Applied Demographic Data Analysis with R

Corey S. Sparks

2022-02-04

Contents

Introduction

0.1 Why a book on statistics for demographers?

Demographers have always been a mixture of sociologists, economists, statisticians, health researchers, and other broad sub-disciples of social science. As such, we bring with us a large amount of baggage from our respective academic life courses, and we often are trained by a wide variety of mentors and professors. It's my perspective that our interdisciplinary experience is one of our greatest strengths as a group. Given that our training is often in one of a core set of home disciplines, we often have methodological training from said discipline, and this may not be a broad enough perspective to firmly ground us in the types of methods that demographers commonly employ. This is not the fault of the departments that trained us, it's just a historical fact. So, why am I writing a book on statistics and data analysis aimed at demographers? I will give you three reasons:

1. Demographers have to go beyond the sample. This is to say that our results and research is generally representative of a larger national or international population, and we do this explicitly in our models.
2. We demographers don't use random samples for our analysis. Statistics books the world over are based on assumptions of random sampling and independence, while the data that we often have to, or desire to use, comes more than likely from a data source that was collected using a complex survey design. This is a big deal and we have to have training materials that instill this in our students early on in their careers.
3. Weird data. As demographers, we often use data from lots of different places and if you were trained up to this point to believe that the linear model is the end-all be-all of statistical inference, I've got news for you friends, you've been misled. Categorical outcomes, counts, hierarchically structured, longitudinally collected, spatially referenced, just to name a few of such oddities, are ubiquitous in our field, and part of what makes our discipline so cool and interesting to newcomers.

My goal for this book is to take the lessons I've learned teaching statistics to a diverse and often cursorily trained group of students who have problems they care about, that they need to bring demographic data to bear upon. This is a challenge, and I have always been a stalwart proponent of teaching statistics and data analysis in a *very* applied manner. As such, this book won't be going into rigorous proofs of estimators or devoting pages to expositions of icky algebra; instead it will focus on exploring modern methods of data analysis that in used by demographers every day, but not always taught in our training programs.

As someone who has learned much more of these methods by personal exploration than by formal study, I find that many of these methods are absent from the canon of social science statistics, but are both in great demand from people who hire us, and absolutely necessary to the demographer's analytic toolkit. It's a major goal of this book to de-mystify the process and to make it accessible to a wide audience, so I will always strive to illustrate the key aspects of the methods described herein, and ground the discussion of methods in applications.

0.1.1 is this a cookbook?

0.1.2 Broader picture of what i'll cover

0.1.3 What's a demographer?

0.1.4 What is applied demography?

0.1.5 Why we need to see this stuff?

0.1.6 Why R is a good option for applied demography?

0.1.6.1 why write code?

0.1.7 Mention packages earlier - talk about later

1 R in applied demography

1.1 Why R?

I've used R for twenty years. I was also trained in SPSS and SAS along the way, by various mentors. Some tried to get me to learn more general purpose languages like Delphi (of all things) or Perl, or Basic, and I've been chastised for not knowing the depths of Python, but R presents a nimble and rigorous platform to *do* demography. My top three reasons for teaching and using R are:

1. It's free - This is important, because, why should we pass along more costs to people, especially our students? This also make R code accessible to people, worldwide.
2. It's the hotbed of methodological development. The R ecosystem has thousands of packages that represent the bleeding edge of data analysis, visualization and data science. This makes R attractive because it can pivot quickly to adopt new methods, which often lag in their development in other environments.
3. It has a supportive community of users. While there are some debates over how friendly some R users are to new users, overall, after spending 20 years in the R community, I've personally assisted hundreds of users, and been personally helped by many others. The open source nature of R lends itself to sharing of ideas and collaboration between users.

1.1.1 My assumptions in this book

In statistics we always make assumptions, often these are wrong, but we adapt to our mistakes daily. My assumptions about who is reading this book are:

1. You are interested in learning more about R.
2. You are likely a student or professional interested in demography or population research.
3. You have likely been exposed to other statistical platforms and are curious about R, in conjunction with 1 and 2 above.
4. You may be an avid R user from another strange and exotic discipline, but are interested in how demographers do research.
5. You want to see *how* to do things instead of being bombarded with theoretical and often unnecessary gate-keeping mathematical treatments of statistical models.

I think if any of these assumptions are true, you're in the right place. That being said, this book *is not* a review of all of statistics, nor is it an encyclopedic coverage of the R language and ecosystem. I image the latter being on the same scale of hopelessness as the search for the Holy Grail or the fountain of youth. People have died for such fool hearty quests, I'm not falling on my sword here folks.

1.2 Who is this book for?

This book has come from several courses that I teach in our Applied Demography program at the University of Texas at San Antonio. **MORE NEEDED**

2 Introduction to R

This chapter is devoted to introducing R to new users. R was first implemented in the early 1990's by Robert Gentleman and Ross Ihaka, both faculty members at the University of Auckland. It is an open source software system that specializes in statistical analysis and graphics. R is its own programming language and the R ecosystem includes over 18,000 user-contributed additions to the software, known as packages.

2.1 Welcome to R.

If you're coming to R from SAS, there is no data step. There are no procs. The SAS and R book **?** is very useful for going between the two programs.

If you're coming from SPSS and you've been using the button clicking method, be prepared for a steep learning curve. If you've been writing syntax in SPSS, you're at least used to having to code. There's a good book for SAS and SPSS users by Bob Meunchen at the Univ. of Tennessee here, which may be of some help.

Stata users have fewer official publications at their fingertips to ease the transition to R, but I have always thought that the two were very similar. If you search the internet for information related to R and Stata, you will find a myriad of (somewhat dated) blog posts on which one is better for "data science", how to "get started" with either and so forth. What is really needed is a text similar to **?** which acts as a crosswalk between the two programs.

2.2 R and Rstudio

The Rgui is the base version of R, but is not very good to program in. Rstudio is much better, as it gives you a true integrated development environment (IDE), where you can write code in one window, see results in others, see locations of files, and see objects you've created. To get started, you should download the R installer for your operating system. Windows and Mac have installer files, and Linux users will install R using their preferred package manager.

Download R from CRAN. If you're on Windows, I would also highly recommend you install Rtools, because it gives you c++ and Fortran compilers, which many packages need to be installed.

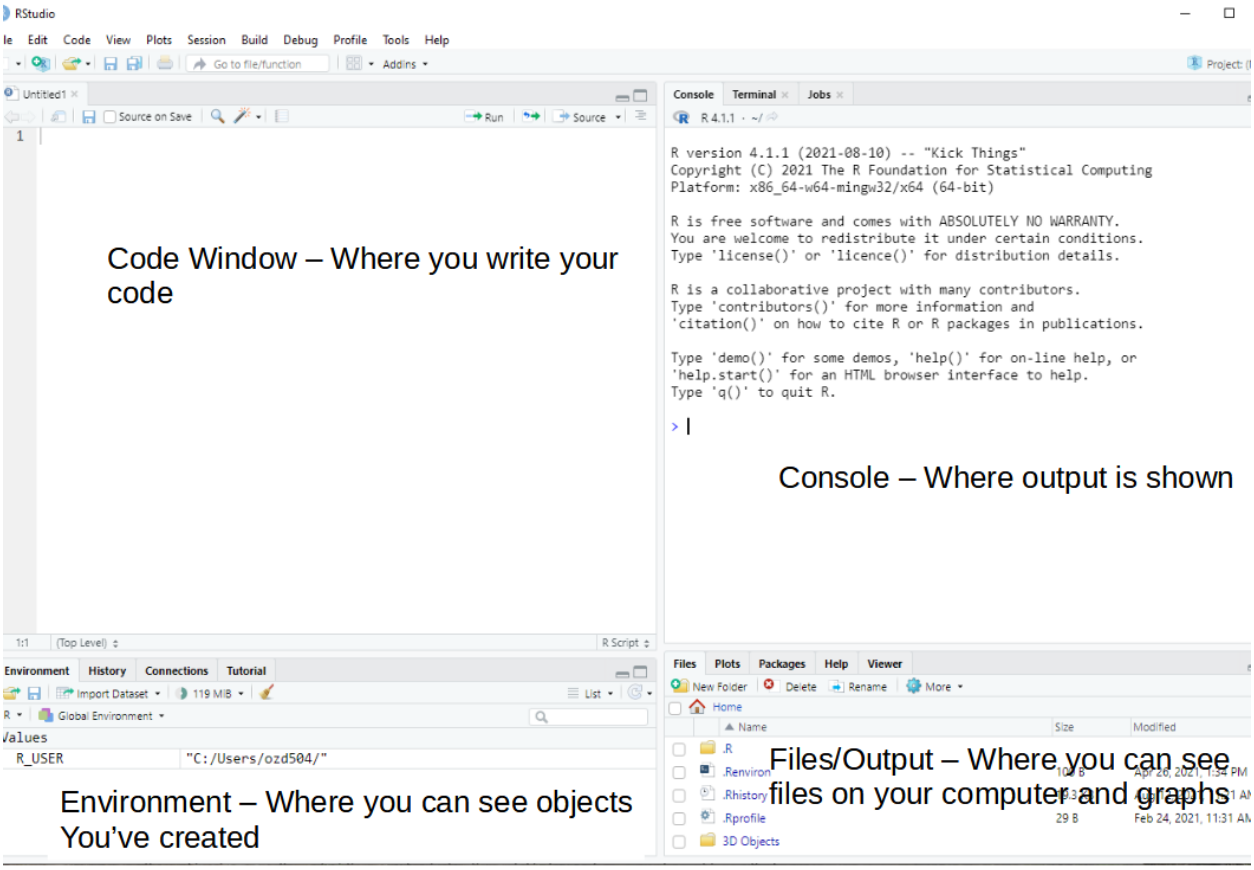
Rstudio can be downloaded for free here.

I would recommend installing the base R program from CRAN first then (for Windows users) install Rtools, then install Rstudio, in that order.

2.3 Introduction to Rstudio

Again, each operating system has its own binary for Rstudio, so pick the one that matches your operating system. Rstudio typically has 4 sub-windows open at any given time.

Rstudio is an open source Integrated Development Environment (IDE) for R. It is a much better interface for using R because it allows you to write code in multiple languages, navigate your computer's files, and see your output in a very nice single place. The Rstudio IDE has several components that we will explore.



2.3.1 Code window/Source editor pane

- This is where you write your R code. You can write R code in a few different file types (more on this later), but the basic one is an R script, with file extension .R
- The code window allows you to write and execute your code one line at a time, or to run an entire script at once. I use this to develop new code and when I want to test if things work (a VERY common exercise when writing any code).
- To run a single line of code, put your cursor on the line and hit Ctrl-Enter (on Mac CMD-Enter also does this)
- To run multiple lines of code, highlight the lines you want to run and do the same thing

2.3.2 Console Pane

- This is where most of your non-graphical output will be shown. Any numeric output will appear here, as well as any warnings or error messages. In R a warning doesn't necessarily mean something went wrong, its just R's polite way of telling you to pay attention.
- An Error means something did go wrong. This is often because you left off a) or a, sometimes because you misspelled something. I routinely spell `length` as `lenght` which causes R to print an error message. If you see an error, don't worry, R will print some kind of message telling you what went wrong.
- R's output is in plain text, although we can produce much prettier output using other output methods, and we'll talk more about that later.

- You can type commands or code into the console as well, and you'll immediately get the result, versus if you write it in the Source/Code window, you have to run it to see the result. I will often work in the console when I want to get "fast" answers, meaning little checks that I will often do to see the value of something.

2.3.3 Environment or Workspace browser pane

- The R environment is where any object you create is stored. In R, anything you read in or create with your code is called an object, and R is said to be an object oriented programming language.
- Depending on the type of object something is, you may be able to click on the object in the environment and see more about it.
- For instance if the object is a data frame, R will open it in a viewer where you can explore it like a spreadsheet, and sort and filter it as well.
- Other objects may not do anything when you click on them.
- There is also a useful History tab here that shows you recently executed lines of code from the console or the code pane.

2.3.4 Files/Output/Help pane

- The files and output area is where you can interact with files on your local computer, such as data files or code files, or images that R can open.
- This area also has a plots window that will display plots you create in R either via typing directly into the console or by running a line(s) of code from the source/code pane.
- There is also a very valuable part of this pane that lets you access the help system in R. If you are either looking for something, or you just want to explore the functions, you can get access to all of this here.

2.3.5 R file types

.R files R uses a basic text file with the .R extension. This type of file is useful if you're going to write a function or do some analysis and don't want to have formatted output or text. You can use these files for everything, but they are limited in their ability to produce reports and formatted output, so I recommend people work with R Markdown files instead.

.Rmd files Rstudio uses a form of the markdown formatting language, called R Markdown, for creating formatted documents that include code, tables, figures and statistical output. **This book is written in R Markdown!**

R Markdown is nice for lots of reasons, such as the ability to insert latex equations into documents.

$$y_i \sim Normal(x'\beta, \sigma_2)$$

or to include output tables directly into a document:

```
library(broom)
library(pander)
fit <- lm(imr~tfr+pcturban+pctlt15_2018+pctwomcontra_all,
          data = prb)
pander(broom::tidy(fit))
```

term	estimate	std.error	statistic	p.value
(Intercept)	6.209	6.551	0.9478	0.3446

term	estimate	std.error	statistic	p.value
tfr	3.392	2.006	1.691	0.09274
pcturban	-0.0923	0.04553	-2.028	0.04425
pctlt15_2018	0.8699	0.2441	3.564	0.0004798
pctwomcontra_all	-0.2114	0.06018	-3.512	0.0005757

This allows you to make tables in Rmarkdown without having to do non-repeatable tasks in Word or some other program. You can basically do your entire analysis, or a sideshow for a presentation, or an entire paper, including bibliography, in Rstudio.

2.3.6 R projects

Rstudio allows you to create a R project, which basically sets up a specific location to store R code for a given project you may be doing. For instance, this book is a single R project, which helps me organize all the chapters, bibliographies, figures, etc.

R projects also allow you to use version control, including Git and SVN, to collaborate and share code and data with others.

2.3.7 R data files

R allows you to read and write its own *native* data formats, as well as read and write text formatted files and data files from other statistical software packages. Two native R data formats are `.rds` and `.rdata` formats. `.rds` files allow you to save a single R object to an external files, while `.rdata` files allow you to save one or more objects to a file.

Here is a short example of doing this, where I create 2 vectors, `x` and `y` and save them.

```
x <- c(1, 2,3)

y <- c(4, 5, 6)

saveRDS(x,
        file="~/x.rds")

save(list=c("x","y"),
     file="xy.rdata")
```

I can also load these into R again:

```
readRDS(file = "~/x.rds")
```

```
## [1] 1 2 3
```

```
load("xy.rdata")
```

Standard methods for importing text data such as comma separated value or tab delimited files can be read into R using `read.csv()` or `read.table()` and similar writing functions are available.

To read in a dataset from another statistical package, I recommend using the `haven` package. It allows you to read and write SAS (both sas7bdat and xpt files), Stata, SPSS (both .por and .sav files).

For example, here I write out a dataframe containing `x` and `y` from above to a SAS version 7 file:

```
xy <- data.frame(x = x, y = y)
xy
```

```
##   x y
## 1 1 4
## 2 2 5
## 3 3 6
```

```
library(haven)

write_sas(data = xy,
          path = "~/xy.sas7bdat")
```

I will describe dataframes more later in the chapter.

R also has packages for reading/writing such data formats as JSON, ESRI Shapefiles, Excel spreadsheets, Google Spreadsheets, DBF files, in addition to tools for connecting to SQL databases, and for interfacing with other statistics packages, such as Mplus, OpenBUGS, WinBUGS and various Geographic Information Systems.

2.4 Getting help in R

I wish I had a nickel for every time I ran into a problem trying to do something in R, that would be a lot of nickles. Here are some good tips for finding help in R:

1) If you know the name of a function you want to use, but just need help using it, try `?`

```
?lm
```

2) If you need to find a function to do something, try `??`

```
??"linear model"
```

3) You can also search the history of other R users questions by tapping into the RSiteSearch website, which is an archive of user questions to the R list serve. This can be used by tying `RSiteSearch()`

```
RSiteSearch("heteroskedasticity")
```

4) Speaking of which, there are multiple R user email list serves that you can ask questions to, or subscribe to daily digests from. These typically want an example of what you're trying to do, referred to as a *reproducible example*. I wish I also had nickles for each question I've asked and answered on these forums.

5) A good source for all things programming is the statistics branch of Stack Exchange, which has lots of contributed questions and answers, although many answers are either very snarky or wrong or for an old version of a library, so *caveat emptor*.

6) Your local R guru or R user group. You would be surprised at how many people are R users, there may be one just down the hall, or in the cubicle next door. I relish the opportunity to talk to other R users, mostly because, even though I've used R for more than 20 years, I still learn so much by talking to others about how they use R.

Lastly, I want to be clear that there are often **more than one way to do everything** in R. Simple things like reading and writing a CSV data file can be accomplished by any of a handful of different functions found in different packages. If someone tells you that there is only one way to do something, they are usually wrong in such a statement, regarding R at least.

2.5 R packages

R uses packages to store functions that do different types of analysis, so we will need to install lots of different packages to do different things. There are over 20,000 different packages currently for R. These are hosted on one of a number of *repositories*, such as the Comprehensive R Archive Network, or CRAN, which