The Political Methodologist

Newsletter of the Political Methodology Section
American Political Science Association
Volume 21, Number 1, Winter 2014

Editor:
Justin Esarey, Rice University
justin@justinesarey.com

Associate Editors:
Royce Carroll, Rice University
rcarroll@rice.edu

Randolph T. Stevenson, Rice University
stevenso@rice.edu

Rick K. Wilson, Rice University
rkw@rice.edu

Editorial Assistant:
Ahra Wu, Rice University
ahra.wu@rice.edu

Contents

Notes From the Editors

This is the first issue of *The Political Methodologist* produced under a editorial team based at Rice University. We extend our congratulations and thanks to Jake Bowers, Wendy K. Tam Cho, Brian Gaines, and the Department of Political Science at the University of Illinois at Urbana-Champaign for their years of successful stewardship of TPM, and for their support during this transition period.

While TPM is still produced in the same biannual printed format that it has used for years, we now also present articles in an on-line blog format at http://thepoliticalmethodologist.com. Our blog allows for more immediate publication, and for conversations between our authors and readers in the way of moderated comment threads. Between the time that TPM went on-line in September 2013 and the time of this publication (late January 2014), the articles on http://thepoliticalmethodologist.com have been viewed over 10,000 times. Furthermore, 44 comments have been posted across all these articles.

Our inaugural print edition of TPM contains four articles that were originally posted on-line and attracted especially large interest. We hope that you will enjoy them as much as our blog readers did, and that you will sign up for e-mail, RSS, Facebook, or Twitter alerts for new postings on http://thepoliticalmethodologist.com to stay abreast of the latest research published in *The Political Methodologist*.

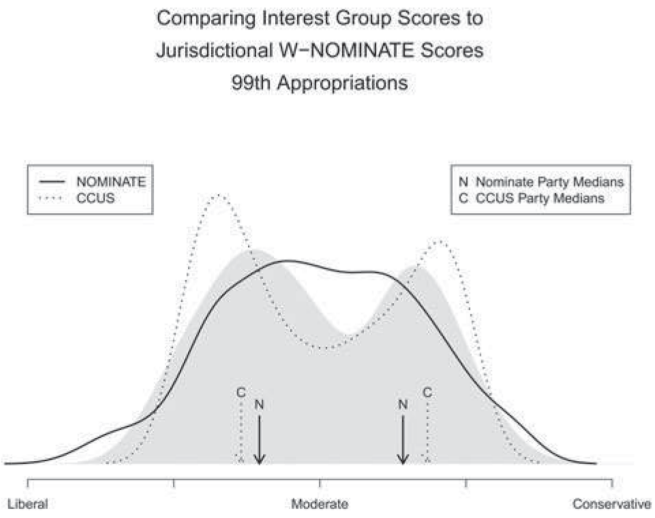
The Editors

Articles

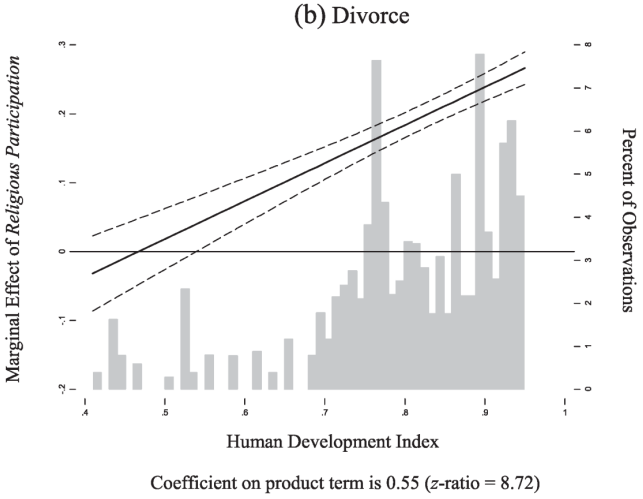
Making High-Resolution Graphics for Academic Publishing

Thomas J. Leeper
Aarhus University
thosjleeper@gmail.com

Political scientists increasingly use data visualizations to communicate statistical results. These visualizations make those results clear and intuitive, and perhaps beautiful. Yet the quality of the data story is sometimes lost in pixelated, low-resolution images. While the visualization itself might be strong, the physical rendering of that visualization is less than ideal. Though we are trained to make statistical graphics, the instructions we receive from publishers about how to make those graphics look their best in published form can be vague or confusing. This post discusses the formatting requirements for image files in journal publishing, points out some examples of high- and low-resolution graphics in a recent journal issue, and describes how to produce high-resolution image files from R, Stata, and Excel (possibly with the help of one or more simple add-on utilities).



Example of a Low-Resolution Image



Example of a High-Resolution Image

File Formats and Resolution

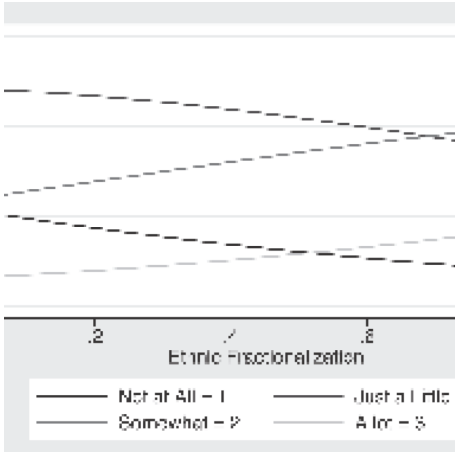
Journal publishers have reached an industry consensus on how image files should be formatted for academic journals. As a few examples, Cambridge, Elsevier, Oxford, SAGE, Taylor & Francis, and Wiley agree that all image files should be in one of three formats: TIFF (Tagged Image File Format), EPS (Encapsulated PostScript), or PDF (Portable Document Format). All other formats are discouraged or not allowed, due to their lossy image compression.

Two of the recommended file formats (EPS and PDF) are vector graphics formats, which store an image as a series of lines of specified shape, color, and size between relative points in a space. Thus the image is rendered by drawing each of those vectors from the instructions encoded in the file. Vector graphics look sharp at any size because the vectors are simply drawn larger or smaller, depending on the requested size of the image. This differs from raster graphics, like TIFF, BMP, JPEG, and GIF. Unlike vector graphics, raster graphics are stored as a matrix of colored pixels.

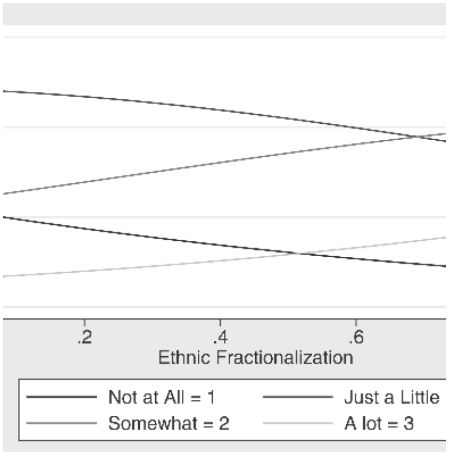
Due to the scalability of vector graphics, they are the logical choice for producing high resolution graphics in academic publishing, and R, Stata, and Excel all produce these formats. When a vector graphic format is not available or for some reason not accepted, graphics need to be saved as high-resolution TIFF files of an appropriate size in order to avoid the grainy or pixelated images sometimes found in journals.

Academic publishers have a clear and shared standard for raster image resolution: a minimum 600 dots per inch (DPI) for most images, and 300 DPI for "halftone" figures, such as multicolor photographs (rare in political science

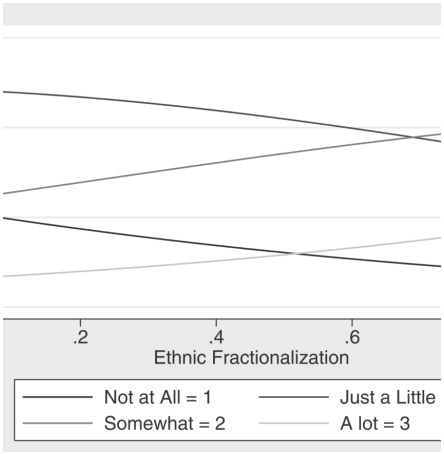
journals). Many computer applications output image files at a default DPI far below this, often 72-96 DPI, which is the standard resolution for internet graphics. Thus, the default file settings for saving to TIFF are almost never appropriate for academic publishing.



96 DPI resolution



200 DPI resolution



600 DPI resolution

Exporting Graphics from R

R offers drivers for all three of the required file formats in the `pdf`, `postscript`, and `tiff` functions. In RGui, one can also save images to any of these formats (or other formats) using the `File > Save as` menu on the graphics device window.

For LaTeX users, the PDF format is probably the most familiar due to its easy incorporate into `.tex` files. Indeed outputting an R plot to PDF is simple. For all of the below examples, we'll consider the simple case of plotting a scatterplot. To produce a PDF, one simply needs to precede any plotting function(s) by a call to `pdf`, with an appropriate filename specified:

```
pdf("image.pdf")
plot(x, y)
dev.off()
```

Because PDF is a vector format, one need not specify `height` and `width` arguments in order for the image to have desirable resolution. That said, the plot as shown in the R console will not map directly onto the plot saved to file. Specifying a larger image size (with `height` and/or `width`) will yield a PDF with smaller text and more space between points. These arguments can also be used to control the aspect ratio (i.e., to produce a wider or taller rectangular plotting region).

The other approved vector format is EPS, which we can produce by replicating the above commands almost verbatim, replacing `pdf` with `postscript`. The R documentation recommends, however, that you first call `setEPS()` in order to instruct the `postscript` device to render a single-page image (i.e., `postscript` can produce multi-page files and using `setEPS()` restricts the result to a single page). We can see this code at work below:

```
setEPS()
postscript("image.eps")
plot(x, y)
dev.off()
```

Another strategy is to use the Cairo device (`cairo_ps`) without the call to `setEPS()`:

```
cairo_ps("image.eps")
plot(x, y)
dev.off()
```

Again, because EPS is a vector format, one need not specify a resolution, but both `postscript` and `cairo_ps` support `height` and `width` arguments for controlling aspect ratio.

Producing TIFF images with the TIFF device (`tiff`) is similar, but here we must explicitly specify a resolution (with the `res` argument), as well as `height` and `width` in order to yield an appropriately high resolution image at the intended output size (and for `height` and `width` to be interpretable, one should also specify `units`). To save space, these files can be compressed using lossless compression (i.e., with no damage to the resulting image). The preferred compression algorithm is called LZW, which can be specified by the `compression` argument to `tiff`. Here's an example:

```
tiff("image.tif", res=600, compression = "lzw", height=5, width=5,
     units="in")
plot(x, y)
dev.off()
```

An alternative workflow to the above methods is to use `dev.print` to send a currently open plot window to an image file:

```
plot(x, y)
dev.print(pdf, "image.pdf")
dev.print(cairo_ps, "image.eps")
dev.print(tiff, "image.tiff", res=600, height=5, width=5, units="in")
```

This can be especially useful for saving to multiple formats (as above) or for separately saving layers of a multi-layer

plot (e.g., to use in a presentation where parts of the figure are revealed sequentially). This is the same general approach used by the ggplot2 graphics package, which offers a `ggsave` function that is called *after* the plotting call. For TIFF formats, the `dpi` argument should also be specified to produce an appropriately high resolution. Here are some examples:

```
qplot(x,y)
ggsave("image.pdf", height=5, width=5, units='in')
ggsave("image.eps", height=5, width=5, units='in')
ggsave("image.tiff", height=5, width=5, units='in', dpi=600)
```

Exporting Graphics from Stata

Saving graphics interactively in Stata is straightforward. From the graphics window, one can simply press the save icon or select `File > Save` and be presented with a simple menu to save the file. PDF, EPS, and TIFF formats are all available by default. Plots can also be saved from the console using `graph export` after a plotting command:

```
graph twoway scatter x y
graph export image.pdf
graph export image.eps
graph export image.tif
```

Using the menu or icon, the TIFF format prints only at 96 DPI by default. Thus we should turn to the console and use the `width` option in our call to `graph export`. Unfortunately, it is not possible to specify a resolution directly, so we need to do some hackery. You need to calculate the width of the graph (in pixels) necessary to produce the desired resolution at final printed size. In a full-page two-column journal, a one-column width is about 3 inches and two-column width is about 6.5 inches, whereas in a small-format journal, a one-column width is about 4.25 inches. Thus, if we want 600 DPI, we would need pixel widths of 1800, 3900, or 2550, respectively:

```
graph twoway scatter x y
graph export image1.tif, width(1800)
graph export image2.tif, width(3900)
graph export image3.tif, width(2550)
```

The result of each of these is, however, a very large image with 96 DPI resolution (i.e., Stata simply produces a larger image with the same low resolution). Of course, because size and resolution are related, the images are visually equivalent, but you will need to modify the file with another utility to actually see the 600 DPI resolution at the intended output size.

Exporting Graphics from Excel

While versions of Microsoft Excel prior to 2007 allowed users to directly save charts as image files. This is no longer possible. Instead, one needs to use the `File > Save As` menu to output a chart to PDF (the only built-in file format for printing charts). To do this, select the chart (in need not be on its own tab), then follow `File > Save As`. In the pop-up menu, select PDF from the Save as type drop-down menu and specify an appropriate filename. Clicking the `Options...` menu will open another small window that allows one to confirm that only the `Selected chart` will be output to PDF. Attempting to save a chart (as an object in a spreadsheet tab) without first selecting the chart will cause the `Options...` pop-up to display a different set of options, none of which include `Selected chart`. Another method is to move the chart to its own sheet, then under `File > Save As`, select `Options...`, and choose `Active sheet(s)` to save just the chart to a PDF file. If, for some reason, a publisher will not accept a PDF file, one can use any of the options described in the next section for converting the PDF to TIFF.

Image Utilities

With our files exported from R, Stata, Excel, or another software application, we may still need to make changes. For example, because Excel can only output PDF, we may need to convert our image to TIFF; or, because Stata can only output TIFF at 96 DPI, we may want to rescale the image to the appropriate size and resolution. Most publishers recommend using Adobe Photoshop to do this. Unfortunately, Adobe Photoshop is proprietary, expensive, and may not be readily accessible. Luckily several free, open-source, and easy-to-use alternatives exist.

For the most direct analogue to Photoshop, one should try GIMP (GNU Image Manipulation Program). For command line manipulation of images, GhostScript works well for PostScript (EPS) and PDF formats and ImageMagick offers diverse functionality for manipulating almost all image formats in addition to EPS and PDF. All of these programs should work on all modern operating systems.

GIMP: GNU Image Manipulation Program

GIMP allows you to easily adjust the resolution of images as well as save images into other file formats. For example, to convert a Stata graph saved at 96 DPI to 600 DPI, we can open GIMP, choose `File > Open` to select and import the TIFF file. With the TIFF open, we can choose `Image > Print Size...` and a small window will open describing the size and resolution of the file. From that menu, changing the resolution from its default to 600 makes GIMP adjust the image size accordingly.

We can then save the file using `File > Export...`, specifying a filename, and selecting TIFF image from the file format drop-down. (We could also save the file in any other format.) If saving to TIFF, a small pop-up window will offer the option to choose file compression such as LZW. The new file will have the intended dimensions and resolution. GIMP is a fully featured image manipulation program, so can also be helpful for converting color to grayscale, cropping, etc.

Because GIMP can read almost any image format, if we have files in other formats, we can easily open them in GIMP and export them to any of the supported formats (e.g., to convert a TIFF to a PDF or vice versa). These features are straightforward and simply require opening the input file and using `File > Export...` to save in an appropriate output format. Note, however, that GIMP does not – by default – support EPS format. To convert to or from EPS, we need to have the GhostScript command line utility installed first. Then, when opening an EPS or PDF in GIMP, you can specify the size and resolution at which to render the vector image (because GIMP will coerce the vector graphics to a raster).

Command-Line Utilities: GhostScript and ImageMagick

File conversion operations can also be performed on the command line. To convert PDF or EPS to TIFF, one can use GhostScript, a command-line utility for working with PDF and PostScript files. To use GhostScript, it must be installed and its directory must be on the system path. On Windows, you will probably have to manually add GhostScript to the system path after it is installed. You can check your path by opening Command Prompt (run it as an administrator) and typing:

```
echo %PATH%
```

That will output a long string of delimited directories that point to particular applications. The directory for GhostScript should be among them. For a current version of GhostScript on Windows 7, this directory is listed as: `C:/Program Files/gs/gs9.10/bin`. If this (or a similar directory) is not listed in the path, one can easily add it to the Windows path by typing:

```
set PATH=%PATH%;C:\Program Files\gs\gs9.10\bin
```

You can also update the Windows system path by visiting `Control Panel > System`, clicking on `Advanced System Settings`, and pressing the `Environment Variables...` button. This will open a small pop-up window where you can edit the `PATH` variable for your user account and the `Path` variable for all users on the machine. You can simply select the variable, press the `Edit...` button, and paste the path to GhostScript (preceded by a semicolon) at the end of the current path.

With GhostScript on the path, we can reopen Command Prompt (or Terminal, on a UNIX-alike) and navigate to the directory containing our images. Let's say we want to create `image.tif` from `image.pdf`, we can simply type the following:

```
gs -r600 -sDEVICE=tiffg4 -sOutputFile=image.tif image.pdf -dBATCH
```

The result is a high resolution TIFF file called `image.tif` created in our working directory. Here's a breakdown of the command:

- `gs` refers to GhostScript (On Windows, `gs` may instead need to be `gswin64` or `gswin32`, referring to the name of the actual GhostScript application.)
- `-r600` requests a 600 DPI resolution
- `-sDEVICE=tiffg4` says what image device to use (in our case, one for monochrome TIFF files, though if working with color graphics, another device might be appropriate such as `tiff24nc` for 24-bit color or `tiff12nc` for 12-bit color)
- `-dBATCH` cleans up a little bit after everything is done.

Replacing `image.pdf` with `image.eps`, we can do the same conversion from EPS to TIFF:

```
gs -r600 -sDEVICE=tiffg4 -sOutputFile=image.tif image.eps -dBATCH
```

Thus both GIMP and GhostScript can convert between relevant formats. By installing both GhostScript and ImageMagick, you should also be able to do the conversion even more easily. To use ImageMagick, it must also be on the system path (and you can follow the above directions to ensure it is available on the path).

The following code for ImageMagick is equivalent to the above line for GhostScript to convert PDF to TIFF or to convert EPS to TIFF:

```
convert -density 600 image.pdf image.tif
convert -density 600 image.eps image.tif
```

We can also reverse the process to turn an EPS or TIFF into a PDF:

```
convert -density 600 image.eps image.pdf
convert -density 600 image.tif image.pdf
```

Note that this final conversion will not improve the resolution of a TIFF or convert it to a vector image. Instead it will simply encode the original raster into a PDF file. ImageMagick can also be used to perform a large number of other image manipulations.

Git/GitHub, Transparency, and Legitimacy in Quantitative Research

Zachary M. Jones
Pennsylvania State University
zmj@zmjones.com

The decreasing cost of computing power and the increase in the availability of a variety of public data has made quantitative research much more attractive to quantitative social scientists. I would argue that the increasing availability of public data and the availability of enormous computational power has generally been a good thing for the discipline. It has not been without cost however. Since researchers now have enormous flexibility in data collection and manipulation, as well as model selection, estimation, and reporting, it is often difficult to evaluate the internal and external validity of published findings. In other disciplines (notably psychology and medicine) there has been a perceived and actual increase in the false-positive rate of published quantitative research (Simmons, Nelson and Simonsohn 2011). Increases in the actual and/or perceived false-positive rate may have policy implications, as politicians and grant-giving institutions decide how to allocate limited resources.

What is the most effective way to deal with these issues? Many of the most prominent journals in political science now require replication materials. Replication archives are still not ubiquitous, and progress in this area is an important step towards decreasing the actual and perceived false-positive rate. There is also evidence of a disconnect between requirements and practice, at least in economics (Andreoli-Versbach and Mueller-Langer 2013). There has been a broader push to make research more transparent. For example, a recent issue of *Political Analysis* focused on the advantages of study pre-registration (Lupia 2008; Monogan 2013; Humphreys, de la Sierra and van der Windt 2013; Anderson 2013).

I propose an addition to the idea of a replication archive that lends credibility in a way similar to study registration, makes all data and model related decisions at all points in time reproducible (if the author so chooses), and serves a pedagogical purpose as well. Simply keep your project (data, tranformation code, model code, and the manuscript) in a Git repository, and post said repository publicly. Git is a distributed revision control system which allows the user to track changes to any file that is text (R and Stata code, LaTeX code, delimited data, etc.), revert to any previous version easily, visualize changes between versions, and a variety of other emminently useful things (creating branches of a project, asynchronous collaboration, etc.). GitHub is an enormously popular web-service that allows the user to host Git repositories publicly (or privately for a price, though they offer free student accounts for 2 years). There are a number of excellent resources for aquainting yourself with Git and GitHub, in particular Pro Git, Try Git, and this excellent answer on StackOverflow. GitHub also has graphical applications available for Mac and Windows machines, as well as integration with Eclipse, Vim, Emacs, and most other text editors with an active community. In addition to the aforementioned offical Git GUIs, there are a number of 3rd party applications that make using Git quite easy (see here for a list). The (justifiably) popular integrated development environment (IDE) for R, RStudio, also provides integrated support for Git through RStudio "projects." While Stata does not provide integration with Git, it would be trivial to keep `.do` files in revision control using any one of the above resources. It is worth noting that Git is not the only revision control system (though it is probably the most popular). Subversion and Mercurial are two popular alternatives which could be used for a similar purpose.

A complete research project hosted on GitHub is reproducible and transparent by default in a more comprehensive manner than a typical journal mandated replication archive. With a typical journal replication archive, the final data and code to run the final set of models is provided. This leaves to the imagination most of the details of the data collection and/or data manipulation that produced the final data set, what model specifications preceded the ones present in the final script, and how the manuscript changed during its journey from idea to publication. With a public Git repository the data, any manipulation code, and the associated models are available at any time that a change was "committed" to a file tracked in said Git repository. Keeping data, data manipulation code, model code, code for visualizations (tables and graphs), along with the manuscript in a Git repository on GitHub (or a similar site such as Bitbucket) thus subsumes and extends the advantages of journal maintained replication archives.

Maintaining your research project on GitHub confers advantages beyond the social desireability of the practice and the the technical benefits of using a revision control system. Making your research publicly accessible in this manner makes it considerably easier to replicate, meaning that, all else equal, more people will build on your work, leading to higher citation counts and impact (Piwowar, Day and Fridsma 2007). Hosting your work on GitHub, because of its popularity in and outside of academia, also increases the probability of your work being seen by people that aren't actively involved in academic political science. Worries about being "scooped" may also be allayed by using a public revision control system, since there is then a public record of your work on the project (as previously noted, you can also keep repositories private).

Additionally, there are pedagogical advantages to this sort of open research. The process by which research ideas are generated, formalized, empirically evaluated, written up, and then (hopefully) published is often opaque to those who have not participated in it. Published papers often seem to

have sprung forth from the head of Zeus, absent previous, more imperfect forms. Much of graduate school revolves around learning how to navigate the idea to publication pathway, and all the pitfalls it entails. Greater knowledge of how it is navigated would undoubtedly help in this process. With Git and GitHub, illuminating this would be low-cost.

If open research of this sort was to become a norm in political science, it is hard to imagine that the field would not advance more quickly. Using Git and Github confers non-trivial technical advantages, has a low startup cost given the array of modern software that interfaces with Git, is desireable from a social perspective and an individual perspective, and provides a helpful pedagogical service as well. Although adoption across the field is unlikely (or at least will be a long time in coming), political methodologists are the ideal group of people to be leaders in pushing for transparent, reproducible research, in political science and in related disciplines.

References

Anderson, Richard G. 2013. "Registration and Replication: A Comment." *Political Analysis* 21 (1): 38-39.

Andreoli-Versbach, Patrick and Frank Mueller-Langer. 2013. "Open Access to Data: An Ideal Professed but not Practised." *RatSWD Working Paper Series* 215: 1-10.

Humphreys, Macartan, Raul Sanchez de la Sierra, and Peter van der Windt. 2013. "Fishing, Commitment, and Communication: A Proposal for Comprehensive Nonbinding Research Registration." *Political Analysis* 21 (1): 1-20.

Lupia, Arthur. 2008. "Procedural Transparency and the Credibility of Election Surveys." *Electoral Studies* 27 (4): 732-739.

Monogan, James E. 2013. "A Case for Registering Studies of Political Outcomes: An Application in the 2010 House Elections." *Political Analysis* 21 (1): 21-37.

Piwowar, Heather A., Roger S. Day, and Douglas B. Fridsma. 2007. "Sharing Detailed Research Data is Associated with Increased Citation Rate." *PloS ONE* 2 (3): e308.

Simmons, Joseph P., Leif D. Nelson, and Uri Simonsohn 2011. "False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant." *Psychological Science* 22 (11): 1359-1366.

Giving a (Job) Talk: Notes from the Field

Arthur Spirling
Harvard University
aspirling@gov.harvard.edu

There can be few activities more anxiety-inducing than going on the job market. Once on the market, there can be few activities that are more anxiety-inducing than giving a job talk. This post is an attempt to share information and advice, and hopefully reduce the stress that many feel.

Given its importance as a decider of fates, it is perhaps surprising that so little attention is paid to teaching students about the job talk's value, structure and content as they work their way through grad school. What follows is my advice, some of which I was given, and some which I learned on my own, or following discussion with colleagues: the usual caveat most definitely applies. Inevitably, different people will have different thoughts, and students preparing for the market should ask local experts for input. Quite who those "experts" actually are should be decided on a case-by-case basis, but here is a pro-tip on something to look for: the 'expert' has given a successful job talk (recently). Relaxing the conditions slightly, if you have never seen them give an interesting or intelligible talk about their own research such that you would wish to emulate their performance, they should probably not be your primary source on "what works" for closing the deal at the university where you seek employment.

In my current role as a faculty member, I've given several talks on talks to students and others, and have prepared a slideshow that discusses everything from the nature of the talk, its typical contents, how to display data, how to deal with questions, and so on. To be clear, I come at this problem (mainly) from the perspective of a political methodologist seeking a job at a research orientated institution, but I think the contents will helpful for most candidates navigating this difficult process. You can find my slides here.

For those looking for quicker tips on strategy, here are the five most important general points I wish to convey:

1. It is a job ***talk***, not a paper, not a recitation, not a performance, not a lecture.

Though it is tempting to 'script' your talk, this is always bad news: candidates who have memorized (to the letter) what they "must" say on each slide come across as stilted and unintelligent. An even lower opinion results for those who have notes that they read for each slide. As an audience member, it is very hard to understand why speakers don't know their own work well enough to discuss it in an unaided

way for the 45 minutes or so that a typical talk lasts. Sometimes, candidates think they can get around this problem by packing their slides with text such that they can simply "read off" their presentation. This may be worse: see next point(s).

2. Visuals matter, so think about your slides from the perspective of an audience member.

If you have to use the phrase "I'm not sure if everyone can see that" because your table or font or diagram is too small, with probability 1 you have a "bad" slide. In methods talks, this seems especially common when candidates present regression results. If you *must* have a regression table, have it as a back-up, and go to it only if someone requests the whole thing. Very generally speaking, clean slides are better than busy ones, and slides with less technical detail are better than those loaded up with Greek letters. If you must use equations, be sure to introduce them intuitively and explain everything with words rather than symbols. If you are not sure how to write nice slides, go to other people's talks and think about what worked and what didn't in terms of the presentation. It's fine to use tricks or displays that you've been impressed by. If you struggle to know what should and should not be included in your presentation, ask yourself "what question does this slide answer?" If, in fact, you can not think of the corresponding interrogative, you probably have a slide that should be edited – and maybe even removed.

Remember: it is often difficult to be succinct rather than verbose, but it will make the world of difference to an audience.

3. Take questions as they come up: don't say you will only be taking "clarifying" questions, or that you will deal with "substantive questions" only "at the end", or something similar.

When you give a talk, you are attempting to showcase your research to the audience, along with your ability to teach and to be a good colleague. Luckily, I have lots of good colleagues; tellingly, not one of them has ever asked me to wait 40 minutes while he or she completes a monologue before I am allowed to seek clarification on something they've said. My point here is that audience members have every right to question your work, and they have the right to expect an answer. If the next few slides (or one in reserve) will make something clearer, simply say that. If you want to give a short, helpful answer and return to the issue later, say that. If you haven't thought about the issue, try to say something intelligent that makes it clear you are intellectually flexible. It is a dull experience when a speaker is so uninterested in the audience's comprehension of the material being presented that no one may seek to check their understanding along the way. Notice further, that you should be willing to take questions from everyone: grad students, junior faculty, senior faculty, visiting faculty etc. The committee that hires you wants to see that you can "speak" (literally) to the broad range of people in a department – certainly, most committees will solicit feedback from all their colleagues as to the merits of the candidate.

It is worth noting that candidates often, perhaps naturally, worry about getting caught off-guard: that a question will be so devastating or complicated that they will never get their talk back on track. This is unlikely to be the case if you practice your talk and Q-and-A (see next item). Moreover, never forget that you really are *the* expert on your own work: you have spent years on this project, and no one knows it as well as you. Subsequently, the way that you craft your presentation, including what you seek to focus on and how, will go someway to funneling questions in terms of their topic.

Precisely because you will take questions, you should budget the time for the talk accordingly: it would not be an unreasonable goal for the talk to run around 35 minutes straight through, expanding to 45 minutes with interruptions. Quite how many slides that requires is up to you, though spending around 1.5-2 minutes a slide is a good (starting) benchmark.

4. Practice, practice, practice.

Your job talk should be prepared as early as possible: two weeks before the market opens (so, around the time of APSA) is not unreasonable for a first draft. Then, you should practice its delivery. A lot. For a grad student, practicing the talk everyday they are on the market is prudent: simply being in a room on your own and running through the slides out loud is a good start, but it's even better if you can have people (grad students, friends, junior and senior faculty) there to give you feedback and just generally simulate an audience. I have found that practicing goes a long way to reducing anxiety for me, and for students. It's fine, and even good, to be nervous about doing the best job you can: but once the talk begins, and you see the (by now very familiar) slides, you will hopefully find yourself relaxing and (even!) enjoying yourself.

5. Don't obsess over how the talk went: get on with business.

Under the stress of giving a talk (and the relief of having completed one), it is natural that your mind will obsess about how the audience responded. In truth, this is very difficult for the speaker to tell: there have been occasions

(including times when I was on the market) when I thought the talk had gone well, though it hadn't. Similarly, there were times that the audience seemed listless, quiet and unreceptive, though they had (or, perhaps, the key members had) actually enjoyed it very much. In the moment, it is much better to stick to the mantra that you did your best, and you now need to move on the next item on the agenda: typically one-on-ones with faculty, a lunch, or a dinner. It does no good to over-think these things after the event – except to learn for next time.

What Courses Do I Need to Prepare for a PhD in Political Science?

Justin Esarey
Rice University
justin@justinesarey.com

I recently had a discussion with some of my graduate students about what an ideal preparation for a PhD program in Political Science would look like. They were discussing the issue because they felt that very little of their undergraduate Political Science education prepared them for what they'd be learning in graduate school, especially in terms of methodological tools and design approaches to applied research. I felt it might be valuable for undergraduates thinking of pursuing the PhD – or new graduate students who hadn't realized what they were getting themselves into – to post the question to the community at large and have the responses on TPM as a resources.

When undergraduates ask me this question, I usually tell them that someone hoping to study a substantive area (International Relations, Comparative Politics, American Politics, or Policy) would ideally have taken:

- two semesters of calculus, including differentiation, integration, and infinite series;
- one semester of matrix linear algebra;
- one semester of (a) undergraduate econometrics or (b) probability theory from a statistics department;
- one semester of programming in a relevant language, such as Python, MATLAB, or R;
- some kind of serious research design/epistemology class; and
- as many courses as you can take that include reading published academic literature in your subject area (look to see that the syllabus assigns academic journals or university press books, not textbooks)

Some courses may kill two birds with one stone if, for example, they use MATLAB or R as a part of teaching some other subject.

Those hoping to work in methods or formal theory should consider pursuing a Math minor or double major, including all of the above courses plus:

- a semester of three-dimensional calculus;
- a semester of real analysis;
- a semester of differential equations;
- a semester of discrete math;
- a semester of some form of mathematical microeconomics class at the advanced undergraduate or introductory graduate level;
- as many econometrics or applied statistics courses as you can fit into your schedule on top of the above.

Designing an appropriate preparation for people who plan on being area specialists and spending a lot of time in the field using qualitative methods is somewhat outside of my area of special expertise. With that proviso, I usually recommend the following courses in place of the lists above:

- at least one semester of calculus, covering differentiation and integration;
- one semester of matrix linear algebra;
- fluency in at least one of the following: Spanish, Chinese, Russian, Arabic (chosen to best-suit your area of interest)
- reading and writing proficiency in another language relevant to your area
- as many courses as you can take that include reading published academic literature in your subject area (look to see that the syllabus assigns academic journals or university press books, not textbooks)

This list replaces most of the math with language training.

It is, of course, worth noting that very few students – including those who are very successful – come into a PhD with this amount of training. But my own undergraduate adviser told me that the more of this stuff that I could get out of the way before I got to graduate school, the more that I could focus on learning the substance of the field rather than picking up mathematical tools. I think that was basically good advice.

What courses would you add to or subtract from this list?

Rice University
Department of Political Science
Herzstein Hall
6100 Main Street (P.O. Box 1892)
Houston, Texas 77251-1892

The Political Methodologist is the newsletter of the Political Methodology Section of the American Political Science Association. The support of the Department of Political Science at Rice University in helping to defray the editorial and production costs of the newsletter is gratefully acknowledged.

Subscriptions to *TPM* are free for members of the APSA's Methodology Section. Please contact APSA (202-483-2512) if you are interested in joining the section. Dues are $29.00 per year and include a free subscription to *Political Analysis*, the quarterly journal of the section.

Submissions to *TPM* are always welcome. Articles may be sent to any of the editors at thepoliticalmethodologist@gmail.com. LaTeX format files are especially encouraged.

TPM was produced using LaTeX.