StatR 502 Homework 3

Gregor Thomas

Due Thursday, Jan. 28, 2016, 6:30 pm

Submission guidelines: please submit a knitted PDF or Word document, and optionally your `.Rmd` file. As always, ask in the discussion forum if you're having trouble!

1: Webscraping

Last week you imported data from an Excel workbook with data on King County population and crime. This week, you'll build on last week's work and get similar data for all the Washington counties.

Here's a vector with all the county names:

```r
counties = c("Adams", "Asotin", "Benton", "Chelan", "Clallam", "Clark",
"Columbia", "Cowlitz", "Douglas", "Ferry", "Franklin", "Garfield",
"Grant", "Grays Harbor", "Island", "Jefferson", "King", "Kitsap",
"Kittitas", "Klickitat", "Lewis", "Lincoln", "Mason", "Okanogan",
"Pacific", "Pend Oreille", "Pierce", "San Juan", "Skagit", "Skamania",
"Snohomish", "Spokane", "Stevens", "Thurston", "Wahkiakum", "Walla Walla",
"Whatcom", "Whitman", "Yakima")
```

The source of the last week's Excel workbook is the Office of Financial Management. Their Criminal Justice Databooks are all linked off of this page: http://www.ofm.wa.gov/sac/cjdatabook/

```r
base.url = "http://www.ofm.wa.gov/sac/cjdatabook/"
```

Mousing around on the site, you can see that each county's databook is available at the url "http://www.ofm.wa.gov/sac/cjdatabook/aaaa.xlsx", where the "aaaa" is the first four letters of the county, in lower case (excluding spaces, in the case of San Juan county).

(a) Using string manipulation functions, make a vector of the first four letters of each county, in lower case. Call it `county4`. There *many* ways to do this, functions such as `stringr::str_replace`, `base::tolower` or `stringr::str_to_lower`, `substring`, and `stringr::str_extract` will help.

Note: in `pattern` arguments that use regular expressions, `"."` will match any single character. If you actually want to match a period—which you don't for this exercise—you would need to escape it as `"\\."`. We'll cover just a little bit of regular expressions in lab. I often use this regular expressions cheat sheet.

(b) (The really easy part.) Using `paste` or `paste0`, create a new vector with all the workbook names by sticking ".xlsx" on the end of your `county4` variable. Call it `county4.xlsx`. Then create a vector `full.url` by sticking the `county4.xlsx` on the end of the `base.url` above. (The first element of the full URL vector should be `http://www.ofm.wa.gov/sac/cjdatabook/adam.xlsx`.)

(c) Create a new folder in your working directory to house the data. (If you wanted to do this programmatically, you could use the `dir.create()`. I recommend doing it manually since you may be running this code more than once and you probably don't want 100 new directories.) Use R to download the databooks for each county into the directory. You could do this with an `apply` family function, but I would just use a `for` loop.

Two options for downloading (inside the for loop):

- With `base::download.file()` If you're on Windows, you'll need to set `mode = "wb"` (explained deep in the details of `?download.file`).

- With the `httr` package (more complicated... but can be used to access APIs)
  - use `GET()` on the full URL and assign it to an R object (I called it `sheet`).
  - call `writeBin`. The `con` argument needs to be `"your_new_data_folder/aaaa.xlsx"`, with your actual folder name and the appropriate "aaaa" for each county subbed in. The `what` argument of `writeBin` should be `content(sheet, as = "raw")` to keep R from trying to parse the workbook.

For this part of the problem, **show your code, but set the knitr option `eval = FALSE`** so that the code is not executed *every time you compile your homework.* (After the 3 backticks to start your code chunk, change `{r}` to `{r, eval = FALSE}`.

(d) Based on either your solution or my solution from the key (or even someone else's solution) from HW 2, *write a function that pulls just the murder data out of a workbook* (combine the NIBRS and SRS data as in last week's homework). Your function should take two arguments, the filepath of the workbook and the name of the county. It should (1) read in the workbook, (2) manipulate/subset the data down to the year and number of murders and (3) add a new column of class `character` named "county", filled in with the name of the particular county. Make sure it's a `character`, not a `factor`, or you'll have trouble combining all the data!

(e) Initialize a list `crime_data = list()`. In a for loop, use your function from (d) to fill in `crime_data` so that each list element is a `data.frame` of murders over time for a county. When using lists, to access or assign a single element, use **two brackets**, e.g., `crime_data[[i]] <- my_function(...)`. (A lot of the R community has general disdain for `for` loops. This is generally overblown. A `for` loop is a poor substitute for something truly *vectorized*, but something like this is a perfectly fine use for a `for` loop. If you *really* want to be fancy, you can use `mapply`, which is like `lapply` but it works with multiple arguments.)

(f) Combine the data. If your import was successful and all the data frames have the same columns in the same order, this should be as easy as `crime = do.call(rbind, crime_data)` (from `base`), or `bind_rows(crime_data)` (from `dplyr`).

(g) Pick 12 counties. Make a plot of murders by year for each of your 12 selected counties. (You probably want to `melt` the data, either in this step or inside the import function you wrote.)

More Obesity!

Use the YRBS obesity data from Lecture 3.

- **(a)** Fit a model using all the available predictors. Using that as a starting point, take out inputs that don't seem to be helping much, and explore adding interactions. Show the code for your two top models.
- **(b)** Evaluate and compare the models from part (a), comment on their differences. Which one has better predictive accuracy?

Problems from Gelman & Hill

Section 5.10 (pp. 105-107), **do problems 3, and 5**. In your write-up, please label them as G&H X, where X is the problem number.