Intro to R and Rstudio

POST 8000 – Foundations of Social Science Research for Public Policy

Steven V. Miller

Department of Political Science

Goal for Today

Introduce you to R and Rstudio.

https://github.com/svmiller/post8000/tree/master/lab-scripts

What Is R?

R is an open source programming language with origins in C and FORTRAN. Advantages:

- Flexibility
- It’s free (and open source)!
- Ease of handling advanced computational models
- Ease of handling multiple data sets in one session
- Higher demand in industries.

But more importantly, it’s free.

What Is R?

Some disadvantages:

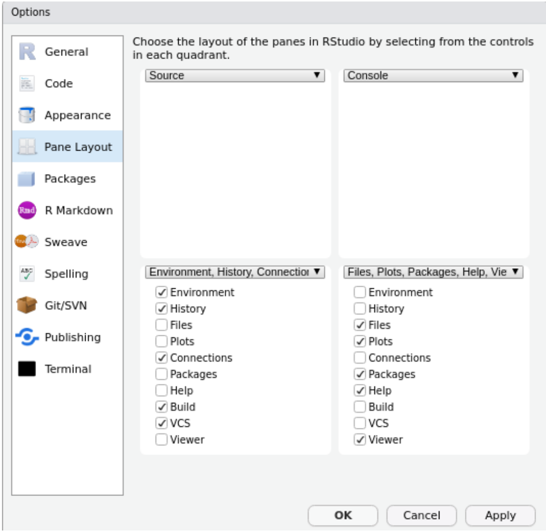
- “Bleeding” edge? (Even then...)
- Higher learning curve
- A “programming language” and not a “program.”

Rstudio will help with the learning curve component.

Getting Started in R and Rstudio

Let's get started in Rstudio first. Select "Tools" in the menu.

- Scroll to "Global Options" (should be at the bottom)
- On the pop-up, select "pane layout."
- Rearrange so that "Source" is top left, "Console" is top right", and the files/plots/packages/etc. is the bottom right.
- Save



Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.

Source ▼

Console ▼

Environment, History, Connectior ▼

- ☑ Environment
- ☑ History
- □ Files
- □ Plots
- ☑ Connections
- □ Packages
- □ Help
- ☑ Build
- ☑ VCS
- □ Viewer

Files, Plots, Packages, Help, Vie ▼

- □ Environment
- □ History
- ☑ Files
- ☑ Plots
- □ Connections
- ☑ Packages
- ☑ Help
- □ Build
- □ VCS
- ☑ Viewer

Getting Started in R and Rstudio

Hit Ctrl-Shift-N (Cmd-Shift-N if you're on a Mac) to open up a new script.

- Minimize the "Environment/History/Connections/Git" pane in the bottom left.
- Adjust the console output to your liking.

This should maximize your Rstudio experience, esp. as you'll eventually start writing documents in Rstudio.

- That should maximize your Rstudio experience, esp. as you begin to write documents in Rstudio as well.

A Few Commands to Get Started

`getwd()` will spit out your current working directory.

```r
getwd()
```

```
## [1] "/home/svmille/Dropbox/teaching/post8000/intro-r-rstudio"
```

By default, assuming your username is “Steve”:

- Windows: `"C:/Users/Steve/Documents"` (notice the forward slashes!)
- Mac: `/Users/Steve`
- Linux: `/home/Steve`

Creating Objects

R is an “object-oriented” programming language.

- i.e. inputs create outputs that may be assigned to objects in the workspace.

For example:

```r
a <- 3
b <- 4
this_is_a_long_object_name_and_you_should_not_do_this <- 5
d <- pi # notice there are a few built-in functions/objects
```

Sometimes it’s useful to see all the mess you’ve created in your workspace

```r
ls()
```

```
## [1] "a"
## [2] "b"
## [3] "d"
## [4] "this_is_a_long_object_name_and_you_should_not_do_this"
```

Install Packages

R depends on user-created libraries to do much of its functionality. We're going to start with a few for the sake of this exercise.

```r
# This will take a while, mostly for tidyverse
install.packages(c("tidyverse","devtools"))

# Once it's installed:
library(tidyverse)
library(devtools)

# Where I'll be putting some example data sets.
devtools::install_github("svmiller/post8000r")

library(post8000r)
```

Load Data

You can load data from your hard drive, or even the internet. Some commands:

- `haven::read_dta()` for Stata .dta files
- `haven::read_spss()` for SPSS files
- `read_csv()` for CSV files
- `readxl::read_excel()` for MS Excel spreadsheets
- `read_tsv()`for tab-separated values.

Just make sure to apply it to an object.

```r
# Note: hypothetical data
Apply <- haven::read_dta("https://stats.idre.ucla.edu/stat/data/ologit.dta")
# County unemployment
Cunemp <- read_tsv("https://download.bls.gov/pub/time.series/la/la.data.64.County")
```

Load Data

Some R packages, like my `post8000r` package, has built-in data. For example:

```r
data(pwt_sample)
names(pwt_sample)
```

```
## [1] "country" "isocode" "year"    "pop"      "hc"       "rgdpna"   "lab
```

```r
# also: help(pwt_sample)
```

Brief description: macroeconomic data from select rich countries from PWT

- 23 countries
- `pop`: population in millions
- `hc`: index of human capital per person (based on years of schooling/returns to education)
- `rgdpna`: real GDP at constant 2011 prices.
- `labsh`: labor share of income at current national prices.

Tidyverse

The tidyverse is a suite of functions/packages that totally rethink base R. Some functions we'll discuss:

- `%>%` (the pipe)
- `glimpse()` and `summary()`
- `select()`
- `group_by()`
- `summarize()`
- `mutate()`
- `filter()`

I cannot fully discuss everything from the tidyverse. That's why there's Google/Stackexchange. :P

%>%

The pipe (%>%) allows you to chain together a series of tidyverse functions.

- This is especially useful when you're recoding data and you want to make sure you got everything right before saving the data.

You can chain together a host of tidyverse commands within it.

glimpse() and summary()

glimpse() and summary() will get you some basic descriptions of your data. For example:

```
pwt_sample %>% glimpse() # notice the pipe
```

```
## Observations: 1,428
## Variables: 7
## $ country <fct> Australia, Australia, Australia, Australia, Australi
## $ isocode <fct> AUS, AUS, AUS, AUS, AUS, AUS, AUS, AUS, AUS, AUS, AU
## $ year    <dbl> 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958
## $ pop     <dbl> 8.386674, 8.633449, 8.816668, 8.985786, 9.194855, 9.
## $ hc      <dbl> 2.667302, 2.674344, 2.681403, 2.688482, 2.695580, 2.
## $ rgdpna  <dbl> 119510.4, 122550.0, 117533.8, 130284.5, 140700.2, 14
## $ labsh   <dbl> 0.6804925, 0.6804925, 0.6804925, 0.6804925, 0.680492
```

glimpse() and summary()

summary() is technically not a tidyverse function, but it works within the pipe.

```r
pwt_sample %>% summary()
```

```
##       country        isocode          year           pop
##  Australia:  68   AUS    :  68   Min.   :1950   Min.   :  0.1432
##  Austria  :  68   AUT    :  68   1st Qu.:1967   1st Qu.:  7.3530
##  Belgium  :  68   BEL    :  68   Median :1984   Median : 11.2006
##  Canada   :  68   CAN    :  68   Mean   :1984   Mean   : 36.8008
##  Chile    :  68   CHE    :  68   3rd Qu.:2000   3rd Qu.: 52.7539
##  Denmark  :  68   CHL    :  68   Max.   :2017   Max.   :324.4595
##  (Other)  :1020   (Other):1020                  NA's   :2
##        hc            rgdpna              labsh
##  Min.   :1.242   Min.   :    1098   Min.   :0.3286
##  1st Qu.:2.440   1st Qu.:  137609   1st Qu.:0.5761
##  Median :2.809   Median :  302889   Median :0.6313
##  Mean   :2.784   Mean   : 1044426   Mean   :0.6137
##  3rd Qu.:3.165   3rd Qu.: 1021393   3rd Qu.:0.6565
##  Max.   :3.758   Max.   :17711024   Max.   :0.7701
```

select()

`select()` will grab (or omit) columns from the data.

```
# grab everything
pwt_sample %>% select(everything())
```

```
## # A tibble: 1,428 x 7
##    country   isocode  year   pop    hc  rgdpna labsh
##    <fct>     <fct>   <dbl> <dbl> <dbl>   <dbl> <dbl>
##  1 Australia AUS      1950  8.39  2.67 119510. 0.680
##  2 Australia AUS      1951  8.63  2.67 122550. 0.680
##  3 Australia AUS      1952  8.82  2.68 117534. 0.680
##  4 Australia AUS      1953  8.99  2.69 130285. 0.680
##  5 Australia AUS      1954  9.19  2.70 140700. 0.680
##  6 Australia AUS      1955  9.41  2.70 146250. 0.680
##  7 Australia AUS      1956  9.64  2.71 146586. 0.680
##  8 Australia AUS      1957  9.85  2.72 149796. 0.680
##  9 Australia AUS      1958 10.1   2.73 159957. 0.680
## 10 Australia AUS      1959 10.3   2.74 169756. 0.680
## # ... with 1,418 more rows
```

select()

```r
# grab everything, but drop the labsh variable.
pwt_sample %>% select(-labsh)
```

```
## # A tibble: 1,428 x 6
##    country   isocode  year   pop    hc  rgdpna
##    <fct>     <fct>   <dbl> <dbl> <dbl>   <dbl>
##  1 Australia AUS      1950  8.39  2.67 119510.
##  2 Australia AUS      1951  8.63  2.67 122550.
##  3 Australia AUS      1952  8.82  2.68 117534.
##  4 Australia AUS      1953  8.99  2.69 130285.
##  5 Australia AUS      1954  9.19  2.70 140700.
##  6 Australia AUS      1955  9.41  2.70 146250.
##  7 Australia AUS      1956  9.64  2.71 146586.
##  8 Australia AUS      1957  9.85  2.72 149796.
##  9 Australia AUS      1958 10.1   2.73 159957.
## 10 Australia AUS      1959 10.3   2.74 169756.
## # ... with 1,418 more rows
```

select()

```r
# grab just these three columns.
pwt_sample %>% select(country, year, rgdpna)
```

```
## # A tibble: 1,428 x 3
##    country    year  rgdpna
##    <fct>     <dbl>   <dbl>
##  1 Australia  1950 119510.
##  2 Australia  1951 122550.
##  3 Australia  1952 117534.
##  4 Australia  1953 130285.
##  5 Australia  1954 140700.
##  6 Australia  1955 146250.
##  7 Australia  1956 146586.
##  8 Australia  1957 149796.
##  9 Australia  1958 159957.
## 10 Australia  1959 169756.
## # ... with 1,418 more rows
```

group_by()

group_by() might be the most powerful function in tidyverse.

- tl;dr: it allows you to perform functions within specific subsets (groups) of the data.

```
# Notice we can chain some pipes together
pwt_sample %>%
    # group by country
    group_by(country) %>%
    # Get me the first observation, by group.
    slice(1)
```

```
## # A tibble: 21 x 7
## # Groups:   country [21]
##    country   isocode  year   pop    hc  rgdpna  labsh
##    <fct>     <fct>   <dbl> <dbl> <dbl>   <dbl>  <dbl>
##  1 Australia AUS      1950  8.39  2.67 119510.  0.680
##  2 Austria   AUT      1950  6.98  2.55  47147.  0.637
##  3 Belgium   BEL      1950  8.63  2.20  76035.  0.651
##  4 Canada    CAN      1950 13.8   2.48 179072.  0.768
```

group_by()

Notice what would happen in the absence of `group_by()`

```
pwt_sample %>%
    # Get me the first observation for each country
    slice(1) # womp womp. Forgot to group_by()
```

```
## # A tibble: 1 x 7
##   country   isocode  year   pop    hc  rgdpna labsh
##   <fct>     <fct>   <dbl> <dbl> <dbl>   <dbl> <dbl>
## 1 Australia AUS      1950  8.39  2.67 119510. 0.680
```

Caveat: if you're applying a group-specific function (that you need once), it's generally advisable to "ungroup" (i.e. `ungroup()`) the data when you're done.

summarize()

summarize() creates condensed summaries of the data, for whatever it is you want.

```
pwt_sample %>%
    # How many observations are in the data?
    summarize(n = n())
```

```
## # A tibble: 1 x 1
##        n
##    <int>
## 1  1428
```

summarize()

```
# Note: works *wonderfully* with group_by()
pwt_sample %>%
    group_by(country) %>%
    # Give me the max real GDP observed in the data.
    summarize(maxgdp = max(rgdpna, na.rm=T))
```

```
## # A tibble: 21 x 2
##    country     maxgdp
##    <fct>        <dbl>
##  1 Australia 1215688
##  2 Austria    380620.
##  3 Belgium    453158.
##  4 Canada    1647159.
##  5 Chile      399417.
##  6 Denmark    274272.
##  7 Finland    217679.
##  8 France    2565994.
##  9 Germany   3805884
```

mutate()

mutate() creates new columns while retaining original dimensions of the data (unlike summarize()).

```
pwt_sample %>%
    # Convert rgdpna from real GDP in millions to real GDP in billions
    mutate(rgdpnab = rgdpna/1000)
```

```
## # A tibble: 1,428 x 8
##    country   isocode  year   pop    hc  rgdpna labsh rgdpnab
##    <fct>     <fct>   <dbl> <dbl> <dbl>   <dbl> <dbl>   <dbl>
##  1 Australia AUS      1950  8.39  2.67 119510. 0.680    120.
##  2 Australia AUS      1951  8.63  2.67 122550. 0.680    123.
##  3 Australia AUS      1952  8.82  2.68 117534. 0.680    118.
##  4 Australia AUS      1953  8.99  2.69 130285. 0.680    130.
##  5 Australia AUS      1954  9.19  2.70 140700. 0.680    141.
##  6 Australia AUS      1955  9.41  2.70 146250. 0.680    146.
##  7 Australia AUS      1956  9.64  2.71 146586. 0.680    147.
##  8 Australia AUS      1957  9.85  2.72 149796. 0.680    150.
##  9 Australia AUS      1958 10.1   2.73 159957. 0.680    160.
```

mutate()

Note: this also works well with group_by()

```
pwt_sample %>%
    group_by(country) %>%
    # divide rgdpna over the country's max, for some reason.
    mutate(rgdpnaprop = rgdpna/max(rgdpna, na.rm=T))
```

```
## # A tibble: 1,428 x 8
## # Groups:   country [21]
##    country   isocode  year   pop    hc  rgdpna labsh rgdpnaprop
##    <fct>     <fct>   <dbl> <dbl> <dbl>   <dbl> <dbl>      <dbl>
##  1 Australia AUS      1950  8.39  2.67 119510. 0.680     0.0983
##  2 Australia AUS      1951  8.63  2.67 122550. 0.680     0.101
##  3 Australia AUS      1952  8.82  2.68 117534. 0.680     0.0967
##  4 Australia AUS      1953  8.99  2.69 130285. 0.680     0.107
##  5 Australia AUS      1954  9.19  2.70 140700. 0.680     0.116
##  6 Australia AUS      1955  9.41  2.70 146250. 0.680     0.120
##  7 Australia AUS      1956  9.64  2.71 146586. 0.680     0.121
##  8 Australia AUS      1957  9.85  2.72 149796. 0.680     0.123
```

filter()

filter() is a great diagnostic tool for subsetting your data to look at specific observations.

- Notice the use of double-equal signs (==) for the filter() functions.

```
pwt_sample %>%
    # give me just the USA observations
    filter(isocode == "USA")
```

```
## # A tibble: 68 x 7
##    country                  isocode  year   pop    hc  rgdpna labsh
##    <fct>                    <fct>   <dbl> <dbl> <dbl>    <dbl> <dbl>
##  1 United States of America USA      1950  156.  2.58 2246944. 0.628
##  2 United States of America USA      1951  158.  2.60 2428017  0.634
##  3 United States of America USA      1952  161.  2.61 2526887. 0.645
##  4 United States of America USA      1953  164.  2.62 2645510. 0.644
##  5 United States of America USA      1954  167.  2.63 2630592. 0.637
##  6 United States of America USA      1955  170.  2.65 2817940  0.627
##  7 United States of America USA      1956  173.  2.66 2878023  0.640
```

filter()

```
pwt_sample %>%
    # give me the observations from the most recent year.
    filter(year == max(year))
```

```
## # A tibble: 21 x 7
##    country     isocode  year   pop    hc    rgdpna labsh
##    <fct>       <fct>   <dbl> <dbl> <dbl>     <dbl> <dbl>
##  1 Australia   AUS      2017 24.5   3.52 1215688   0.586
##  2 Austria     AUT      2017  8.74  3.36  380620.  0.573
##  3 Belgium     BEL      2017 11.4   3.14  453158.  0.610
##  4 Canada      CAN      2017 36.6   3.71 1647159.  0.651
##  5 Switzerland CHE      2017  8.48  3.69  527023.  0.650
##  6 Chile       CHL      2017 18.1   3.11  399417.  0.440
##  7 Germany     DEU      2017 82.1   3.67 3805884   0.618
##  8 Denmark     DNK      2017  5.73  3.56  274272.  0.613
##  9 Spain       ESP      2017 46.4   2.94 1557162.  0.574
## 10 Finland     FIN      2017  5.52  3.47  216303.  0.576
## # ... with 11 more rows
```

Don’t Forget to Assign

When you’re done, don’t forget to assign what you’ve done to an object.

```
pwt_sample  %>%
    # convert real GDP to billions
    mutate(rgdpnab = rgdpna/1000) -> NewObjectName
```

tidyverse’s greatest feature is the ability to see what you’re coding in real time before commiting/overwrting data.

Table of Contents