Linguistics 337: Introduction to R/Rstudio

K.Bott - kbott@reed.edu

8 September 2020

Introduction

Welcome to R/RStudio – a great add to your analytical/methods toolset. We will work with R/Studio a handful of times as a class before fall break, and you may continue to use these methods later in this class or other classes.

In case you are not familiar with R/Rstudio, here are some basic first steps and reminders. If you are comfortable with R/Rstudio, use this as an opportunity to review what you know and strengthen your skills.

One of the benefits of working with R is that your work will be *reproducible*. When working in R, I would strongly recommend that you use R scripts (*.R files), so that your code is in one place and can easily be used (and reviewed) by both you and collaborators.

To start, open up a new R script (open RStudio > File > New File > RScript). You will use this to document your code.

For each of the lines below that `look like this` (different font = code), type the code into your R script, then highlight the code in R and run the code by pressing *command* and *enter* simultaneously.

If you would prefer, you can also use the “run” button you see in the interface. For the purpose of this lab, we will assume that you are working on the Rstudio Server (rstudio.reed.edu). To access the server, use your “normal” Reed/Kerberos login (same for Moodle, etc).

Set-up: Install packages + load libraries

You should not need to do this on the RStudio server, because we have installed packages for you. If you are working on a desktop version of R, then you will need to do the following:

```r
install.packages("tidyverse")
install.packages("gapminder")
```

Whether you are on the server or a desktop verion of R, every time you use a library, you must load it:

```r
library(tidyverse)
library(gapminder)
```

Look at your data

Try each of the below for different ways inspecting your data.

```r
gapminder
View(gapminder)
glimpse(gapminder)
head(gapminder)
```

Each presents the data slightly differently. It is *highly recommended* that you look at your data before working with it, although how you look at your data is a matter of personal preference.

Summary measures

Descriptive statistics are fairly easily calculated in R, and can provide useful information about your data.

For example, how many observations do you have in the `gapminder` dataset?

```r
gapminder %>%
  count()
```

How many observations do you have per *continent* in the `gapminder` dataset?

```r
gapminder %>%
  count(continent)
```

Here is an example, finding the mean GDP per capita across the `gapminder` dataset:

```r
gapminder %>%
  summarize(mean(gdpPercap))
```

The above code takes dataframe `gapminder`, sends it to the next line, which is calculates `mean` (a tool within the `summarize` toolset) on specified variable `gdpPercap`.

Some other handy tools are below:

```r
summarize(max())
summarize(median())
summarize(var())
summarize(sd())
summarize(min())
summarize(IQR())
```

your turn

Means and medians are differently affected by outliers. Calculate the median GDP per capita and compare the two numbers.

The power of group_by()

We're looking at GDP data for the entire globe. Are there any patterns by continent? As a function, `group_by` puts the data into groups before R can do anything else to it. This can be very useful in data work. For example, the below code calculates mean GDP per capita by continent.

```r
gapminder %>%
  group_by(continent) %>%
  summarize(mean(gdpPercap))
```

Note:: Combining this with the `count` *command we saw above, you can use this to answer questions like "how many observations do I have per continent, per year?"*

You could also calculate median for each continent, to compare the mean + median values:

```r
gapminder %>%
  group_by(continent) %>%
  summarize(median(gdpPercap))
```

Note that you can use multiple group_by() arguments. Run the below code and consider the output:

```r
gapminder %>%
  group_by(continent, year) %>%
  summarize(median(gdpPercap))
```

. . . that is quite a LOT of numbers. Perhaps it's time to look at some visualization.

Data visualization

We will look at this more next class; here's a preview. We will make graphs using package `ggplot2`, which is part of the `tidyverse` tools you used earlier.

All ggplots have three component: data + aes(thetics) + geom(etry of the graph).

Histograms

Try an example for the `gapminder` data:

```r
ggplot(data = gapminder, aes(x = gdpPercap)) +
  geom_histogram(bins = 60)
```

`facet_wrap()`

Here's a useful trick (aka 'small multiples'): like we used `group_by()` for summary statistics above, you can use faceting for variables:

```r
ggplot(data = gapminder, aes(x = gdpPercap)) +
  geom_histogram(bins = 60) +
  facet_wrap(~continent)
```

Boxplots

Boxplots can be useful for showing the distribution of data. Here's GDP for the whole dataset:

```r
ggplot(data = gapminder, aes(y = gdpPercap)) +
  geom_boxplot()
```

. . . .and by continent. . .

```r
ggplot(data = gapminder, aes(x = continent, y = gdpPercap)) +
  geom_boxplot()
```

Can you imagine some use for this graphic?

```r
ggplot(data = gapminder, aes(y = gdpPercap)) +
  geom_boxplot() +
  facet_wrap(~continent)
```

Practice on your own

Load and examine data

Download pnw.csv from Moodle. We will use the tool `read_csv()` from the tidyverse tools to read in the CSV, and name the data `pnw`.

Loading: Rstudio server

If you are on Rstudio server, first upload the file (lower right hand panel), then "Import data" in the upper right hand panel (using "Text (readr)"). Rstudio will load in your data and name the data what your *.csv was named (in this case, your data will be named "pnw")

Loading: Desktop

If you are on desktop, note that you *will need to change the filepath* to help R/Rstudio find where your file is. (Recommended: Find file in finder, right-click [option-click] on the file, and you'll be able to see the filepath. We will walk through this in class.)

```r
pnw <- read_csv("/Users/bottk/Downloads/pnw.csv")
```

Once the data is loaded, use one of the four tools referenced in the *Introduction* section to look at your data.

```r
pnw
View(pnw)
glimpse(pnw)
head(pnw)
```

Use `summary` to give you some overall, data-set-wide metrics like minimum, maximum, median, mean…

```r
summary(pnw)
```

Note that some variables are characters (words) – `vowel` and `word` summaries are counts. Variable `stress` is categorical so the numbers are somewhat meaningless; the rest of the variables have more meaningful numbers.

Arrange (sort) the data by vowel category using `arrange` - ascending (default) or descending (option `desc`). Both options shown below.

```r
pnw %>%
  arrange(vowel)

pnw %>%
  arrange(desc(vowel))
```

Subsetting data

Create a dataset that is JUST observations (rows) with F2 values above 2000Hz. Tool `filter` works on rows:

```r
f2above2k <- pnw %>%
  filter(F2 > 2000)
```

You can filter on multiple conditions, too. Below, F2 values above 2000 where the vowel is 'ER'…

```r
ER2k <- pnw %>%
  filter(F2 > 2000 & vowel == "ER")
```

… or everything *not* ER and above 2k:

```r
ER2k <- pnw %>%
  filter(F2 > 2000 & vowel != "ER")
```