What They Don't Tell You About Data Science 2: Data Analyst Roles Are Poison

Posted by nadbor Dec 10th, 2017 11:46 am

This is the second of a series of posts about things I wish someone had told me when I was first considering a career in data science. Part 1.

For the purposes of this post I define a data analyst as someone who uses tools like Excel and SQL to interrogate data to produce reports, plots, recommendations but crucially doesn't *deliver code*. If you work in online retail and create an algorithm recommending tiaras for pets - I call you a data scientist. If you query a database and discover that chihuahua owners prefer pink tiaras and share this finding with the advertising team - you are a data analyst.

Let me get one thing out of the way first: this post is not bashing analysts. **Of course** data analyst's work is useful and rewarding in its own right. And there is more demand for it (under various names) than there is for data science. But that is beside the point. The point is that a lot of people will tell you that taking a job as a data analyst is a good way to prepare for data science and that is a lie. In terms of transferable skills you may as well be working as a dentist.

Misconception 1: you can take a job as a data analyst and evolve it into data science as you become more experienced

A data analyst is not a larval stage of a data scientist. They are completely different species.

Data Analyst	Data Scientist
	Sits with engineers (but talks to the

Sits with the business	business)
Produces reports, presentations	Produces software

Interestingly, the part about sitting in a different place (often a different floor or a different building!) is the bigger obstacle to moving into data science. Independent of having or not having the right skills, a data analyst can't just up and start doing data science because they don't have the physical means to do it! They don't have:

- access to full production data
- access to tools to do something with that data (hadoop, spark, compute instances)
- access to code repositories

While those things can be eventually gotten hold of with enough perseverance, there are other deficits that even harder to make up for:

- lack of familiarity with the company's technological stack
- lack of mandate to make necessary changes to that stack/implement features etc.

This should be obvious to anyone who has ever worked in a big company. You don't simply walk into an software team and start making changes. It sometimes takes months of training for a new developer on the team make first real contribution. For an outsider from a different business unit to do it remotely is unheard of.

Misconception 2: data analysis is good training for data science

As a data analyst:

- you will not be gaining the most important kind of experience - in software engineering
- you will not be learning about modern machine learning/statistical

techniques either - because they are optimised for accuracy and efficiency, not interpretability (which is the analyst's concern)

You will on the other hand do:

- exploratory data analysis
- excel, SQL, maybe some one-off R and python scripts

So that doesn't sound all bad, right? Wrong.

I think a case can be made that the little technical work a data analysts do actually does more harm than good to their data science education. A data scientist and an analyst may be using some of the same tools, but what they do with them is very, very different.

Data analyst's code	Data Scientist's code
Manually operated sequence of scripts, clicking through GUIs etc.	Fully automated pipelines
Code that only you will ever see	Code that will be used and maintained by other people
One-off, throwaway scripts	Code that is a part of an live app or a scheduled pipeline
Code tweaked until it runs this one time	Code optimised for performance, maintainability and reusability

Doing things a certain way may make sense from a data analyst's perspective, but the needs of data science are different. When former analysts are thrown into data science projects and start applying the patterns they have developed through the years, the results are not pretty.

Horror story time

Let me illustrate with an example which I promise is not cherry-picked and a fairly typical in my experience.

I joined a project led by analysts-turned-data scientists. We were building prototype of a pipeline doing some machine learning on the client's data and displaying pretty plots. One of my first questions when I joined was: how are you getting your data from the client? (we needed a new batch of data at that time). The answer was:

1. Email X in Sweden with a query that he runs on the client's database. X downloads a csv with results and puts it on an ftp server.
2. Download the csv from ftp to your laptop.
3. Upload it to the server where we have Python.
4. Run a python script on the server to clean the data (the script is in Y's home directory).
5. Download the results on your laptop.
6. Upload results to our database through a GUI.
7. Run a SQL script in the GUI to join with our other tables (you will find the script in an attachement to some old email).
8. Download the results.
9. Upload to our dev MySQL database.
10. Run another SQL (Y has the script on her laptop).
11. Pull the data from MySQL into RStudio on the server.
12. Do actual analytics on the server in R (all of it consists of a single gigantic R script).

Needless to say, this workflow made it impossible to get anything done whatsoever. To even run the pipeline again on fresher data would take weeks (when it should be seconds) and the results were junk anyway because the technologies they used forced them to only use 1% of available data.

On top of that, every single script in the pipeline was extremely hacky and brittle - and here's why:

When faced with a task, an analyst would start writing code. If it doesn't work at first, they add to it and tweak it until it does. As soon as a result is

produced (a csv file usually), they move on to the next step. No effort is made to ensure reproducibility, reusability, maintainability, scalability. The piece of code gets you from A to B and that is that. A script made this way is full of hard-coded database passwords, paths to local directories, magic constants and untested assumptions about the input data. It resembles a late-game Jenga tower - weird and misshapen, with many blocks missing and others sticking out in weird directions. It is standing for now but you know that it will come crashing down if you as much as touch it.

The tragic part is that none of the people involved in this mess were dumb. No, they were smart and experienced, just not the right kind of experienced. This spaghetti of manual steps, hacky scripts and big data on old laptops is not the result of not enough cleverness. Way too much cleverness if anything. It's the result of intelligent people with no experience in making software realising too late that they're out of their depth.

If only my colleagues were completely non-technical - never having written a SAS or SQL script in their lives - they would have had to hire an engineer to do the coding and they themselves would have focused on preparing the spec. This kind of arrangement is not ideal but I guarantee that the result would have been much better. This is why I believe that the data analyst's experience is not just useless but actively harmful to data science.

Ultimately though the fault doesn't lie with the analysts but with the management for mismatching people and tasks. It's time managers understood that:

1. Data science is software engineering
2. Software engineering is hard
3. Software engineering community has developed tools and practices to make it less hard

4. You need a software professional to wield those tools
5. Having written a script in SAS doesn't make one a software professional

Closing remarks

In case I wasn't clear about this: I am emphatically **not** saying that analysts can't learn proper software engineering and data science. If miners can do it, so can analysts. It's just that an analyst's experience makes it harder for them (and their managers!) to realise that they are missing something and easier to get by without learning a thing.

If you're an analyst and want to switch to data science (And I'm not saying that you should! The world needs analysts too!) I recommend that you forget everything you have learned about coding and start over, like the miners.

If you're a grad considering a data analyst role as training for data science I strongly recommend that you find a junior software developer job instead. If you're lucky, you may get to do some machine learning and graduate into full-on data science. But even if not, practically everything you learn in an entry-level engineering position will make you a better data scientist when you finally become one.