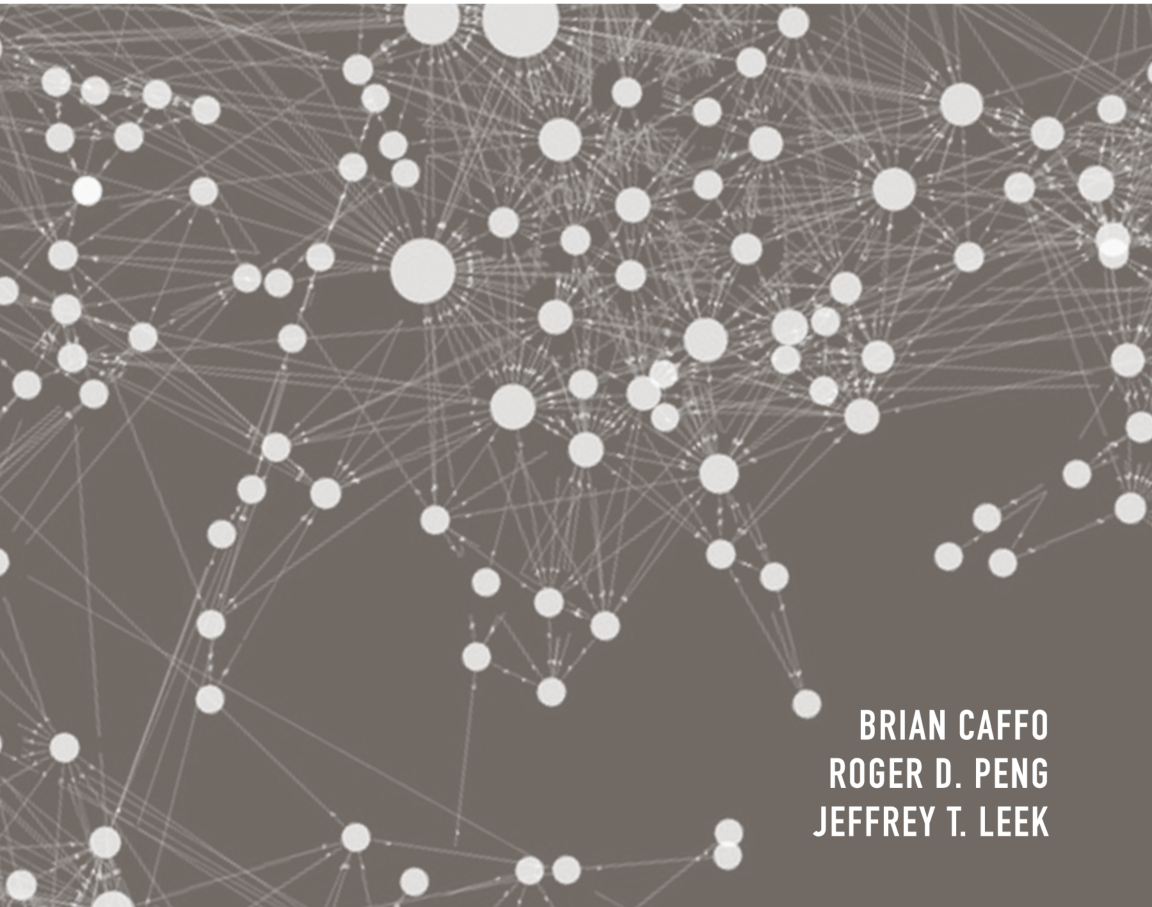
EXECUTIVE DATA SCIENCE

A GUIDE TO TRAINING AND MANAGING THE BEST DATA SCIENTISTS



BRIAN CAFFO
ROGER D. PENG
JEFFREY T. LEEK

Executive Data Science

A Guide to Training and Managing the Best Data Scientists

Brian Caffo, Roger D. Peng and Jeffrey Leek

This book is for sale at http://leanpub.com/eds

This version was published on 2018-12-12

ISBN 978-1-365-12197-5

Leanpub

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By These Authors

Books by Brian Caffo

Statistical inference for data science

Regression Models for Data Science in R

Advanced Linear Models for Data Science

Developing Data Products in R

Methods in Biostatistics with R

Books by Roger D. Peng

R Programming for Data Science

The Art of Data Science

Exploratory Data Analysis with R

Report Writing for Data Science in R

Advanced Statistical Computing

The Data Science Salon

Conversations On Data Science

Mastering Software Development in R

Putting It All Together: Essays on Data Analysis

Books by Jeffrey Leek

The Elements of Data Analytic Style

How to be a modern scientist

Material for Introduction to Chromebook Data Science

Material for Organizing Data Science Projects

Material for How to Use a Chromebook

Material for Google and the Cloud

Material for Version Control

Material for Introduction to R

Material for Data Tidying

Material for Data Visualization

Material for Getting data

Material for Data Analysis

Material for Written and Oral Communication in Data Science

Material for Getting a job in data science

Contents

1. **What is Data Science?** **1**
   Moneyball 2
   Voter Turnout 3
   Engineering Solutions 3

2. **What is Statistics Good For?** **5**

3. **What is Machine Learning?** **7**

4. **What is Software Engineering for Data Science?** **10**

5. **Structure of a Data Science Project** **14**

6. **Output of a Data Science Experiment** **18**

7. **Defining Success: Four Secrets of a Successful Data Science Experiment** **21**

8. **Data Science Toolbox** **23**

9. **Separating Hype from Value** **27**

1. What is Data Science?

Note: Some of the material in this section appeared on the Simply Statistics Blog

Because this is a book about data science, it might be reasonable to first ask, "What is data science?". Most people hyping data science have focused on the first word: *data*. They care about volume and velocity and whatever other buzzwords describe data that is too big for you to analyze in Excel. This hype about the size (relative or absolute) of the data being collected fed into the second category of hype: hype about *tools*. People threw around EC2, Hadoop, Pig, and had huge debates about Python versus R.

But the key word in data science is not "data"; it is *science*. Data science is only useful when the data are used to answer a question. That is the science part of the equation. The problem with this view of data science is that it is much harder than the view that focuses on data size or tools. It is much easier to calculate the size of a data set and say "My data are bigger than yours" or to say, "I can code in Hadoop, can you?" than to say, "I have this really hard question, can I answer it with my data?"

A few reasons it is harder to focus on the science than the data/tools are:

- John Tukey's quote: "The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data." You may have 100 Gb and only 3 Kb are useful for answering the real question you care about.

- When you start with the question you often discover that you need to collect new data or design an experiment to confirm you are getting the right answer.
- It is easy to discover structure or networks in a data set. There will always be correlations for a thousand reasons if you collect enough data. Understanding whether these correlations matter for specific, interesting questions is much harder.
- Often the structure you found on the first pass is due to a phenomeon (measurement error, artifacts, data processing) that isn't related to answer an interesting question.

The hype around big data/data science will flame out (it already is) if data science is only about "data" and not about science. The long term impact of data science will be measured by the scientific questions we can answer with the data.

Moneyball

One of the examples that you hear about a lot when you hear about data science is Moneyball. With Moneyball, the question was, can we build a winning baseball team if we have a really limited budget? They used quantification of player skills, and developed a new metric that's more useful to answer that question. But the key underlying question that they were asking, the key reason why this was a data science problem, was "Could we use the data that we collected to answer this specific question, which is *building a low budget baseball team*?"

Voter Turnout

A second question would be, “How do we find the people who vote for Barack Obama and make sure that those people end up at the polls on polling day?” And so this is an example from a study of Barack Obama’s data team, where they went and they actually tried to run experiments and analyze the data to identify those people. They ended up being a surprising group of people that weren’t necessarily the moderate voters that everybody thought they would be, that could be swayed to go out and vote for Barack Obama.

This is again an example where there was a high-level technical issue that had been used–A/B testing on websites and things like that–to collect and identify the data that they used to answer the question. But at the core, the data science question was “Can we use data to answer this question about voter turnout, to make sure a particular candidate wins an election.

Engineering Solutions

We’ve talked a lot about how data science is about answering questions with data. While that’s definitely true there are also some other components to the problem. Data science is involved in formulating quantitative questions, identifying the data that could be used to answer those questions, cleaning it, making it nice, then analyzing the data, whether that’s with machine learning, or with statistics, or with the latest neural network approaches. The final step involves communicating that answer to other people.

One component of this entire process that often gets left out in these discussions is the engineering component of it.n A good example of where the engineering component is critical came up with the Netflix prize. With the Netflix prize, Netflix had a whole bunch of teams competing to try to predict how best to show people what movies to watch next. The team that won blended together a large number of machine learning algorithms. But it turns out that's really computationally hard to do, and so Netflix never actually ended up implementing the winning solution on their system, because there wasn't enough computing power to do that at a scale where they could do it for all their customers.

In addition to the actual data science, the actual learning from data and discovering what the right prediction model is, there's the implementation component (often lumped into data engineering) which is how you actually implement or scale that technology to be able to apply it to, say, a large customer base or to a large number of people all at once.

There are trade-offs that always come up in data science. The trade-offs between interpretability and accuracy or interpretability and speed, or interpretability and scalability, and so forth. You can imagine that there are all these different components to a model: whether it's interpretable, simple, accurate, fast, and scalable. You have to make judgments about which of those things are important for the particular problem that you're trying to solve.

2. What is Statistics Good For?

Get the lecture notes used in the videos.

Statistics is the discipline of analyzing data. As such it intersects heavily with data science, machine learning and, of course, traditional statistical analysis. In this section, we orient you to statistics by covering a few key activities that define the field. These are:

- Descriptive statistics
- Inference
- Prediction
- Experimental Design

Descriptive statistics includes exploratory data analysis, unsupervised learning, clustering and basic data summaries. Descriptive statistics have many uses, most notably helping us get familiar with a data set. Descriptive statistics usually are the starting point for any analysis. Often, descriptive statistics help us arrive at hypotheses to be tested later with more formal inference.

Inference is the process of making conclusions about populations from samples. Inference includes most of the activities traditionally associated with statistics such as: estimation, confidence intervals, hypothesis tests and variability. Inference forces us to formally define targets of estimations or hypotheses. It forces us to think about the population that we're trying to generalize to from our sample.

Prediction overlaps quite a bit with inference, but modern prediction tends to have a different mindset. Prediction is the process of trying to guess an outcome given a set of realizations of the outcome and some predictors. Machine learning, regression, deep learning, boosting, random forests and logistic regression are all prediction algorithms. If the target of prediction is binary or categorical, prediction is often called classification. In modern prediction, emphasis shifts from building small, parsimonious, interpretable models to focusing on prediction performance, often estimated via cross validation. Generalizability is often given not by a sampling model, as in traditional inference, but by challenging the algorithm on novel datasets. Prediction has transformed many fields include e-commerce, marketing and financial forecasting.

Experimental design is the act of controlling your experimental process to optimize the chance of arriving at sound conclusions. The most notable example of experimental design is randomization. In randomization a treatment is randomized across experimental units to make treatment groups as comparable as possible. Clinical trials and A/B testing both employ randomization. In random sampling, one tries to randomly sample from a population of interest to get better generalizability of the results to the population. Many election polls try to get a random sample.

3. What is Machine Learning?

The lecture notes used for this section.

Machine learning has been a revolution in modern prediction and clustering. Machine learning has become an expansive field involving computer science, statistics and engineering. Some of the algorithms have their roots in artificial intelligence (like neural networks and deep learning).

For data scientists, we decompose two main activities of machine learning. (Of course, this list is non-exhaustive.) These are are

1. **Unsupervised learning** - trying to uncover unobserved factors in the data. It is called "unsupervised" as there is no gold standard outcome to judge against. Some example algorithms including hierarchical clustering, principal components analysis, factor analysis and k-means.
2. **Supervised learning** - using a collection of predictors, and some observed outcomes, to build an algorithm to predict the outcome when it is not observed. Some examples include: neural networks, random forests, boosting and support vector machines.

One famous early example of unsupervised clustering in the computation of the *g*-factor. This was postulated to be a measure of intrinsic intelligence. Early factor analytic models were used to cluster scores on psychometric questions to create the *g*-factor. Notice the lack of a gold standard outcome. There was no true measure of intrinsic intelligence to train an algorithm to predict it.

For supervised learning, an early example is the development of regression. In this, Francis Galton wanted to predict children's heights from their parents. He developed linear regression in the process. Notice that having several children with known adult heights along with their parents allows one to build the model, then apply it to parents who are expecting.

It is worth contrasting modern machine learning and prediction with more traditional statistics. Traditional statistics has a great deal of overlap with machine learning, including models that produce very good predictions and methods for clustering. However, there is much more of an emphasis in traditional statistics on modeling and inference, the problem of extending results to a population. Modern machine learning was somewhat of a revolution in statistics not only because of the performance of the algorithms for supervised and unsupervised problems, but also from a paradigm shift away from a focus on models and inference. Below we characterize some of these differences.

For this discussion, I would summarize (focusing on supervised learning) some characteristics of ML as:

- the emphasis on predictions;
- evaluating results via prediction performance;
- having concern for overfitting but not model complexity per se;
- emphasis on performance;
- obtaining generalizability through performance on novel datasets;

- usually no superpopulation model specified;
- concern over performance and robustness.

In contrast, I would characterize the typical characteristics of traditional statistics as:

- emphasizing superpopulation inference;
- focusing on a-priori hypotheses;
- preferring simpler models over complex ones (parsimony), even if the more complex models perform slightly better;
- emphasizing parameter interpretability;
- having statistical modeling or sampling assumptions that connect data to a population of interest;
- having concern over assumptions and robustness.

In recent years, the distinction between both fields have substantially faded. ML researchers have worked tirelessly to improve interpretations while statistical researchers have improved the prediction performance of their algorithms.

4. What is Software Engineering for Data Science?

Software is the generalization of a specific aspect of a data analysis. If specific parts of a data analysis require implementing or applying a number of procedures or tools together, software is the encompassing of all these tools into a specific module or procedure that can be repeatedly applied in a variety of settings. Software allows for the systematizing and the standardizing of a procedure, so that different people can use it and understand what it's going to do at any given time.

Software is useful because it formalizes and abstracts the functionality of a set of procedures or tools, by developing a well defined interface to the analysis. Software will have an interface, or a set of inputs and a set of outputs that are well understood. People can think about the inputs and the outputs without having to worry about the gory details of what's going on underneath. Now, they may be interested in those details, but the application of the software at any given setting will not necessarily depend on the knowledge of those details. Rather, the knowledge of the *interface* to that software is important to using it in any given situation.

For example, most statistical packages will have a linear regression function which has a very well defined interface. Typically, you'll have to input things like the outcome and the set of predictors, and maybe there will be some other inputs like the data set or weights. Most linear regression functions kind of work in that way. And importantly, the user does not have to know exactly how the linear regression calculation is done underneath the hood. Rather, they only need to know that they need to specify the outcome, the predictors, and a couple of other things. The linear regression function abstracts all the details that are required to implement linear regression, so that the user can apply the tool in a variety of settings.

There are three levels of software that are important to consider, going from kind of the simplest to the most abstract.

1. At the first level you might just have some code that you wrote, and you might want to encapsulate the automation of a set of procedures using a loop (or something similar) that repeats an operation multiple times.
2. The next step might be some sort of function. Regardless of what language you may be using, often there will be some notion of a function, which is used to encapsulate a set of instructions. And the key thing about a function is that you'll have to define some sort of interface, which will be the inputs to the function. The function may also have a set of outputs or it may have some side effect for example, if it's a plotting function. Now the user only needs to know those inputs and what the outputs will be. This is the first level of abstraction that you might encounter, where you have to actually define and interface to that function.
3. The highest level is an actual software package, which will often be a collection of functions and other things. That will be a little bit more formal because there'll be a very specific interface or API that a user has to understand. Often for a software package there'll

be a number of convenience features for users, like documentation, examples, or tutorials that may come with it, to help the user apply the software to many different settings. A full on software package will be most general in the sense that it should be applicable to more than one setting.

One question that you'll find yourself asking, is at what point do you need to systematize common tasks and procedures across projects versus recreating code or writing new code from scratch on every new project? It depends on a variety of factors and answering this question may require communication within your team, and with people outside of your team. You may need to develop an understanding of how often a given process is repeated, or how often a given type of data analysis is done, in order to weigh the costs and benefits of investing in developing a software package or something similar.

Within your team, you may want to ask yourself, "Is the data analysis you're going to do something that you are going to build upon for future work, or is it just going to be a one shot deal?" In our experience, there are relatively few one shot deals out there. Often you will have to do a certain analysis more than once, twice, or even three times, at which point you've reached the threshold where you want to write some code, write some software, or at least a function. The point at which you need to systematize a given set of procedures is going to be sooner than you think it is. The initial investment for developing more formal software will be higher, of course, but that will likely pay off in time savings down the road.

A basic rule of thumb is

- If you're going to do something **once** (that does happen on occasion), just write some code and document it very well. The important thing is that you want to make sure that you understand what the code does, and so that requires both writing the code well and writing documentation. You want to be able to reproduce it down later on if you ever come back to it, or if someone else comes back to it.

- If you're going to do something **twice**, write a function. This allows you to abstract a small piece of code, and it forces you to define an interface, so you have well defined inputs and outputs.
- If you're going to do something **three** times or more, you should think about writing a small package. It doesn't have to be commercial level software, but a small package which encapsulates the set of operations that you're going to be doing in a given analysis. It's also important to write some real documentation so that people can understand what's supposed to be going on, and can apply the software to a different situation if they have to.

5. Structure of a Data Science Project

A typical data science project will be structured in a few different phases. There's roughly five different phases that we can think about in a data science project.

The first phase is the most important phase, and that's the phase where you ask the question and you specify what is it that you're interested in learning from data. Now, specifying the question and kind of refining it over time is really important because it will ultimately guide the data that you obtain and the type of analysis that you do. Part of specifying the question is also determining the type of question that you are gonna be asking. There are six types of questions that you can ask going from kind of descriptive, to exploratory, to inferential, to causal, to prediction, predictive and mechanistic. And so figuring out what type of question you're asking and what exactly is the type of question is really influential. You should spend a lot of time thinking about this.

Once you've kind of figured out what your question is, but typically you'll get some data. Now, either you'll have the data or you'll have to go out and get it somewhere or maybe someone will provide it to you, but the data will come to you. The the next phase will be exploratory data analysis. So this is the second part, there are two main goals to exploratory data analysis. The first is you want to know *if the data that you have is suitable for answering the question that you have*. This will depend on a variety of questions ranging from basic ones—"Is there enough data?", "Are there too many missing values?"—to more fundamental ones, like are you missing certain variables or do you need to collect more data to get those variables, etc?

The second goal of exploratory data analysis is to start to develop a sketch of the solution. If the data are appropriate for answering your question, you can start using it to sketch out what the answer might be to get a sense of what it might look like. This can be done without any formal modeling or any kind of the statistical testing.

The next stage, the third stage, is formal modeling. If your sketch seems to work, you've got the right data and it seems appropriate to move on, the formal modeling phase is the way to specifically write down what questions you're asking and to lay out what parameters you're trying to estimate. It also provides a framework for challenging your results. Just because you've come up with an answer in the exploratory data analysis phase doesn't mean that it's necessarily going to be the right answer and you need to be able to challenge your results to examine their sensitivity to different assumptions. Challenging your model and developing a formal framework is really important to making sure that you can develop robust evidence for answering your question.

Once you've done your analysis and your formal modeling you want to think about how to interpret your results. There are a variety of things to think about in the interpretation phase the data science project. The first is to think about how your results comport with what you expected to find when you where first asking the question (before you had data). You also want to think about totality of the evidence that you've developed. You've probably done many different analyses, you probably fit many different models. And so you have many different bits of information to think about. Part of the challenge of the interpretation phase is to assemble all of the information and weigh each of the different pieces of evidence. You know which pieces are more reliable, which are are more uncertain than others, and which more important than others to get a sense of the totality of evidence with respect to answering the question.

The last phase is the communication phase. Any data science project that is successful will want to communicate its findings to some sort of audience. That audience may be internal to your organization, it may be external, it may be to a large audience or even just a few people. But communicating your findings is an essential part of data science because it informs the data analysis process and a it translates your findings into action.

So that's the last part (which is not necessarily a formal part of a data science project)—often there will be some *decision* that needs to be made or some action that needs to be taken. And the data science project will have been conducted in support of making a decision or taking an action. That last phase will depend on more than just the results of the data analysis, but may require inputs from many different parts of an organization or from other stakeholders. Ultimately if a decision is made, the data analysis will inform that decision and the evidence that was collected will support that decision.

To summarize, the five phases of a data science project are

1. Question
2. Exploratory data analysis
3. Formal modeling
4. Interpretation
5. Communication.

Now, there is another approach that can be taken, it's very often taken in a data science project. This approach starts with the data and an exploratory data analysis. Often there will be a data set already available, but it won't be immediately clear what the data set will be useful for. So it can be useful to do some exploratory data analysis, to look at the data, to summarize it a little bit, make some plots, and see what's there. This process can you lead you to generate interesting questions based on the data. This process is sometimes called hypothesis generating because the goal is to produce questions as opposed to answers. Once you've produced the questions that you want to ask it may be useful to get more data or other data to do an exploratory data analysis that's more specific to your question. After that, you can continue with the formal modeling, interpretation and communication.

One thing that you have to be wary of is to do the exploratory data analysis in one data set, develop the question, and then go back to the *same* data set, pretending like you hadn't done the exploratory data analysis before. This is often be a recipe for bias in your analysis because the results were derived from the same dataset that was used to generate the question. It's important to be careful about not doing that and to always try to obtain other independent data when you're using the data to generate the questions in the first place. That said, this approach to data science can be very useful and can often result in many interesting questions.

6. Output of a Data Science Experiment

The lecture notes can be found here.

The potential set of outputs of a data science experiment are pretty much limitless. However, four general types of outputs pop up most frequently. Those are:

- Reports
- Presentations
- Interactive web pages
- Apps

(Interactive graphics are important enough to merit their own category. However, they're usually embedded in a web page, so I'm lumping them in there.) Let's discuss each of the categories in turn.

Reports are easily the most common output of a data science experiment. Since the goals of reports varies wildly across settings, let's simply discuss a few hallmarks of a good data science report. It should:

- Be clearly written
- Involve a narrative around the data
- Discuss the creation of the analytic dataset
- Have concise conclusions
- Omit unnecessary details
- Be reproducible

By and large, these points are obvious. However, this latter point is one that we discuss a lot throughout the specialization. Reproducible reports have mechanisms under the hood to recreate the data analysis. The number of benefits of report writing in this fashion are many. They include: getting the data scientist to think about the output of the process (since the code is embedded in the eventual output), very clear documentation that extends beyond code commenting, automation of the report generation process and then, of course, reproducibility. The main tools for producing reproducible reports are kntir and ipython notebooks. You should create a culture of using these tools, or similar ones, in your organization, as they will dramatically improve reproducibility.

Oddly enough, the same rules apply to presentations, though reproducible presentation software is less well adopted. For R, there's slidify and rStudio's presenter. These tools automate presentations in the same way that knitr and ipython notebooks automate report generation.

Interactive web pages and apps are similar enough to combined in the discussion. Again, as the requirements will vary so much across applications, we will only discuss a few hallmarks of good output. These include:

- Good ease of use / design
- Good documentation
- Code commented
- Code version controlled

Good ease of use and design are a discipline unto themselves. Since your data scientists are likely also not software engineers or designers, their design is probably not going to be optimal. However, modern tools allow data scientists to prototype apps and interactive web pages quickly and relatively easily. Your data scientists should then pay some attention to ease use and design. Good documentation is mostly effort, though the same caveats apply with design.

On the other hand, having well commented code and version control should be standard practice for data scientists. Well commented code is easy to return to for new employees or the original coder returning to the project after a long hiatus. Version control is similar good practice. With version control, data scientists will be able to return to any checked in version of the project. The comments are useful for documenting the evolution of the project as well. Tools such as git and github make version control easy and are in the standard toolkit of data scientists.

7. Defining Success: Four Secrets of a Successful Data Science Experiment

Here's a link to the lecture notes.

Defining success is a crucial part of managing a data science experiment. Of course, success is often context specific. However, some aspects of success are general enough to merit discussion. My list of hallmarks of success includes:

1. New knowledge is created.
2. Decisions or policies are made based on the outcome of the experiment.
3. A report, presentation or app with impact is created.
4. It is learned that the data can't answer the question being asked of it.

Some more negative outcomes include: decisions being made that disregard clear evidence from the data, equivocal results that do not shed light in one direction or another, uncertainty prevents new knowledge from being created.

Let's discuss some of the successful outcomes first.

New knowledge seems ideal to me (especially since I'm an academic). However, new knowledge doesn't necessarily mean that it's important. If it produces actionable decisions or policies, that's even better. (Wouldn't it be great if there was an evidence-based policy like the evidence-based medicine movement that has transformed medicine.) That our data science products have great (positive) impact is of course ideal. Creating reusable code or apps is great way to increase the impact of a project.

Finally, the last point is perhaps the most controversial. I view it as a success if we can show that the data can't answer the questions being asked. I am reminded of a friend who told a story of the company he worked at. They hired many expensive prediction consultants to help use their data to inform pricing. However, the prediction results weren't helping. They were able to prove that the data couldn't answer the hypothesis under study. There was too much noise and the measurements just weren't accurately measuring what was needed. Sure, the result wasn't optimal, as they still needed to know how to price things, but it did save money on consultants. I have since heard this story repeated nearly identically by friends in different industries.

8. Data Science Toolbox

The data scientist toolbox is the collection of tools that are used to store, process, analyze and communicate results of data science experiments. Data are typically stored in a database. For a smaller organization that might be single, small MySQL database. And for a large organization, it might be a large distributed database across many servers. Usually, most of the analysis that takes place and most of the production doesn't actually happen in the database—it happens elsewhere. You usually have to use another programming language to pull the data out of that database to analyze it.

There are two common languages for analyzing data. The first one is the R programming language. R is a statistical programming language that allows you to pull data out of a database, analyze it, and produce visualizations very quickly. The other major programming language that's used for this type of analysis is Python. Python is another similar language that allows you to pull data out of databases, analyze and manipulate it, visualize it, and connected to downstream production.

The other thing that you need to do to be able to use these languages is some kind of computing infrastructure that you can use to run those programming languages on. You have the database, which stores the data, and then you have the servers which you will use to analyze the data. One useful example is Amazon Web Services. This is a set of computing resources that you can actually rent from Amazon, so many organizations that do data analysis actually just directly rent their computing resources rather than buy and manage their own. This is particularly true for small organizations that don't have a large IT budget.

Once you've actually done some low-level analysis and maybe made some discoveries or done some experiments, and decided actually how you're going to use data to make decisions for your organization you might want to scale those solutions up. There's a large number of analysis tools that can be used to provide more scalable analyses of datasets, whether that's in a database or by pulling the data out of the database. So two of the most popular right now are the Hadoop framework and the Spark framework. And both of these are basically ways to analyze, at a very large scale, data sets. Now it is possible to do interactive analysis with both of these, particularly with Spark. But it's a little bit more complicated and little bit more expensive, especially if you're applying it to large sets of data. Therefore, it's very typical in the data science process to take the database, pull out a small sample of the data, process it and analyse it in R or Python and then go back to the engineering team and scale it back up with Hadoop, or Spark, or other tools like that.

The next tool in the data scientist toolbox is actually communication. A data scientist or a data engineer has a job that's typically changing quite rapidly as new packages and new sort of tools become available, and so the quickest way to keep them up to speed is to have quick communication and to have an open channel of communication. A lot of data science teams use tools like Slack to communicate with each other, to basically be able to post new results, new ideas, and be able to communicate about what the latest packages are available.

There are a large number of help websites like Stack Overflow, that allow people go out and search for the questions that they need to answer. Even if they're quite technical, and quite detailed, it's possible to get answers relatively quickly. And that allows people to keep the process moving, even though the technology is changing quite rapidly.

Once the analysis is done and you want to share it with other people in your organization, you need to do that with reproducible or literate documentation. What does that mean? It basically means a way to integrate the analysis code and the figures and the plots that have been created by the data scientist with plain text that can be used to explain what's going on. One example is the R Markdown framework. Another example is iPython notebooks. These are ways to make your analysis reproducible and make it possible that if one data scientist runs an analysis and they want to hand it off to someone else, they'll be able to easily do that.

You also need to be able to visualize the results of the data science experiment. So the end product is often some kind of data visualization or interactive data experience. There are a large number of tools that are available to build those sorts of interactive experiences and visualizations because at the end user of a data science product is very often not a data scientist themselves. It's often a manager or an executive who needs to handle that data, understand what's happening with it, and make a decision. One such tool is Shiny, which is a way to build data products that you can share with people who don't necessarily have a lot of data science experience.

Finally, most of the time when you do a science data experiment, you don't do it in isolation—you want to communicate your results to other people. People frequently make data presentations, whether that's the data science manager, the data engineer, or the data scientist themselves, that explains how they actually performed that data science experiment. What are the techniques that they used, what are the caveats, and how can their analysis be applied to the data to make a decision.

9. Separating Hype from Value

You've probably heard a lot about data science and big data. Frankly, there's been a lot of hype around these areas. What that's done is it has inflated expectations about what data science and data can actually accomplish. Overall that has been a net negative for the field of data science and for big data. It's useful to think for just a bit about what are the questions you can ask to separate *data science hype* from *data science hope*.

The first question is always "What is the question you are trying to answer with the data?" If someone comes to talk to you about a big data project, or a data science project, and they start talking about the hottest new cool technology they can use to do distributed computing, and analyze data with machine learning and they throw a bunch of buzz words at you, the very first question you should ask is "What is the question you're trying to answer with data?" Because that really narrows the question down and it filters out a lot of the hype around what tools and technologies people are using which can often be very interesting and fun to talk about. We like to talk about them too, but they're not really going to add value to your organization all by themselves.

The second question to ask is, once you've identified what that question is you're trying to answer with the data, is "Do you have the data to actually answer that question?" So often the question that you want to answer and the data that you have to answer it with are not actually very compatible with each other. And so you have to ask yourself "Can we get the data in a shape where we can actually answer the question we want to answer?" Sometimes the answer is just no, in which case, you have to give up (for now). Bottom line—if you want to decide on whether a project is hype or whether it's reality, you need to decide whether the data the people are trying to use are actually relevant for the question they are trying to answer.

The third thing you need to ask is "If you could answer the question with the data you have, could you even use the answer in a meaningful way?" This question goes back to that idea from the Netflix prize where there was a solution to the problem of predicting which videos people would like to watch. And it was a very very good solution but it wasn't a solution that could be implemented with the computing resources that Netflix had in a way that was financially expedient. Even though they could answer the question, even though they did have the right data, even though they were answering a specific question, they couldn't actually apply the results of what they figured out.

If you ask these three questions, you can very quickly decipher whether a data science project is about hype or whether it's about a real contribution that can really move your organisation forward.