MICHAEL CLARK
CENTER FOR SOCIAL RESEARCH
UNIVERSITY OF NOTRE DAME

R & SOCIAL SCIENCE

GETTING STARTED WITH APPLIED USE OF R IN THE SOCIAL SCIENCES

Contents

Preface

Current draft August 11, 2014.

My Journey to R

A little background. While I could go into boring detail, suffice it to say, and as much as it pains me to do so, I was born an SPSS menu-clicker, as far as my stats training goes. I started making the switch over 10 years ago thanks to an acquaintance who had spent more time in the stats world and was using R. Unlike SPSS, whose base offering seemed to center around the content of statistics textbooks of the 1970s, I found that R could easily pull off the modern methods statistically minded individuals in social science and psychology were requesting that applied researchers do. But it could do a lot more, and the only limiting factor was me. I never learned a thing about statistics using SPSS, while I was learning something new all the time with R. When I began using it to teach, and despite the students' having the very same grumblings I had about programming, there would actually be light bulbs coming on when they used it. Again, this isn't something I witnessed with SPSS. It was also nice to have a tool they and anyone could afford. Due to R's rise in popularity you can now get a free student, i.e. crippled, version from SPSS, as well as use R code within it's syntax, as well as offering a developer pack so that you can use R to make SPSS better (for a price of course). If nothing else, *even SPSS's own founder jumped ship to R years ago*, saying "R is an absolutely massive advancement on the kind of analytics I invented. It's an opportunity to change the game in the fastest-growing field in software."[1].

[1] I'm not sure what analytics he's referring to as having invented.

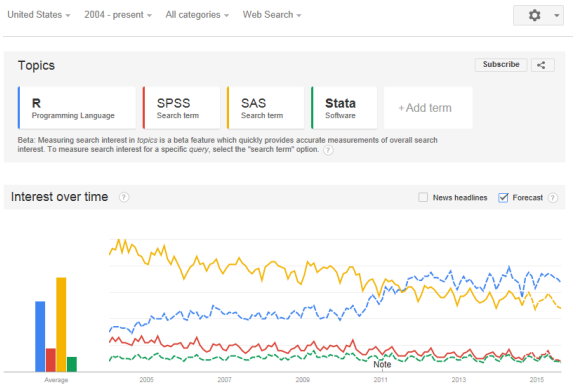
R is both free and freeing. It is not tied down to any particular discipline or industry persuasion. It is easily modified, expanded, and enhanced. In short, R is there to do with what you want statistically, and the possibilities are endless. You will learn more about your data, your models, and about statistics, by programming in R than you would using canned routines in a typical statistics package. Is it perfect? No, no programming language is. But it will do what you want, and what else really matters aside from that?

While personally I feel the writing was on the wall for applied researchers years ago, the following document will attempt to provide specific reasons for someone else to make a similar switch, particularly those in the social sciences. By the way, this document was created *with R* in the same environment used for statistical programming[2]. For other examples you can see my documents here.

[2] Using knitR and other packages, mixed with LaTeX

R on the Rise

R's popularity has been meteoric over the past 10 or more years. But you don't have to take my word for it. As a starting point we can note a Google trends plot over the past 10 years. Note that for the following, since there was no special topic for SPSS programming or statistics, or SAS outside of SAS Institute, so their absolute numbers may be inflated by other things (especially SAS), but the absolute values aren't as important for our purposes as the trends. The only one notably on the rise is R.



R Popularity Summarized

In fact there is a regularly updated website devoted to the popularity of R, maintained by the author of books that help people to migrate from SAS, SPSS, and Stata[3]. I will provide a brief summary[4].

Job Advertisements For analytics jobs, R is on the rise while SAS and SPSS are stagnate and other statistical packages aren't even common. You'd be better off with Python than SPSS skills in the modern analytics world.

Google Scholar Not exactly telling as most articles seem to want you to think that their statistics arise magically from the authors' skulls and the tools used are left uncited, but SPSS has simply plummeted in Google scholar hits since 2005, while SAS has lost notable ground since 2008. R, and even Stata, have gained.

Published Books R has provided SAS and SPSS a 20 year head start, but has already caught up to SPSS, while SAS is still a ways off.

Website Popularity R is second only to SPSS in terms of links to its homepage, though really that is a comparison to IBM as IBM.com is

[3] I should note that I've never come across a book with a title that goes in the reverse direction. If you are proficient at R, you have enough programming skill to learn the others, but there is also no need to.

[4] Several of these are based on results from two years ago.

SPSS's homepage. SAS is a distant second, and Stata notably lower than SAS.

Blogs It's not even remotely close here. There are 550 blogs currently listed at R-Bloggers.com, while SciPy (Python library) has only a tenth devoted to it, which is still more than SAS and Stata combined.

Discussion Lists From their own to discussion forums to popular programming sites like CrossValidated, StackOverflow and TalkStats, R dominates. SAS catches up at places like LinkedIn and Quora. SPSS and Stata are way off the pace.

Popularity Measures Several (obviously limited) surveys at sites like KDNuggets ask big data and analytics folks what tools they're using, and R is again more widely used than traditional statistics packages, followed by SAS and SPSS. Stata isn't even on the map here.

There are plenty of bones to pick about trying to assess any of the above outside of a few of the measures, but I think it's safe to say that R is very popular right now. Companies like Google, aside from general programmers, aren't looking for SPSS and SAS proficiency, they're hiring R programmers. For data driven graphics, the NY times graphics team is using R and other programming languages like JavaScript on the backend, not any other statistical packages[5]. So if industry is shifting to R and other programming tools, and the statistical community long ago adopted R as their primary tool of choice, it stands to reason applied researchers might want to use it as well, and the fact is a great many of them already are.

R has really become the second language for people coming out of grad school now, and there's an amazing amount of code being written for it. You can look on the SAS message boards and see there is a proportional downturn in traffic. Max Kuhn, associate director of nonclinical statistics at Pfizer in a New York Times article on R from *2009*.

[5] See the blog of their Graphics Editor here and here for more insight into their process.

In the following I'll show some specific R tools framed within social science applications. Nowadays it's not uncommon to find someone doing social science research might be a physicist, biologist, or computer scientist, though I have more traditionally trained social scientists in mind. I'll specifically note a few applications in political science, psychology and sociology. After that I'll provide specific programming examples to show how easy it is relative to other programs. Then I'll suggests some additional thoughts that go beyond the standard approaches.

Overview of R

I'm not going to provide a full introduction to R. If you want that you can see my introduction here. However an overview is in order to give one the proper context for comparison to alternatives.

R is not a stats package

R is not a "stats package". R is a true *programming language*[6]- it is a dialect of S, which has won awards and has almost as long a history as SAS and SPSS. Beyond that it is best thought of as a programming environment within which statistics is conducted. In a sense it has more in common with Python than it does with the syntax used in stats packages, and that has far-reaching effects for data manipulation and exploration, post-processing of models, and visualization.

Base R has most of the functionality you'd find in a traditional statistics package and beyond, plus a system for creating fantastic visualizations. R is open-source, which means that one can inspect and modify its contents freely. Because of its programmability, it is easy for others to provide additional functionality in the form of *packages*, to date of which there are almost 6000. So with R you have a fully functioning statistical system to start with, and its open nature and programming language allow others to add more features all the time. This is part of the reason for its popularity, as well as it always being able to provide cutting edge techniques to more applied users.

You will have to beef up some programming skills to use R effectively, but serious statistical analysis has always required that, at the very least for the initial data preparation. Only now it will be easier. R is what is referred to in programming languages as an *object-oriented* programming, and once you get used to it for your statistical and data needs, it will be hard if not impossible to go back to the inefficient methods of statistical package syntax[7]. Yes you can use those to do many of the same commonly employed approaches to data manipulation, and it is easy to get basic model results. However with R it is no more difficult to work with 1 vs 10 or 100 data sets, you don't have to create explicit representations of variables to use them in modeling or for simple data exploration, you can avoid loops entirely for common iterative tasks, prediction is built in to the vast majority of modeling functions, you have many plotting methods for results of analyses built-in, you can easily call in or even write and use other languages for certain tasks, and you can easily write your own functions (even implicitly) and modify others, engage in interactive graphics, write dynamic, publish-ready documents. etc. One key idea to note- *you do not have to be a great R programmer to use it effectively and do some pretty amazing things, but you do have to be an excellent programmer to do those things in other statistics packages*.

Much of the apprehension I see from non-R users is that because of the handcuffing of their own tools, they simply cannot imagine the sorts of things they *could* be doing with R. As I started using it, my initial attempts would make me wonder if I could take it further. Once

[6] Contrast with a syntax developed for using punch cards on mainframe computer.

[7] Some offer more developed programming languages now, e.g. Stata's Mata language, but the user communities are relatively smaller because those who program like that have long had other alternatives.

I did, I could imagine what else I could do, and tried that. And this cycle continues to this day. As mentioned, the only limitation to what you can do with R is you.

The later demonstrations will hopefully provide greater insight on the R approach in the social sciences. For now it is simply important to note that R is fundamentally different from traditional statistics packages. It will take some getting used to, but the payoff is worth it, and you wouldn't be doing research in the first place if you were afraid of hard work.

R for Political Science

A lot of political scientists seem to like Stata, and for good reason, it's simply a good statistical tool. But what might R offer to a political scientist? The answer is plenty.

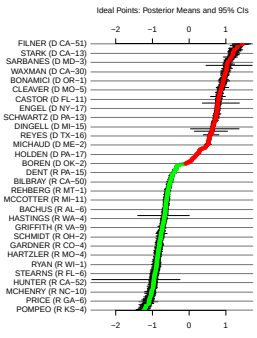
As a starting point Much of political science methodology is based on econometrics, and there is a slew of packages that would fall within that realm. There are political scientists who have provided specific packages such as Gary King and the IQSS group at Harvard, the Political Science Computational Lab at Stanford, and even from the President of the Society for Political Methodology. On the data side one can use a packages such as WDI for world bank data, dvn for access to the Dataverse Network, and a more recent offering, psData, that also looks

Example

Let's provide a political science demonstration. Recently I was interested in looking at investigating cliques within the congress, and as a starting point we can examine how they vote on bills and where they line up within their party. Let's say I would like rollcall data from the current House of Representatives and make map that reflects their spatial position in terms *ideal point estimation*. This is not the place for details but instead merely a place to show how easy it can be to go from something to nothing.

Getting the data from voteview.com and estimating a quick model is easy enough with the *pscl* package, which has specific functionality to create a rollcall class object ready for analysis, and the function to do analysis based on MCMC. I'll choose the last completed legislature, and stick to defaults for ease of presentation. The default plot is to the right.

```
library(pscl)
rc112 = readKH('ftp://voteview.com/hou112kh.ord',
```



```
               desc='112th U.S. House of Representatives')
idealModel = ideal(rc112)
plot(idealModel)
```

The idealModel object has the estimated values for each member of the House. Now we can create a map that reflects them in geographic space as well. I downloaded the shapefile from the UCLA political science department, but know that there is plenty of mapping functionality already available for state, county and even census tract/block level mapping in R.

```
library(maptools)
congressDist = readShapePoly('PolySci/districtShapes112/districts112.shp')
plot(congressDist)
NA
```

```
## pdf
##   2
```

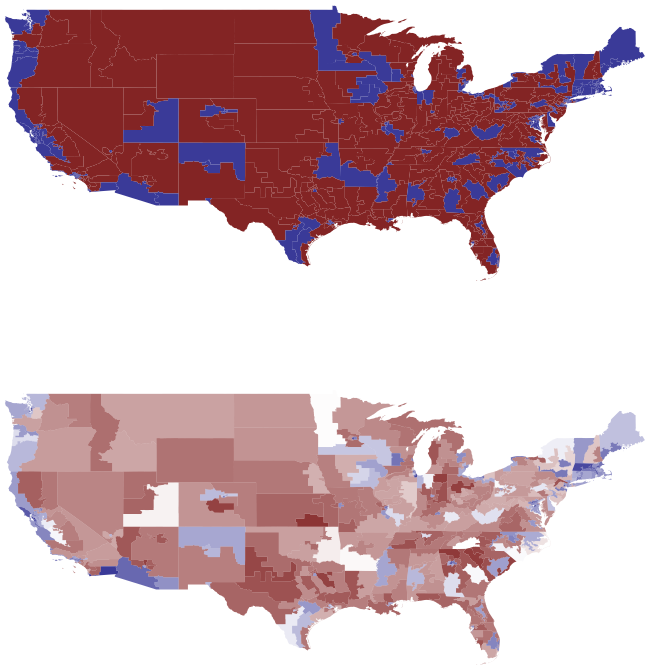
Trying to transfer data values onto maps is rarely a straightforward process. While there are a several issues to contend with, we'll note one particular example I came across. Within both the map object `congressDist` and the ideal object `rc112` I created above resides a data set with district codes. These match for the most part but we need to create a unique identifier so that, e.g. the first district in each state goes to the correct state. This can be easily accomplished by creating a new unique identifier that just combines state and district. Unfortunately the data associated with the map has long form state names, while the roll call data uses abbreviations, so we'll need to find a way around this.

Turns out R has state long form names and abbreviations in its base installation in the objects called `state.abb` and `state.name`, and changing from one form to the other in our own data set then becomes easy. After extracting the rollcall data.frame (`rcdat`), we can more easily play with it directly, then merge it into the map's data set.

The final map I show drops Alaska and Hawaii for easier inspection. Colors fade to white as they get closer to overall mean of the legislators. I also add a comparison of party identification for comparison. One can see, for example, that Democratic representatives (blue) in otherwise Republican states (e.g. in Oklahoma and Arkansas) and regions appear to be drawn more away from their typical party line.

```
# create a new identifier that combines the state with the
rcdat$cd2 = paste0(rcdat$state, rcdat$cd)

# do the same for the map data, simultaneously converting the full name to
# postal abbreviation
congressDist@data$cd2 = paste0(state.abb[match(congressDist@data$STATENAME, state.name)],
                               congressDist@data$DISTRICT)
```

```
# examine and see that they have the same form

##      [,1]  [,2]
## [1,] "GA4" "AL1"
## [2,] "GA5" "AL2"
## [3,] "HI1" "AL3"
## [4,] "IL1" "AL4"
## [5,] "IL2" "AL5"
## [6,] "IL3" "AL6"

# merge with the map data
congressDistwithIdeal = merge(congressDist, rcdat)
```



All told it took about 13 required lines of code[8] to go from nothing to a point where I was ready for final mapping, and that includes dropping out Obama's and replacement votes a couple other minor manipulations to make the data fully compatible. I was able to easily extract points of interest from various objects, manipulate multiple data sets and easily bounce back and forth among them, make changes to them on the fly, import map objects immediately ready for visualization and customization, and take the output from completely independent packages to produce a final product. Also, I have no formal training or background in political science or GIS mapping, though R has allowed me to play with many tools that such disciplines utilize.

[8] Including lines that just load packages, which technically could all be done in one line.

Enhancement

But what if you want a little more from this? Say you want to grab the last 10 Congresses? In R you wouldn't need an explicit loop to do this. The following code would produce a list of 10 rollcall objects, ready for you to examine whichever one you wanted to at anytime. The first line creates a vector of appropriate urls, the next feeds them to the readKH function.

```
congress = paste0('ftp://voteview.com/hou', 103:112, 'kh.ord')
cong103_112 = sapply(congress, readKH, simplify=F)
```

This takes a few seconds per file, but it's easy to do in parallel with the *parallel* package[9]. If you have your cluster `clus` setup, just change the second line to the following.

[9] No purchase of a non-crippled version of the software or additional module required.

```
cong103_112 = parSapply(clus, congress, readKH, simplify=F)
```

I created at function that takes this approach to get either House or Senate (or both), automatically generates descriptions, can write out the file results, and does the process in parallel to create either a rollcall class object or a binary matrix where 1 is a yes vote and 0 a no vote. You can examine it here. On my own machine, it would download and create rollcall objects for all legislatures in about twice the time it takes to download just those 10 using the approach above, plus you'd have the other options available. It also wouldn't take much to add the model and maps as far as code goes, but the models do take several minutes for the House.

R for Psychology

Psychological methodologists jumped on the R bandwagon early, and now have a very wide range of tools to work with from social sciences broadly speaking, to psychometrics specifically, and others that would be of notable interest such as multivariate analysis, experimental design, meta-analysis, and even brain imaging.

Example

I'll start with a latent variable model, as many psychologists use such models to study the underlying constructs they attempt to measure, such as for personality and clinical diagnoses. To begin, there are a great many tools in the *psych* package[10] to engage in exploratory factor analysis, even beyond the standard way most approach such

[10] I use this package frequently for nicely displayed summary statistics and other data exploration.

an analysis. Furthermore, the author of the package provides a freely available online text in psychometrics.

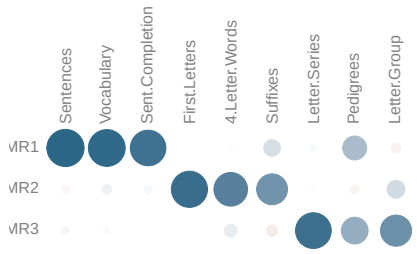
I show an example of some data that has 3 underlying factors. It takes one line to run the model, summarize it or visualize it (latter not shown). To extract factor scores or loadings for further processing takes no effort at all.

```
library(psych)
standardFA = fa(Thurstone, nfactors = 3)
summary(standardFA)

##
## Factor analysis with Call: fa(r = Thurstone, nfactors = 3)
##
## Test of the hypothesis that 3 factors are sufficient.
## The degrees of freedom for the model is 12  and the objective function was  0.01
##
## The root mean square of the residuals (RMSA) is  0.01
## The df corrected root mean square of the residuals is  0.01
##
##  With factor correlations of
##      MR1  MR2  MR3
## MR1 1.00 0.59 0.54
## MR2 0.59 1.00 0.52
## MR3 0.54 0.52 1.00

fa.diagram(standardFA) # visualize
faScores = factor.scores(Thurstone, standardFA) # extract factor scores
faLoadings = loadings(standardFA) # extract the loadings

# modified version using my own function is shown.
## cor.plot(standardFA)
```

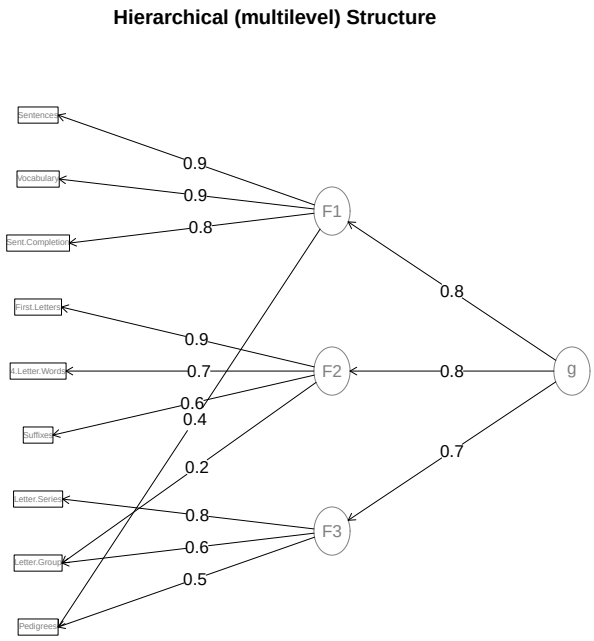
We can see that most items load nicely on 3 correlated factors, but a couple appear to have slight crossloadings on other factors.

Enhancement

Let's say I wanted to simply explore the possibility of a hierarchical factor structure. This isn't something I'm sold on, I don't necessarily want to get carried away analytically, just explore. R allows one to do this sort of thing all the time, because it is easy to extract the pieces of model output you would like for further processing, or simply use other functionality and visualization.




```
omega.diagram(omega(Thurstone))
```



Other tools in this area include the *MplusAutomation* package, which will write out different iterations of an Mplus syntax template, run the files in Mplus, extract the output, and provide the summary of all models, *and you never have to leave the R environment*. However, now you may not even need Mplus. The *lavaan* package provides most of the functionality in Mplus as well as output formatted in the same fashion.

R for Sociology

Sociologists have plenty of ways to benefit from using R. Along with general social science packages, there are packages pertaining to official statistics and survey design, and graphical modeling including social networks. I will provide an example of the latter.

Example

For network analysis there are a great many tools in R such as the packages *sna*, *igraph* and so forth. As a simple example we can begin with a data matrix, and from it create an adjacency matrix in which a 1 (or more) represents a connection for two observations, and finally plot the network graph.

Let's say we have a data set for 10 individuals and 6 individuals. First we take a look.

```
colnames(snData) = paste0('V',1:6)
snData

##            V1  V2  V3  V4 V5  V6
## Yelberton  No  No  No  No No  No
## Billy      No  No  No  No No  No
## Fred       No  No  No  No No  No
## Henrietta  No Yes  No  No No  No
## Janie     Yes  No  No  No No  No
## Bertha     No Yes Yes  No No  No
## Willie     No  No  No  No No  No
## Iggy       No  No Yes  No No  No
## Flozelle   No  No Yes Yes No  No
## Louise     No  No  No Yes No Yes
```

For our purposes, let's say that if anyone has 'Yes' as a value for a particular variable, then, based on that, individuals are connected to anyone else with a 'Yes', and more so the more variables in which this occurs. Now there various distance metrics we could employ for such data that would only take a single line of code to create an adjacency/distance matrix (e.g. binary distance, Hamming's distance etc.). For demonstration though, we'll do it ourselves.

For a moment you might think about how you would go about this in a standard statistics package. I can imagine your thoughts might take you through a double loop to calculate adjacency, figure out where to store or write out the matrix, do all this as a completely separate enterprise, then bring in your matrix to possibly another specialized package entirely that can better handle such things. This is an area traditional statistical packages have only recently begun to even offer much regarding, but regardless there would be a number of steps

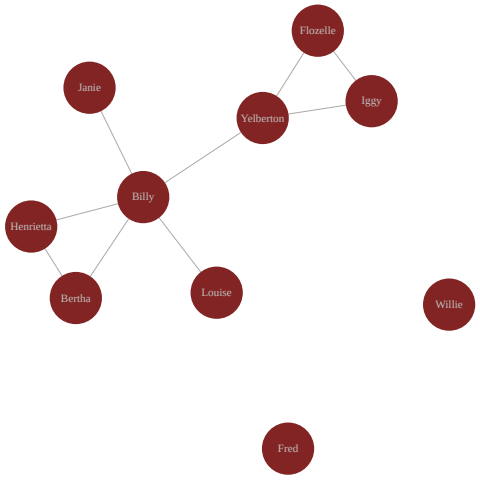
In the following, I create an object containing all possible pairwise connections among the rows 1 through 10. I then make an empty matrix that will eventually become the adjacency matrix. In one line of code I'm able make the all calculations by implicitly creating my own function that will sum all occasions in which both inputs are 'yes', and this is applied to the object with all the pairwise combinations, whose elements serve as a row index for the original matrix. I then fill in the lower triangle of the matrix with those values, and as a shortcut, use the dist function (for distance matrices) to fill in the rest.

```
pairwise = combn(nrow(snData), 2)
adjmat = matrix(NA, 10, 10)
adj = apply(pairwise, 2, function(x) sum(snData[x[1],]=='Yes' & snData[x[2],]=='Yes'))
adjmat[lower.tri(adjmat)] = adj
adjmat = as.matrix(as.dist(adjmat, diag=T, upper=T))
adjmat

##           Yelberton Billy Fred Henrietta Janie Bertha Willie Iggy Flozelle Louise
## Yelberton         0     0    0         0     0      0      0    0        0      0
## Billy             0     0    0         0     0      0      0    0        0      0
## Fred              0     0    0         0     0      0      0    0        0      0
```

```
## Henrietta         0     0     0         0     0      1      0     0         0      0
## Janie             0     0     0         0     0      0      0     0         0      0
## Bertha            0     0     0         1     0      0      0     1         1      0
## Willie            0     0     0         0     0      0      0     0         0      0
## Iggy              0     0     0         0     0      1      0     0         1      0
## Flozelle          0     0     0         0     0      1      0     1         0      1
## Louise            0     0     0         0     0      0      0     0         1      0
```

Now we can plot the graph. The plot shown has some additional arguments added.

```
library(igraph)
g = graph.adjacency(adjmat)
plot(g)
```

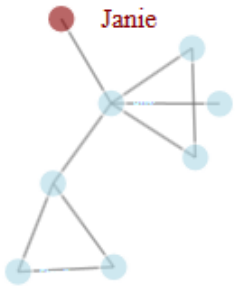


Note that we were able to not only able to do this efficiently in our code, we never had leave the current R environment, use multiple data files or multiple programs, and we have plenty enough functionality to calculate network metrics and model the network itself. We also have code that would be trivial to change to other parameters of determining adjacency.

Enhancement

What if you want more? Maybe you want to put an interactive, force-directed graph of the network on your website. The following code will create the html you need, at which point you can either use it directly as a webpage or embed it in another. The interactive version is here; click to add a name and change the color.

```
library(d3Network)
d3SimpleNetwork(data.frame(get.edgelist(g)), file='network.html')
```



Brief Summary

At this point we've seen R in action in engaged in typical social science data and methodological activity. We've also seen that it can be very easy to take the process further, and generally, that's how it is with R. You get a feel for things, and then realize the potential, take your process further. It allows for far easier data exploration[11], allows one to easily specify and compare the output from alternative models, and use that for visualization.

What might not have been obvious is how easily the different packages work with the objects you create. The packages might be created by different people, but they work on the same classes of R objects, and so many of them work very much in the same fashion even for quite different goals. But even for different classes of objects, different methods for the same function will produce a different product for an object of a different class.

One might also make the claim 'Yeah but, you already know R, I couldn't have done those things!'. And that's where you'd be incorrect. There is so much information about R on the web and elsewhere with working demonstrations and usable code that you merely have to find who else did something, and tweak it for your own needs[12].

In the following I'll show how to do common tasks such as import data, get basic descriptive statistics and visuals, and run traditional models. The point will be to show how easy it is to use R just as you would SAS, SPSS or Stata for common tasks irrespective of discipline or analysis. For most of those things it would take very little transition

[11] Let's say you had a list object containing 1000 simulated data sets and you wanted to run a model on each. This would take one line of code in R. Let's say you wanted to run 1000 separate models on each data set. Assuming a corresponding list of formulas specifying the models, this too would take one line of code.

[12] For the state naming issue, I hadn't done that before, but in looking for an R function that created state abbreviations, I came acros a stack overflow example.

time if you've done any programming in another language.

Working with Data

It is no more difficult to get data into R than other programs. The following offer different ways to read in data in standard formats and from common statistical packages.

```
myCsvDatainR = read.csv('mydata.csv')
myTabDelimDatainR = read.table('mydata.dat')

# for statistical packages
library(foreign)

mySPSSDatainR = read.spss('mydata.sav')
myStataDatainR = read.dta('mydata.dta')
```

Probably one of the most powerful advantages R has over other statistical packages is its indexing capabilities. To provide an arbitrary but extreme example, I will subset a data set as follows: if the fitted value from a regression (R object `modlm`) is above 10 and, following that, if Sex is male or their Mood, which is in another data set (mydata2), is 'blue', but only the variables whose name begins with 'Alf' or ends with 'Webster', and we only want to keep the last 10 rows, and finally, we want summary statistics for those that are retained. Note that none of these particular variables are already available except those in the data sets. Think of the ways in which you might accomplish this in your statistical package of choice. Mostly it would involve steps to create the fitted values, create the conditional variables, create a variable list, run the initial conditionals, subset to the last 10 rows, and finally run a separate command to summarize the output. In the following, we're going to do all of this on the fly.

```
subsetData = summary(tail(mydata[fitted(modlm)>10 && mydata$Sex=='male' | mydata2$Mood=='blue',
                                 grep('^Alf|Webster$', colnames(mydata))], 10))
```

One line of code. This would be a bad way to code and not recommended due to clarity, but it serves to illustrate a point, namely *that you can*. Furthermore, once you get used to these capabilities you end up taking advantage of them in ways you just simply wouldn't have thought to do in other statistical packages because it either wouldn't have been possible or would have been overly difficult. Once you start doing this, data manipulation and exploration, part and parcel of *initial data analysis*, might actually even be fun. While you might spend weeks cleaning data using traditional stat packages, you'll already have made new discoveries from venturing down paths you otherwise wouldn't have bothered with due to time constraints.

A student or faculty member that only occasionally uses a statistical package from time to time when they have some data they want to analyze in all likelihood doesn't do enough programming to be overly wed to a statistical package, and this is because they likely don't do a lot of advanced programming anyway. That said, why not instead use one that will save you time in the long run even if you don't use all of its power?

RData

Another efficiency feature R has over other statistical packages is the RData file itself. When you save your workspace, you aren't just saving a data set, you are able to save *every object you've created*. When you load the RData file, *you don't even have to rerun any code*. All objects are available, including graphs if saved as objects. Even if one could program as flexibly in another statistical package, in those you'd always have to reimport data and rerun code to get back to the point you were. R allows you to pick up at the exact point you left off.

Models

Let's start by examining some non-R approaches to the basic modeling case of standard regression.

SPSS

```
regression
 /dependent science
 /method = enter math female socst read.
```

SAS

```
proc reg;
model science = math female socst read;
run;
```

Stata

```
reg science math female socst read
```

R

```
lm(science ~ math + female + socst + read)
```

This is as basic as modeling gets, and maybe it's just me, but as far as code goes, even in this simple setting I think most would prefer the latter two for their legibility and efficiency[13]. The following are some standard models in the social sciences in R code.

[13] Though unless you know Stata, you wouldn't necessarily know what model is being run.

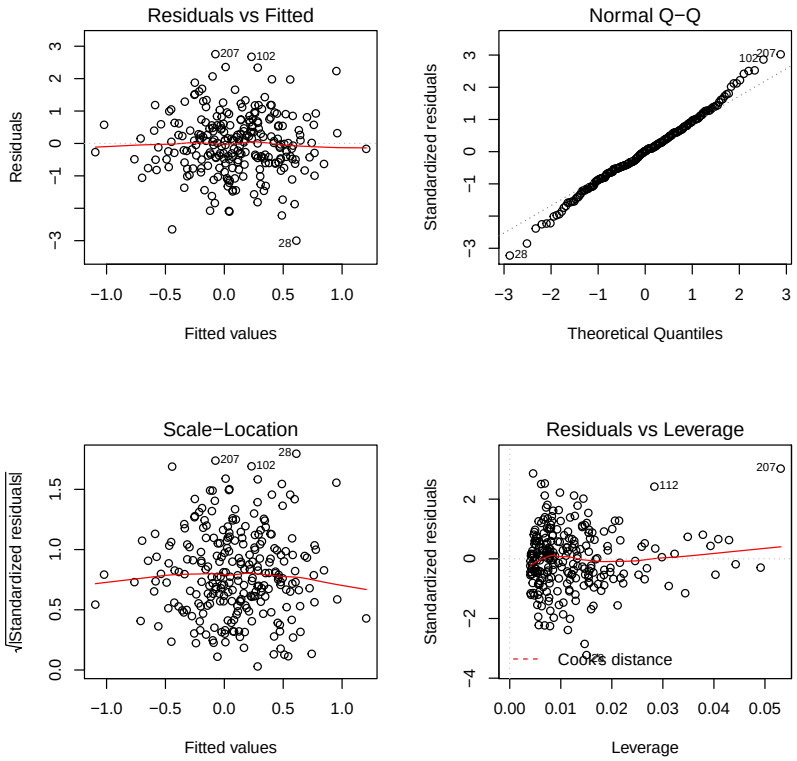
```
# standard linear model regression
lm(y ~ x1 + x2)

# generalized linear model (poisson)
glm(y ~ x1 + x2, family='poisson')

# mixed model, random intercept (lme4 package)
lmer(y ~ x1 + x2 + (1|group))

# generlized mixed model (with a random slope added)
glmer(y ~ x1 + x2 + (1 + x1|group), family='poisson')

# survival model
coxph(Surv(time, status) ~ x1 + x2)

# principal components
princomp(mydata)
```

Hopefully the general theme is clear. After data manipulation running a typical model in R is just as or even more easy to do in R, typically one needs to just specify a formula object (i.e. y~x) and note the data (if the variables are not otherwise in the environment). But let's look at what we might do with a standard regression. The following runs a simple model and shows the results.

```
modlm = lm(y ~ x1 + x2, data=mydata)
summary(modlm)

##
## Call:
## lm(formula = y ~ x1 + x2, data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9985 -0.5104  0.0139  0.5602  2.7546
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1000     0.0592    1.69  0.09261 .
## x1            0.2000     0.0593    3.37  0.00087 ***
## x2            0.3000     0.0593    5.05  8.4e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.937 on 247 degrees of freedom
## Multiple R-squared:  0.13,Adjusted R-squared:  0.123
## F-statistic: 18.5 on 2 and 247 DF,  p-value: 3.39e-08
```

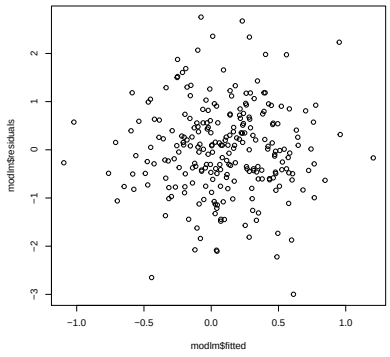
But it's not enough to merely get results. Every model must be examined to see how well it fits the data, how tenable assumptions are etc. It also aids interpretation to see predictive plots and so forth. A set of diagnostic plots is readily available in for many standard models by simply using the `plot` function on the lm object.

```
plot(modlm, ask=F)
```



Or do it yourself.

```
plot(modlm$fitted, modlm$residuals)
```



Note that in the first place we have several diagnostics ready to go for all lm, glm, and additive models, as well as many others in various packages. R developers know you need these sorts of things and so make it easy to get, but even then R objects make it easy to extract them for your own purposes. The following shows everything that can be extracted from the model object such as coefficients, model matrix, residuals etc. You can extract other things from the summary object

(not shown).

```
str(modlm)
## str(summary(modlm))

## List of 12
##  $ coefficients : Named num [1:3] 0.1 0.2 0.3
##   ..- attr(*, "names")= chr [1:3] "(Intercept)" "x1" "x2"
##  $ residuals    : Named num [1:250] 0.964 0.086 0.344 0.395 0.818 ...
##   ..- attr(*, "names")= chr [1:250] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:250] -1.581 3.156 4.734 0.305 0.739 ...
##   ..- attr(*, "names")= chr [1:250] "(Intercept)" "x1" "x2" "" ...
##  $ rank         : int 3
##  $ fitted.values: Named num [1:250] 0.2563 -0.2559 0.2314 -0.596 -0.0228 ...
##   ..- attr(*, "names")= chr [1:250] "1" "2" "3" "4" ...
##  $ assign       : int [1:3] 0 1 2
##  $ qr           :List of 5
##   ..$ qr   : num [1:250, 1:3] -15.8114 0.0632 0.0632 0.0632 0.0632 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:250] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:3] "(Intercept)" "x1" "x2"
##   .. ..- attr(*, "assign")= int [1:3] 0 1 2
##   ..$ qraux: num [1:3] 1.06 1.01 1.02
##   ..$ pivot: int [1:3] 1 2 3
##   ..$ tol  : num 1e-07
##   ..$ rank : int 3
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 247
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = y ~ x1 + x2, data = mydata)
##  $ terms        :Classes 'terms', 'formula' length 3 y ~ x1 + x2
##   .. ..- attr(*, "variables")= language list(y, x1, x2)
##   .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:3] "y" "x1" "x2"
##   .. .. .. ..$ : chr [1:2] "x1" "x2"
##   .. ..- attr(*, "term.labels")= chr [1:2] "x1" "x2"
##   .. ..- attr(*, "order")= int [1:2] 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(y, x1, x2)
##   .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:3] "y" "x1" "x2"
##  $ model        :'data.frame': 250 obs. of  3 variables:
##   ..$ y : num [1:250] 1.22 -0.17 0.576 -0.201 0.795 ...
##   ..$ x1: num [1:250] -0.495 -0.253 0.931 -0.886 1.231 ...
##   ..$ x2: num [1:250] 0.851 -1.018 -0.183 -1.729 -1.23 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula' length 3 y ~ x1 + x2
##   .. .. ..- attr(*, "variables")= language list(y, x1, x2)
##   .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:3] "y" "x1" "x2"
##   .. .. .. .. ..$ : chr [1:2] "x1" "x2"
##   .. .. ..- attr(*, "term.labels")= chr [1:2] "x1" "x2"
##   .. .. ..- attr(*, "order")= int [1:2] 1 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(y, x1, x2)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:3] "y" "x1" "x2"
##  - attr(*, "class")= chr "lm"
```

Prediction is straightforward, and works the same in many modeling packages. Again, R developers know this is part and parcel of statistical analysis, and it is made easy for standard models, and most modeling packages take the same approach. In the following we merely have to tell it what values of the explanatory variables you want to use[14]. Here we look at x1 values from -1 to 1, keeping x2 at its 90th percentile. Note that we aren't explicitly creating new x1 or x2 variables, though it would be better form to create a separate 'newdata' object

[14] If you didn't you'd just be duplicating the fitted values.

```
predict(modlm, data.frame(x1=seq(-1,1, by=.25), x2=quantile(mydata$x2, p=.9)))

##      1      2      3      4      5      6      7      8      9
## 0.2901 0.3401 0.3901 0.4401 0.4901 0.5401 0.5901 0.6401 0.6901
```

Here are some examples of other typical modeling exercises.

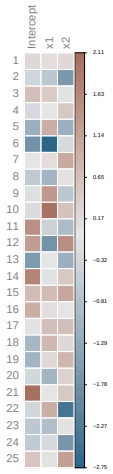
```
# interactions (including main effects)
lm(y ~ x1*x2)

# dealing with categorical factors; nothing special needed
lm(y ~ f1 + x2)

# transform a variable on the fly; no need to explicitly create the variables if
# you don't want to. The last one creates a logical variable that's true if x is
# greater than 5.
lm(log(y) ~ sqrt(x1) + x2>5)

# add a predictor
update(modlm, ~. + x3)

# model comparsion
anova(mod1, mod2)

# automatic model comparison of added or dropped terms
drop1(modlm, ~ x2, test='Chisq')
add1(modlm, ~ x1*x2, test='Chisq')

# y predicted by all other variables
lm(y ~ .)

# write out results to latex (xtable package)
xtable(summary(modlm))
```

Let's say as a precursor to a mixed model, you want to run separate regressions on the levels of the grouping factor within which observations are nested. Furthermore, you want to make a graph that reflects the coefficients values, fading them out and make them smaller as they tend toward zero. First think about the steps you'd have to take in your own package, then note that this technically can be a one line operation in R[15]. The plot is shown to the right with additional options specified, and further down to it I show a similar plot, which looks at model for two dependent variables and 46 countries.

```
library(lme4); library(corrplot)
corrplot(scale(coef(lmList(y ~ x1 + x2 | g, mydata))), is.corr=F)
```

It's assumed if folks could do this sort of thing easily in other statistical packages they would be doing so. R users by contrast are doing things like this all the time as standard practice, and R allows them to take their models further much more easily. Now let's look at some other advanced models.

```
# multiple regressions, same predictors
lm(cbind(y1, y2, y3) ~ x1 + x2)

# generalized additive model (mgcv package)
gam(y ~ s(x1) + s(x2))

# mixtures of regessions (flexmix package)
flexmix(y ~ x1 + x2, k=3)
```



[15] For visualization purposes only, I scale the matrix of coefficients so that they can be on equal visual footing. The purpose here would never be for inference.

```
# bayesian regression (bayesglm package)
bayesglm(y ~ x1 + x2, family='poisson')

# optimization with likelihood function
lmLL = function(par){
  beta = par[-1]
  sigma = par[1]
  mu = X%*%beta
  -sum(dnorm(y, mu, sigma, log=T))
}
optim(par=startingvalues, fun=lmLL)

# classical time series
arima(y, order=c(0, 1, 1))
```

You get the picture. Modeling is as easy in R as it would be in other packages, if not easier in many cases. In addition, you get access to more models and more flavors of the models[16], not to mention various approaches that are at the cutting edge of statistical science. Furthermore, you can simply do more with your model results once you have them. Users of other statistical packages are stuck with older approaches, commands that won't work if you try to use them in any nonstandard fashion, and modeling functions for which one has no way to see what the commands are actually doing.

[16] As an example, the `ols` in the *Hmisc* package would do even more for your standard regression (and logistic,s survival and other) models.

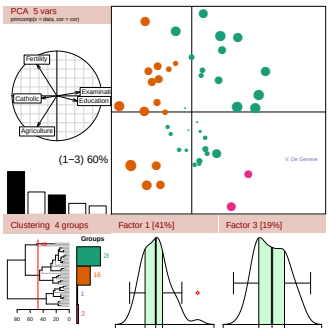
Visualization

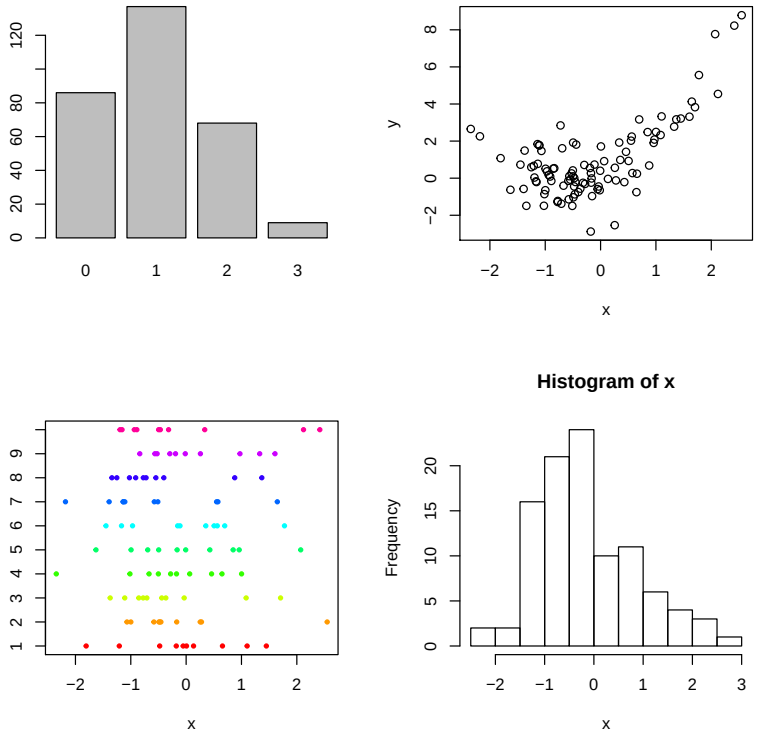
I'm not going to talk a lot about visualization in R, it's been ahead of other statistics packages in that department at least since version 1.0[17]. Admittedly traditional statistics packages have finally started to make gains in this area in recent years, but even as they do R continues to move the goalposts.

I will show how easy it is to get standard plots, but it's more or less a waste of R's capabilities if this is all you want to use it for.

```
barplot(table(d))
plot(x, y)
stripchart(x ~ g, col=rainbow(10), pch=20)
hist(x)
```

[17] Look at R's homepage, this plot is over 10 years old, and would still be impossible for most statistics packages (code available if you click on the image at the website).

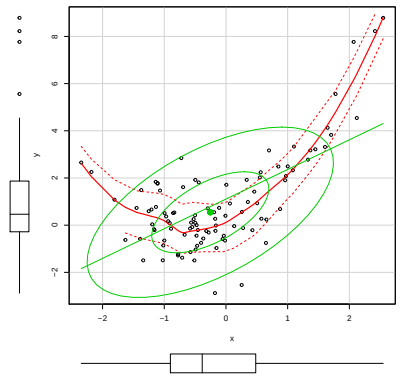




For one, these convey too little information. You don't need a visualization for something you can describe in a short sentence. *But even if this is all you're doing visually*, you'd still be better off in R because you would have fine control over every aspect of the plot and be able to tweak anything about it that you'd want to.

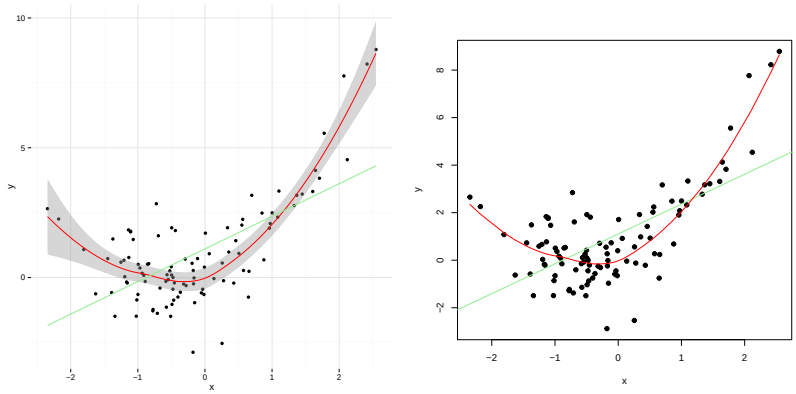
But let's stick with standard plots and show how easy it is to get something better. As an example I'll show a scatterplot using the car package.

```
library(car)
scatterplot(x, y, ellipse=T)
```



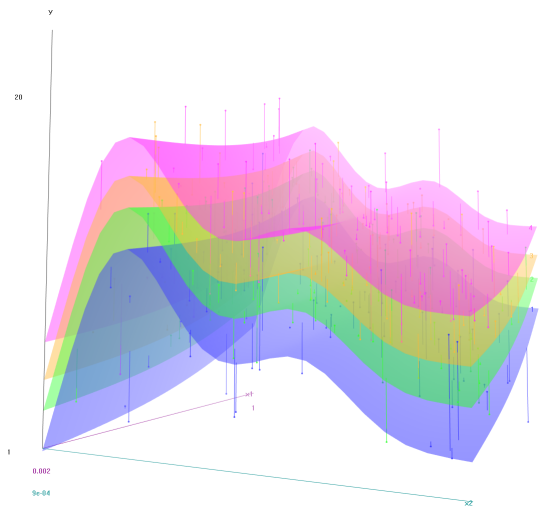
With no additional coding effort you have an enhanced scatterplot, a smooth and linear function imposed on it, a data concentration ellipse based on a bivariate normal distribution, univariate boxplots in the margins, and the ability to identify points of interest interactively. It actually would take little effort to reproduce most of the scatterplot with the ggplot2 package or even the base R plotting capabilities.

```
library(ggplot2)
ggplot(aes(x, y), data=gd) +
  geom_point() +
  geom_smooth(col='red', se=T) +
  geom_smooth(col='lightgreen', method='lm', se=F) +
  theme_minimal()

plot(x, y, pch=19)
abline(lm(y~x), col='lightgreen')
lines(data.frame(x, predict(loess(y~x)))[order(x),], col='red')
```



Maybe you want a 3d scatterplot with a with smooths for different groups? Again this is one line of code.

```
scatter3d(y ~ x1 + x2 | x0, data=d, fit='additive')
```

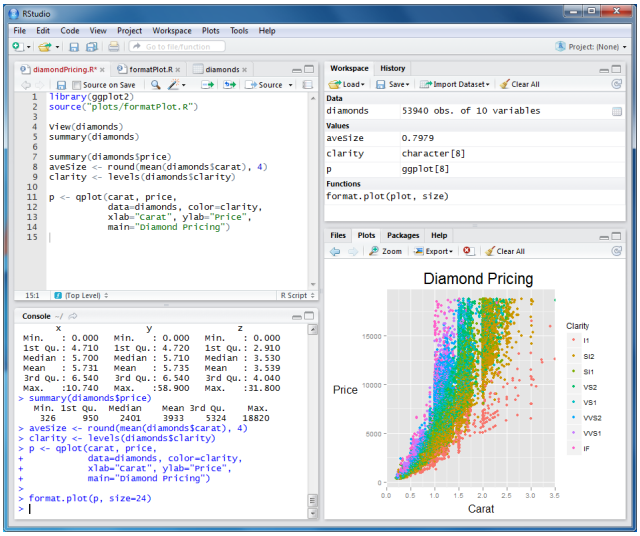


I provide the above examples not to show that R can do this or that visualization while statistical packages can't. The point is not that other statistical packages can't do the same kinds of visualizations you want, although that is sometimes, if not often, going to be the case. It's just going to typically be easier, often much easier, in R to get far better versions of standard plots, so that you'll be spending most of your time tweaking the visualization you actually wanted to get it just the way you want it, rather than fighting with a program that hasn't put much relative effort into visualization and in the end eventually settling on whatever it gives you.

For a comparison I looked up "'statistics package' visualization" and the results are as follows. SAS, has an unnecessarily shaded bar plot as the first thing it shows regarding its visualization module. SPSS as usual wants you to pay for this capability, and its screenshots suggest it's way too much. I couldn't find much on the topic with Stata that actually showed much beyond the standard graphics, and in any case it has never been known for its prowess there. Honestly, you'd be better of using other visualization-specific tools freely available on the web even if you were wed to SAS, SPSS, Stata etc. But as a comparison, here are the sorts of things R visualization entails- shiny, ggvis. Modern visualization must be interactive and web ready to go along with producing high quality static graphics, and R has tools ready for you to go there.

Programming in R

Something that would surprise those coming from traditional statistics packages is the fact that it is actually much easier to code in R. It's already the case that data slicing, dicing, wrangling, cleaning etc. is easier in R because of the flexibility it gets from being a true, object-oriented programming language. However even the act of coding itself is made easier with RStudio, the integrated development environment for R. I'll list some features and you can ask yourself if you'd rather program in RStudio or the syntax editor of your statistical package.



Basic Benefits

With RStudio you get various base functions, strings, logicals and so forth highlighted as is common with many syntax editors, which makes reading code notably easier. In addition, RStudio has a searchable history, auto-completes object names, closes all manner of brackets, and closes quotation marks, auto-indents etc. making it far less likely to have, e.g. a left off parenthesis, to break your code. The auto-indent helps keep loops and so forth tidy and more legible automatically. And finally, it's nice to know you'll never lose anything, as RStudio auto-saves every few seconds. You can open 10 scripts, not save any of them, close RStudio, open it back up, and it'd all be just as you left it. Many other core features can be found at RStudio's website.

Code Maneuverability

Let's say you are staring at 100 lines of code on one of several R scripts you are working on, and your cursor is currently at line 50 and you want to do the following steps: run everything to that line, pop to the console for a quick one-off graph, go back to where you were in the script, open the help file for the function you're currently using, look at the source code for it, run the rest of the code from that line on, switch to the script 3 tabs down, switch back, rerun, what you the last bit of code you ran, switch to the next tab (which is a markdown script), check the spelling and compile the script to pdf or html. To do this in RStudio would take about 20 keystrokes and no mouse movement at all[18]. This is actually a common sort of capability in IDEs, and you'll

[18] You can see all the shortcuts in RStudio here

find similar functionality in others that are geared toward other programming languages (just not statistical packages). And for those that don't care much for programming, they should want to spend less time doing it, but my guess is that the easier it is for someone to program the more likely they will start to enjoy it.

Project Management

RStudio allows one to create projects, which are like self-contained collections of scripts housed in their own directory. Say you're a faculty member and you're working on a book, and have two current research projects. You could be writing the book in one project that has, for example, separate scripts for each chapter. In another are the analysis scripts pertaining to that research, and the same for your other research endeavor. You're working on your book when a colleague calls up and asks you to rerun part of an analysis for her. You simply switch to that project, and RStudio brings up all the scripts you had when you were working on it. When done you just go back to the book project and it will be as though you never left[19]. Furthermore, these projects are capable of being tied to Git or Subversion for version control, making it more easy to collaborate on these projects with colleagues. You can also have RStudio settings specific to each project, and have package management specific to each project.

[19] I'll point out you wouldn't even need the mouse to do this.

Support for Literate Programming & Reproducible Research

The time has come for researchers to get more serious about their data, how they produce the results of their research, and conducting analysis in a fashion that is reproducible. Research results in publication form are almost entirely thought of as a finished product among social scientists, but those days are over. Better work results from assuming the final product will be dynamic, easily updated, and make it as easy as possible to reproduce the results.One never has to leave the R environment to go from data to journal ready publication, while engaging in best practices for reproducible research. R has long been in this game, as tools like Sweave have been around since the days of S.

I found it telling that searching for "reproducible research" and "SPSS" has a first link at IBM to calling R from SPSS.

Document Production

RStudio makes it easy to use R to take your code and analysis and produce high-quality, publish-ready documents, and in a variety of ways. As mentioned, RStudio takes LaTeXand with various R tools like the *knitr* package, and compiles the text, code and results into a final pdf. If I were producing a an article based on statistical results, it would

be trivial to update the models and thus update results. The following code uses the *stargazer* package write model results out to a file in LaTeX, using the style of the American Journal of Political Science, which can then be imported directly into the document file.

```
mod = lm(y ~ x1 + x2)
stargazer::stargazer(mod, out='metaLaTeX.Rnw', style='ajps')
```

	y
x1	0.867***
	(0.105)
x2	1.024***
	(0.099)
Constant	0.135
	(0.096)
N	100
R-squared	0.633
Adj. R-squared	0.625
Residual Std. Error	0.951 (df = 97)
F Statistic	83.610*** (df = 2; 97)

***p < .01; **p < .05; *p < .1

In this way we can always update the document based on new data, discovered bugs, or maybe try enhanced analysis. The document doesn't have to stay frozen in time with all its possible foibles. This is something that wasn't possible when print outlets were the primary way to dispense scientific information. Now the web and digital presentation are not only desired they are assumed. Now we can think of research articles having version numbers just like software, with relevant updates and enhanced 'features'.

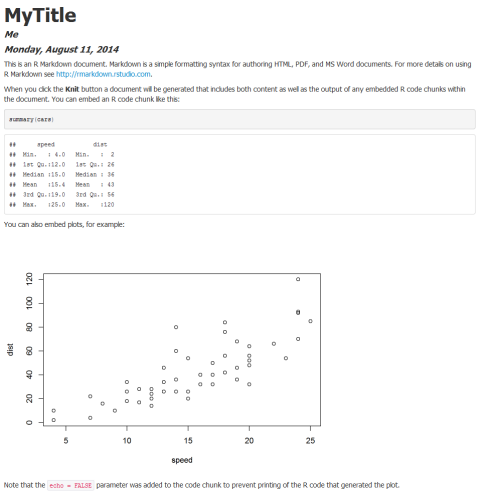
Web-based Presentation

It might actually be even easier with RStudio to produce a web-ready document. R has its own flavor of Markdown, text-to-HTML conversion tool widely used to produce content for the web.

```
---
title: "MyTitle"
author: "Me"
date: "Monday, August 11, 2014"
output: html_document
---

This is an R Markdown document. Markdown is a simple formatting syntax for authoring
HTML, PDF, and MS Word documents. For more details on using R Markdown
see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both
content as well as the output of any embedded R code chunks within the document.
You can embed an R code chunk like this:
```

````
```{r}
summary(cars)
```

You can also embed plots, for example:

```{r, echo=FALSE}
plot(cars)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent
printing of the R code that generated the plot.
````

That code will produce the following webpage.



In addition, R can make html5 presentations. I have an example here that I use for my R short course. So now you can do you research presentations with code and results included as well, and again you've never had to leave the RStudio. I guess I should mention you can use markdown for MS Word documents, but I haven't yet figured out why one would want to.

Version Control

Along with all of this, RStudio projects incorporate version control, enabling collaborative research where you don't have to worry about the data, and can easily keep track of changes to data and code, revert if necessary etc. If you're wondering why you'd need version control, I'll let the following list from stackoverflow.

> Have you ever:
>
> - Made a change to code, realized it was a mistake and wanted to revert back?
> - Lost code or had a backup that was too old?

- Had to maintain multiple versions of a product?
- Wanted to see the difference between two (or more) versions of your code?
- Wanted to prove that a particular change broke or fixed a piece of code?
- Wanted to review the history of some code?
- Wanted to submit a change to someone else's code?
- Wanted to share your code, or let other people work on your code?
- Wanted to see how much work is being done, and where, when and by whom?
- Wanted to experiment with a new feature without interfering with working code?

In these cases, and no doubt others, a version control system should make your life easier.

I'll add that most of those apply to both data and documents as well, and version control works for those just like it does for code. It's indispensable for collaborations. You and co-authors can work on documents, data, and analysis without worry, and always know what you're working with.

Functions & Debugging

R makes even makes it easy to write your own functions, which can make many operations easier and contributes to reproducibility as well. But you don't have be an advanced R programmer. You could simply change the defaults for some function you like to use, for example. However, in the following we'll demonstrate writing an original function, one that takes 3 inputs (arbitrarily named 'a' 'b' and 'c'), and takes the sum of the first two and multiplies it by the third input.

```
myfunc = function(a, b, c){
  result = (a + b) * c
  result
}

myfunc(1, 2, 3)

## [1] 9
```

Many R functions are quite complex, but yours just has to do what you need to. You can also do so within other functions implicitly if it's very simple. The following uses the `sapply`, which applies some function to all elements of a vector or list, and in this case we'll make that same calculation to the list elements[20].

[20] Some of you may be wondering where the loop is. In R, it is typically easier, and often clearer, to not use explicit loops.

```
mylist = list(c(1,2,3), c(3,2,1), c(1,1,1))

sapply(mylist, function(listelem) (listelem[1] + listelem[2]) * listelem[3])

## [1] 9 5 2
```

In this case the implicit function takes a an element from the list (now arbitrarily named listelem), each of which contains three values, and does the same operation as before. Unlike other statistical packages where you practically have to be an expert to create and program your own commands to use, or it's simply an otherwise discouraging process, it's pretty easy to do in R, and you'll save a lot of programming time the more you do it.

My own rule of thumb is that if I'm using a chunk of similar code more than twice, it's time to write a function that does what the code does.

Furthermore, R and RStudio have a debugging system that makes it easy to spot problems, find time bottlenecks, cycle through all iterations of loops etc. So for those that get more involved with coding or want to write a complex function, it will likely be a lot easier to do in R.

Summary

Before wrapping things up, let's talk about some of the other things R can do. These are just some of things that should/would be of interest to any applied social scientist, but are not the sorts of things typically covered in a social science stats course, nor are they easily done or common in statistics packages they normally use. With R you have a lot of functionality for things like web scraping, machine learning, text analysis, Bayesian estimation, mapping and geospatial modeling, direct access to data repositories, accessing social media streams, the ability to interface with newspaper and journal APIs, and a whole host of tools from other sciences that might be of use for their particular research. R's audience is very general, and its users build tools to meet their needs and make them available for others. You will typically have a lot of choice in how you do any particular thing in R modeling or otherwise, unlike other packages for which you may have only one or two commands that don't provide much different content. It may be overwhelming at first introduction, but it is very freeing. You'll soon that the question isn't 'Can R do this?', it's 'What package in R does this?'.

Hopefully this document will give the SAS, SPSS, Stata etc. users in social science some things to think about. Those are fine programs in their own right, but unless you are a great programmer with their syntax, using them is less efficient than what you can do with only a basic knowledge of R. A lot of folks think there is a steep learning

curve just to get into R, but for purely applied use of just importing data and running a model, it can be used in pretty much the same way as one does the other statistical packages, so you can pop over to it to do the things your own package doesn't do the way you would like. But even with purely applied use, there are efficiency gains due to its object-oriented programming features, RData files, R Projects, visualization etc., and the more you get used to it the more you'll get out of your data.

Modern science is dynamic, interactive, web-based, and reproducible. R has all the tools you need for modern scientific research in any discipline. Spend some time at R-bloggers to see what people are doing with R. It will make your head spin. But know that many of them aren't expert programmers or statisticians either, and if they can use R to do those things, you can too.