New Haven Scrape Week 1

Sep. 1, 2017

Administrative Stuff

The 1 HTBA will likely start in Week 3, so that we have some time to see how enrollment numbers will settle.

Xin Xu (xin.xu@yale.edu) is our TF for this course. Office hours will start next week.

Agenda

1. Submit .csv files.
2. Look at a few script submissions.
3. Talk about best practices for coding in R/RMarkdown.
4. Reconcile .csv's.
5. Proceed to next part of scrape.

Submitting .csv Files

Upload your .csv files, named 'hw1_netid.csv' to Canvas. The .csv should have exactly 1000 rows and 3 columns, with the appropriate column names.

I will be reading in your .csv file with something like:

```
x <- read.csv("hw1_netid.csv", as.is = TRUE)
```

Best Practices

- Set up organized folders on your computer for your work in this class (and other classes in which you will be using R).

It can be extremely tempting to put all your R-related files into one place, such as in the default **working directory**. This could cause all sorts of chaos as the number of files increases throughout the semester.

- Always use <- for variable assignment. Ensure that all operators are preceded and followed by a space.

```
a <- 5 # this is the right way to do it!

a<-5  # this seems harmless, does it?
a< -5 # how about this?
```

```
## [1] FALSE
```

Here's an example of where it could all go wrong...

```
a <- -5:5   # shorthand for seq(-5, 5, by=1)
a
```

```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

```
a[a<-5]
```

```
## [1] -1
```

```
a
```

```
## [1] 5
```

For instances where you want to explicitly check whether, say, `a` is less than -5, put a space before and after the operator to remove the ambiguity.

```
a < -5
```

```
## [1] FALSE
```

Don't use the equal sign to assign values:

```
a = 5
```

This works, but we like to reserve the equal sign to include arguments to functions:

```
rnorm(5, mean = 3, sd = 10)
```

```
## [1] -3.6852653 -0.4174778 10.2675496  5.4789736 -1.9866521
```

- Don't let yourself write lines of code longer than 80 characters. It does not project well.

Configure RStudio options. Go to 'Tools' -> 'Global Options...'. There are a lot of options here you can select to help you form good coding habits.

For example, I choose to add a vertical edge at 80 characters, so that I get into the practice of not writing extremely long lines of code. I also select to indent via 2 spaces instead of a tab to save myself space. You're far less likely to get text that trails off the page after compiling things if you stick to this rule.

What happens if you need to write commands that are longer than 80 characters? Add returns and align appropriately.

Here's what not to do:

```
x <- data.frame(name=c("Anna", "Bob", "Casper", "Doris", "Ethan"), age=runif(5, min=0, max=50))
```

Better is:

```
x <- data.frame(name=c("Anna", "Bob", "Casper", "Doris", "Ethan"),
                age=runif(5, min=0, max=50))
```

You may have read from other places that it's possible to put multiple R commands on the same line when separated by a `;`. Avoid doing this because it makes the code less readable.

```
x <- 5*exp(5); x  # there's no need to save yourself space. Just use a new line!
```

```
## [1] 742.0658
```

- When using control statements like `if()` and `for()`, include a space before the `()`. No space is needed between function names and parentheses.

```
if (grade > 90) {
  print("YAY")
}
```

- Always include a space before and after mathematical/comparison operators (`+`, `-`, `/`, `*`, `>`, `<`, `==`, `!=`, etc.):

```
if (x == 3) {
  x <- x - 3
}
```

- While we're on Global Options, let's also choose to **never** `Save the  workspace to .RData on exit`.

I see some of you starting your scripts with `rm(list=ls())`. This is unnecessary if you choose to always start with a clean slate.

- Set the default encoding of all saved files to UTF-8 or ASCII.

Global Options -> Code -> Saving -> Default text encoding -> UTF-8.

If you use an operating system in a different language, this can potentially resolve issues with unusual characters getting inserted/interpreted randomly within your files.

- Indent your code.

Proper indentation will significantly improve the readability of your code. (Thankfully, RStudio does auto-code reformat for you, and a Cmd+Shift+A or Ctrl+Shift+A can help clean-up an existing mess. This doesn't always produce the cleanest output. Cmd + I or Ctrl + I, which purely assists with indentation, also deserves an honorable mention.)

Reconciling the .csv Files

Continue the Scrape

HW2: For Wednesday, let's add in some new variables. These are going to be more challenging, so keep your eyes open for unusual cases within the dataset.

- `totval`: 2016 total appraisal value
- `bedrooms`: number of bedrooms
- `bathrooms`: number of bathrooms
- `halfbaths`: number of half-bathrooms

For property with pid 600, you should have `totval` = 266600 (note the absence of dollar signs and commas), `bedrooms` = 8, `bathrooms` = 3, `halfbaths` = 1.

Let's now expand our scope to all pids (not just the first 1000). By Wednesday 11AM, submit a `hw2_netid.csv` file containing a total of 7 columns (3 old ones first, then 4 new ones) and 27,308 rows, with appropriate column headings. Also include a `hw2_netid.R` script file that puts together this .csv file.