

# Data Science Moves – Some thoughts and provocations

Amelia McNamara

September 2016

## 1 Introduction

Bill asked me to spend time this month thinking about what ‘data science moves’ might be. Data science is generally considered to be at the intersection of statistics, computer science, an application area, and communication of results. There is a famous Venn diagram to that effect by Drew Conway, and we’ve used the basic idea in writing our major in Statistical and Data Sciences at Smith (Smith College, 2016).

In R, where I spend most of my time, there is a collection of packages called the ‘tidyverse,’ so-named because they work with tidy data (Wickham, 2014) and have a consistent, pipeable syntax. The tidyverse attempts to make it possible to work through an entire data science workflow, of tidying, transforming, visualizing, modeling, and communicating results about data. These tasks feed back into one another, as shown in Figure 1.

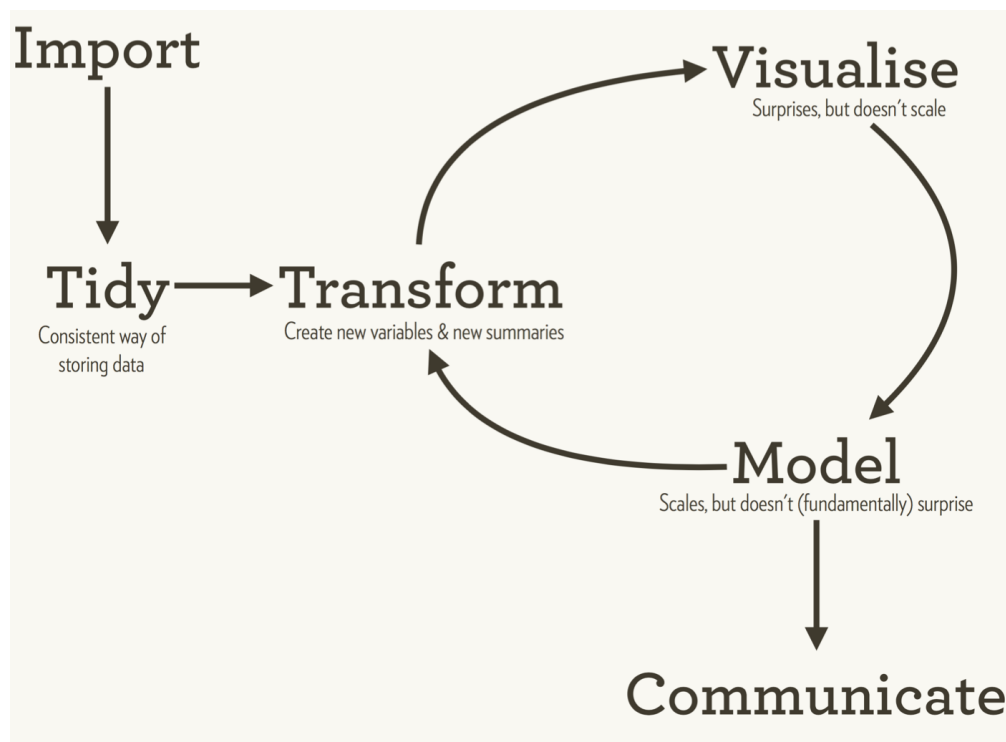


Figure 1: The data science cycle, via Hadley Wickham.

My argument is that all the tasks from Figure 1 should be possible in CODAP. Let's think about them each in more depth.

## 2 Import

Importing encompasses reading in many types of data, including data in different file types and from APIs. It seems like the goal of CODAP is to allow for import of data from local files and from URLs. I wasn't able to get a URL to work for me, but since the feature is already under the dropdown it seems like it's coming. One issue I discussed with Bill is that the data import functions expect to get flat files (tidy data) but then allow you to create hierarchical structures. The system won't allow you to import already-hierarchical data like JSON. I know the CODAP documents are stored as JSON, so my guess is that the underlying data representation for hierarchical data you make out of flat data is JSON. It would be nice to support the import of those files.

## 3 Tidy

This is the one goal from the tidyverse I'm not sure will be relevant for CODAP. Tidy data has one observation per row, one variable per column, and all observations are of the same type (Wickham, 2014). (Wickham notes this a version of Codd's 3rd normal form.) One common category of un-tidy data is education data. For whatever reason, educational outcome data often comes with some rows representing counties, some representing school districts, and some individual schools. Because the observations aren't all at the same observational unit, this data isn't tidy. To tidy it, a user would need to separate it into multiple tables, one for each level of observation. Or, perhaps it could be stored hierarchically.

Like I discussed above, CODAP supports hierarchical data as long as it is made within the system. I think this is great! Hierarchical data is perhaps more natural for people. But, it does require some more thought about what would make it 'tidy,' and it will likely change both the technical requirements for data transformation and perhaps the visual metaphors for the same.

## 4 Transform

This is the area where 'data science moves' are most clearly different from tasks in other domains, and also where there is the most work needed in CODAP. Again, I'm looking toward the tidyverse for inspiration. The R package **dplyr** was written to capture many possible data manipulations with a small number of 'verbs.' These verbs were inspired by SQL. They are:

- **mutate** (allows you to make a new variable or transform an existing one)
- **summarize** (does a many-to-one mapping to summarize variables, perhaps grouped in a particular way)
- **filter** (logical filtering on data)
- **select** (pulls out specific variables)

- `slice` (pulls out specific rows)
- `group_by` (groups data by a particular variable or variables)
- `arrange` (sorts the data)
- `sample_n` (samples from the data)

The idea of the small set of verbs is that they are like building blocks that can be put together in many different ways, to powerful effect. A canonical example from the **dplyr** docs uses some data on flight delays (Wickham, 2016). It groups flights by year, month, and day (essentially grouping by unique days without having to convert the date variables), then selects the arrival and departure delay variables, and finally summarizes both delays to show average arrival and departure delays per day. Results with average delays greater than 30 minutes in either arrival or departure are filtered out and shown. Notice that the `summarize()` call didn't need to specify that the averages be computed per day, because the data being piped in from before was already grouped by day, the summaries were done on those groups.

```
library(nycflights13)
library(dplyr)
flights %>%
  group_by(year, month, day) %>%
  select(arr_delay, dep_delay) %>%
  summarize(
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)
  ) %>%
  filter(arr > 30 | dep > 30)

## Source: local data frame [49 x 5]
## Groups: year, month [11]
##
##   year month   day      arr      dep
##   <int> <int> <int>   <dbl>   <dbl>
## 1  2013     1    16  34.24736  24.61287
## 2  2013     1    31  32.60285  28.65836
## 3  2013     2    11  36.29009  39.07360
## 4  2013     2    27  31.25249  37.76327
## 5  2013     3     8  85.86216  83.53692
## 6  2013     3    18  41.29189  30.11796
## 7  2013     4    10  38.41231  33.02368
## 8  2013     4    12  36.04814  34.83843
## 9  2013     4    18  36.02848  34.91536
## 10 2013     4    19  47.91170  46.12783
## # ... with 39 more rows
```

Some of these ‘moves’ are possible in CODAP already— you can summarize a variable, for example. But I think the chaining of operations is important for data science. I believe CODAP would allow a set of operations to be nested inside one another, but reading that sort of code is often difficult.

## How could CODAP support data transformations?

There are several possible levels here. One is to imagine a local maximum based on the functionality currently possible, and the other is to think truly blue-sky and go for the global maximum. Let's start with the local max. Data manipulations all seem to appear in as part of the ruler menu on the data table. All the functions are accessible through a menu. Assuming this is going to remain the same, let's consider what would make it more dynamic to work with data manipulations. Currently, once you have selected a function, you must know the syntax (no code completion, hovering hints, or sidebar documentation) and click the Apply button to get anything to happen. If you made a mistake, you have to either edit the formula for the attribute or delete it and start again. Essentially, this is a blind process. Bret Victor has thought a lot about the future of programming, and one of his tenets is that anything you do in the program should have an immediate result in the outcome (Victor, 2012). Applying this to CODAP, I think you should be able to see a live preview of what your code will do. Then, you can see if the operation you are applying does what you think it does before you hit apply.

If we are thinking bigger, we should begin by considering one of the strengths of CODAP: its visual nature. I love that when you click on the graph button, it creates a visual block that already has something going on in it. That block encourages you to go further with the text "Click here, or drag an attribute here." Could we imagine something similar for data manipulation?

If there was a tool palette (like that in Photoshop or KidPix) with a small number of data manipulation operations, that would reduce cognitive load on users by allowing them to outsource remembering what the operations were to the palette. There is some debate on how many things humans can hold in their short-term memory. Some say  $7 \pm 2$  (Miller, 1955), some exactly 4 (Cowan, 2000). Either way, it is a small number, and we want to provide as much support as possible. The palette could contain a button for summarize (for example), with a graphic image illustrating a summary. Since summaries are a many-to-one or many-to-few operation, it seems that there could be a general image for this. The RStudio cheatsheet on data manipulation provides some graphical examples (RStudio Team, 2015). Their image for summarizing can be seen in Figure 2. Another great source of inspiration is a shiny app (shiny is an way to create interactive web graphics from within R) to visualize data manipulations (Ribeiro, 2014).

Of course, `summarize()` doesn't just work on its own. The user must specify how they want their data summarized. Some common summaries are `mean()`, `max()`, and `count()`. I could imagine the palette having several levels (again, taking inspiration from KidPix) (McNamara, 2014; Media Leaders, 2014). If you are not familiar with KidPix, a screenshot of the program is shown in Figure 3.

Once someone had clicked on the summary button, they would be exposed to another palette (perhaps at the bottom of the screen) with a variety of common summaries and a ... to show more. If this could be done quickly, it would be nice to have a preview of each summary as the user hovered over the button.

These data manipulation verbs are typically many-to-many (for example, making a new variable in an existing dataset) or many-to-few/many-to-one (like summarizing a variable down to a mean or means for each of a number of groups). I haven't figured out how to make many-to-few operations create a new data table in CODAP (or, Fathom, but I think it must be possible there). Several times, I've done many-to-one operations that have added a new variable to my data that is just the same value repeated for every row. This is not desirable behavior.

## Summarise Data



**dplyr::summarise(iris, avg = mean(Sepal.Length))**

Summarise data into single row of values.

**dplyr::summarise\_each(iris, funs(mean))**

Apply summary function to each column.

**dplyr::count(iris, Species, wt = Sepal.Length)**

Count number of rows with each unique value of variable (with or without weights).



Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

**dplyr::first**

First value of a vector.

**min**

Minimum value in a vector.

Figure 2: Excerpt from RStudio cheatsheet on data wrangling

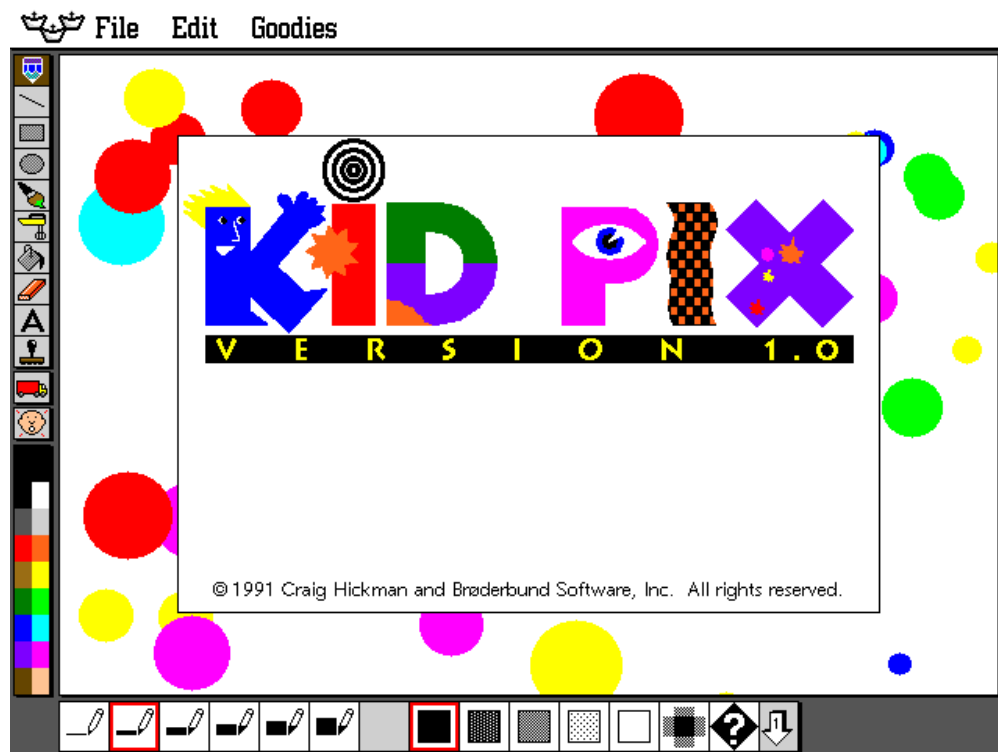


Figure 3: Screenshot of KidPix, via Wikipedia

## 5 Visualize

This is another area where CODAP is well on the way. Like I said earlier, the fact that the graph button doesn't create a completely empty window, but rather a draft of a plot with encouragement to edit, is so powerful. The linked brushing that comes for free is also a huge plus.

The room for improvement is in the implementation of more plot types. In general, I think there should be at least two ways to visualize any data. Because Fathom had many plot types, I'm guessing this is coming in CODAP, too.

## 6 Model

As with the visualize section, I'm guessing that modeling will be coming because it is in Fathom.

## 7 Communicate

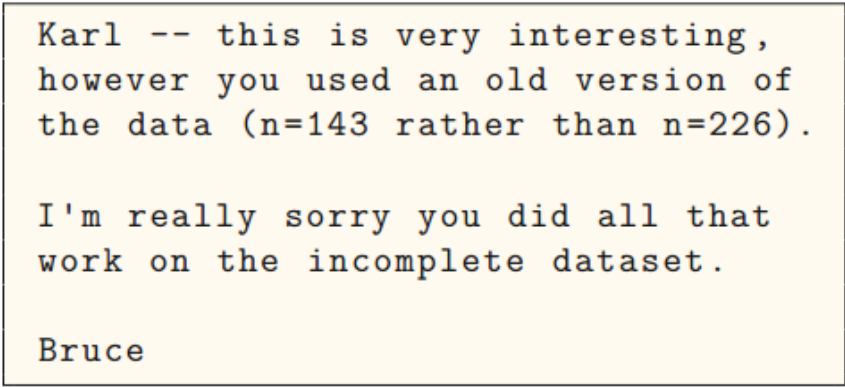
CODAP is solving some of the problems Fathom had with communication, particularly because CODAP is open and on the web. CODAP documents can be shared with people without them needing specialized software. I'm also excited about the idea that CODAP documents could be embedded in other pages (using iframes?). And, I like the ability to add text boxes to explain what is going on.

Even with these features, it is sometimes difficult for a reader to decipher what is going on when they open an existing CODAP document. They have to play with graphs to see what they are linked to, and generally begin interacting with the document (is that good or bad?). There is no way for them to see where an attribute came from without clicking to edit the formula, and if there are several datasets and several graphs, no visual link between the data and the graph(s) it created. I wonder if there could be another view on a document that would visualize the links between blocks (tables, graphs, models, etc). A data-flow view.

Because everything is being stored in JSON, I'm sure there is a way to see what all these connections are, but it is not user-friendly yet. Providing visual links between pieces would help support reproducibility.

Speaking of reproducibility, there are many layers to full reproducibility, but I'm stuck on the motivating use case illustrated in Figure 4, from a talk Karl Broman gave about reproducible research (Broman, 2016).

Because CODAP is already keeping track of links (to be able to provide reactive behavior) it seems like it should be possible to solve this issue, and I can imagine users actually using it. "Oh, this document does some cool maps, I wonder how my data would look in there?" and then being able to swap in a different dataset is very powerful. Again, a data-flow view could help facilitate this.



```
Karl -- this is very interesting,  
however you used an old version of  
the data (n=143 rather than n=226).  
  
I'm really sorry you did all that  
work on the incomplete dataset.  
  
Bruce
```

Figure 4: A motivation for reproducible research, via Karl Broman.

## 8 Concluding thoughts

If we are considering the components of data science to be import, tidy, transform, visualize, model, and communicate, CODAP is well on its way to facilitate the cycle. The largest area for improvement is in data transformation, which could particularly shine if the tools had great visual metaphors for the actions. In my dissertation, I came up with a list of attributes for a modern statistical computing tool, and one of them was “inherent visual documentation” (McNamara, 2015). Again, I think CODAP shines in this regard. Plots show you what they are going to do, they don’t start with a blank screen, and they are reactive. If CODAP had facilities for even simple data manipulations with inherent visual documentation, I think the case could be easily made to use CODAP as a tool for teaching data science.

## 9 Questions for discussion

Of course, this is all meant to be a provocation, to spark discussion. You may have come up with questions of your own while you were reading this, but if not, here are a few to get us started:

1. Should CODAP support the data-analytic cycle of the tidyverse? Is it appropriate for a tool for learning?
2. Are the elements of the cycle fully inclusive? Are there other components that have been left out?
3. Should CODAP provide all the data transformations of a professional tool like R?
4. How does variability get expressed through the entire process? One of the goals of statistical thinking is often to consider trends and variability, but sometimes making things more formal makes it harder to focus on that.
5. Who would use CODAP if it had these features? Are those the users we are targeting?
6. How could CODAP better support its users? What other already-strong characteristics can be translated to data science moves?

## References

- Broman, K. (2016). Steps toward reproducible research. [https://www.biostat.wisc.edu/~kbroman/presentations/repro\\_research\\_ChicagoASA2016\\_withnotes.pdf](https://www.biostat.wisc.edu/~kbroman/presentations/repro_research_ChicagoASA2016_withnotes.pdf).
- Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185.
- McNamara, A. (2014). Kid Pix. <http://www.science.smith.edu/~amcnamara/blog/tools/2014/10/01/KidPix.html>.
- McNamara, A. (2015). *Bridging the Gap Between Tools for Learning and for Doing Statistics*. PhD thesis, University of California, Los Angeles.
- Media Leaders (2014). Kid Pix paint program went from idea to mass success with Craig Hickman. <https://www.youtube.com/watch?v=LqVhK6JgbSU>.
- Miller, G. A. (1955). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 101(2):343–352.
- Ribeiro, B. B. (2014). Visualizing data manipulation operations. <http://rstudio.calvin.edu:3838/rpruim/data0ps/>.
- RStudio Team (2015). Data wrangling with dplyr and tidyr: Cheat sheet. <https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf>.
- Smith College (2016). The major in statistical & data sciences. <https://www.smith.edu/statistics/major.php>.
- Victor, B. (2012). Inventing on principle. <http://worrydream.com/#!/InventingOnPrinciple>.
- Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, 59(10).
- Wickham, H. (2016). Introduction to dplyr. <https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html>.