Lessons from between the white lines for isolated data scientists

Benjamin S. Baumer
Program in Statistical and Data Sciences, Smith College

Abstract

Many current and future data scientists will be "isolated"—working alone or in small teams within a larger organization. This isolation brings certain challenges as well as freedoms. Drawing on my considerable experience both working in the professional sports industry and teaching in academia, I discuss troubled waters likely to be encountered by newly-minted data scientists, and offer advice about how to navigate them. Neither the issues raised nor the advice given are particular to sports, and should be applicable to a wide range of knowledge domains.

Keywords: applications and case studies, sports statistics, statistical computing, education, other

1 Introduction

Remember *Moneyball* (Lewis, 2004)? The book (and the movie) is a rags-to-riches story of how the Oakland A's and their contrarian general manager, Billy Beane, used analytics to win the most games in the American League with one of the smallest payrolls. While there are several holes in this story (Baumer and Zimbalist, 2014), it is undeniable that the influx of what we now call *data science* has brought dramatic changes to the sports industry. Most notably, every team in Major League Baseball now has an analytics department within baseball operations (Baumer, 2015), as do most teams in the National Basketball Association, many teams in the National Hockey League, some teams in the National Football League, and many other sports franchises in leagues across the world.

Moneyball was written about the 2002 season and published in 2003. I was hired by the New York Mets to be their first full-time analyst in January 2004. My title was "Statistical Analyst," but that was only because the title "Data Scientist" wasn't in the common parlance at that time—that was my job. After completing my Ph.D. in 2012 I left the Mets to join the faculty at Smith College. Like many faculty who have spent time working in industry, my experiences there have informed my teaching, and in turn, our curriculum has shifted to include more data science topics, courses, and even a new major. Having spent significant time both in the sports analytics industry (8.5 years) and now in academia (5 years), I know a number of things now that I wish I knew then, and similarly many things that I learned then continue to be useful now. This article is my attempt to synthesize that material into coherent advice for newly-minted or aspiring data scientists. While my reference application domain is baseball, this advice should be applicable across a wide array of industries. My advice will be particularly relevant to those working alone or in small teams of data scientists for organizations that are not in the data science business themselves—I will call these people *isolated data scientists*[1]. All of this advice is consistent with the principles explicated in American Statistical Association Undergraduate Guidelines Workgroup (2014) and De Veaux et al. (2017).

[1]There is a well-read *Isolated Statistician* mailing list for statisticians in industry and academia in similar positions.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

Aside: The use of baseball as an accessible setting for data science is neither original nor coincidental. Efron and Morris (1975) use the problem of predicting batting averages in baseball as their motivation for the use of Stein's estimator. Albert (2003)'s idea of teaching statistics using baseball remains influential. More recently, Donoho (2015) suggests that using the data science procedures described in Marchi and Albert (2013) to explore and verify the sabermetric arguments made in Tango et al. (2007) would be an exemplary learning exercise for data scientists of all flavors.

2 Creating a sustainable, efficient workflow

As a professional data scientist, whether you join a cutting-edge team at a major tech firm, or become the sole data analyst for a non-profit or Mom-and-Pop shop, creating a sustainable, efficient workflow will be critical to maintaining productivity, and your sanity. Every data scientist will face different constraints, but the following concepts should inform your choices.

2.1 Reproducibility

The ongoing reproducibility and replicability crisis in science is well-documented (Marwick et al., 2017), but even if you aren't publishing your research, there are strong incentives to adopt a reproducible workflow. Even from a purely selfish point-of-view, you will be forced to revisit projects that you completed—by yourself—months, and even years ago. Most likely you will not be able to remember why, or even how, you made each of the many choices you made during that process. Where did you put the original data? How and why did you recode that variable? Where is the authoritative documentation for these data? Why did you choose that model? How did you make that plot? The frustration of not being able to reconstruct work that you clearly did yourself is shared among data scientists, but needn't be an ongoing part of your career. Adopting a reproducible workflow provides a sensible path forward.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

A reproducible workflow is one in which each step of the analytical process is clearly documented in such a way that someone—and here it is better to imagine that person is **not** you—can retrace your steps and verify the exact results that you presented. Since your work necessarily involves computing, that means that your computing workflow needs to be reproducible, and this immediately necessitates *scriptable* programs, as opposed to point-and-click, menu-driven software. There are many reasons not to use spreadsheet software (e.g. Microsoft Excel) (see Broman and Woo (2017) for guidelines on how to use them responsibly), but chief among them is the fact that spreadsheet operations cannot be scripted. This means that it is generally impossible to produce truly reproducible work in that environment. A fundamental problem with spreadsheets is that they fail to distinguish between the *data* and the presentation of those data. In a spreadsheet, everything is fair game: data can be overwritten or reformatted in a way that destroys the original precision, or simply garbled by automatic type conversion tools. Limitations imposed on the number of "Undo" commands further restricts one's ability to retrace steps.

If a program is scriptable, then the precise sequence of commands that load and transform the data, perform the analysis, and produce the plots can be recorded. This needn't be a history or transcript of the entire session, but rather should be the *minimal* set of commands needed to reproduce your analysis. For even the most thoughtful programmer, a complete history will contain many irrelevant or incorrect commands that are not necessarily recorded in a sensible order. What reproducibility demands is a carefully edited recipe. Tools such as **rmarkdown** (Allaire et al., 2016) and **knitr** (Xie, 2015) have made achieving reproducibility in R quite painless.

Back in the aughts when I was working for the Mets, I used to send my reports to my colleagues and boss via email (which was through Outlook). This had the advantage of being a plain-text editor that allowed me to avoid using Word and its aggravating formatting rules, but had the disadvantage of resulting in a very plain presentation. Modern tools such as **rmarkdown** provide a perfect solution: plain-text authoring and rich HTML output.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

2.2 Collaboration

While you likely won't be working in total isolation, there may be times when you work on a project alone. In either case, use a formal *version control system* like **git** to manage the development of your work. Informal version control systems (e.g., renaming a file every time you change it), can provide useful backups and allow you to go back to a previous version, but are difficult to maintain and navigate (Bryan, 2017). Furthermore, front-ends such as GitHub[2] allow you to collaborate with yourself or others on multiple projects, while keeping track not only of changes to files, but also debates about the issues that arise and the choices you make to resolve them. Having milestones and releases, even for software that will only be used internally, will bring professionalism to your work.

2.3 Clone, fork, and extend modular packages

Recognize that you are not the greatest programmer in the world. In fact, you are a *data scientist* and not a programmer! You can do much more (and faster) by making use of software that other people have written—so use it liberally. Before writing any code, search for a suitable tool that is already out there. If it is on GitHub, fork it, contribute, and send pull requests upstream to bring your changes into the main branch. There are many good reasons for doing this: first, it makes you a good citizen in this open-source community that has given so much to so many; second, by bringing your changes into the main branch you relieve yourself of the burden of having to maintain the code base; third, it creates a public record of your contributions that you can show off at your next job interview.

In general, you want to write as few lines of (readable) code as possible. The main reason for this is that every line of code that you write becomes a line of code that you have to maintain, and you are not in the code maintenance business. On the other hand, if you have features that you want, and are willing and able to write the code, then you can push upstream and get the maintenance off of your plate.

Your code should focus on connecting existing, well-established tools in ways that serve

[2]Please see GitHub Help for a list of resources for learning how to use git and GitHub.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

your purpose—not on developing new, large pieces of software that millions of people will use. [If that is your goal, then you should be a software developer, not a data scientist.]

2.4 ETL and medium data

For many isolated data scientists, maintaining a database that changes daily (or monthly) provides an important mechanism for an ongoing project. In baseball, games are played every day, and so part of my job was to maintain several databases that required daily updating. As the number of data sources grows, the complexity of maintenance grows. Given the frequency with which these tasks must be completed, automating them is a powerful time-saver.

The process of converting data from its raw form into something that can be inserted into a database is called **Extract-Transform-Load (ETL)** and it can occupy the lion's share of your time. It is a worthwhile endeavor—keeping your data in a well-maintained database is an appropriate solution, and it will allow you to share your data with your colleagues in a convenient and safe manner (see below). Whether you are working with a Spark cluster for big data, or a SQL installation for medium data, a well-documented, efficient workflow can save many hours of your time.

A typical solution to this problem might look something like this: raw data comes from your vendor into an FTP site each morning. You write a shell (or Python) script that downloads the data and transforms it into a (series of) CSV(s). You write an import script that imports the data into your database. You set up a **cron** job on your server to run this process automatically. This can work well, but it is a bit clunky since you've written several bits of code in several languages stored in several files. More importantly, your solution is likely platform-dependent, since your shell script won't run on Windows. It's also likely to be idiosyncratic, since it's a one-off solution, and it will likely be difficult for someone else to use or maintain in the long run.

A slightly different solution that I have been working on is to write an R package to do all of the above. The **etl** package provides a framework—a "grammar", generously—for

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

performing ETL operations. It abstracts the ETL process into four "verbs" (Wickham and Francois, 2016): `etl_extract()`, `etl_transform()`, `etl_load()`, and `etl_init()`. These are S3 generic functions that can be chained to form a pipeline (Robinson et al., 2017), making it easy to perform these operations either in sequence or in isolation. Since the code is packaged for R, it is cross-platform, self-contained, easily portable, and can be well-documented and properly-versioned. Using this unified framework allows one to maintain separate codebases for each data source, while providing a common user experience and underlying functionality.

3 Hacking your budget

If you're lucky, you have a budget, but even if you don't, your manager does. In the previous section, I discussed how to make efficient use of your time. Here, I discuss how to use your financial resources effectively.

3.1 Use open source software

The benefits of open source software are well-documented, and there is no need to use this space for a polemic about why you might prefer a GNU/Linux operating system to one licensed from Microsoft and Apple. There are two very practical reasons for using open source software: 1) it is likely to be more robust, extensible, modular, and secure than its proprietary competitors; and 2) it's free. When working inside a budget, the importance of the latter cannot be overstated. By replacing Windows with Ubuntu, SAS with R, Oracle with MySQL/PostgreSQL, and .NET with PHP, you will save your department tens of thousands of dollars in licensing fees per year.

Others in your organization may try to spread fear, uncertainty, and doubt about the quality of open source software. They will try to tell you that "you get what you pay for." Do not believe them. These people are most often just afraid of the trade-off that comes with using open source software: while it may be well-documented, it usually doesn't come with

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

"support." This means that you can't just pick up the phone and call someone to fix your problems—you have to figure them out for yourself. This can be hard, it can be frustrating when you can't figure things out. But the silver lining is that this process will force you to learn more, which will make you a better data scientist in the long run.

3.2 Working with IT

If figuring things out for yourself by reading documentation, experimenting, searching StackOverflow and other messageboards, and talking to colleagues doesn't sound like your cup of tea, then you might not be a data scientist—you might be an IT person.

Generally—but especially at large, established companies that are not in tech—the IT department is an obstacle that you will have to overcome. It's not really their fault—their job is to keep the network up, the servers running, the software and hardware up-to-date, and to maintain security. Your job is to learn from data, and unfortunately this means that you will want to do things that no other employee in the organization wants to do or has ever done before. You will want query access to a database, you will want to spin up a new web and/or database server, you will want to bypass the VPN, use cloud services, run Linux, and have administrative access to your own computer. All of these special requests make life more difficult for IT, so they will likely say "no" if you ask. My advice is to be friendly, emphathetic, supportive, patient, and trust in the process. Assuming that you aren't actually trying to do anything bad[3], it is IT's job to provide you with what you need to do your job well.

3.3 Don't buy the interface, get the raw data

Vendors will come to your organization and try to sell you data or software, or both. If the data are worthwhile, buy them. Unless the software is really compelling, write your own

[3]For a terrifying story of a rogue data scientist, read about Chris Correa (The Associated Press, 2016). Correa held a job similar to mine with the St. Louis Cardinals, but was convicted of hacking into the Houston Astros' proprietary database and was subsequently sentenced to nearly four years in prison.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

instead. Your organization has specific needs that are different from the other organizations that each vendor is targeting. No matter how much the vendor tells you that they will be responsive to your specific needs, they will never be as responsive as you. You understand your organization's needs in a way that they never will, and because they are simultaneously servicing many organizations, they aren't going to quickly change their product just to make you happy. Thus, in most cases, proprietary software and web applications are unlikely to be worth the steep costs they will incur. A better strategy is to build proof-of-concept prototypes yourself, and then use the savings to hire a developer who can expand and maintain your application in the long-term.

Vendors love to sell organizations the full suite that comes with a big sticker price, because it allows them to control everything and maximize their profit. But you are a data scientist—you like to get under the hood. You were hired to convey information to your boss, so spend your resources on getting access to the raw data that you need and can't create yourself, and build your own delivery mechanisms. For simplicity, build interactive web applications (using Shiny (Chang et al., 2016), Javascript, or PHP) that you and your colleagues can use on an ongoing basis.

4 Reaching out

It's easy for an isolated data scientist to get marginalized within the organization's larger mission. Making your presence felt and having ongoing connections with other departments will help to solidify your place in the hierarchy.

4.1 Don't hoard your data and skills, share them

If you are the only employee in your organization with certain technical skills and access to certain raw data, then it can be tempting to try and preserve these unique attributes by keeping them to yourself. Resist this urge. You may be thinking: "Since I am the only one who knows SQL here, then whenever anyone needs to get some data they have to come to

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

me, and that makes me valuable and means I can't be fired." This may be true, but it is a short-sighted and self-limiting view. This stance will result in two far larger problems. First, it will make more work for you, because you have to do the fishing for everyone. Second, it will get you pigeonholed as the fisher, and because you are the only fisher you won't be considered for promotion to the more lucrative position of Manager of Fish. Instead, what you should do is teach your colleagues how to fish. This will expand your influence and make the data science group into a larger entity—with you at the center of it. It will empower your colleagues to do more data science work on their own, freeing you to do even higher-quality data science work. It will help your colleagues understand what you do, how valuable it can be, and they will become more invested in data science. You will become the obvious candidate to become Manager of Fish—because now there is something (and hopefully someone) to manage.

Once you have your database set up, give your colleagues access to it. Teach them SQL, which may be hard to master, but is not hard to learn, even for someone with no programming experience. Put your work in places whether other people can see it (e.g., an internal Wiki). Encourage your colleagues to learn new skills through Coursera and DataCamp, and mentor them.

4.2 Brand your baseball information system

In the aughts, the pioneering sabermetric organizations in baseball were building *baseball information systems*. These were (generally), web or standalone applications that were backed by a database. While the details are proprietary, most were likely focusing on automated reporting, while other features included daily-updated forecasts, aggregated lists of prospects, and other predictions for various things. The Indians had DiamondView, the Red Sox had Carmine, and later, the Pirates had MITT and the Astros had Ground Control. The Padres (under the direction of Chris Long), the Yankees (Michael Fishman), and the Mets (yours truly) also built baseball information systems that were not publicly branded. In retrospect, this was a mistake.

Imagine a company that has a great product but no name for it. It goes nowhere because

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

there is nothing for consumers to latch onto. The same holds for the data science enterprise that you are growing within your organization. Give it a name so that others have a way to talk about your work. You don't want to be a Svengali whispering in the ear of the CEO—you want to be an *architect* who builds things that are of value to the whole organization.

4.3 Getting buy-in

Even in this day and age, you may encounter some resistance to your data-driven work. It should not be surprising that there is a larger group of people who will remain skeptical of what we can learn from data, especially in domains where experience and intuition have historically been highly-valued.

4.4 Learn from your elders

In baseball, this conflict arose as scouts vs. stats. The job of a scout is to evaluate baseball players by watching them play and comparing them to their internal database of other players that they have accumulated over a lifetime of playing and watching. My job was to evaluate baseball players by analyzing their performance and comparing them to a SQL database of other players accumulated over the history of baseball. Naturally, many scouts were skeptical of the data scientists, and nervous that scouting would become obsolete. However, over the decade of the aughts it became clear that scouts were not obsolete, and to the contrary several organizations hired more scouts once they recognized some of the limitations of what they could learn from data. While some scouts have had to modernize their workflow, there is no evidence that there are any fewer scouts than there were twenty years ago.

Other curators of information have been less fortunate (remember travel agents?), and in your organization, there are likely people who have built their careers on their ability to make good decisions—often in the absence of data—who will feel threatened by your role as a provider of data for making decisions. Do not discount the collective wisdom of these

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

people. Do not be so arrogant as to think that you can replace them without first making an earnest effort to learn what they know. At a minimum, you will have better working relationships with these people if you have made honest attempts to learn from them. You might also actually learn things of value that will improve the quality of your own work.

4.5 Speak the language

Building *domain knowledge*—including domain-specific dialect—is often impossible to do from outside of an organization, but it is a necessary condition for your integration. Some of this lingo may be unimaginably vapid, but you don't want to be the tech person who doesn't speak the language—you want to an unassailable insider.

4.6 Making presentations

As noted earlier, outreach is an important step towards avoiding marginalization. A time-efficient way to do this is by giving presentations to your colleagues. View these as opportunities to spread your knowledge. Don't worry about keeping your insights secret or getting credit for them. If you have discovered an actionable insight, you want that idea to permeate your organization so that it becomes part of the conversation the next time that issue comes up.

For example, in the early aughts the insight that a batter's plate discipline was an important skill was not canon. Most scouts focused on skills that were demonstrable to the eye (e.g., how fast you ran, how far you hit the ball, and how hard you threw) as opposed to invisible skills that had to be counted over time (e.g., how often you walk and strike out). In the early days, we had to rehash this argument over and over again, as there were some scouts who would consistently tout players with poor plate discipline. Over time, as we had these conversations more and more, and my colleagues and I gave presentations to player development and scouting personnel, plate discipline became a canonized skill that everyone learned to value. This is a sign of organizational progress.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

5 Never stop learning

One of the most frustrating parts of being an isolated data scientist in a competitive industry is not being able to share ideas and get help from fellow isolated data scientists at competitor organizations. Consulting books and online resources can help, but your questions are often very specific and it may be hard to find the answers you need.

While many non-academic jobs assume a relatively fixed skill set, a data scientist can never stop learning. Whether you have research experience as an undergraduate or graduate student, you must continue to learn about new technologies, methods, and ideas. You will have colleagues who think that because they don't already know something (e.g., programming), they can't or shouldn't have to learn it. You must reject this self-nullifying attitude. You job requires continual personal and professional growth. Read blogs (e.g., R-bloggers), experiment with new software, take courses online, and continue to augment your burgeoning skill set.

5.1 Assumption-free statistical modeling

As an isolated data scientist, you will likely be the most technical person in your organization. As such, there may not be more senior people who check your work the way that your professors did in school. Your boss isn't going to ask you about multicollinearity or the heterogeneity of your residuals. However, it is your ethical duty to exercise rigor anyway. Do not give in to the temptation to practice "assumption-free statistical modeling."[4] The consequences of faulty statistical modeling can be catastrophic—not just for your reputation, but for your organization itself and anyone affected by the actions of your organization.

5.2 Mentoring

One of the keys to long-term success at a research university is the development of a pipeline of capable graduate students who can assist you with research projects. If you have the

[4] I was introduced to this term by Andrew Bray of Reed College.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

ability to multitask, this can allow you to parallelize your workload, which can result in a wider stream of output. As an isolated data scientist, you may have access to a limited set of entry-level employees and interns, but mentoring these people can be rewarding for all involved.

During my time with the Mets, we had a steady flow of interns, many of whom are already in high-ranking positions in baseball (David Stearns is the general manager (GM) of the Milwaukee Brewers, Jonathan Strangio is the assistant GM of the Los Angeles Angels of Anaheim, and Scott Freedman is director of baseball operations with the Philadelphia Phillies, to name a few). Among other things, most of these interns learned how to query our internal SQL database, which enabled them to carry out research projects independently. I am sure that having these skills, and being able to demonstrate such research, was instrumental in their success upon leaving the Mets.

In 2009, I was asked to give a talk about defensive evaluation in baseball at the Joint Statistical Meetings. I enlisted the help of two then-interns, Rob Sebastian and Andy Galdi. Our goal was to survey methods for defensive evaluation in baseball, ranging from the earliest metrics created by Henry Chadwick in the early 1900s (Schwarz, 2004) to the most recent hierarchical Bayesian methods (Jensen et al., 2009). To do this, we had to not only understand these metrics—which was especially challenging given the mathematical sophistication in Jensen et al. (2009)—but also to implement the computation of these statistics using our database. Even though it was not peer-reviewed, the resulting proceedings publication (Baumer et al., 2009) gave these interns something concrete to show prospective employers. Rob got a job at Google, while Andy worked for the NBA, earned a master's degree in statistics at Stanford, joined Google, and is now the director of baseball research for the Phillies. Their success reflects well on me and the Mets organization, and so too can it be for you and your organization. This early mentoring experience has also been foundational for me as a faculty member.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

6 Conclusion

Given the breadth of industries to which data science can contribute, many data scientists will be isolated—working either alone or in small teams as part of a larger company. These jobs require profound independence and self-motivation, but can be exciting opportunities wherein creativity and ingenuity are rewarded. I hope these pieces of advice—gleaned through years of experience—can help the next generation of isolated data scientists make meaningful contributions to their respective organizations.

7 Acknowledgements

I am grateful to Nicole Lazar, Lance Waller, Nicholas Horton, Luke Bornn, Jenny Bryan, and Hadley Wickham for helpful comments on previous drafts of this manuscript.

References

Albert, J. (2003). *Teaching statistics using baseball*. MAA: Washington, DC.

Allaire, J., J. Cheng, Y. Xie, J. McPherson, W. Chang, J. Allen, H. Wickham, A. Atkins, and R. Hyndman (2016). *rmarkdown: Dynamic Documents for R*. CRAN. R package version 1.0.

American Statistical Association Undergraduate Guidelines Workgroup (2014). *2014 Curriculum Guidelines for Undergraduate Programs in Statistical Science*. American Statistical Association. http://www.amstat.org/education/curriculumguidelines.cfm.

Baumer, B. (2015, 2). In a Moneyball world, a number of teams remain slow to buy into sabermetrics. In R. Webb (Ed.), *The Great Analytics Rankings*. ESPN.com.

Baumer, B., A. Galdi, and R. Sebastian (2009, August). A Survey of Methods for the Statistical Evaluation of Defensive Ability in Major League Baseball. In *JSM Proceedings*, Statistics in Sports. American Statistical Association: Alexandria, VA.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

Baumer, B. and A. Zimbalist (2014). *The Sabermetric Revolution: Assessing the Growth of Analytics in Baseball*. University of Pennsylvania Press: Philadelphia.

Broman, K. and K. Woo (2017). Data organization in spreadsheets. *The American Statistician 71*(3). submitted for this TAS collection.

Bryan, J. (2017). Happy Git and GitHub for the useR. *The American Statistician 71*(3). submitted for this TAS collection.

Chang, W., J. Cheng, J. Allaire, Y. Xie, and J. McPherson (2016). *shiny: Web Application Framework for R*. CRAN. R package version 0.13.2.

De Veaux, R. D., M. Agarwal, M. Averett, B. S. Baumer, A. Bray, T. C. Bressoud, L. Bryant, L. Z. Cheng, A. Francis, R. Gould, A. Y. Kim, M. Kretchmar, Q. Lu, A. Moskol, D. Nolan, R. Pelayo, S. Raleigh, R. J. Sethi, M. Sondjaja, N. Tiruviluamala, P. X. Uhlig, T. M. Washington, C. L. Wesley, D. White, and P. Ye (2017). Curriculum guidelines for undergraduate programs in data science. *Annual Review of Statistics and Its Application 4*(1), 1–16.

Donoho, D. (2015, 9). 50 years of data science. Technical report, Stanford University. http://courses.csail.mit.edu/18.337/2015/docs/50YearsDataScience.pdf.

Efron, B. and C. Morris (1975). Data analysis using Stein's estimator and its generalizations. *Journal of the American Statistical Association 70*(350), 311–319.

Jensen, S. T., K. E. Shirley, and A. J. Wyner (2009). Bayesball: A Bayesian hierarchical model for evaluating fielding in major league baseball. *The Annals of Applied Statistics 3*(2), 491–520.

Lewis, M. (2004). *Moneyball: The art of winning an unfair game*. WW Norton & Company: New York.

Marchi, M. and J. Albert (2013). *Analyzing Baseball Data with R*. CRC Press: Boca Raton.

Marwick, B., C. Boettiger, and L. Mullen (2017). Packaging data analytical work reproducibly using R (and friends). *The American Statistician 71*(3). submitted for this TAS collection.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3160v1 | CC BY 4.0 Open Access | rec: 17 Aug 2017, publ: 17 Aug 2017

Robinson, D., S. Bache, and Z. Ross (2017). dplyr + tidyr + broom. *The American Statistician 71*(3). submitted for this TAS collection.

Schwarz, A. (2004). *The numbers game: Baseball's lifelong fascination with statistics*. Macmillan.

Tango, T., M. Lichtman, and A. Dolphin (2007). *The Book: Playing the Percentages in Baseball*. Potomac Books.

The Associated Press (2016, 7). Christopher Correa, former Cardinals executive, sentenced to four years for hacking Astros' database.

Wickham, H. and R. Francois (2016). *dplyr: A Grammar of Data Manipulation*. CRAN. R package version 0.5.0.

Xie, Y. (2015). *Dynamic Documents with R and knitr*, Volume 29. CRC Press: Boca Raton.