Mapping basics in R, a brief tutorial

Hilary Prichard

January 21, 2014

1 Purpose

This tutorial will guide you through the basics of mapping in R, from creating a geocoded data frame to mapping values as points. Links describing more advanced mapping techniques for you to explore are also included at the end of this tutorial. There are two accompanying files: `telsur-example.csv` is a file with sample data taken from the Atlas of North American English which we will be using to produce an ANAE-style map; `telsur-example.R` is a file with the R code for creating the map.

2 Data structure

In order to create a map, you first need to collect two different kinds of data: linguistic data, and geographic data.

"Linguistic data" is the dialect data you will find in the dialect atlas you choose, and it can take whatever form you like – phonetic transcriptions, coded variables, percentages, etc. "Geographic data" is the latitude and longitude data that allows R to put your dialect data on a map. In the unlikely event that your dialect data is already coded with the latitude and longitude of each location, you're good to go! Most of you, however, will need to *geocode* your data, using whatever location identifiers you have on hand, like city, state, county, or zip code.

The end goal is a spreadsheet with (minimally) columns for latitude, longitude, and linguistic variable. For example, the data we'll be working with in this tutorial looks like this:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Longitude	Latitude	Telsur_no	Pseudonym	Age	Gender	SpCom	State	IV_Date	o_oh	F1_o	F2_o	F1_oh	F2_oh
2	-149.74633	61.204466	380	Louise S	35	F	Anchorage	AK	5/18/95	0	886	1268	869	1214
3	-149.74633	61.204466	381	Iris K	32	F	Anchorage	AK	5/18/95	10	806	1239	830	1240
4	-113.48211	53.618419	653	Leila A	55	F	Ottawa	ON	5/24/97	0	734	1233	722	1224
5	-113.48211	53.618419	654	Brent M	25	M	Edmonton	AB	5/24/97	0	832	1186	779	1147

We have columns for latitude and longitude, and five columns for linguistic information. Column "o_oh" is a perception test score, while "F1_o" through "F2_oh" contain vowel formant measurements.

If you have more data than is feasible to geocode by hand, one option is to use http://www.gpsvisualizer.com/convert_input.

If you already have zip codes for your data, and you're comfortable working in R, another option is to use the `zipcode` package. This package contains a function which checks the format of your zip codes, and a data frame of latitudes and longitudes for each zip code. You can use `merge` to get latitude and longitude for your zip codes.

Once your data is collected and geocoded, save it as a .csv (comma-separated values) file so that you can easily work with it in R.

3 Getting started with R

If you've never used R before, I strongly recommend having a look at Joe Fruehwald's study group materials, at http://www.ling.upenn.edu/~joseff/rstudy/index.html. While the example code provided here does not assume any prior experience with R, it would be helpful to have a basic understanding of how R works for following along and later adapting the code to your specific needs. Weeks 1 and 2 of Joe's tutorial should get you up to speed on the basics.

4 The `maps` package

The example code attached shows you how to create maps in R using base graphics and the `maps` package: http://cran.r-project.org/web/packages/maps/maps.pdf. We will be using the "state" map in this tutorial; this gives us a pre-defined map of the U.S. with state boundaries. Other pre-defined maps are available via this package (see documentation), but should the area you're working on not be included in this way, you can always use the "world" map and set your x and y axis limits to only plot the part of the world you're interested in.

An alternative to `maps` is `mapdata`, which contains higher-resolution world maps but fewer pre-defined country maps. The documentation for `mapdata` is here: http://cran.r-project.org/web/packages/mapdata/mapdata.pdf.

5 On to the code!

Now that we've covered the preliminaries, let's look at an example of how to build a basic map in R. You will need:

1. The current version of R (http://lib.stat.cmu.edu/R/CRAN/) or RStudio (a much nicer user interface for R: http://www.rstudio.com/)

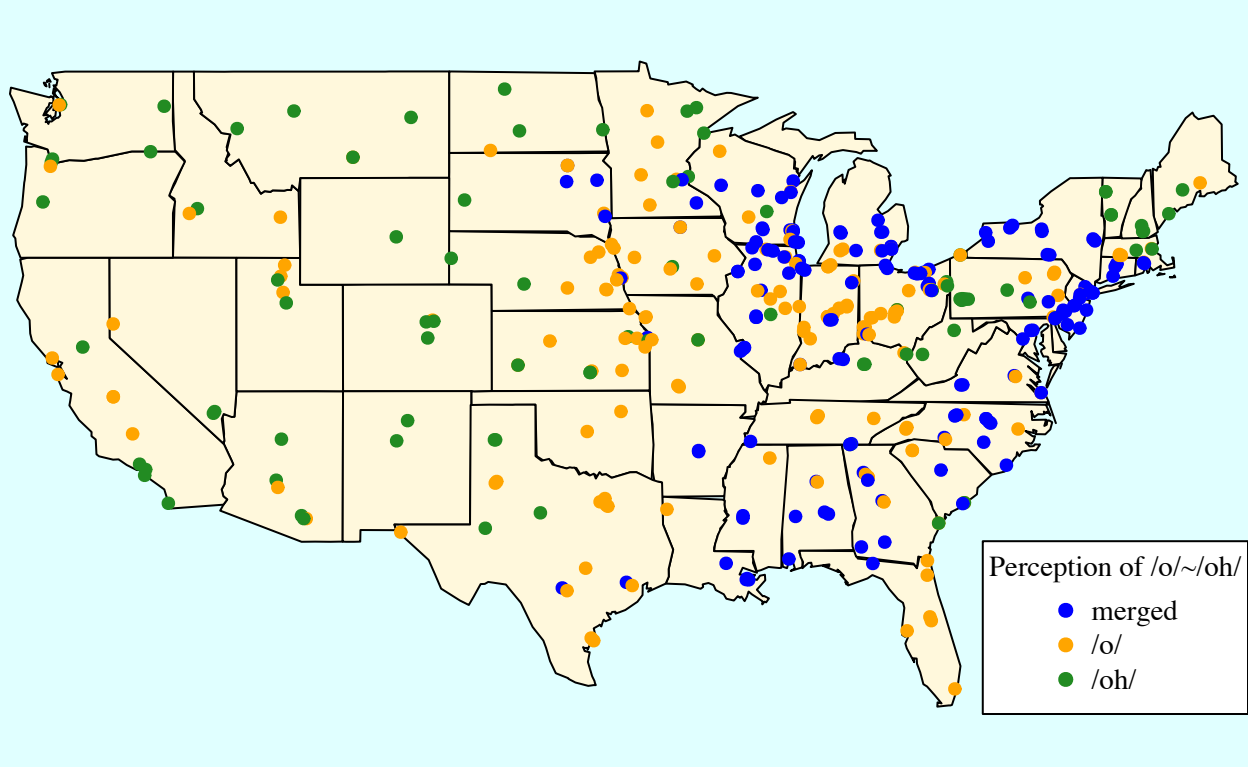
2. The `maps` package. In R, do: `install.packages("maps")`

Once you have those, open up `telsur-example.R` in R. For anyone unfamiliar with what R code looks like, all the grey lines with a pound symbol (#) at the beginning are my commentary and instructions. All other lines are for you to copy/paste into your R prompt. On a Mac, you can also highlight a line of code and hit Command + Enter to execute it.

Read through `telsur-example.R`, executing the lines of code as instructed. I've included lots of comments explaining what each piece does, but if you want to read more about the functions used in

the code, you can access the R help documents by typing a question mark in front of a function name. For example, type in `?points` for more information on different point plotting options.

When you get to the end, if everything has gone well, you should have a pdf that looks like this:



Hooray!

6 Further resources

6.1 You have questions, the internet has answers

If you ever have a question about how to do something in R that isn't covered in the help documents, chances are someone else has had the same question, and posted it on http://stackoverflow.com/. Checking Stack Overflow and googling error messages will most often get you the answer you need.

I've already mentioned Joe Fruehwald's R Study Group materials, but I'll mention it again. Week 4 has a thorough introduction to how to work with base graphics, which is worth reading for understanding what parameters of your plots you can control.

6.2 More advanced mapping to explore

1. Topography: Terrain maps can be created with `RgoogleMaps`:
   http://www.molecularecologist.com/2012/09/making-maps-with-r/

2. Rivers: or just add rivers to a map using the "rivers" map in `mapdata`.

3. ggplot: The `ggplot2` package offers an alternative to mapping in base graphics, with all the features we love about regular ggplotting:
   http://docs.ggplot2.org/current/geom_map.html
   Or explore `ggmap` for integration with Google Maps and OpenStreetMap:
   http://cran.r-project.org/web/packages/ggmap/ggmap.pdf

4. Choropleth maps are another attractive option, when it would be more appropriate to color-code entire counties or states rather than plotting individual points. This blog post has a variety of solutions to the same mapping problem:
   http://blog.revolutionanalytics.com/2009/11/choropleth-challenge-result.html

5. Animated gifs! Not just for buzzfeed anymore. Aaron Ecay has been exploring the use of animated maps to display diachronic change in syntactic variables. You should talk to him if this seems like something you'd be interested in doing.