Lecture notes:

Reproducibility:https://docs.google.com/presentation/d/1hrzer7bpGn8jzN9QlPEfgq3LhiVh25XEmiJuCYOedNk/edit?usp=sharing

Version control: https://docs.google.com/presentation/d/1jfLVMX-OK8u4T4pzT5mT4pJ75g3yhng1G-rS40Md1sM/edit?usp=sharing

In this lecture we consider the products of the analysis. This is usually some sort of report or presentation. If the analysis is far enough along, then it might be an app or web page. We have a coursera class exactly on topic of the technical development of data products https://www.coursera.org/course/devdataprod

If the product is a report, ideally, it would clear and concisely written, with a nice narrative and interesting results. However, the needs of data science managers are variable enough that we focus on two components that make for good final products that are ubiquitous across all settings. These are making the report **reproducible** and making the report and code **version controlled**.

Analysis reproducible considers the fact that if we ask people to replicate their own analysis, let alone someone else's, they often get different results, sometimes, very different. We have a coursera class on making reproducible reports https://www.coursera.org/course/repdata The benefits of using the right tools for reproducible reports are many. They include dramatically helping achieve the goal of reproducibility, but also: helping organize ones thinking by blending the code and the narrative into a single document, they help document the code in a way that commenting doesn't achieve and they help automate the report writing process.

Thus if you're managing data scientist, we suggest knitr http://yihui.name/knitr/ and ipython notebooks http://ipython.org/notebook.html. These tools knit the analysis and the report together. And they're *easy* to learn.

As for the content of the report, some other recommendations that come to mind.

Check the signs, magnitudes and units. Checking the signs means checking that your effects are in the direction that you expect. It's also helps enforce asking the people that you manage to do more than just reporting coefficients. Checking the magnitudes by comparison with other known effects (covered in an earlier lecture) is a really good thing to encourage. Finally, put units on everything (graph axes, coefficients, means, ...) And make sure to understand the units. I can't tell you how many times this small step has helped.

It's important to get the data scientists and analysts that you mange out of technical jargon speak and into interpretation and interpretability. For the former, I keep harping on this idea of comparison with other known effects. Secondly, I encourage the idea of **effect critiquing**. This is the idea of, instead of getting excited about an effect, become its biggest critic. Try to figure out every possible reason why it could be spurious. This almost always leads to new analysis that should be conducted. Finally, for interpretability, encourage parsimonious models with interpretable parameters. That is, place a premium on simplicity. This is, of course, if you're more interested in a data science experiment where you are trying to create new parsimonious knowledge. The situation changes if one is trying to only improve prediction (see the first lecture).

Finally, **version control** is just general good practice. Version control is the process of keeping versions of your software, code, data and reports. Modern version control software makes this process easy. Using good version control, the project can be reverted to any previous step, and all of the steps are commented. Tools like **git** make this possible in massively distributed settings where lots of people are working on the same project. Git is the version control software, and a **git repository** is the actual version control database. Collaboration on a git

repository occurs on a server. Fortunately, there are tons of hosted git servers out there for general and business use. The biggest one is github https://github.com/ though others like bitbucket are also great https://bitbucket.org/ . These offer nice web interfaces to the server as well. There's an entire free book on git to use https://git-scm.com/book/en/v2 and a million tutorials online.

Of course, git is one of many version control systems. The main point is recommend (or demand/force) a version control culture in your organization.