *date* value with year being 1967 to equal 1960. To convert the *date* variable (as formatted in the *economics* and *econPlus* tibbles) to a variable *year*, we can use the *year()* function from the *lubridate* package. Though it is possible to use base **R** functions to manipulate dates, the *lubridate* package makes working with dates much easier!

- Follow these steps to create the *decade* variable:
  - Use the *mutate()* function, together with the lubridate *year()* function, to create a new variable, named *year*.

[2]Why do we set the random seed value? To ensure the report is reproducible. By setting a known seed value prior to generating (pseudo-)random variates on a computer, the drawn values will not change each time the code is run. In contrast, without setting this random seed value, **R** will produce a different sample each time the code is run.

- *Pipe* the result from the above *mutate()* function into a second use of *mutate()* where the *year* value is rounded to a year corresponding to the first year of the decade. Name this second new variable as *decade*. This second step can be achieved using *mutate(decade = round(year - 4.5, -1))*.
- Retain both the *year* and the *decade* variables in the *econPlus* tibble.
- Check that the new variables have been defined correctly by selecting the *decade*, *year* and *date* variables and using the *head()* function to print the first several rows of the resulting *econPlus* tibble. (This table used for checking does not need to be included in your submitted .Rmd file.)

Now the constructed *decade* variable may be used to identify the cases (rows) that share the same *decade* value. The same summary statistics computed in **Exercise 9** can be computed for each group, as determined by the values of the **decade** variable and using the *group_by()* command from the *dplyr* package.

Notice there is a problem here. The groups for 1960 and 2000 have fewer observations than those for 1970, 1980, 1990 and 2000. So the summary statistics involving these two decades with incomplete data should not really be compared with the summary statistics for the intervening decades. Filter out the observations from the decades labelled 1960 and 2010 from your final printed table.

Explicit instructions to complete the above are as follows:

- Copy your code chunk from **Exercise 9** and paste it in position for **Exercise 10*. Note that all named code chunks must have unique names.
- Filter the *econPlus* tibble to exclude rows where *decade* equals either 1960 or 2010. This may be done using a *pipe* operator and *filter((decade > 1960) & (decade < 2010))*.
- Group the filtered *econPlus* data using *group_by(decade)*. This may be done before summarising the relevant variables using a *pipe* operator and *group_by(decade)*.
- Once the summaries have been computed, print the resulting tibble.

Exercise 11. Produce side-by-side boxplots for *pct_unemploy*, for each of the *decade* groups beginning in 1970 and up to 2010 (i.e., excluding 2010).

Together with the *group_by()* function, the *dplyr summarise()* function was able to identify groups using the numerical values of *decade*. However, to designate groups for the *ggplot()* function, we need to convert the *decade* variable to a *factor*, so that the numerical values are reduced to a set of unique *level* values.

The code chunk below produces the desired boxplots.

```
econPlus %>% filter((decade >= 1970) & (decade < 2010)) %>% ggplot(aes(as_factor(decade),
    pct_unemploy)) + geom_boxplot()
```

Insert and modify the code chunk provided above as follows:

- Add an appropriate title to the plot.
- Change to more suitable axis labels.
- Change the colour so that the boxplot for each decade has a different colour. Be sure that a legend appears that identifies the colour with the decade.
- Remove the legend title by adding the layer

  `theme(legend.title = element_blank())`
- While no other changes are formally required for the lab submission, do make any other changes to the resulting plot that you feel would improve it!

C. Submission instructions

Lab 2 submissions must be completed before 12noon on Monday 17 August 2020. To submit the Week 2 Lab, you must complete the Week 2 Lab Submission "quiz" in Moodle (with correct answers), following all instructions provided on Moodle.

A complete submission will include:

i. Completion of the 10 exercises detailed above in the **Lab02_IDnnnnnnnn.Rmd** file, appropriately named as described in part A.

ii. Uploaded copies of the completed **Lab02_IDnnnnnnnn.Rmd** file and a rendered version of the same file in **.html** format.

iii. An uploaded copy of a print of the `Lab02_IDnnnnnnnn.html` document as a .pdf document, named **Lab02_IDnnnnnnnn.pdf**.

iv. Complete the Week 2 Lab Submission "quiz" questions.

You can attempt the submission quiz as many times as you wish, however ONLY the final attempt will be marked.

Note that a quiz that is open but not yet submitted when the quiz is due will be treated as the final attempt.

GOOD LUCK and HAVE FUN!