Scientists Programming

Amelia McNamara

Abstract: Scientists are an important important sub-category of programmers, because their work has implications in government policy, medicine and general knowledge. The three main categories of computer use by scientists are "make models, generate data," "generate data, make models," and general application programming. We will not consider the last category in this talk, as it is the closest to traditional software engineering. Common tools for making models to generate data are Matlab, Sage, Maple, and Mathematica. Scientists who generate data (by field collection, experimentation, etc) and make models, often use Excel, Stata, SPSS, SAS, R or Python. The transition from non-reproducible research to more robust analysis can be challenging, but the benefits are broad. We consider as a case study a marine science group who wrote "Our path to better science in less time using open data science tools" about their movement from Excel to a toolkit including git and GitHub for collaboration and version control, R in the IDE RStudio for analysis, and RMarkdown for reproducibility and narrative.

Introduction

When scientists program, they are typically in one of three categories: "make models, generate data," "generate data, make models," and general application programming. We will not consider the last category, as it is the closest to traditional software engineering.

Make models, generate data

In the category of "make models, generate data," scientists begin with a theory about the world, and then use the computer to generate data under that model to observe whether it is consistent with their understanding.

The most common tools in this paradigm include Matlab, Sage, Maple, and Mathematica.

For example, in (Chang, Fu, and Luo 2012), the authors write "Based on the obtained kinetic model, we used Matlab to simulate the direct DME synthesis reactions over Cu-Zn-Al/HZSM-5 catalyst for syngas from run 6 (air/CO 2 gasification combined with addition of biogas)." In (van der Tol et al. 2009), "Model simulations were evaluated against observations reported in the literature and against data collected during field campaigns. These evaluations showed that SCOPE is able to reproduce realistic radiance spectra, directional radiance and energy balance fluxes. [...] The model has been implemented in Matlab and has a modular design, thus allowing for great flexibility and scalability."

The authors provide their full Matlab SCOPE code online through GitHub (van der Tol 2017). A snippet is reproduced below

```
fid     = fopen(filename,'r');
modname = filename(1:size(filename,2)-4);
outname = [modname '.atm'];

for i = 1:7, fgetl(fid); end

rline = str2num(fgetl(fid));
tts   = rline(4);
cts   = cosd(tts);

s     = fgetl(fid);
rline = sscanf(s,'%10f',3);
wns   = rline(1); wne = rline(2); wstep = rline(3);
```

```
nw    = int32((wne-wns)/wstep)+1;

datarec = zeros(nw,15,4);        % MODTRAN5.2.1 tp7 15 column output format
Tall    = zeros(nw,18);          % 18 output spectra
```

In fact, it was this paradigm of scientists programming that is credited with the idea of reproducible research. In their 1995 paper, Jonathan Buckheit and David Donoho wrote,

> "To avoid sounding like we are pointing the finger at anyone else, we will mention a few problems we have encountered in our own research.
>
> **Burning the Midnight Oil.** Once, writing an article with approximately 30 figures, we had to tweek various algorithms and display options to display clearly the effects we were looking for. As a result, after an IS-hour day we had accumulated a stack of a few hundred sheets of paper, all of which purported to be versions of the figures for the article. We gave up well after midnight. [...]
>
> **A Year is a Long Time in this Business.** Once, about a year after one of us had done some work and written an article (and basically forgot the details of the work he had done), he had the occasion to apply the methods of the article on a newly-arrived dataset. When he went back to the old software library to try and do it, he couldn't remember how the software worked - invocation sequences, data structures, etc. In the end, he abandoned the project, saying he just didn't have time to get into it anymore" (Buckheit and Donoho 1995).

Generate data, make models

The paradigm we will focus more strongly on is "generate data (through experimentation, observation, etc), make models." This is a more statistical perspective, and is used by many of the natural sciences.

Common tools here include Excel, SAS, SPSS, Stata, R, and Python.

As computer scientists, it can be easy to dismiss Excel as a programming tool. However, it has approximately 750 million users, and provides reactive programming for free. It combines elements of data input, wrangling, modeling, visualization, output in one document. People who use Excel often do not think of themselves as programmers, although they are engaging in sophisticated computational thinking. For more about this, see Hermans (2016).

As an example, consider ("GEOL 388 Excel Tips," n.d.), a guide for geology graduate students analyzing data with Excel. The page is filled with screenshots and detailed instructions. The author writes "The best thing about Excel is its automatic fill capabilities. Let's say I wanted to make a column of numbers from 0 to 15. Type in a few entries - enough to define a pattern - into a column. Below, I typed in 0 and 1 into a column and highlighted both numbers."

Of course, Excel is not reproducible, which has implications for transparent science and makes it difficult to collaborate. Jenny Bryan, a noted proponent of reproducibility tweeted, "I'm seeking TRUE, crazy spreadsheet stories. Happy to get the actual sheet or just a description of the crazy. Also: I can keep a secret." (Bryan 2016) The tweet received at least 70 responses.

Excel also has continuing errors in statistical functionality. Because it is a closed-source piece of software, this is hard to suss out, but a group of researchers have persistently tested every version of Excel. In a 2004 paper, they wrote "Some of the problems that rendered Excel 97, Excel 2000 and Excel 2002 unfit for use as a statistical package have been fixed in Excel 2003, though some have not. Additionally, in fixing some errors, Microsoft introduced other errors" (McCullough and Wilson 2004; McCullough and Heiser 2008; Melard 2014).

Beyond these errors, intentional features of Excel can lead to problems in data analysis. The most well-publicized example of this is the conversion of gene names to dates in Excel.

> "We downloaded and screened supplementary files from 18 journals published between 2005 and 2015 using a suite of shell scripts. [...]
>
> Of the selected journals, the proportion of published articles with Excel files containing gene lists that are affected by gene name errors is 19.6 %. [...]
>
> Journals that had the highest proportion of papers with affected supplementary files were Nucleic Acids Research, Genome Biology, Nature Genetics, Genome Research, Genes and Development and Nature (>20 %)." (Ziemann, Eren, and El-Osta 2016)

Even with all these drawbacks, spreadsheets are still very commonly used in science. As Charles Weiss writes, "Spreadsheets are a standard tool in chemistry for simple tasks such as data analysis and graphing. Chemistry students are often introduced to spreadsheets their first year of college, if not earlier, and those who continue on to do research will likely use them as a means of handling and visualizing data. [...]

Software better geared to those earning chemistry degrees or conducting research is readily available. Common examples include MATLAB, Python's SciPy stack, and GNU Octave." (Weiss 2017)

Although scientists may be afraid to make the leap, they almost always see the benefits once they have converted to a more programming-intensive, reproducible workflow. "Andrew Durso can vouch for those upsides. The ecology graduate student at Utah State University in Logan started his research career using programs with graphical interfaces. Whenever he clicked buttons or checked boxes on a computer screen, he would try to write those steps down on paper in case he wanted to redo an analysis — a strategy that was both time-consuming and unreliable." (Baker 2017)

> "If http://www.bioinformatics.org is any indication, most biologists in need of advanced mathematics take the no-cost route. The blog recently asked its readers,"Which math/statistics language/application do you most frequently use?" A plurality (24%) of the survey's 1,675 or so respondents chose R, an open-source statistical language and environment [...]
>
> Matlab garnered second place in the bioinformatics.org survey with 20%.
>
> Five percent of those who responded to the bioinformatics.org poll develop mathematical applications in Mathematica, from Wolfram Research of Champaign, Ill." (Perkel 2005)

As a motivating example, we can consider "Our path to better science in less time using open data science tools," by the Ocean Health Index project (Lowndes et al. 2017). For years, the group had an Excel-based workflow, and thought it was working for them.

> "We thought we were doing reproducible science. For the first global OHI assessment in 2012 we employed an approach to reproducibility that is standard to our field, which focused on scientific methods, not data science methods. Data from nearly one hundred sources were prepared manually—that is, without coding, typically in Microsoft Excel—which included organizing, transforming, rescaling, gap-filling and formatting data. Processing decisions were documented primarily within the Excel files themselves, e-mails, and Microsoft Word documents. We programmatically coded models and meticulously documented their development, (resulting in the 130-page supplemental materials), and upon publication we also made the model inputs (that is, prepared data and metadata) freely available to download. This level of documentation and transparency is beyond the norm for environmental science." (Lowndes et al. 2017)

They took a gradual model for change, iterating over the course of several years. As they write,

> "We decided to base our work in R and RStudio for coding and visualization, Git for version control, GitHub for collaboration, and a combination of GitHub and RStudio for organization, documentation, project management, online publishing, distribution and communication. [...]
>
> **Data preparation: coding and documenting.** Our first priority was to code all data preparation, create a standard format for final data layers, and do so using a single programmatic language, R. Code enables us to reproduce the full process of data preparation, from data download to final model inputs, and a single language makes it more practical for our team to learn and

> contribute collaboratively. We code in R and use RStudio to power our workflow because it has a user-friendly interface and built-in tools useful for coders of all skill levels, and, importantly, it can be configured with Git to directly sync with GitHub online (See 'Collaboration'). [...]
>
> **Sharing methods and instruction.** We use R Markdown not only for data preparation but also for broader communication. R Markdown files can be generated into a wide variety of formatted outputs, including PDFs, slides, Microsoft Word documents, HTML files, books or full websites." (Lowndes et al. 2017)

There are many resources available to scientists who want to improve their workflows. A paper called "Best Practices for Scientific Computing" has gone through several iterations (Aruliah et al. 2012; Wilson et al. 2014). However, this has turned out to be slightly too high a bar, which led to a subset of the authors writing "Good Enough Practices for Scientific Computing" (Wilson et al. 2016).

Another good resource for scientists is the Practical Data Science for Statistics PeerJ collection, curated by Jenny Bryan and Hadley Wickham (Bryan and Wickham, n.d.). In particular, "Data Organization in Spreadsheets" may be relevant. This paper has a reputation for having the best abstract of all time, because it packs in all the most important takeaways from the paper.

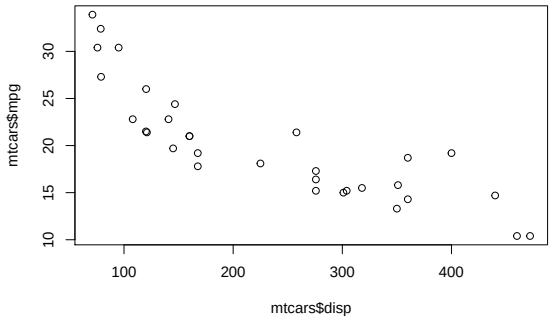
> "Abstract: Spreadsheets are widely used software tools for data entry, storage, analysis, and visualization. Focusing on the data entry and storage aspects, this paper offers practical recommendations for organizing spreadsheet data to reduce errors and ease later analyses. The basic principles are: be consistent, write dates like YYYY-MM-DD, don't leave any cells empty, put just one thing in a cell, organize the data as a single rectangle (with subjects as rows and variables as columns, and with a single header row), create a data dictionary, don't include calculations in the raw data files, don't use font color or highlighting as data, choose good names for things, make backups, use data validation to avoid data entry errors, and save the data in plain text file" (Broman and Woo 2017).

Many of these resources are centered on the use of the statistical programming language R. R is widely used by scientists, but is sometimes criticized by computer scientists for its strange language properties. For some computer scientists, reading "Evaluating the Design of the R Language: Objects and Functions For Data Analysis" helps give them a better perspective on the language (Morandat et al. 2012). They describe R as "dynamic, lazy, functional, object-oriented" (Morandat et al. 2012)

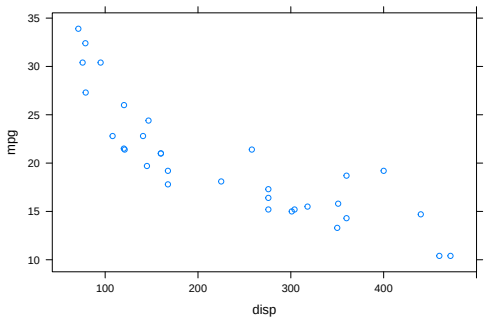
One aspect of the language that is particularly relevant to novices is the fact that R provides the ability to support a variety of syntaxes. Because of this, the language can feel inconsistent. Instructors who teach R try to sandbox students into a subset of the language that is internally consistent. Full courses can be taught in the three main R syntaxes: - base R syntax (uses brackets for indexing, numeric indexing, and dollar signs to access variables within data sets) - formula syntax (builds "formulas" using the tilde, comes from the original modeling syntax in R) - tidyverse syntax (becoming the most popular syntax, allows data to be "piped" through a sequence of functions)

As a pedagogical tool, the author of this paper has written an "R Syntax Cheatsheet" (McNamara 2018). It provides comparisons between all three syntaxes. For example, making a scatterplot:

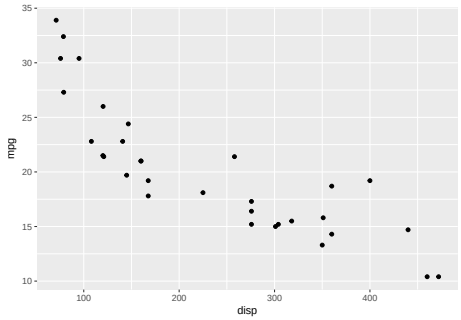
```
plot(mtcars$disp, mtcars$mpg)
```




```
library(lattice)
xyplot(mpg~disp, data=mtcars)
```




```
library(ggplot2)
qplot(x=disp, y=mpg, data=mtcars, geom="point")
```



The three syntaxes produce the same plot (with slightly different default colors, etc) , but are quite differently specified, particularly for novices. Again, the tidyverse syntax is becoming the most popular, both by scientists and statistics educators.

The tidyverse syntax expects "tidy" data (Wickham 2014), and uses the pipe `%>%` (often pronounced "then") to string together verbs like `select()`, `filter()`, `mutate()` and `group_by()`.

Bringing back the Ocean Health Index, here is an example of their code, taken from their GitHub repo (Ocean Health Index 2017):

```
monitoring_indicator_mean_loc= monitoring_indicator_mean %>%
  left_join(., select(coastal_fish_loc, station_cleaned, lat, lon),
            by=c("monitoring_area" = "station_cleaned"))

coastal_fish_loc = coastal_fish_loc %>%
  mutate(lat = DECWGSN, lon = DECWGSE) %>%
  mutate(lat = ifelse(station_cleaned== "ICES SD 31", 64.207281,
                      ifelse(station_cleaned== "ICES SD 30", 61.679815,
                      ifelse(station_cleaned== "ICES SD 29", 59.705518,
                      ifelse(station_cleaned== "ICES SD 32", 60.108130,
                      ifelse(station_cleaned== "Rectangle 23 & 28",  63.505817,lat ))))),
         lon = ifelse(station_cleaned== "ICES SD 31", 23.511161,
                      ifelse(station_cleaned== "ICES SD 30", 21.269112,
                      ifelse(station_cleaned== "ICES SD 29", 21.264879,
                      ifelse(station_cleaned== "ICES SD 32", 25.772410,
                      ifelse(station_cleaned== "Rectangle 23 & 28",  21.351243,lon )))))) %>%
  mutate(station_cleaned = as.character(station_cleaned),
         station = as.character(station))

temp_long = coastal_fish_scores %>%
  select(monitoring_area,core_indicator,taxa, score1,score2,score3) %>%
  group_by(monitoring_area, core_indicator,taxa) %>%
  gather(score_type,score,score1,score2,score3) %>%
  ungroup()

slope2 = slope2 %>%
  group_by(basin_name, core_indicator) %>%
  mutate(slope_mean_basin_indicator = mean(slope_mean_basin_taxa)) %>%
  ungroup()

basin_n_obs = coastal_fish_scores_long %>%
  filter(score_type=="score1") %>% ## only select 1 score type so no duplicates
  select(Basin_HOLAS) %>%
  count(Basin_HOLAS)
```

Beyond the language functionality, R has become popular because of the development of the Integrated Development Environment, RStudio, and related products. A primary tool is the R package RMarkdown, which is a flavor of Markdown that allows a writer to intermingle formatted text, code, and code output for fully-reproducible reports (RStudio Team 2018). (In fact, this manuscript was produced using RMarkdown.)

While R has many great qualities, and is perhaps the most-recommended tool for scientists programming in the "generate data, make models" paradigm, it is still quite difficult for novices to master. (Here, when we say novice we mean someone unaccustomed to programming.) When teaching statistics and data analysis, some instructors choose to use tools designed for teaching and learning statistics (such as software packages TinkerPlots or Fathom, and applet collections like StatKey). This creates a gap of knowledge when users want to move from the learning tools to a tool designed for *doing* data analysis, like R. It is possible to imagine a tool that would better support novices (among them, scientists moving away from Excel). McNamara (2016) outlines 10 "key attributes" for such a modern statistical computing tool:

- Accessibility
- Ease of Entry
- Data as a first-order object
- Support for a cycle of exploratory and confirmatory analysis
- Flexible plot creation
- Support for randomization and the bootstrap

- Interactivity at every level
- Inherent documentation
- Support for narrative, publishing, and reproducibility
- Flexibility to build extensions

Conclusion

The way scientists use computers and program may be quite different than other programmers. Their tasks are primarily related to data, either generating it from a model, or using it to make a model. There has been a movement toward better reproducibility in science, which is pushing more scientists toward programming tools, particularly R. However, programming languages could be designed to better support these types of tasks.

References

Aruliah, D A, C Titus Brown, Neil P Chue Hong, Matt David, Richard T Guy, Steven H D Haddock, Katy Huff, et al. 2012. "Best Practices for Scientific Computing." http://arxiv.org/abs/1210.0530v1.

Baker, Monya. 2017. "Scientific Computing: Code Alert." *Nature Jobs*, 563–65.

Broman, Karl, and Kara Woo. 2017. "Data Organization in Spreadsheets." *PeerJ Preprints*. https://peerj.com/preprints/3183/.

Bryan, Jennifer. 2016. "@JennyBryan 'I'm Seeking True, Crazy Spreadsheet Stories. Happy to Get the Actual Sheet or Just a Description of the Crazy. Also: I Can Keep a Secret."' https://twitter.com/JennyBryan/status/722954354198597632.

Bryan, Jennifer, and Hadley Wickham, eds. n.d. "Practical Data Science for Stats." https://peerj.com/collections/50-practicaldatascistats/.

Buckheit, Jonathan B, and David L Donoho. 1995. "WaveLab and Reproducible Research." *Wavelets and Statistics*.

Chang, Jie, Yan Fu, and Zhongyang Luo. 2012. "Experimental Study for Dimethyl Ether Production from Biomass Gasification and Simulation on Dimethyl Ether Production." *Biomass and Bioenergy* 39: 67–72.

"GEOL 388 Excel Tips." n.d. http://www.geo.cornell.edu/geology/classes/geol388/excel/excel.html.

Hermans, Felienne. 2016. "Functional Programming in Excel." https://vimeo.com/162206549.

Lowndes, Julia Stewart, Benjamin D Best, Courtney Scarborough, Jamie C Afflerbach, Melanie R Frazier, Casey C O'Hara, Ning Jiang, and Benjamin S Halpern. 2017. "Our Path to Better Science in Less Time Using Open Data Science Tools." *Nature Ecology & Evolution* 1.

McCullough, B D, and David A Heiser. 2008. "On the Accuracy of Statistical Procedures in Microsoft Excel 2007." *Computational Statistics & Data Analysis* 52: 4570–8.

McCullough, B D, and Berry Wilson. 2004. "On the Accuracy of Statistical Procedures in Microsoft Excel 2003." *Computational Statistics & Data Analysis* 49 (4): 1244–52.

McNamara, Amelia. 2016. "Key Attributes of a Modern Statistical Computing Tool." https://arxiv.org/abs/1610.00985.

———. 2018. "R Syntax Comparison Cheatsheet." https://github.com/rstudio/cheatsheets/raw/master/syntax.pdf.

Melard, Guy. 2014. "On the Accuracy of Statistical Procedures in Microsoft Excel 2010." *Computational Statistics* 29 (1095).

Morandat, Floréal, Brandon Hill, Leo Osvald, and Jan Vitek. 2012. "Evaluating the Design of the R Language:

Objects and Functions for Data Analysis." In *ECOOP'12 Proceedings of the 26th European Conference on Object-Oriented Programming*.

Ocean Health Index. 2017. "Ocean Health Index for the Baltic Sea." https://github.com/OHI-Science/bhi.

Perkel, Jeffrey. 2005. "Biology by the Numbers." *The Scientist*, June.

RStudio Team. 2018. "R Markdown." https://rmarkdown.rstudio.com/.

van der Tol, Christiaan. 2017. "SCOPE." https://github.com/Christiaanvandertol/SCOPE.

van der Tol, Christiaan, W Verhoef, J Timmermans, A Verhoef, and Z Su. 2009. "An Integrated Model of Soil-Canopy Spectral Radiances, Photosynthesis, Fluorescence, Temperature and Energy Balance." *Biogeosciences* 6: 3109–29.

Weiss, Charles J. 2017. "Perspectives: Teaching Chemists to Code." *Chemical & Engineering News* 95 (35): 30–31.

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10).

Wilson, Greg, D A Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven H D Haddock, et al. 2014. "Best Practices for Scientific Computing." *PLoS Biology* 12 (1).

Wilson, Greg, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K Teal. 2016. "Good Enough Practices for Scientific Computing." https://swcarpentry.github.io/good-enough-practices-in-scientific-computing/.

Ziemann, Mark, Yotam Eren, and Assam El-Osta. 2016. "Gene Name Errors Are Widespread in the Scientific Literature." *Genome Biology* 17 (177).