Interview with a Data Scientist (Hadley Wickham)

I recently interviewed Hadley Wickham (https://en.wikipedia.org/wiki/Hadley_Wickham) the creator of Ggplot2 and a famous R Stats person. He works for RStudio (https://www.rstudio.com) and his job is to work on Open Source software aimed at Data Geeks. Hadley is famous for his contributions to Data Science tooling and inspires a lot of other languages! I include some light edits.

1. What project have you worked on do you wish you could go back to, and do better? ggplot2! ggplot2 is such a heavily used package, and I knew so little about R programming when I wrote it. It works (mostly) and I love it, but the internals are pretty hairy. Even I don't understand how a lot of works these days. Fortunately, I've had the opportunity to spend some more time on it lately, as I've been working on the 2nd edition of the ggplot2 book.

One thing that I'm particularly excited about is adding an official extension mechanism, so that others can extend ggplot2 by creating their own geoms, stats etc. Some people have figured out how to do that already, but it's not documented and the current system causes hassles with CRAN. Winston (http://github.com/winston/r-source) and I have been working on this fairly heavily for the last couple of weeks, and it's also been good for us. Thinking about what other people will need to do to write a new geom has forced us to think carefully about how geoms should work, and has resulted in a lot of internal tidying up. That makes life more pleasant for us!

2. What advice do you have to younger analytics professionals and in particular PhD students in the Sciences? My general advice is that's better to be really good at one thing rather than pretty good at a few things. For example, I think you're better off spending your time mastering either R or python, rather than developing a middling understanding of both. In the short term, that will cost you some time (because there are something that are much easier to do in R than in python and vice versa), but in the long term you end up with a more powerful toolset that few others can match.

That said, you also need to have some surface knowledge of many other areas. You need to accept that there are many subjects that you *could* master if you put the time into them, but you don't have the time. For example, if you're interestied in data, I think it's really important that you have a working knowledge SQL. If you have any interest in working in industry, the chances are that your data will live in a database that speaks SQL, and if you don't know the basics you won't be useful.

Similarly, if you work a lot with text, you should learn regular expressions; and if you work with html or xml, you should learn xpath.

3. What do you wish you knew earlier about being a data scientist? To be honest, I'm not sure that I am a data scientist. I spend most of my time making tools for data scientists to use, rather than actually doing data science myself.

Obviously I do some data analysis, but its mostly exploratory and for fun. I do worry that I am out of the loop when it comes to users needs in Data Science.

4. How do you go about framing a data problem – in particular, how do you avoid spending too long, how do you manage expectations etc. How do you know what is good enough? Again, I'm not sure that I'm a particularly skilled data analyst, but I do do a little. Mostly it's on personal data, or data about R or RStudio that's of interest to me. I’m also on the look out for interesting datasets to use for teaching and in documentation and books (which is one of the reasons I make data (http://blog.rstudio.org/2014/07/23/new-data-packages/) packages (http://blog.rstudio.org/2014/07/23/new-data-packages/))

A couple of interesting datasets that I’ve been playing around with lately are on liquor sales in (https://gist.github.com/dannguyen/18ed71d3451d147af414) Iowa (https://gist.github.com/dannguyen/18ed71d3451d147af414) and parking violations in (https://www.opendataphilly.org/dataset/parking-violations) Philadelphia (https://www.opendataphilly.org/dataset/parking-violations) These datasets are particularly interesting to be because they’re almost 1 GB. I want to make sure that my tools scale to handle this volume of data on a commodity laptop.

Mostly I’m done with them once I get bored, which makes me a poor role model for practicing data scientists!

5. How do you respond when you hear the phrase ‘big data’? Big data is extremely overhyped and not terribly well defined. Many people think they have big data, when they actually don’t.

I think there are two particularly important transition points:

* From in-memory to disk. If your data fits in memory, it’s small data. And these days you can get 1 TB of ram, so even small data is big! Moving from in-memory to on-disk is an important transition because access speeds are so different. You can do quite naive computations on in-memory data and it’ll be fast enough. You need to plan (and index) much more with on-disk data

* From one computer to many computers. The next important threshold occurs when you data no longer fits on one disk on one computer. Moving to a distributed environment makes computation much more challenging because you don’t have all the data needed for a computation in one place. Designing distributed algorithms is much harder, and you’re fundamentally limited by the way the data is split up between computers.

I personally believe it’s impossible for one system to span from in-memory to on-disk to distributed. R is a fantastic environment for the rapid exploration of in-memory data, but there’s no elegant way to scale it to much larger datasets. Hadoop works well when you have thousands of computers, but is incredible slow on just one machine. Fortunately, I don’t think one system needs to solve all big data problems.

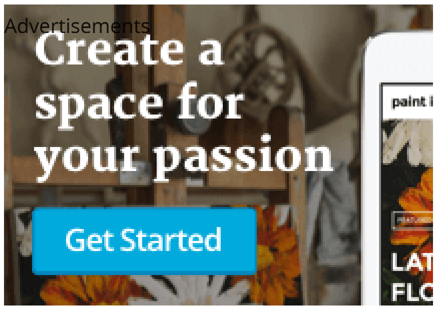
To me there are three main classes of problem:

1. Big data problems that are actually small data problems, once you have the right subset/sample/summary. Inventing numbers on the spot, I’d say 90% of big data problems fall into this category. To solve this problem you need a distributed database (like hive, impala, teradata etc), and a tool like dplyr to let you rapidly iterate to the right small dataset (which still might be gigabytes in size).

2. Big data problems that are actually lots and lots of small data problems, e.g. you need to fit one model per individual for thousands of individuals. I’d say ~9% of big data problems fall into this category. This sort of problem is known as a trivially parallelisable problem and you need some way to distribute computation over multiple machines. The foreach (https://cran.r-project.org/web/packages/foreach/) package is a nice solution to this problem because it abstracts away the backend, allowing you to focus on the computation, not the details of distributing it.

3. Finally, there are irretrievably big problems where you do need all the data, perhaps because you fitting a complex model. An example of this type of problem is recommender systems which really do benefit from lots of data because they need to recognise interactions that occur only rarely. These problems tend to be solved by

dedicated systems specifically designed to solve a particular problem.



AUGUST 2, 2015
ANALYTICS, ARTIFICIAL INTELLIGENCE, BIG DATA, DATA SCIENCE, DATA-MINING
LEAVE A COMMENT