Getting started with Rbrul for the completely clueless[1]: A basic illustrated guide to the quantitative analysis of categorical linguistic variables

Meredith Tamminga
University of Pennsylvania
Last updated 9/16/11

Introduction

This guide is about the basic steps you need to take to carry out statistical analyses of categorical linguistic variables using Rbrul, a program that runs within the statistics program R. It is not about understanding or interpreting statistics. It skips most aspects of exploratory data analysis and does not cover continuous linguistic variables. If you don't know whether your linguistic variable is continuous or categorical, you need a different guide.

A note about terminology

R is actually a programming language but you can think of it as a program that you will download onto your computer. R is different than most programs because you have to type in commands, instead of clicking on buttons, to make it do things. This is called a "command line interface" (because you type commands into lines, I guess). Many people find that this makes it difficult to use at first, since you have to learn what to type and when to type it.

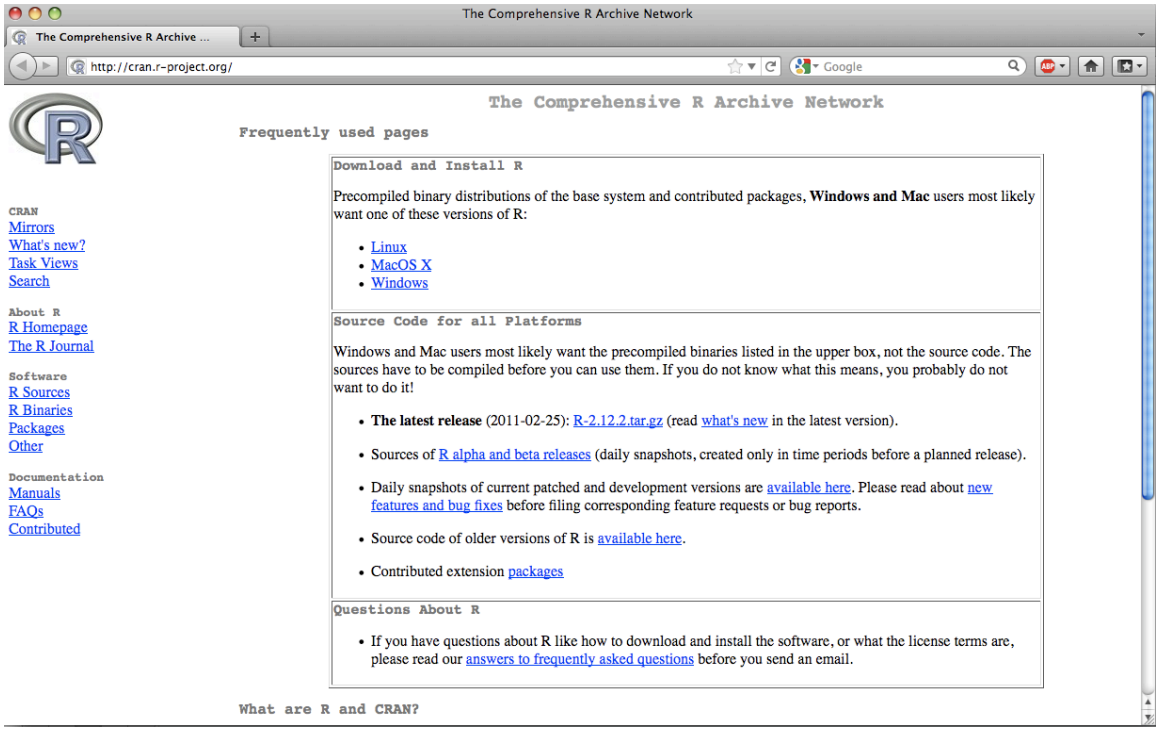
Rbrul is a program written in R. Whenever you want to use it, you will first open R and then open Rbrul within R. The good thing about Rbrul is that once it's up and running it will ask you questions with multiple choice answers, so that you don't have to know the right things to type into R.

GoldVarb is a different program that doesn't use R but does many of the same things Rbrul does. There are several downsides to GoldVarb, the most practical being that it requires a file in a particular format that isn't very flexible or easy to work with. In my opinion Rbrul is much easier to use, which is why this guide is about Rbrul and not GoldVarb. If you want to learn how to use GoldVarb, try the book *Analysing Sociolinguistic Variation* (Tagliamonte 2006).

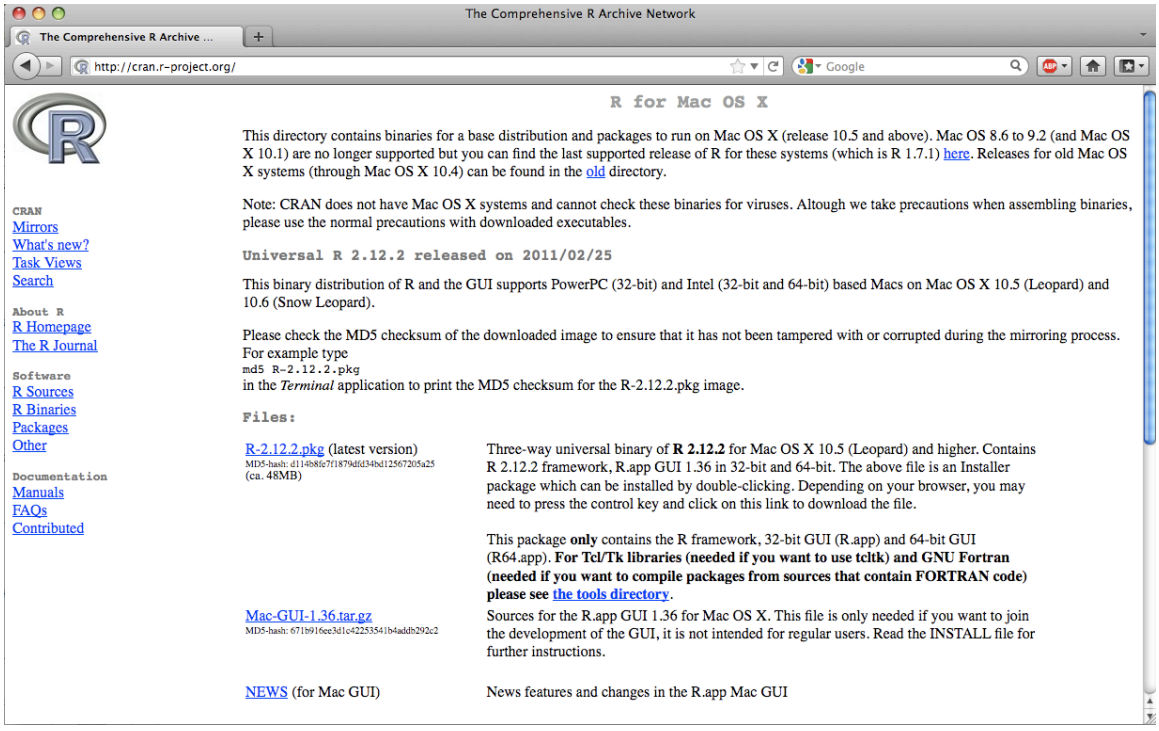
Getting started with R

First you need to download R. Go to http://cran.r-project.org/. It should look like this:

[1] I mean this in the very nicest way possible. This guide reflects the detail-oriented hand-holding I wished for when I first started learning various computer-y skills.

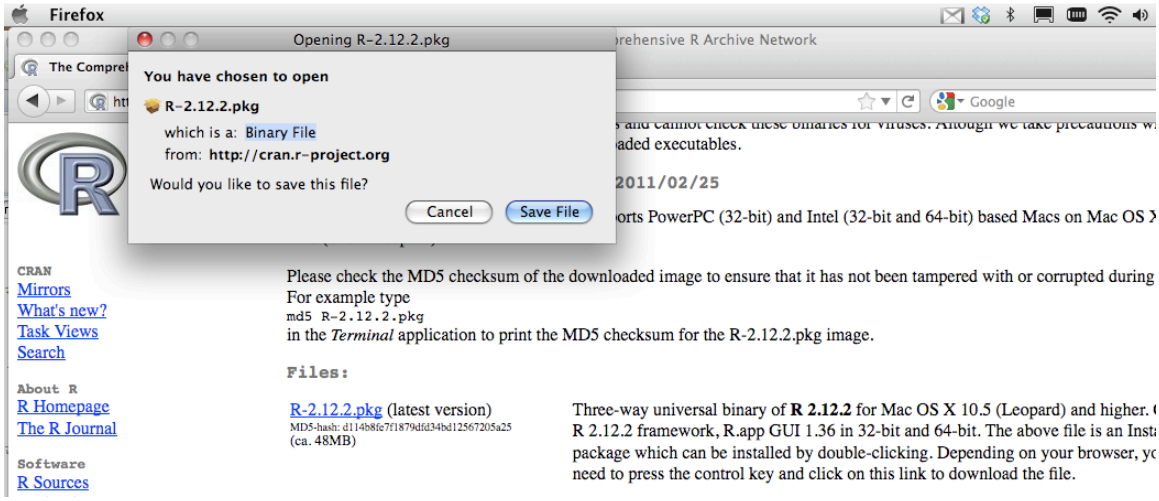


In the box that says "Download and Install R" at the top of it, click on the link for whichever operating system you're using. I'm using MacOS 10.5 so I would click on MacOS X. The MacOS X page has a lot of technical looking language on it, most of which you can ignore:



If you're not sure what version of MacOS X you have, click on the little apple symbol in the very upper left-hand corner of your screen and choose "About This Mac" from the drop-down menu.

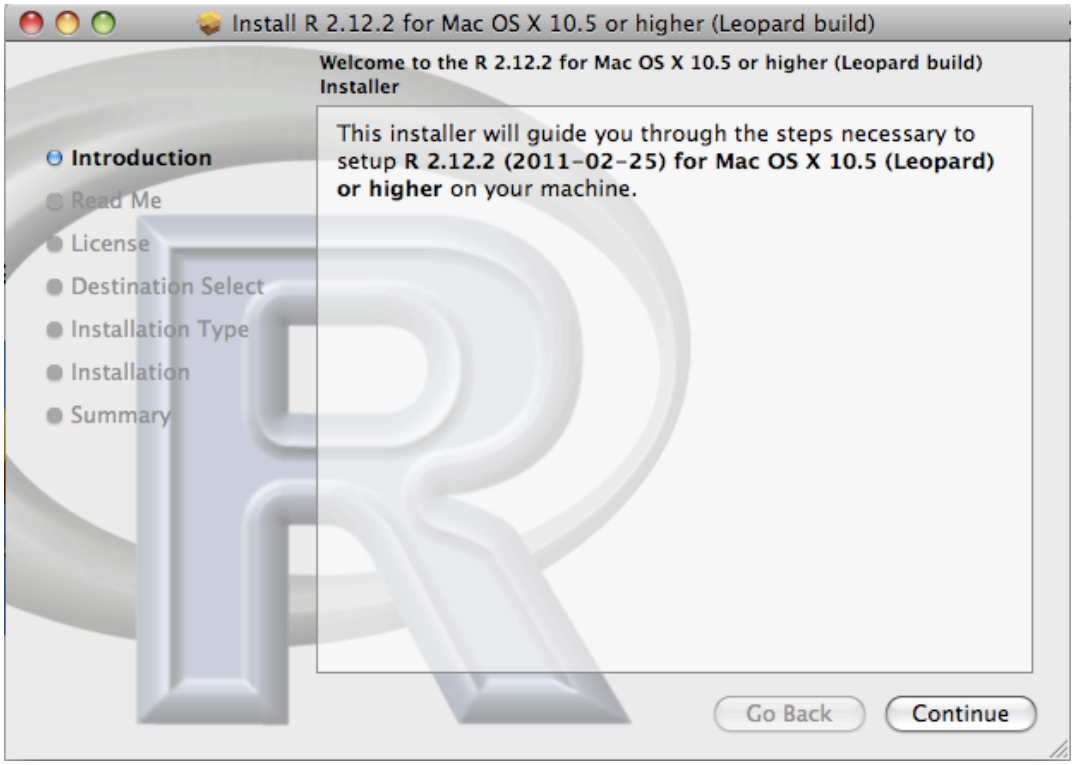
Here you can see that I'm running Version 10.5.8. This means that I can use the latest version of R. Under the heading that says "Files:", click on the link that says "R-2.12.2.pkg". My browser (Firefox) asks me if I want to save this file, which I do.



Make sure you know where your browser is putting this download! (Mine asks me where I want it to go, but yours might be set to put it in Downloads or somewhere else.)

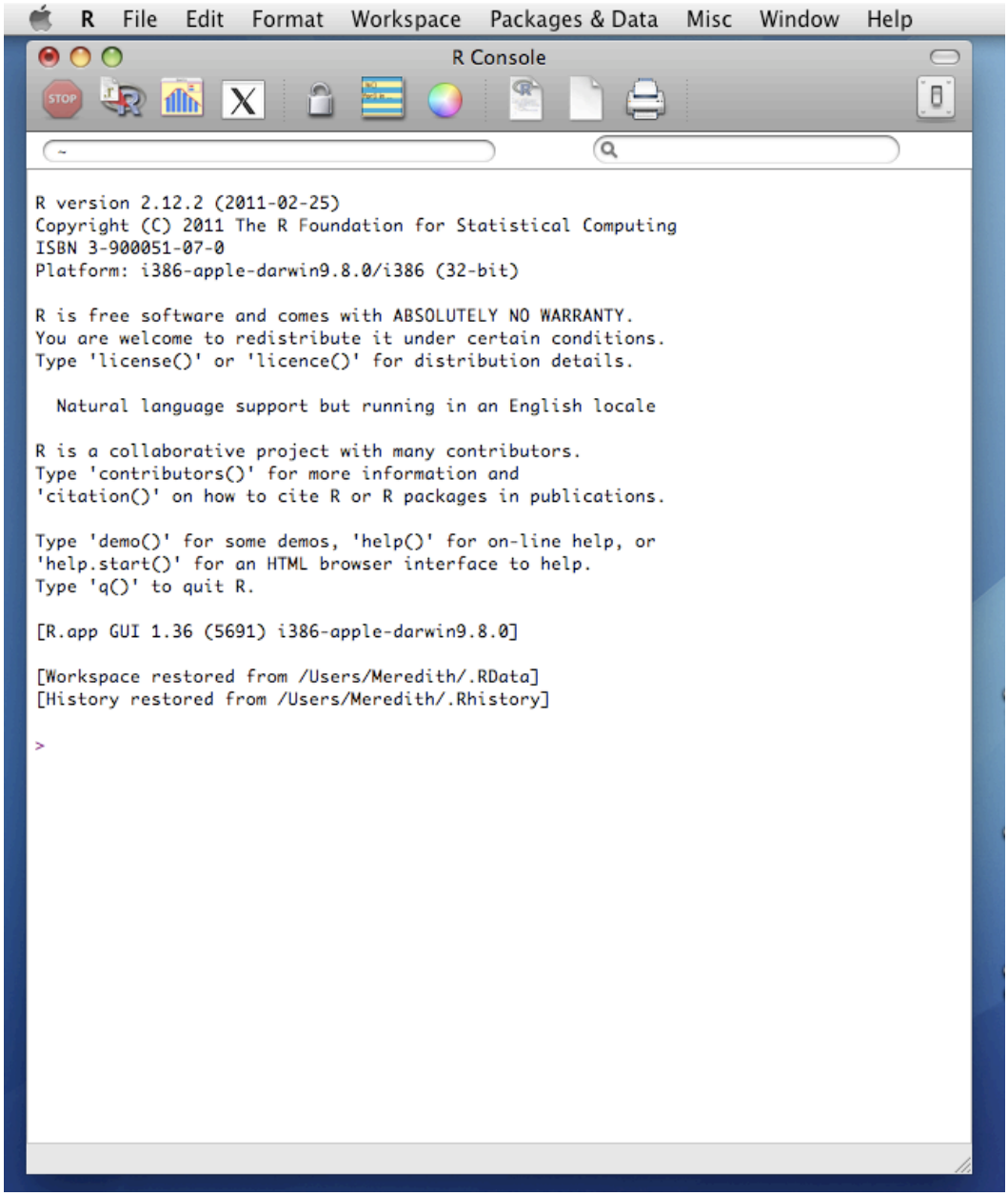
Now you need to find the downloaded file and double-click it. I put mine on my desktop and it looked like this:

When you double-click R-2.12.2.pkg, it will open up the installer, like this:



I'm not going to take you through the installer since it's completely intuitive—just answer all of the questions affirmatively (Continue, Agree, Install, etc.).

Now that R is installed, you should be able to find it in your Applications folder. Double click to open it and what will open should look like this:



This window is called the R Console. You should see your cursor blinking next to the > arrow. This is where you type things in. When I show you something that you'll be typing into R or Rbrul, I'll put the text in green. When I show you something that will pop up on display in the console without you typing it in, I'll put the text in blue (like the > arrow). Sometimes you might see R code given with the > arrow included in the example. I'm going to leave out that > arrow for two reasons:

1. So that you can copy-paste easily from this document right into R, and
2. So that you know that you don't have to type in the > arrow.

You can ignore everything above the first > arrow. Now you're ready to start using Rbrul.

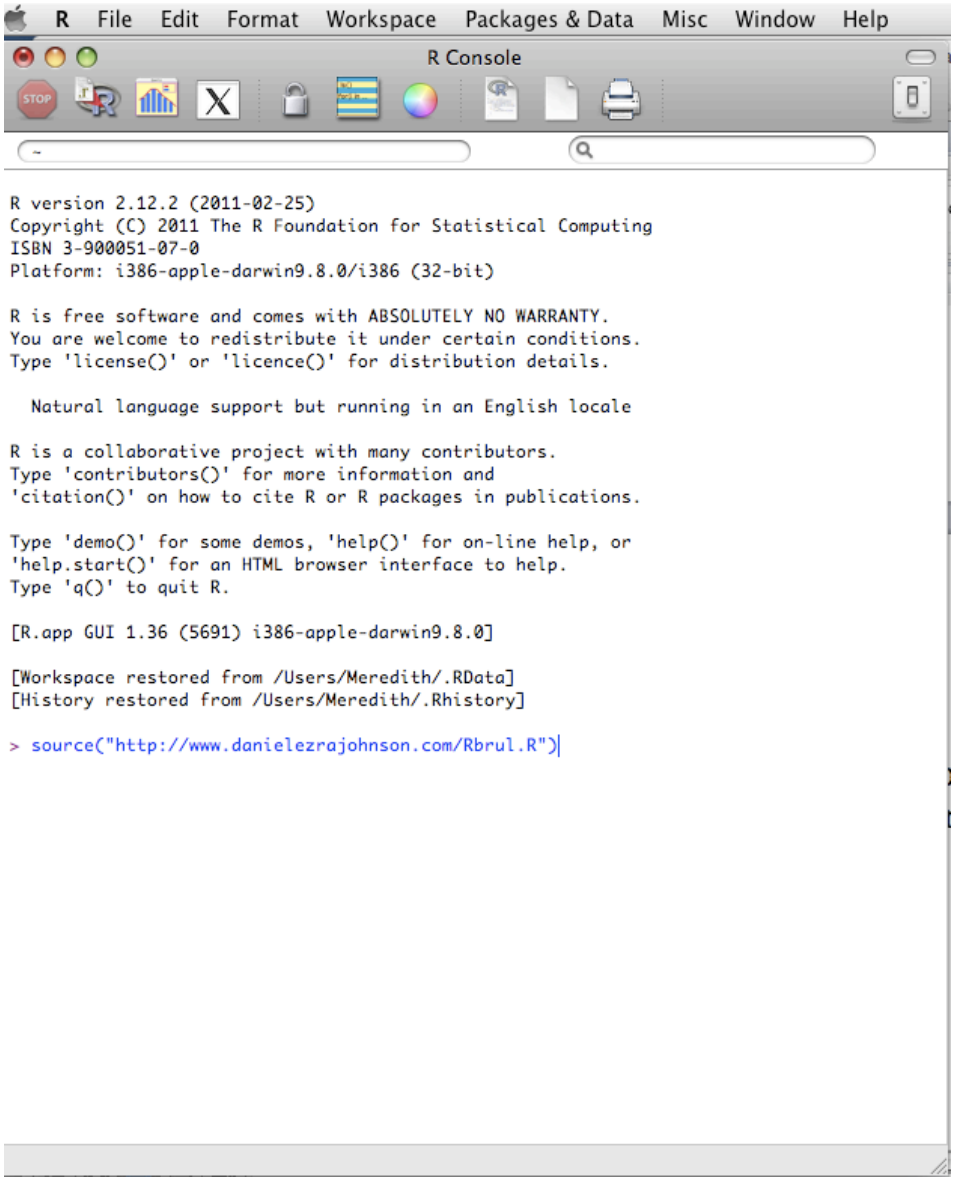
Getting started with Rbrul

There are two steps you need to take every time you start using Rbrul. The first is to *source* it and the second is to *run* it. These steps both require you to know what to type into the R console, because you're still only inside R. Once you're inside Rbrul, though, you won't have to remember things to type in any more.

When you *source* Rbrul, it means you tell R where to get the Rbrul program (the code for it is stored on a website). To source Rbrul, type into the R console, next to the > arrow:

```
source("http://www.danielezrajohnson.com/Rbrul.R")
```

This is an R command. The source part tells R what to do (go looking for a program) and the part inside the parentheses tells it where to do it (at Daniel Ezra Johnson's website, where the Rbrul code is stored). Make sure you type it in exactly like above. For example, the very first letter "s" has to be lowercase, then you have to use normal parentheses and have quotation marks inside them. There can't be any stray spaces or punctuation marks. An easy way to get it right is to copy-paste it from this document. Your console should look like this:

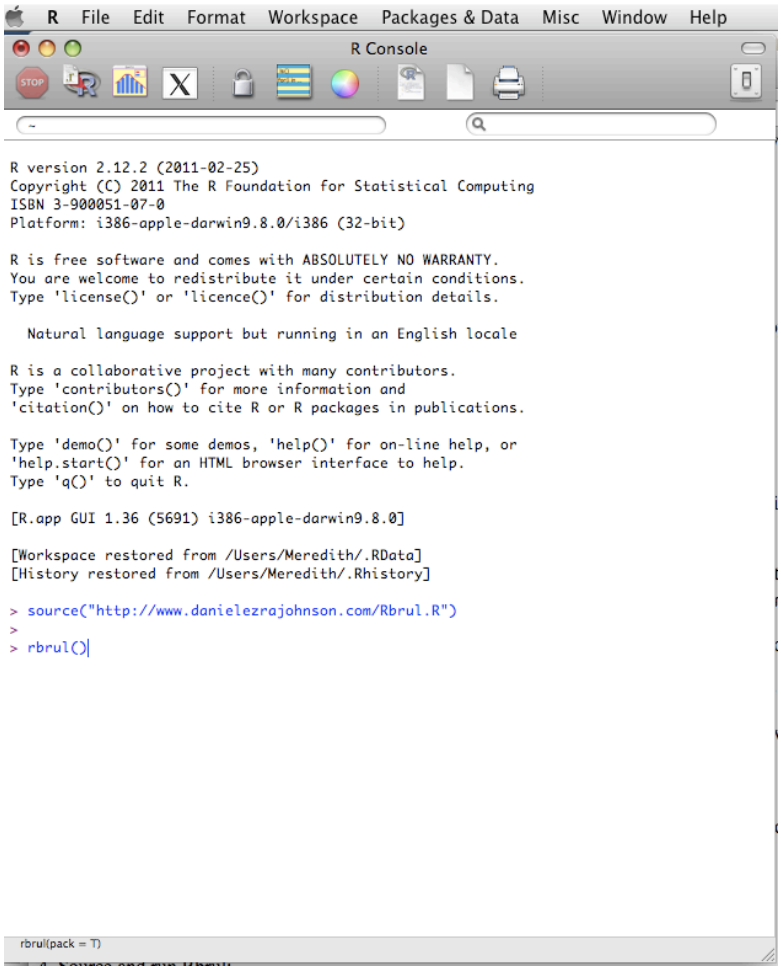


Now hit the return key to tell R to go ahead and do the command you're giving it. R will start thinking for a moment. You'll know it's done thinking when a new > pops up at the beginning of the next line down. A browser window will pop up with a message about Google Analytics. You can ignore that and close the browser window.

Now that you've sourced Rbrul, you just need to run it – to turn it on, essentially. All you have to do to run it is type:

```
rbrul()
```

This is shown below.

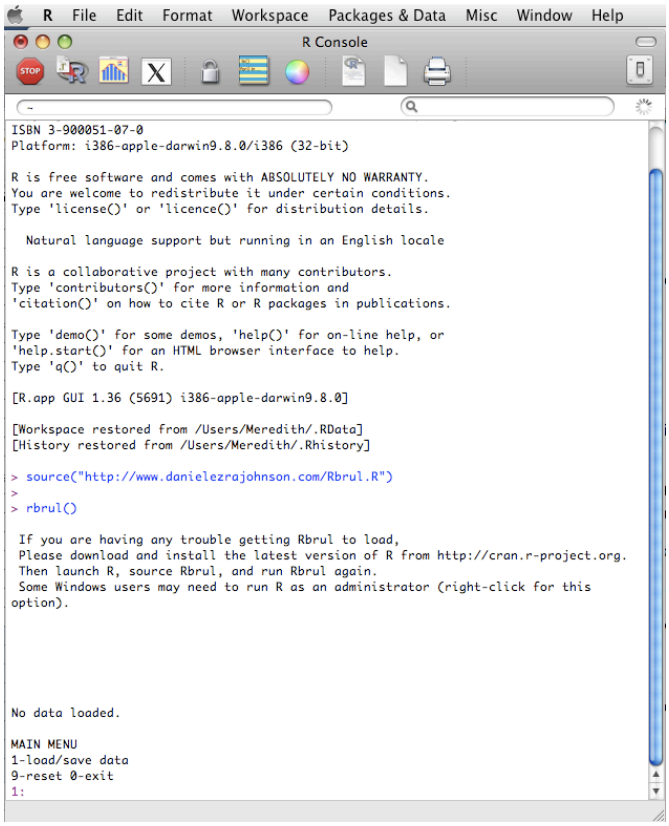


Hit return and it should look like below:[2]

[2] If you get an error that looks like this:

```
Error in source("http://www.danielezrajohnson.com/Rbrul.R") :
  http://www.danielezrajohnson.com/Rbrul.R:1:1: unexpected '<'
1: <
   ^
```

…just try the same thing over again. It should work the second time (this is just a harmless bug that pops up once in a while).



Now you can see that we're at the MAIN MENU down at the bottom of the console. The MAIN MENU has three options, each of which has a number:

- 1 – load/save data
- 9 – reset
- 0 – exit

Below these options it says 1: in a different color. This is where you will type in the *number* of the option you want from the menu. From the options available right now, for example, we can see that if we wanted to exit the program, we would type in a 0.

As it says above the MAIN MENU, no data has been loaded yet. That will be our first step now that Rbrul is running. First, though, we need to have data that Rbrul will know how to deal with.

Making sure your data structure is right

Let's assume that your data is in an Excel spreadsheet, since that's a very common format for data to be in. The spreadsheet will need to be set up in a specific way so that Rbrul will understand it:

- Every row of the spreadsheet needs to be a token (also known as an observation – a single instance of your dependent variable).
- One column must contain the dependent variable, or *response*.
- The rest of the columns must contain the independent variables, or *predictors*.

You cannot have a spreadsheet where each row represents a single speaker and multiple tokens/observations from that speaker are in separate columns.[3]

Here's an example of how your spreadsheet should be set up:

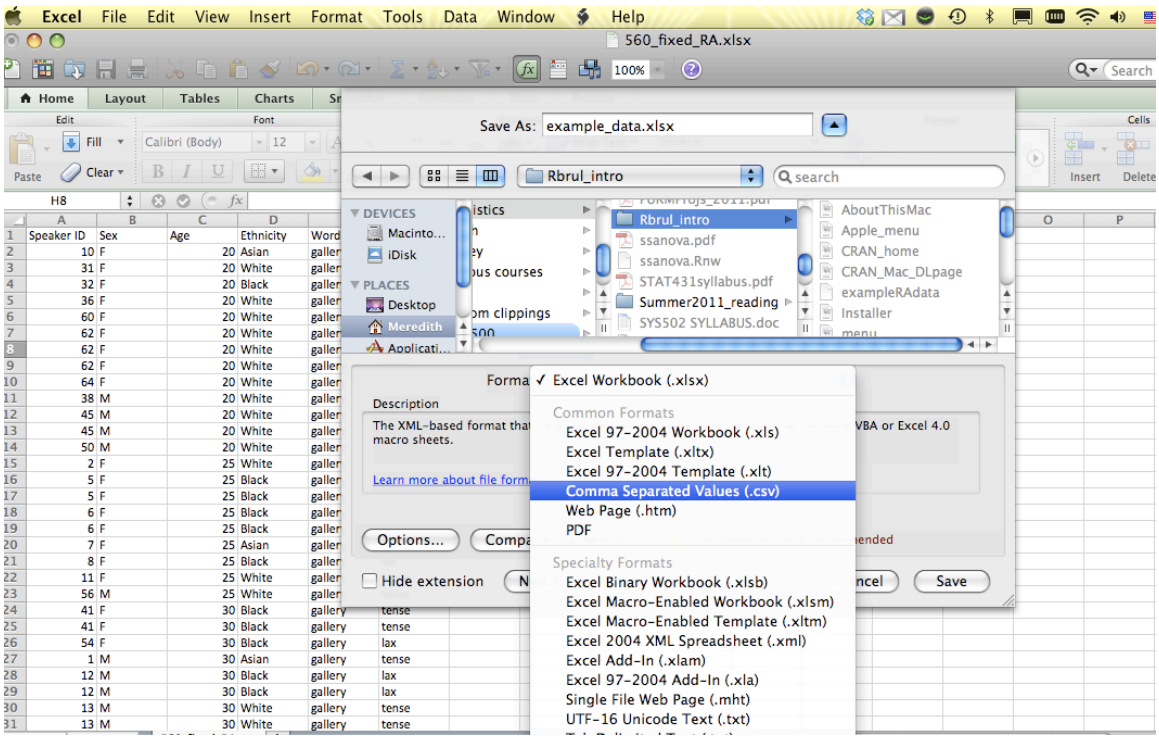
	A	B	C	D	E	F	G
1	**Speaker ID**	**Sex**	**Age**	**Ethnicity**	**Word**	**short_a**	
2	1	M	30	Asian	gallery	tense	
3	2	F	25	White	gallery	tense	
4	3	F	40	White	gallery	lax	
5	4	F	35	Black	gallery	lax	
6	5	F	25	Black	gallery	tense	
7	5	F	25	Black	gallery	lax	
8	6	F	25	Black	gallery	tense	
9	6	F	25	Black	gallery	tense	
10	7	F	25	Asian	gallery	lax	
11	8	F	25	Black	gallery	lax	
12	9	F	45	Black	gallery	lax	
13	9	F	45	Black	gallery	lax	
14	9	F	45	Black	gallery	lax	
15	10	F	20	Asian	gallery	tense	
16	11	F	25	White	gallery	tense	
17	12	M	30	Black	gallery	lax	
18	12	M	30	Black	gallery	lax	
19	13	M	30	White	gallery	tense	
20	13	M	30	White	gallery	tense	
21	13	M	30	White	gallery	tense	

The dependent variable here is given in column F: short-a is either tense or lax. The other five columns are independent variables. Notice that both the variable names ("Ethnicity") and values ("Asian") are informative (instead of, say "E" and "A"). This will make your life easier and is one of the nice things about Rbrul. When I have more than one short-a token from a certain speaker, the tokens go in separate rows, as in rows 6-7 or 8-9. It doesn't matter what order your rows or columns are in, so long as each row is a token and each column is a variable.

Finally, you should turn your Excel file into a .csv file. This is a comma separated value file. It's just a plain text file that can act like it's a spreadsheet because there are commas marking off the columns in each row. To convert your Excel

[3] If your data *is* in this non-Rbrul-friendly format, that's another case where you need a different tutorial.

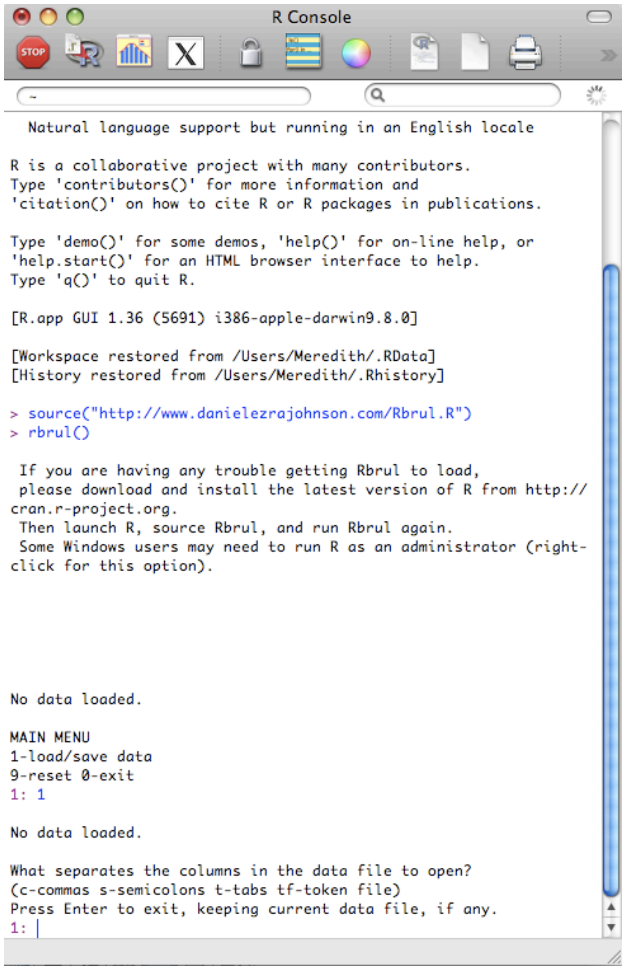
spreadsheet into a .csv file, go to File → Save as… in Excel, then choose Comma Separated Values (.csv) from the Format: dropdown menu.



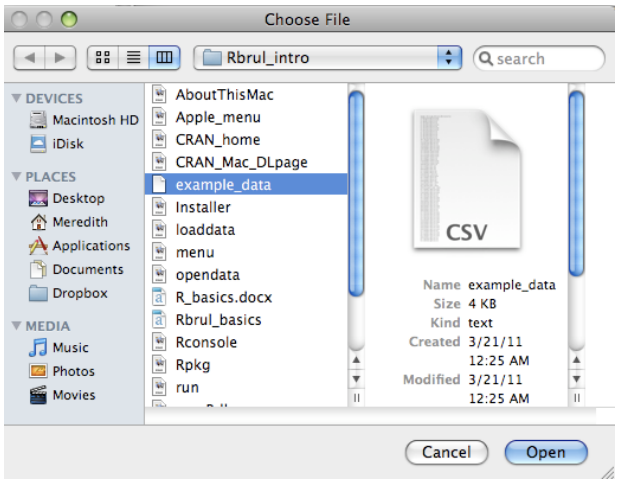
Make sure you keep track of what you name your data file and where you put it!

Loading data

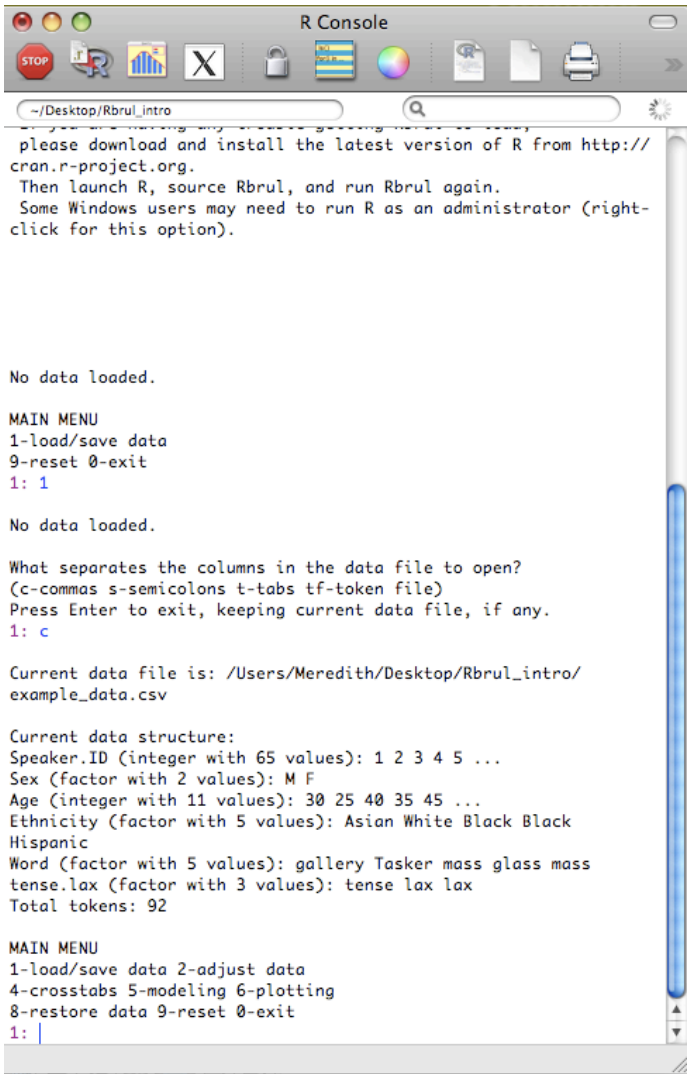
Now you want to load your data in Rbrul. As you'll recall, we were looking at three options in the MAIN MENU. Luckily, one of them was 1 - load/save data. To choose this option, just type in a 1 at the prompt:



When I typed in 1, it told me that there was no data loaded (so I would know that "save" isn't an option) and then asked me what separates the columns in the data file I want to open. We just saved our data file as a .csv, so the correct answer is "commas". Type in a c and hit return. A box should pop up for you to navigate to wherever you saved your data file. Hit Open.



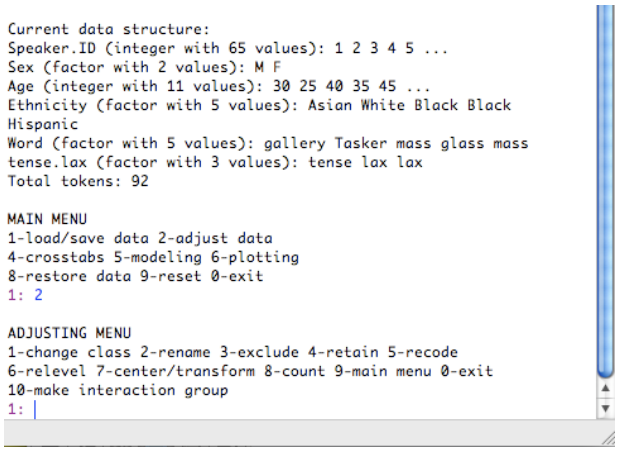
And just like that, Rbrul will have your data in memory. It will prove it by telling you a bit about it under the heading Current data structure:



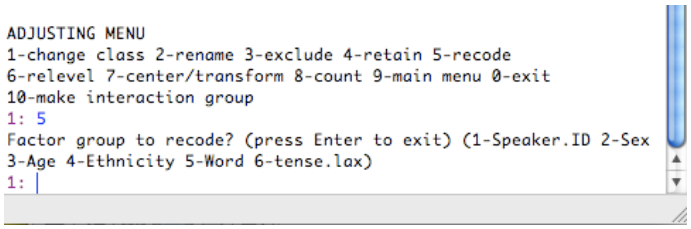
This is a very good point to pause and look closely at what it says about your data structure. The "factors" are the columns from your spreadsheet, and the "values" are the unique things contained within each column. But if you look at mine closely, you can see that something funny is going on. It seems to list "Black" twice as a value of the "Ethnicity" factor, "mass" twice as a value of the "Word" factor, and "lax" twice as a value of the "tense.lax" factor. The problem here is that Rbrul is very, very precise when it reads your data file. If you have an extra space at the end of what you type in a cell, Rbrul will treat it differently than if you typed the same thing but without a space at the end. You could go back to your original data file to fix this, but you can also fix it right here in Rbrul. So we'll do that next.

Adjusting data

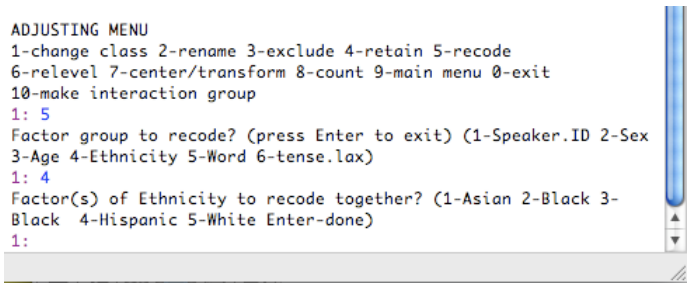
Now that you have data loaded, you'll notice there are more options in the MAIN MENU. Right now we want to adjust the data, so type in a 2 (I'm going to stop showing the whole R console in every screenshot at this point, and just show the relevant part we're working on at the bottom).



Now we're looking at the ADJUSTING MENU. We're going to **recode** the seemingly-identical values we noticed so that we combine the duplicates and end up with only one value for "Black", "mass", and "lax". So, type in 5 to recode.

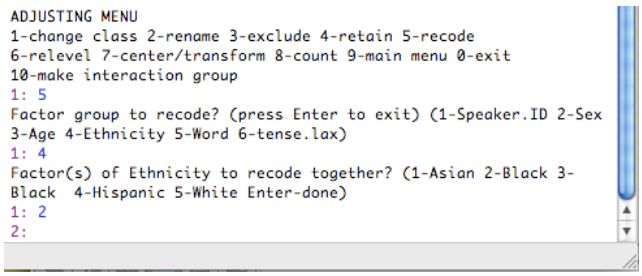


Now it asks us what factor we're recoding. Let's start with "Ethnicity", so type in 4.

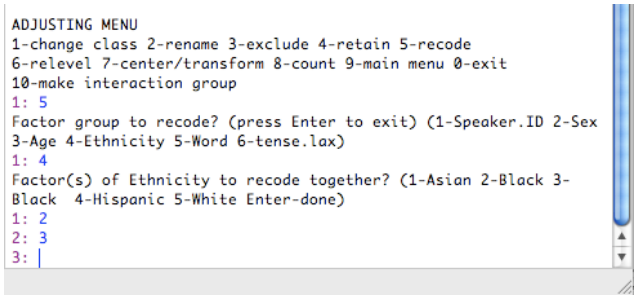


Now we have to choose which things to recode together, and we're about to learn an important detail about using Rbrul. When you want to choose two or more different options (here, the two instances of "Black" that we want to recode together), do not type them both into the same line. Instead, first type one at 1:, then hit enter, then type the second one at the 2: that will pop up. Like this:

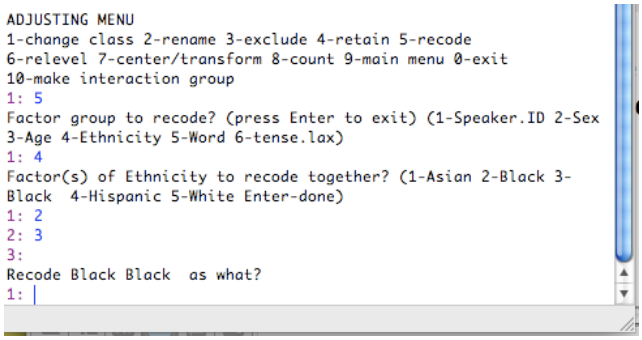
First do one…:



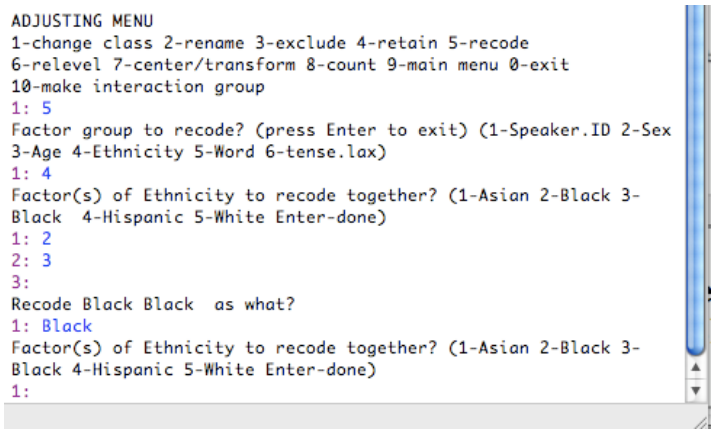
…then the other:



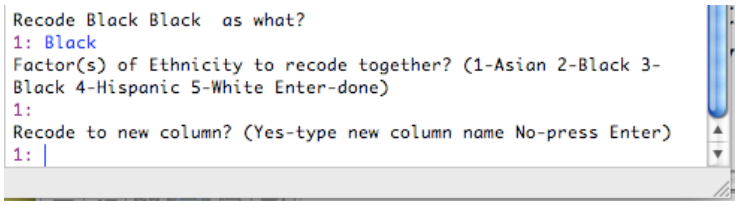
I entered 2 and 3 because those are the numbers assigned to the two instances of "Black". When you're done entering things, just hit enter one more time without typing in anything.



It's asking what to recode "Black" and "Black" as. The logical answer here is "Black", so go ahead and type in Black at the prompt.



Now it brings us back to the same recode option for the same factor. Instead we want to hit enter to say we're done with recoding "Ethnicity". It will ask whether you want to Recode to new column?

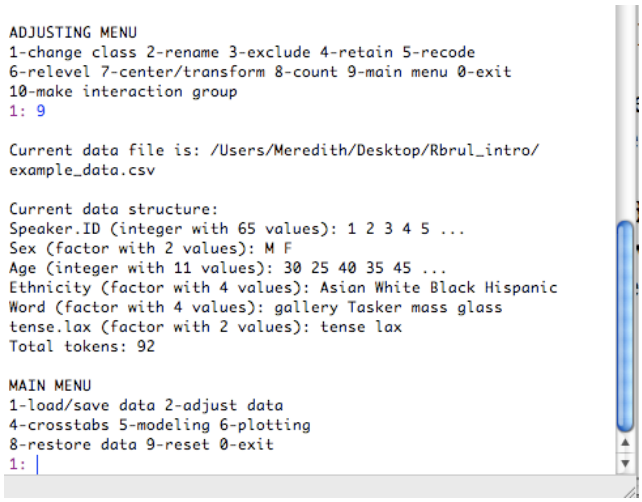


Because we're using recoding to fix a mistake in the data structure, we just want the recoded (correct) version to replace the old (incorrect) version, so we can hit enter.[4] Have you noticed that Rbrul is basically telling you all the same things I am? That's one of the reasons it's so easy to use. Instead of having to remember what to type the way you do in R, you just choose from the options Rbrul gives you at every step. It's less like writing a novel and more like doing a choose-your-own-adventure novel.[5]

At this point, Rbrul will bring you back to the ADJUSTING MENU.

I went ahead and recoded the problems with "mass" and "lax" as well, but I won't show that here. Instead, we'll go back to the MAIN MENU. From the ADJUSTING MENU, we get back to the MAIN MENU by typing in 9.

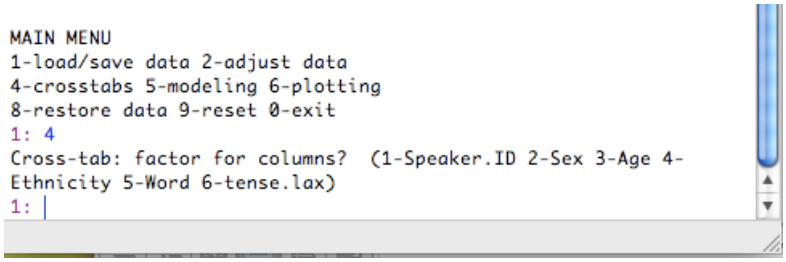
[4] If instead we were playing around with alternative ways of coding our data and might at some point want to revisit an older coding scheme, we would recode to a new column every time so we had access to each coding scheme we had tried.
[5] To extend the analogy, working in R gives you the freedom to do whatever you want with your data, while Rbrul limits you to a certain set of options and outcomes.



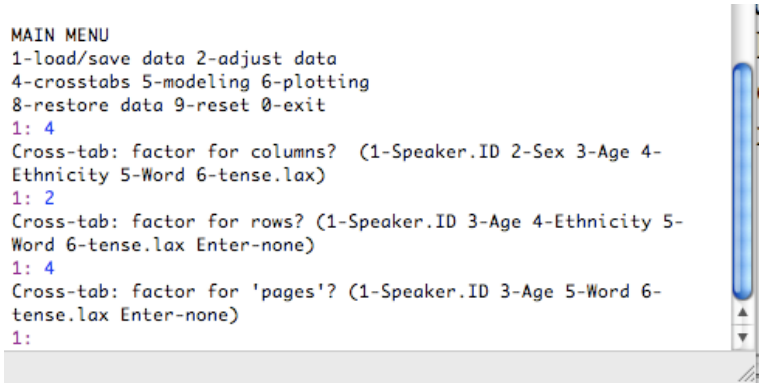
This should be starting to look familiar.

Doing cross-tabs

Next, we'll start actually looking at our data using a cross-tabulation. Type in 4.

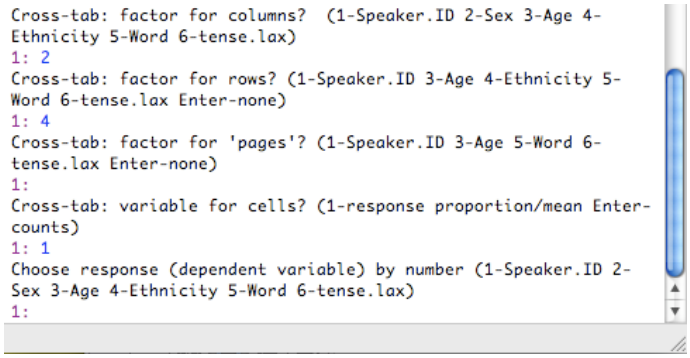


When we look at a cross-tab, we're looking at two independent variables at once. I'm going to cross-tabulate "Sex" and "Ethnicity". I'm going to make each row an ethnicity value since there are more of them (and I have endless vertical space in the console) and put the sex values in columns since there are only two (and I have limited horizontal space).

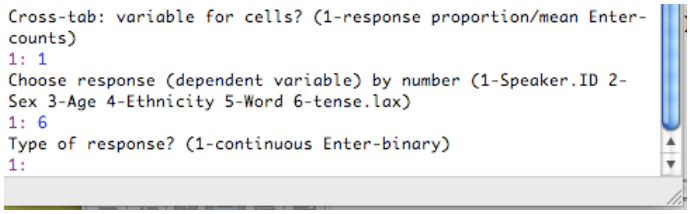


We'll ignore the option to do 'pages' for now (this is how you cross-tab more than two independent variables at a time), so just hit enter.

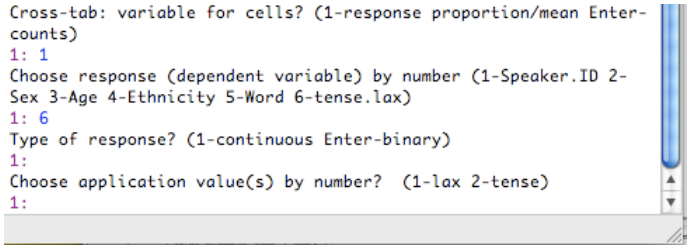
My dependent variable in this data set is "tense.lax" (that is, whether the speaker pronounced a short-a token with a tense or lax vowel). I'd like to see the response proportion/mean in my cells right now, because I want to be able to easily compare the rates of short-a tensing across the different cross-tabulated categories, such as Asian females or White males.[6]



So finally I get to choose my response/dependent variable! I'll have to say whether it's continuous or discrete:

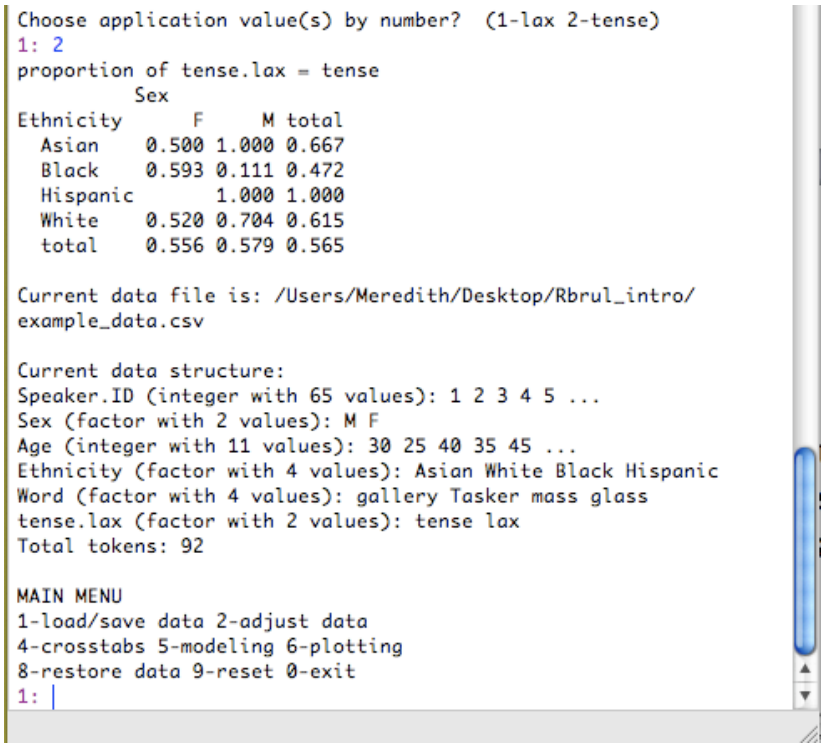


As the title of this tutorial says, I'm dealing with categorical data, so I'll hit enter to indicate that the "tense.lax" factor is binary. Next up is a question about the application value:



[6] You might want to get the counts instead if you were going to do a Chi-square Test. If you're reporting the results of a cross-tab to someone else, though, you generally want to use proportions.

The application value is this: do I want to know the proportion of lax tokens, or the proportion of tense tokens? It doesn't really matter so long as I always know which one I've chosen. I'm going to choose "tense" in this case.



And now I have my cross-tab! Sometimes if Rbrul gives you a lot of output at once, it's easy to not even realize it and wonder how you ended up back at the MAIN MENU all of a sudden. Make sure you scroll back up if needed to see where you were last and all of the output you've gotten.

Looking at this cross-tab, I can conclude that Asian males have more tense short-a's (100%) than Asian females (50%), and white males have more tense short-a's (70.4%) than white females (52%), but black males have *less* tense short-a's (11.1%) than black females (59.3%). In other words, I've discovered an interaction of sex and ethnicity, which is just the kind of thing cross-tabs are meant to do.