IQSS Workshops

September 2019

Contents

Introduction

Table of Contents

Materials for the Data Science Services statistical software workshops from the Institute for Quantitative Social Science at Harvard.

1. Introduction to R
2. Regression models in R
3. Graphics in R using ggplot2
4. R data wrangling
5. Introduction to Python
6. Python web-scraping
7. Introduction to Stata

These workshops are a work-in-progress, please provide feedback! Email: `help@iq.harvard.edu`

Authors and Sources

These content of these workshops are almost entirely the work of Ista Zahn, now at Harvard's School of Public Health. The current workshop materials have been modified by Steve Worthington, Jinjie Liu, and Yihan Wang, at Harvard's Institute for Quantitative Social Science.

Chapter 1

Welcome

1.1 Materials and setup

NOTE: skip this section if you are not running R locally (e.g., if you are running R in your browser using a remote Jupyter server)

You should have R installed –if not:

- Download and install R from `http://cran.r-project.org`
- Download and install RStudio from `https://www.rstudio.com/products/rstudio/download/#download`

Notes and examples for this workshop are available at

Start RStudio create a new project: - On Windows click the start button and search for rstudio. On Mac RStudio will be in your applications folder. - In Rstudio go to `File -> New Project`. - Choose `New Directory` and `New Project`. - Choose a name and location for your new project directory.

1.2 Workshop goals and approach

In this workshop you will

- learn R basics,
- learn about the R package ecosystem,
- practice reading files and manipulating data in R

A more general goal is to get you comfortable with R so that it seems less scary and mystifying than it perhaps does now. Note that this is by no means a complete or thorough introduction to R! It's just enough to get you started.

This workshop is relatively informal, example-oriented, and hands-on. We won't spend much time examining language features in detail. Instead we will work through an example, and learn some things about the R along the way.

As an example project we will analyze the popularity of baby names in the US from 1960 through 2017. Among the questions we will use R to answer are:

- In which year did your name achieve peak popularity?
- How many children were born each year?
- What are the most popular names overall? For girls? For Boys?

Chapter 2

Graphical User Interfaces (GUIs)

There are many different ways you can interact with R. See the Data Science Tools workshop notes for details.

For this workshop I encourage you to use RStudio; it is a good R-specific IDE that mostly just works.

2.1 Launch RStudio (skip if not using Rstudio)

Note: skip this section if you are not using Rstudio (e.g., if you are running these examples in a Jupyter notebook).

- Start the RStudio program
- In RStudio, go to **File => New File => R Script**

The window in the upper-left is your R script. This is where you will write instructions for R to carry out.

The window in the lower-left is the R console. This is where results will be displayed.

2.2 Exercise 0

The purpose of this exercise is to give you an opportunity to explore the interface provided by RStudio (or whichever GUI you've decided to use). You may not know how to do these things; that's fine! This is an opportunity to figure it out.

Also keep in mind that we are living in a golden age of tab completion. If you don't know the name of an R function, try guessing the first two or three letters and pressing TAB. If you guessed correctly the function you are looking for should appear in a pop up!

---

1. Try to get R to add 2 plus 2.

```
##
```

2. Try to calculate the square root of 10.

```
##
```

3. R includes extensive documentation, including a manual named "An introduction to R". Use the RStudio help pane. to locate this manual.

2.3 Exercise 0 solution

```
## 1. 2 plus 2
2 + 2
## or
sum(2, 2)
```

```
## 2. square root of 10:
sqrt(10)
## or
10^(1/2)
```

```
## 3. Find "An Introduction to R".
```

```
## Go to the main help page by running 'help.start() or using the GUI
## menu, find and click on the link to "An Introduction to R".
```

Chapter 3

R basics

3.1 Function calls

The general form for calling R functions is

```
## FunctionName(arg.1 = value.1, arg.2 = value.2, ..., arg.n - value.n)
```

Arguments can be matched by name; unnamed arguments will be matched by position.

3.2 Assignment

Values can be assigned names and used in subsequent operations

- The "gets" `<-` operator (less than followed by a dash) is used to save values
- The name on the left gets the value on the right.

```
sqrt(10) ## calculate square root of 10; result is not stored anywhere
x <- sqrt(10) # assign result to a variable named x
```

Names should start with a letter, and contain only letters, numbers, underscores, and periods.

3.3 Asking R for help

You can ask R for help using the `help` function, or the `?` shortcut.

```
help(help)
```

The `help` function can be used to look up the documentation for a function, or to look up the documentation to a package. We can learn how to use the `stats` package by reading its documentation like this:

```
help(package = "stats")
```

Chapter 4

Getting data into R

R has data reading functionality built-in – see e.g., `help(read.table)`. However, faster and more robust tools are available, and so to make things easier on ourselves we will use a *contributed package* called `readr` instead. This requires that we learn a little bit about packages in R.

4.1 Installing and using R packages

A large number of contributed packages are available. If you are looking for a package for a specific task, `https://cran.r-project.org/web/views/` and `https://r-pkg.org` are good places to start.

You can install a package in R using the `install.packages()` function. Once a package is installed you may use the `library` function to attach it so that it can be used.

```
## install.packages("readr")
library(readr)
```

4.2 Readers for common file types

In order to read data from a file, you have to know what kind of file it is. The table below lists functions that can import data from common plain-text formats.

Data Type	Function
comma separated	`read_csv()`
tab separated	`read_delim()`
other delimited formats	`read_table()`
fixed width	`read_fwf()`

Note You may be confused by the existence of similar functions, e.g., `read.csv` and `read.delim`. These are legacy functions that tend to be slower and less robust than the `readr` functions. One way to tell them apart is that the faster more robust versions use underscores in their names (e.g., `read_csv`) while the older functions us dots (e.g., `read.csv`). My advice is to use the more robust newer versions, i.e., the ones with underscores.

4.3 Baby names data

The examples in this workshop use US baby names data retrieved from `https://catalog.data.gov/dataset/baby-names-from-social-security-card-applications-national-level-data`
A cleaned and merged version of these data is available at `http://tutorials.iq.harvard.edu/data/babyNames.csv`

4.4 Exercise 1: Reading the baby names data

Make sure you have installed the `readr` package and attached it with `library(readr)`.

Baby names data are available at `"http://tutorials.iq.harvard.edu/data/babyNames.csv"`.

1. Open the `read_csv` help page to determine how to use it to read in data.
2. Read the baby names data using the `read_csv` function and assign the result with the name `baby.names`.
3. BONUS (optional): Save the `baby.names` data as a Stata data set `babynames.dta` and as an R data set `babynames.rds`.

4.5 Exercise 1 solution

```
## read ?read_csv
```

```
baby.names <- read_csv("http://tutorials.iq.harvard.edu/data/babyNames.csv")
```

Chapter 5

Popularity of your name

In this section we will pull out specific names and examine changes in their popularity over time.

The `baby.names` object we created in the last exercise is a `data.frame`. There are many other data structures in R, but for now we'll focus on working with `data.frames`.

R has decent data manipulation tools built-in – see e.g., `help(Extract)`. However, these tools are powerful and complex and often overwhelm beginners. To make things easier on ourselves we will use a *contributed package* called `dplyr` instead.

```
## install.packages("dplyr")
library(dplyr)
```

5.1 Filtering and arranging data

One way to find the year in which your name was the most popular is to filter out just the rows corresponding to your name, and then arrange (sort) by Count.

To demonstrate these techniques we'll try to determine whether "Alex"" or "Jim" was more popular in 1992. We start by filtering the data so that we keep only rows where Year is equal to `1992` and Name is either "Alex" or "Mark".

```
am <- filter(baby.names,
             Year == 1992 & (Name == "Alex" | Name == "Mark"))
am
```

Notice that we can we can combine conditons using `&` (AND) and `|` (OR).

In this case it's pretty easy to see that "Mark" is more popular, but to make it even easier we can arrange the data so that the most popular name is listed first.

```
arrange(am, Count)
```

```
arrange(am, desc(Count))
```

5.2 Other logical operators

In the previous example we used `==` to filter rows. Other relational and logical operators are listed below.

Operator	Meaning
`==`	equal to
`!=`	not equal to
`>`	greater than
`>=`	greater than or equal to
`<`	less than
`<=`	less than or equal to
`%in%`	contained in

These operators may be combined with `&` (and) or `|` (or).

5.3 Exercise 2: Peak popularity of your name

In this exercise you will discover the year your name reached its maximum popularity.

Read in the "babyNames.csv" file if you have not already done so, assigning the result to `baby.names`. The file is located at `"http://tutorials.iq.harvard.edu/data/babyNames.csv"`

Make sure you have installed the `dplyr` package and attached it with `library(dplyr)`.

1. Use `filter` to extract data for your name (or another name of your choice).

```
##
```

2. Arrange the data you produced in step 1 above by `Count`. In which year was the name most popular?

```
##
```

3. BONUS (optional): Filter the data to extract *only* the row containing the most popular boys name in 1999.

```
##
```

5.4 Exercise 2 solution

```
# 1.  Use `filter` to extract data for your name (or another name of your choice).
```

```
george <- filter(baby.names, Name == "George")
```

```
# 2.  Arrange the data you produced in step 1 above by `Count`.
#     In which year was the name most popular?
```

```
arrange(george, desc(Count))
```

```
# 3.  BONUS (optional): Filter the data to extract _only_ the
#     row containing the most popular boys name in 1999.
```

```
boys.1999 <- filter(baby.names,
                    Year == 1999 & Sex == "Boys")
```

```
filter(boys.1999, Count == max(Count))
```

Chapter 6

Plotting baby name trends over time

It can be difficult to spot trends when looking at summary tables. Plotting the data makes it easier to identify interesting patterns.

R has decent plotting tools built-in – see e.g., `help(plot)`. However, To make things easier on ourselves we will use a *contributed package* called `ggplot2` instead.

```
## install.packages("ggplot2")
library(ggplot2)
```

For quick and simple plots we can use the `qplot` function. For example, we can plot the number of babies given the name "Diana" over time like this:

```
diana <- filter(baby.names, Name == "Diana")
```

```
qplot(x = Year, y = Count,
     data = diana)
```

Interetingly there are usually some gender-atypical names, even for very strongly gendered names like "Diana". Splitting these trends out by Sex is very easy:

```
qplot(x = Year, y = Count, color = Sex,
      data = diana)
```

6.1 Exercise 3: Plotting peak popularity of your name

Make sure the `ggplot2` package is installed, and that you have attached it using `library(ggplot2)`.

1. Use `filter` to extract data for your name (same as previous exercise)

```
##
```

2. Plot the data you produced in step 1 above, with `Year` on the x-axis and `Count` on the y-axis.

```
##
```

3. Adjust the plot so that is shows boys and girls in different colors.

```
##
```

4. BONUS (Optional): Adust the plot to use lines instead of points.

6.2 Exercise 3 solution

```
# 1. Use `filter` to extract data for your name (same as previous exercise)
```

```
george <- filter(baby.names, Name == "George")
```

```
# 2.  Plot the data you produced in step 1 above, with `Year` on the x-axis
#     and `Count` on the y-axis.
```

```
qplot(x = Year, y = Count, data = george)
```

```
# 3. Adjust the plot so that is shows boys and girls in different colors.
```

```
qplot(x = Year, y = Count, color = Sex, data = george)
```

```
# 4.  BONUS (Optional): Adust the plot to use lines instead of points.
```

```
qplot(x = Year, y = Count, color = Sex, data = george, geom = "line")
```

Chapter 7

Finding the most popular names

Our next goal is to find out which names have been the most popular.

7.1 Computing better measures of popularity

So far we've used `Count` as a measure of popularity. A better approach is to use proportion or rank to avoid confounding popularity with the number of babies born in a given year.

The `mutate` function makes it easy to add or modify the columns of a `data.frame`. For example, we can use it compute the log of the number of boys and girls given each name in each year:

```
baby.names <- mutate(baby.names, logCount = Count/1000)
baby.names
```

7.2 Operating by group

Because of the nested nature of out data, we want to compute rank or proportion within each `Sex X Year` group. The `dplyr` package makes this relatively easy.

```
baby.names <- mutate(group_by(baby.names, Year, Sex),
                     Rank = rank(Count))
```

Note that the data remains grouped until you change the groups by running `group_by` again or remove grouping information with `ungroup`.

7.3 Exercise 4: Most popular names

In this exercise your goal is to identify the most popular names for each year.

1. Use `mutate` and `group_by` to create a column named "Proportion" where `Proportion = Count/sum(Count)` for each `Year X Sex` group.

```
##
```

2. Use `mutate` and `group_by` to create a column named "Rank" where `Rank = rank(-Count)` for each `Year X Sex` group.

```
##
```

3. Filter the baby names data to display only the most popular name for each `Year X Sex` group.

```
##
```

4. Plot the data produced in step 4, putting `Year` on the x-axis and `Proportion` on the y-axis. How has the proportion of babies given the most popular name changed over time?

```
##
```

5. BONUS (optional): Which names are the most popular for both boys and girls?

7.4 Exercise 4 solution

```
## 1.  Use `mutate` and `group_by` to create a column named "Proportion"
##     where `Proportion = Count/sum(Count)` for each `Year X Sex` group.
```

```
baby.names <- mutate(group_by(baby.names, Year, Sex),
                     Proportion = Count/sum(Count))
```

```
## 2.  Use `mutate` and `group_by` to create a column named "Rank" where
##     `Rank = rank(-Count)` for each `Year X Sex` group.
```

```
baby.names <- mutate(group_by(baby.names, Year, Sex),
                     Rank = rank(-Count))
```

```
## 3.  Filter the baby names data to display only the most popular name
##     for each `Year X Sex` group.
```

```
top1 <- filter(baby.names, Rank == 1)
```

```
## 4. Plot the data produced in step 3, putting `Year` on the x-axis
##    and `Proportion` on the y-axis. How has the proportion of babies
##    given the most popular name changed over time?
```

```
qplot(x = Year, y = Proportion, color = Sex,
      data = top1,
      geom = "line")
```

```
## 5. BONUS (optional): Which names are the most popular for both boys
##    and girls?
```

```
girls.and.boys <- inner_join(filter(baby.names, Sex == "Boys"),
                             filter(baby.names, Sex == "Girls"),
                             by = c("Year", "Name"))
```

```
girls.and.boys <- mutate(girls.and.boys,
                         Product = Count.x * Count.y,
                         Rank = rank(-Product))
```

```
filter(girls.and.boys, Rank == 1)
```

Chapter 8

Percent choosing one of the top 10 names

You may have noticed that the percentage of babies given the most popular name of the year appears to have decreases over time. We can compute a more robust measure of the popularity of the most popular names by calculating the number of babies given one of the top 10 girl or boy names of the year.

In order to compute this measure we need to operate within goups, as we did using `mutate` above, but this time we need to collapse each group into a single summary statistic. We can achive this using the `summarize` function. For example, we can calculate the number of babies born each year:

```
bn.by.year <- summarize(group_by(baby.names, Year),
                        Total = sum(Count))
bn.by.year
```

8.1 Exercise 4: Popularity of the most popular names

In this exercise we will plot trends in the proportion of boys and girls given one of the 10 most popular names each year.

1. Filter the baby.names data, retaining only the 10 most popular girl and boy names for each year.

```
##
```

2. Summarize the data produced in step one to calculate the total Proportion of boys and girls given one of the top 10 names each year.

```
##
```

3. Plot the data produced in step 2, with year on the x-axis and total proportion on the y axis. Color by sex.

```
##
```

8.2 Exercise 4 solution

```
## 1.  Filter the baby.names data, retaining only the 10 most
##     popular girl and boy names for each year.
```

```
most.popular <- filter(group_by(baby.names, Year, Sex),
                       Rank <= 10)
```

```
## 2.  Summarize the data produced in step one to calculate the total
##     Proportion of boys and girls given one of the top 10 names
##     each year.
```

```
top10 <- summarize(group_by(most.popular, Year, Sex),
                   TotalProportion = sum(Proportion))
```

```
## 3.  Plot the data produced in step 2, with year on the x-axis
##     and total proportion on the y axis. Color by sex.
```

```
qplot(x = Year, y = TotalProportion, color = Sex,
      data = top10,
      geom = "line")
```

Chapter 9

Saving our Work

Now that we have made some changes to our data set, we might want to save those changes to a file.

9.1 Saving individual datasets

```
# write data to a .csv file
write_csv(baby.names, "babyNames.csv")
```

```
# write data to an R file
write_rds(baby.names, "babyNames.rds")
```

9.2 Saving and loading R workspaces

In addition to importing individual datasets, R can save and load entire workspaces

```
ls() # list objects in our workspace
save.image(file="myWorkspace.RData") # save workspace
rm(list=ls()) # remove all objects from our workspace
ls() # list stored objects to make sure they are deleted
```

```
## Load the "myWorkspace.RData" file and check that it is restored
load("myWorkspace.RData") # load myWorkspace.RData
ls() # list objects
```

Chapter 10

Wrap-up

10.1 Help us make this workshop better!

Please take a moment to fill out a very short feedback form. These workshops exist for you – tell us what you need! `http://tinyurl.com/R-intro-feedback`

10.2 Additional resources

- IQSS workshops: `http://projects.iq.harvard.edu/rtc/filter_by/workshops`
- IQSS statistical consulting: `http://dss.iq.harvard.edu`
- Software (all free!):
  - R and R package download: `http://cran.r-project.org`
  - Rstudio download: `http://rstudio.org`
  - ESS (emacs R package): `http://ess.r-project.org/`
- Online tutorials
  - `http://www.codeschool.com/courses/try-r`
  - `http://www.datacamp.org`
  - `http://swirlstats.com/`
  - `http://r4ds.had.co.nz/`
- Getting help:
  - Documentation and tutorials: `http://cran.r-project.org/other-docs.html`
  - Recommended R packages by topic: `http://cran.r-project.org/web/views/`
  - Mailing list: `https://stat.ethz.ch/mailman/listinfo/r-help`
  - StackOverflow: `http://stackoverflow.com/questions/tagged/r`
- Coming from... Stata : `http://www.princeton.edu/~otorres/RStata.pdf` SAS/SPSS : `http://www.et.bs.ehu.es/~etptupaf/pub/R/RforSAS&SPSSusers.pdf` matlab : `http://www.math.umaine.edu/~hiebeler/comp/matlabR.pdf` Python : `http://mathesaurus.sourceforge.net/matlab-python-xref.pdf`