Data Analysts and Their Software Practices: A Profile of the Sabermetrics Community and Beyond

JUSTIN MIDDLETON, North Carolina State University, USA
EMERSON MURPHY-HILL, Google, USA
KATHRYN T. STOLEE, North Carolina State University, USA

For modern data analytics, practices from software development are increasingly necessary to manage data, but they must be incorporated alongside other statistical and scientific skills. Therefore, we ask: how does a community recontextualize software development through the unique pressures of their work? To answer this, we explore the analytic community around baseball, or sabermetrics. To discover software development's place in the search for robust statistical insight in sports, we interview 10 participants in the sabermetric community and survey over 120 more data analysts, both in baseball and not. We explore how their work lives at the intersection of science and entertainment, and as a consequence, baseball data serves as an accessible yet deep subject to practice analytic skills. Software development exists within an iterative research process that cycles between defining rigorous statistical methods and preserving the flexibility to chase interesting problems. In this question-driven process, members of the community inhabit several overlapping roles of intentional work, in which software development can become the priority to support research and statistical infrastructure, and we discuss the way that the community can foster the balance of these skills.

CCS Concepts: • **Software and its engineering** → *Collaboration in software development*; • **Human-centered computing** → *Empirical studies in collaborative and social computing*; *Collaborative and social computing systems and tools.*

Additional Key Words and Phrases: data analysts, software process, end-user software engineering, end-user communities

ACM Reference Format:
Justin Middleton, Emerson Murphy-Hill, and Kathryn T. Stolee. 2020. Data Analysts and Their Software Practices: A Profile of the Sabermetrics Community and Beyond. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 52 (May 2020), 27 pages. https://doi.org/10.1145/3392859

1 INTRODUCTION

The benefits of software development can be enjoyed by anyone with a computer, a task, and enough patience, whether they are formally trained in engineering or simply content to tinker for personal benefit [32]. One prominent example where we see software's cross-discipline versatility is in data analytics, defined as the interconnection of "theories, technologies, tools, and processes that enable an in-depth understanding and discovery of actionable insight into data" [8]. As the demand for sophisticated and personalized analyses grows across industry and academic alike [12, 18, 50],

Authors' addresses: Justin Middleton, North Carolina State University, Raleigh, North Carolina, USA, jamiddl2@ncsu.edu; Emerson Murphy-Hill, Google, Mountain View, California, USA, emersonm@google.com; Kathryn T. Stolee, North Carolina State University, Raleigh, North Carolina, USA, ktstolee@ncsu.edu.

2573-0142/2020/5-ART52 $15.00
https://doi.org/10.1145/3392859

programming languages like R and Python grow more popular among data analysts, even though many have no background in formal software development [29].

Data analytics takes many forms, employing numerous techniques to address issues in the datasets of business, government, and healthcare alike. One such subcommunity is baseball analytics, or *sabermetrics*, which combines mathematical sciences with athletic sport. As described by Valerdi, sabermetrics "uses data to make objective decisions about which players to draft, which players to play, how much to pay players, and which personnel trades between teams make the most sense" [64]. To us, this provides an opportunity to study a community that take the same data and performs analyses with varying tools and knowledge. Many people participate in it only as a hobby, but the practice is also immersed in the multi-billion-dollar professional baseball industry, yielding careers in team front offices or as independent consultants.

With data analysis and software development converging in populations like this, we explore the extent to which the concepts of software development and engineering (i.e. systematic and disciplined approaches to development to maintain quality [1]) accommodate the behaviors of data analysts, particularly sabermetricians. To gain a deeper understanding of their development processes, we interviewed 10 sabermetricians on their history and participation within this data analytic community. We then expounded their answers by surveying more over 120 more analysts from baseball and general practice to assess generality. Using an approach to generating grounded theory [61], we address specific questions like *how analysts acquire and foster software development skills* and *to what projects and processes they apply such skills*, thereby contextualizing sabermetrics in analysis, development, and engineering. In our exploration of technical practices, however, we necessarily bring a spotlight onto the community and cultural features which shape analytic work. This focus complements collaboration-focused research that details, for examples, how data-centered groups negotiate insight in their practice [4, 46], or how community members navigate the unclear design choices inherent in scientific work [11, 51].

Overall, our results describe tensions and patterns in analyst work, from learning to practice. For example, we note that though sabermetricians enter the practice through common passions and goals, they often specialize in specific responsibilities and inhabit roles from general analyst to developer and data curator. Furthermore, given the volume of public discourse around sports in particular, there are numerous avenues by which sabermetricians learn—personal projects to peer interaction to popular literature—and numerous ways that documentation can support better analyst learning. In practice, software is developed to support a scientific mindset with an exploratory, iterative process which varies between building independent analytical scripts to supporting data infrastructure for larger baseball enterprises. Although some analysts express interest in gaining specific elements from more software engineering rigor, such as in software testing, they face barriers in effectively sharing tools and workflows throughout the community.

This is the first work to explore the software development processes of the sabermetrics community, to our knowledge. This paper makes the following contributions:

- A mixed-methods investigation, including 10 interview participants and over 120 survey respondents, that situates a unique programming community—sabermetricians—in the contexts of data analytics and software development, both professional and end-user.
- An assessment of the educational and social factors that shape analyst skill development.
- An examination of how software development concepts fulfill specific needs (or not) in baseball and data analytic spaces.

2 BACKGROUND

To show how sabermetrics fits into data work in general, we briefly describe, one, the basic rules of baseball and, two, the cultural and technological contexts that make sabermetrics possible.

2.1 The Sport of Baseball

Baseball is a competitive, two-team sport played on a diamond-shaped field with a base at each corner. The defensive team throws the ball past bat-wielding members of the offense, and the offense wants to hit that ball back into the field. When the offense hits the ball, the hitter runs through as many bases as they can before stopping at a base and waiting for the next hitter to start movement again. If a runner makes it back to the starting base, they score their team a point. But if the hitter misses too often, or if the defense catches the hit ball before it touches the ground, or if a running offensive players is tagged with the ball between bases, that player is given an out and sent off the field. Three outs and the teams switch sides. Each team gets nine innings on each side. While a majority of discussion in this research orients around Major League Baseball in the United States, there are also extensive systems of other leagues, including minor and college baseball, as well as international leagues in the Caribbean, Central America, East Asia, and beyond.

2.2 Sabermetrics

Sabermetrics, named for the Society of American Baseball Research, refers to "any mathematical or statistical study of baseball."[1] In practice, it means asking questions like "Do left-handed pitchers have an advantage against right-handed hitters?" or "How much money should a team invest in recruiting a specific player given the weaknesses in their current roster?" These inquiries are made possible by baseball's history of rich data records. Simple records of play have been uniformly organized by Henry Chadwick's introduction to observation-based box scores in the 19th century. Data in 20th and 21st centuries become much more sophisticated by instrumenting fields with physical sensor and analysis technology, such as PITCHf/x, Statcast, TrackMan[2], and Hawk-Eye[3].

With the availability of data around an already popular sport, communities of people interested in its analysis have developed. One on hand, professional baseball management and media have much to gain by supplementing traditional methods with sabermetric insight. However, the popularity and spectacle of the sport make it equally amenable to viewers at home. The uninhibited re-digestion of baseball statistics is exemplified in the writer Bill James' *Baseball Abstract* [24] or in the explosion of fantasy sports from Rotisserie League Baseball among friends [19]. These trends are especially visible with the Internet, where "formerly isolated populations started congregating and experimenting with each other's creations and ideas in digital space" [7]. And, as Millington notes, "as data analysis becomes more valuable to sport franchises, the boundaries between sport consumer and sport (intellectual) laborer grow ever more blurry" [41].

As such, sabermetrics represents especially where cultural and technical dimensions overlap. Given what Hutchins' calls the "quantitative imperative" of sports to constantly analyze and improve, baseball has "produced a labour market for experts with high levels of 'database literacy' and skilled programmers able to design software and original analytical formulas." [23] Sabermetrics' immersion in the baseball industry makes it economically valuable, but its independent side makes it educationally and entertainingly valuable for beginners and teachers in data science looking for accessible, practical datasets on visible and comprehensible behavioral processes.

[1] https://sabr.org/sabermetrics
[2] https://www.trackman.com/
[3] https://www.hawkeyeinnovations.com/sports/baseball

3 RELATED WORK

We focus on the literature that describes the mutual influence that data analysis and software development have on each other.

3.1 Analyst Work Processes

Behavioral and cognitive processes. Building from techniques in cognitive science, statistics, data mining, many researchers have designed step-by-step models of how knowledge workers move from data to insight [8]. Some begin at the cognitive mechanisms of sensemaking. Pirolli, Card [49], Klein [31], and other researchers enumerate the steps and loops of information workers as they forage for and incorporate information into their operating view of the world. These ideas have also been approached empirically [26, 47]; for example, Chin and colleagues [10], ran hypothetical homeland security scenarios at a workshop to observe how intelligence analysts collect, prioritize, and iteratively probe evidence to approach conclusions with national security at stake. Others begin at the discrete behaviors of data mining. The KDD (Knowledge Discovery in Databases) process, described by Fayyad and colleagues, captures nine steps from *learning the application domain* to *using discovered knowledge* in the service of "identifying valid, novel, potentially useful and ultimately understandable patterns in data" [14]. The most popular data-mining methodology in practice today, according to the KDnuggets.com surveys [48], is CRISP-DM in six iterative phases [57] with generally analogous intent: "learning the application domain" is instead "business understanding," for example. Many such data analytic models exist, so rather than proposing new processes, some researchers reconcile differences in existing processes either by comparing their phases side-by-side [2, 34] or proposing revisions to accommodate new technologies [13, 15].

Like us, other researchers acknowledge the growing importance of robust software skills for data analysis, and some have synthesized new hypothetical processes by combining existing processes from each field. For example, Marbán and colleagues compare the CRISP-DM model with IEEE and ISO software engineering processes, thereby demonstrating the insufficiency of CRISP-DM alone for software and suggesting what should be adopted or created from scratch to compensate [37]. Mariscal, Marbán, and Fernandez later expanded this synthesis with a survey of approaches within the KDD and DM literature [39], proposing a speculative 3-process, 17-subprocess Refined Data Mining Process that reconciles phases across standards. Unlike these, however, our concern is not so much to produce a distinct, segmented process but to collect and organize empirical data to illustrate the unique pressures of particular communities of analysts. This data may later be used to shape emerging theories or new categories of work.

Collaborative processes. Beyond individual- or business-oriented processes, CHI and CSCW research also explores how processes are enacted in communities and scientific infrastructures, or the technological, social, and institutional arrangements that bound and define work [27, 58]. For example, Ribes and Finholt outline tensions in long-term scientific work using a matrix of three primary concerns to three scales of work: these tensions include, for example, planning requirements thoroughly beforehand versus letting them emerge, or finishing a project when specific research results are attained versus when the system is generalized and production-ready [51]. The concepts of invisible but omnipresent infrastructures challenge the idea that data and scientific software exists as objective artifacts apart from its context of production, and thus these contexts are worth studying lest we perpetuate the biases that they imply [6, 44].

For data, this means studying the choices that uncover or produce data, as Muller and colleague's IBM interviews explicate [42], and the frames by which to interpret data, as Borgman and colleagues emphasize in their study of environmental science and engineering collaborations [4]. It also means learning how and when to trust it: Rolland and colleagues' work with epidemiology students, for

example, emphasizes the question of data's procedural source, among other germane questions, and how the answer anchors a practical understanding [54]. Passi and colleague study the collaborative emergence of trust in data science at a new media company through a mix of intuition and iterative recontextualization [46].

Likewise for software, this means viewing it not merely as an executable artifact but a facet in the larger system of work, exemplified by how Trainer and colleague's positioning of software in social systems entails numerous forms of "extra work" like community management and responding to user requests [62]. Paine and colleagues, on the other hand, study the visualization practices in a cosmology research group to reframe plots not just as deliverables but as encapsulations of the entire research process from sensors to scripts [45]. Choi and Tausczik, with intentions perhaps most similar to our research, investigated and interviewed data-centered communities for social good to explore collaboration practices and artifacts [11]; their communities comprise primarily small, interdisciplinary teams with heavy turnover and split focuses on making tools and analytical reports. Both our and their work take special interest in the team tools and compositions for a specific kind of data work, but we approach our unique community with a heavier focus on skill development and software process.

3.2 Data Informing Software

With the growth of data analysis as a professional vocation, some researchers study how data analytics reorganizes professional software development specifically [16, 40]. Kim and colleagues explore the identities of data analysts on software engineering teams, identifying the backgrounds that equip industrial data scientists and what they contribute to their software team [29, 30]. Within this research, some discuss how software engineers and data analysts perceive each other in order to improve the collaboration for each: Begel and Zimmermann interviewed software engineers to list questions for data analysts to answer [3], while Li and colleagues interviewed non-software-experts, including data analysts, to uncover the behavior of the socially desirable software engineer [36]. In a study similar to ours, Riungu-Kalliosaari and colleagues held focus groups with experienced data scientists, outlining the work processes and challenges of data work from which software developers might learn [52]. They consulted a ready-made work process from data science literature, outlining the important challenges that crop up during each of its six discrete phases from conceptualization to deployment. Like our work, many of these studies characterize analytic processes and their challenges; in our focus, however, we also include end users and not-professional thinkers whose passion for a data-heavy subject, and not necessarily job responsibilities, leads them to the analytic skillset.

3.3 Software Informing Data

Many profiles of the modern data analyst underscore the importance of knowing programming, to some extent, to perform meaningful work in the industry [8, 20, 50]. In this way, many sabermetricians qualify as end-user developers, as they are information workers who employ tools from software development (e.g. structured querying languages, scripting) as one means to find insight in data, and they program "primarily for personal, rather than public use" [32]. In borrowing from software engineering, end-user software developers have benefited from improved software validation through testing [17, 22, 55], improved debugging to assist with troubleshooting problems [33], improved comprehension and maintenance through refactoring [59, 60], and improved traceability through versioning [35]. The awareness of end-user development allows for a variety of backgrounds that recontextualize what people expect out of software engineering. For example, bioinformatics is one subject in which software and science has resonated: through many studies,

Umarji and colleagues recommend practices for teaching software engineering techniques to non-engineering communities [63], Chilana and colleagues illuminate the challenges in mixed-discipline scientific software development [9], and Verma and colleagues critique the lack of the appropriate quality-assurance processes from professional software engineering [65]. Segal also studied research scientists collaborating with software engineers and probes how the unique demands of non-software fields shapes expectations and challenges with traditional software process, such as articulating requirements, documentations, and testing [56]. These studies exemplify how software development permeates other fields of research, and they attempt to bridge practices for mutual benefit, similar to our intentions with data and baseball analytics.

4 STUDY

In order to perceive the patterns around a community's software practice, we focus on two research directions. The first is to look backward, at the foundations of community members: which perspectives influence or explain their behavior? Following that, we can look forward to see how they apply their foundations and experiences shape the projects they currently work on. The process of answering both will also illuminate the various tensions that contrast this data-analytic community with other communities. Therefore, we enacted two exploratory research questions:

- **RQ1**: **In what contexts do data analysts acquire software development skills?** What motivates a person to become a sabermetrician, what skills do they bring to work, and from where do they acquire such skills?
- **RQ2**: **By what processes do analysts design and fulfill their data questions through software development?** Do they tend towards small-scoped, individual work or collaborative infrastructural work? How do these tendencies influence their workflow?

We applied a mixed-method approach for these questions. First, to better define what dimensions of those questions, we conducted semi-structured interviews with participants in sabermetrics communities and projects, and we distilled our impressions into a 36-question survey to probe the generalizability of their perspectives. On these responses, we apply Corbin and Strauss' approach to generating grounded theory, which describes an versatile but disciplined posture toward organizing and connecting themes from many sources of data, without asserting that the result is the one objective picture that excludes any other interpretation [61]. We include the study material in this section—interview scripts and survey questions—in an associated Zenodo repository.[4]

4.1 Interviews

Structure. We designed a 60-minute semi-structured interview comprising three specific topics:

- Their backgrounds—formal education and other experiences—that inform their current positions with baseball analytics.
- The resources from which they learned the requisite perspectives and skills for their sabermetric work. When we focused on software documentation specifically, we drew sample features from Robillard and DeLine's field study of API designs [53].
- Steps in their software process, furnished with software concepts from the Software Engineering Body Of Knowledge (SWEBoK) [5].

We verified topic relevance through two pilot studies with graduate students in the university baseball analytics club. Through these and all subsequent interviews, we iteratively changed the emphases on specific topics: topics like requirements through coding and testing received more coverage, and others like configuration management and ethics became less prominent. In general, we complied with the direction of the script unless the participants naturally digressed toward

[4] http://doi.org/10.5281/zenodo.3731135

Table 1. Participant designations and a summary of their backgrounds.

ID	Description	Academic Background
P1	Independent consultant for data-driven player development	Mathematics BS
P2	PhD student, member of campus analytics club collaborating with university baseball team	Statistics PhD underway
P3	PhD student, published in sports statistics, sabermetric hackathon participant	Statistics PhD underway
P4	Software developer for MLB Team	Statistics BS, computer science minor
P5	Undergraduate student, participant in sabermetric conferences	Statistics BS underway
P6	Software developer, sabermetric hackathon participant, fantasy leagues	History BS
P7	Data science consultant, contributor to sabermetric software packages	Statistics PhD
P8	Professor, published in sports statistics	Statistics PhD
P9	Professor, former analyst for MLB team	Mathematics PhD
P10	Consultant for baseball data management, professor	Economics PhD, Engineering BS

related experiences; we allowed them to take the direction unless severely deviating from the topic of software engineering according to the interviewer's judgment.

Recruitment. To ensure we study a *community* of people rather than just a collection of individuals with a common trait, we preferred methods of recruitment that relied on existing social networks. We identified potential interview participants through the following sources: attending the annual SABR Analytics conference[5] and university sports analytics groups; contacting presenters at notable sports analytics conferences;[6] joining Slack channels for sabermetric hackathons; and snowball sampling, for which we asked our initial participants whether they knew anyone else who would be relevant to our research.

Through these channels, we emailed 19 people in a few waves of recruitment. Our recruiting emails outlined that we sought people "who have worked both in baseball analytics and programming to some extent (academically or professionally);" we did not offer any compensation on top of this. Overall, 10 people accepted to participate in an interview, 6 declined, and 3 never responded. Every participant came from a different university, team, or company otherwise. Table 1 summarizes their professional and academic information that they discussed explicitly in the interviews. Participants came from a variety of backgrounds in terms of the intensity of experience, ranging from statistics undergraduate acquiring skills through peripheral team involvement to seasoned developers with a dual penchant for numbers and sport. In terms of formal education, four interview participants had doctorates, and all 10 had completed or were completing at least one bachelor's degree. Two participants, P7 and P9, spoke about their experiences around baseball analysis but were no longer directly involved in those communities at the time of the interview.

The first author conducted interviews alone through the teleconferencing softwares Google Hangouts or Skype while the audio was recorded by the same computer, with one exception where the interview was conducted in person. In practice, the average interview was about 65 minutes, with the shortest at 55 minutes and the longest interview 99. The first author manually transcribed all interviews.

4.2 Survey

Structure. To generalize our interview findings to a larger sample, we designed a 36-question survey. The first few questions of the survey collected participants' experience with data analysis, baseball analysis, and software development in general. The definitions which we presented in the

[5] https://sabr.org/analytics
[6] http://www.sloansportsconference.com/

Table 2. Definitions of cornerstone topics by which we interviewed and surveyed analysts.

Concept	Definition
Data Analytics	"Data analytics is the multidisciplinary science of quantitatively and qualitatively examining data for the purpose of drawing new conclusions or insights (exploratory or predictive), or for extracting and proving (confirmatory or fact-based) hypotheses about that information for decision-making and action' [8].
Documentation	"Any documents or resources intended to inform or teach a software user how to use the software, including product manuals, tutorials, and online classes."
Tools	"Any software product that supports you in writing a script or program, including the ready-made software libraries for that language and the environment in which you write your code."

survey were broad, building off Nardi's end-user-inclusive definition of programming as any activity performed "to create an application that serves some function for the user" [43]. To clarify data analysis, we presented Cao's definition, as presented in Table 2. To clarify software development, on the other hand, we provided a broad instructions ranging "from building complex software systems to writing database queries in SQL or spreadsheet formulae." If they had practiced one of these skills, we asked them to distinguish between practicing it professionally or nonprofessionally (e.g. as a hobby or in education).

If a participant had no experience whatsoever with baseball analysis specifically, we asked that they respond in terms of their non-baseball professional data work. If the participant has no experience with both baseball and general data analysis, the survey ends. Even though we are investigating software practice, the survey does not terminate if they report no experience with software development, since we could gather data on their tools in lieu of software and simply exclude that data if we found no insights in the nonprogramming sample.

The survey covered the two general questions of the interviews: the foundations of their data science practice and the process of software creation. Where applicable, the choices to a question were derived from frequent interview responses or topics from earlier research, such as general surveys of data analysts [50]. However, we did not define any strong threshold by which options were chosen from the range of possibilities. To align the concepts that people brought to the survey, we provided definitions of data analytics, software documentation, and software tools which are intentionally broad to accommodate as much variation as participants; we present these in Table 2, as well as others not explicit in the survey which we nevertheless use to organize our work.

Recruitment. We favored channels which broadcast to a large number of people based on a shared interest. For sabermetricians, we posted the survey on an Slack channel and the subreddit for sabermetric projects.[7] We also asked a prominent sabermetrician to post the link on their Twitter feed with, at the time, more than 18,000 followers. Furthermore, though sabermetricians are our primary subject, we are also interested in where we can contrast them with other populations of data analysts, either through existing literature or new survey data. Therefore, to attract general data analysts, we also distributed the survey on internal mailing lists in nearby analytics companies and on alumni mailing lists for a university data science program. In either case, the survey was uncompensated.

Through these means, we attracted 129 participants, although only 75 survey respondents completed the entire survey after a gradual dropoff. Of those who provided a location of work,

[7] https://www.reddit.com/r/Sabermetrics/

Table 3. Analyst experiences in survey. "Pro." is an abbreviation for "Professional(ly)," and "NP" for "Nonprofessional(ly)." Gray-shaded cells constitute our *Baseball Analyst* sample, non-programmers excluded.

Practices Data Analysis...				Practices Programming?					
Professionally?	with Baseball Data?	**Total**		Pro.		NP		No	
		#	%	#	%	#	%	#	%
NP Analysts	None	2	1.6	2	1.6	0	0.0	0	0.0
	NP experience	25	20.5	3	2.5	19	15.6	3	2.5
Pro. Analysts	None	42	34.4	39	32.0	2	1.6	1	0.8
	NP experience	40	32.8	34	27.9	4	3.3	2	1.6
	Pro. experience	13	10.7	11	9.0	1	0.8	1	0.8
	Baseball Analysts	72	59.0	48	39.3	24	19.7	—	—
	Total	122	100.0	89	73.0	26	21.3	7	5.7

all but 4 from 112 respondents were working in North or Central America, the rest from Europe and South America. Additionally, Table 3 summarizes experiences in terms of professionality and baseball specificity. For example, this table shows that 95 of our participants work as professional analysts (40 + 42 + 13 from the first column "#"), 84 of which use coding in their professional responsibilities (34 + 39 + 11 in the column "Practices Programming? Pro."). However, seven participants said they do not program whatsoever, and because of the paucity of nonprogrammers, we do *not* include them in the general reports.

The number of people in the sample who work with sabermetrics professionally is 12, but when including nonprofessional participation, the number grows to 72. Therefore, in this research, we use the label *Baseball Analysts* ($n = 72$, abbreviated "Base." in tables) to specify both professional and nonprofessional interaction with the data; it is a label attained by voluntary interaction, not employment or achievement. *Professional Analysts* ($n = 90$ with only programmers, abbreviated "Pro." in tables after Table 3) is a label defined by employment, and it overlaps with Baseball Analysts by 50. Tables 4 and 5 presents educational backgrounds by degrees according to these subsets. However, we caution against searching for significant differences in degrees between subgroups: we recruited many of the nonbaseball analysts through a university alumni mailing list, methodologically guaranteeing degrees in this subgroup and not in others.

4.3 Analysis

Following Corbin and Strauss' approach, the first author began coding the interviews shortly after they began, using the ATLAS.ti software[8] to assign and organize the codes and a mix of paper notes and ATLAS.ti for memoing. We segmented the transcript on a sentence-by-sentence basis, combining segments wherein the topics for consecutive sentences were consistent. Our approach to defining concepts and categories (i.e. families of concepts) was first based on whatever kind of activity, object, or person they were discussing in their statement, and properties came from whether their statement described the concept in a way that implies variation or judgment. So not to limit theoretical sensitivity, no codes were explicitly forbidden as long as it was relevant to the discussion and could be repeated between interview participant experiences, except when the participant asked that specific statements not be included in the transcripts. As the codes emerged and other data, such as the survey, came in, the first author passed over the interviews and schemes several more times to refine the codes and organize them into categories and themes.

Our termination at 10 interview participants was due to time restraints and not because of theoretical saturation in itself; however, we used the interviews described in this paper as the first

[8] https://atlasti.com/

Table 4. Survey: Formal academic experiences. "Pro." means professional analysts, "Base." means baseball analysts. Subsets overlap.

Subset	Total		Pro.		Base.	
	#	%	#	%	#	%
Less than high school	1	0.9	0	0.0	1	1.5
High school graduate	0	0.0	0	0.0	0	0.0
Some college	10	8.9	6	7.1	9	13.6
2 year degree	1	0.9	0	0.0	1	1.5
4 year degree	26	23.2	20	23.5	21	31.8
Professional degree	2	1.8	1	1.2	1	1.5
Master's	60	53.6	51	60.0	26	39.4
Doctorate	12	10.7	7	8.2	7	10.6
Total	112	100.0	85	100.0	66	100.0

Table 5. Survey: Degrees and self-study topics with >3 responses. Multiple choices permitted. Asterisks (*) indicate custom answer.

	Degree			Self-Study		
Subset	Total	Pro.	Base.	Total	Pro.	Base.
Responses	112	85	66	112	85	66
Statistics	27	21	16	64	49	42
Analytics*	25	24	5	3	1	3
Mathematics	23	20	15	30	23	16
Economics	19	18	13	14	11	8
Business	15	12	9	13	12	6
CS or SE	13	9	9	49	39	33
Journalism*	4	2	4	0	0	0
Psychology*	4	2	3	1	1	0
IT	3	3	2	22	19	14

step in multiple iterations of theoretical sampling via surveys to a broader community, hoping to simultaneously broach new topics of inquiry and further saturate the previous ones. We also made a draft of the results available to interview participants with intermediate results, in case they had feedback on the developing coding scheme.

5 FINDINGS FOR RQ1: ANALYST SOFTWARE SKILL ACQUISITION

In this section, we explore how baseball analysts define and develop their analytic work, focusing on the studies, resources, and communities that influence them.

5.1 What are the backgrounds of baseball analysts?

Before we consider academic backgrounds, the rules of baseball and baseball statistics are their own physical, cultural, and historical phenomenon that must be studied should one want to be a competent sabermetrician. This is perhaps analogous to a general "business understanding" described in prior analyst workflows, but the point here for many baseball analysts are passionate consumers who are "fascinated with the game and they want to learn more about it" (P8), for whom a personal history with baseball precedes, or at least tightly accompanies, any analytic development. P6 described it as "the tail wagging the dog;" they learned development and relational databases through baseball data, just as P2's first foray into visualization and machine learning was spurred by how they "wanted so badly to be able to do stuff with sports data."

Beyond that, we consider the fields of study which frame analysts' formal and self-guided study. These backgrounds are part demographic context as in Tables 4 and 5 and part data for explaining which experiences shape analysts in practice. Though influenced by our recruitment material, our interviewee descriptions in Table 1 contain mostly statistics, some computer science, and occasional reference to indirectly related or unrelated studies. Four participants (P1, P4, P8, P10) referred explicitly to some formal computer science education, but never as the major focus alone. Others without formal CS education clarify their experience as having "been adjacent to computer science and [they] can understand some stuff but [are] not formally educated" (P2). Otherwise, P6 holds a degree in history yet independently developed their skills in professional software engineering, and P10's background in economics allows clarity in contrasting sabermetrics with other disciplines:

> (P10) It's great because you get tons of data [in baseball], you get great high-quality data...and you observe exactly how the data is produced. So contrast that with say, data about the macroeconomy, which is extremely

> hard to measure, subject to all sorts of noise and revisions, coming from an extraordinarily complex process that we only vaguely understand.

Our survey results confirm these focuses. In our Baseball Analyst criteria, 29 of 66 (43.9%) survey respondents studied mathematics, statistics, or analytics at a university, and 11 (16.7%) had a formal information technology (IT), computer science, or software engineering background. These subjects also dominate self-study habits, more than doubling the proportion which had formally studied those topics. Other popular perspectives were provided through economics and business, and minor topics included journalism and psychology. Sports-specific studies were rare but present: four in our sample came from exercise science, biophysics, public health, and sports management.

5.2 How do people specialize in baseball analysis?

Beginning from our interview data, we coded our observations in baseball analyst work within two orthogonal concepts: context and role. Discussing contexts is in some ways just answering the question, "For whom do they work?" or the organization of people and resources around them. This include positions on baseball teams, research institutions, or media entities, working as an independent consultant, and even just dabbling in fantasy leagues or with friends as a hobby. Furthermore, contexts can overlap with each other, and how one acts in a context is likewise influenced by their role in traits such as freedom, deadline rigidity, and compensation.

Table 6. Emergent roles that describe participant priorities

Role	Priority	Sample statements which motivated this category
Analyst	Insight through an informed use of their physical and theoretical tools on data.	(P9) They didn't hire me to be a developer, they hired me to be an analyst. And I just felt like, I needed to do that development work in order to...streamline the process of being an analyst.
Developer	Software or physical tools with which data can be physically transformed.	
Statistician	Mathematical tools by which data can be conceptually transformed.	(P3) I mean I don't view myself particularly as a sports analyst by any stretch of the imagination, I view myself much more as a statistician.
Data Curator	Robust collection, maintenance, and transmission of the data as an end product itself.	(P10) The joke is people like me are arms dealers in the world of analytics, we deal with taking the raw data and packaging it.

Our conception of roles, on the other hand, is based in what the individual is expected to contribute to the breadth of work in their context. Table 6 presents the significant roles sufficient for describing our interview participants. In these descriptions, each role is an emphasis of responsibilities, not necessarily a complete description; our participants often describe their work with elements of all of them, though their specialty and experience with one set of skills are often more developed than others. Consider someone like P2, a university student collaborating with a team. In this example, an individual might inhabit the role of analyst in both contexts, but their responsibilities as a researcher demands coursework and research papers for an academic audience, whereas their context as a team collaborator will demand actionable digests of the data, even if they're working in the same data for both contexts. Likewise, the contexts confer special privileges to those inside them—the student has easy access to other expert scientists in a variety of fields, the team collaborator to more sensitive data at the coach's discretion.

5.3 By what means do analysts learn the practice?

In our survey, we asked sabermetricians to rate 10 different activities or resources, derived from the interviews, on how much each one prepared them for their work in the past year. As shown in

Figure 1, formal education, which we have already covered, is considered more than moderately important by 43% of the sample, There are five responses in the set which claim greater influence: hands-on experience in personal and professional projects, peer interaction, Q&A websites like Stack Overflow, and blogs or social media.

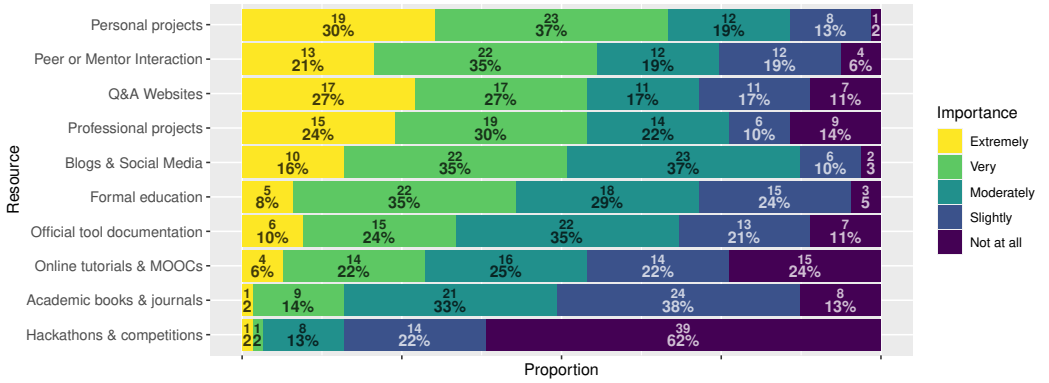

Fig. 1. Survey: "Please rate the following resources on how important they were to prepare you for the analytics work you've done in the last year." (Baseball analysts, n=63)

Of the top five, two of them refer to performing specific projects to practice skills, either for personal benefit or for professional responsibilities. Though general, these responses represent less a specific resource and more an activity which demands intentional practice. Our interviews corroborate that learning by doing is more valued than learning by reading or observation. Participants emphasize that you "need to have a problem that you solve" (P4), because the boundaries defined by a specific problem also contextualize a practical skill in a real-world situation so you "can immediately get something useful out of this" (P7). Personal projects were therefore an effective motivating force, and though no one explicitly defines the scope of worthwhile personal projects, many participants talk about this idea as "tinkering" (P2) or "getting your feet wet" (P3), implying the acceptability of small projects. Others of the top responses, including as "Peer or Mentor Interaction," also represent a form of learning through participation which can support specific projects through exchange; we discuss this more in Section 5.4.

The lower-rated resources include MOOCs, hackathons, and academic texts. In a free-form submission box, two survey respondents identified other resources: while one listed forums and video channels, the other clarified "books that are not textbooks." The interview results also drew a distinction between pedagogical textbooks and popular literature about baseball, but sabermetrics also has resources that blur this distinction. For example, three participants explicitly referred to textbooks that teach data analysis by means of baseball data, such as *Analyzing Baseball Data with R* [38], just as P10 cites Thorn and Palmer's *The Hidden Game of Baseball* as a general audience book "that had a bit of proper statistics and a bit of proper game theory in it."

What distinguishes effective resources for learning analysis? In Figure 2, we present the survey responses to whether baseball analysts want documentation to include specific features or information. Most options are generally favored and none have more than 3 respondents who disfavored them. For example, over 90% of respondents want documentation to include common errors and their fixes as well as sample use cases when the software is appropriate, whereas less than 60% of

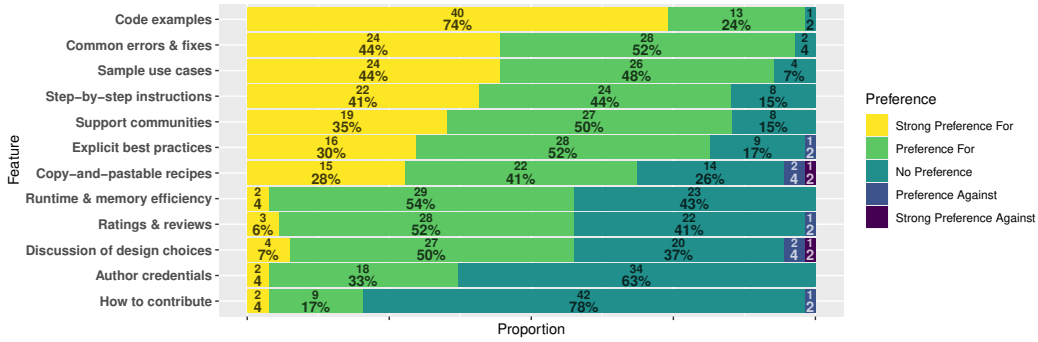

Fig. 2. Survey: "How does the inclusion of the following types of content in documentation affect your choice to use the tool or docs? "Preference for" means you want it included, "preference against" means you'd rather not see it." (Baseball analysts, n=54)

participants need discussions of the software's efficiency. We also asked survey participants what the biggest problems were with learning resources; we list the most popular selections in Table 7.

Every interview participant and nearly all survey respondents wanted demonstrative examples in general software documentation, and over half of survey respondents in Table 9 called the lack of code examples the biggest failure in most documentation. Robillard and DeLine's research [53] found that examples should small be small with multiple methods, and our participants organically corroborate this directive:

> (P7) I would say a good example would just do small things, it would show you how to use, say, like the different function arguments...not only using it in isolation but using it in correspondence with other things.

Table 7. Survey: "What do you see as the biggest failures in most documentation?" Multiple responses permitted. Top 10 selections. (Baseball analysts, n=54)

Barrier	#	%
Not enough included code	29	53.7
Incomplete feature descriptions	20	37.0
Difficult to find online	16	29.6
Too much information / wordiness	15	27.8
Out of date	14	25.9
Expects prior experience	13	24.1
Inconsistent feature descriptions	11	20.4
Not enough community support	10	18.5
Inability to search for keywords	8	14.8
Not open-source	7	13.0

The value of code examples is in how they displace the burden of "trial and error on my end" (P1) off of the learning developers and put it on the documentation writers, saving developer time overall. However, although the copy-and-pastability of an example or larger recipe can contribute to saving time, it can be a dangerous shortcut that reduces reusable skills to "just a bunch of recipes that [learners] try to mimic" (P8).

Some interview participants also consider the familiarity of the documentation and example data used to demonstrate the library, leading some to prefer standard example data over new or randomized data generated just for the example.

> (P2) So, for example, in statistics the *iris* dataset is used a lot... I know that data set, I've seen that dataset in hundreds and hundreds of packages and examples, and so I enjoy it when they use that dataset because I can look at it and understand what is going on in the documentation.

In navigating data analytic software, understanding data and understanding code are connected but distinct tasks to understand why a particular software approach is valid. Choosing a popular, already-understood data source removes the analyst's burden of learning the conditions and

assumptions of data production, freeing them to see with more clarity how the software acts on it. Familiarity also extends to the structural consistency of API documentation, for which "each function or library is described in a similar fashion" (P1) and makes it easy for developers to quickly determine, for example, to what part of document they can jump for the examples or vignettes.

5.4 Community Pressures and Influences

Figure 1 affirms that participation in particular analytical communities is important to an analyst's overall education. For example, every interview participant includes blogs and other websites for baseball and/or analysis, such as Baseball Prospectus and FiveThirtyEight, as a way of staying up to date with the sabermetric state-of-the-art. However, beyond being just for reading other people's work, they also allow analysts to publicize their own research and foster public debates. Some websites like FanGraphs[9] dedicate space to community contributions rather than depending solely on contracted writers. The free access to a lot of data and tools led one participant to analogize with the voluntarily collaborative spirit of open-source software communities: these websites allow "researchers to work their ideas out in public" (P6) before the insight is accepted.

Not everything is free, however, and the extent to which sabermetrics acts as a community is influenced by how openly resources can be shared, as shaped by the context of work. For example, although sabermetric work may be done for personal development, it can also be published with an intent to make career moves into higher baseball institutions.

> (P8) Nowadays, you want to work for a baseball team, you need to have some evidence of what you can do through. It can be a published paper but if you've written a blog, that's useful for the team to think in terms [of how] we can interview them.

Although professional employment improves access to proprietary data, it limits what an analyst can share with the sabermetric community in general. After all, the sophisticated data that teams collect personally could allow other teams to grasp a competitive advantage. As a result, inter-team communication is discouraged: "there are a lot of bright, bright people but you're not hearing about their work because of the nature of the business" (P8). Nevertheless, other sources of data are collaboratively maintained, as in the Retrosheet database[10] with discussion groups for asking questions and quality maintenance.

Furthermore, the dissemination of analytics toolsets is shaped by the way a community frames itself. In a community that frames itself primarily of analytics and statistics and not of the material or programmatic tools to apply those methods, the discourse emphasizes development of the former over the latter:

> (P7) Whereas with something like the baseball community, where the community is just centered around data, not really centered around specific types of software, you have more fragmentation in terms of how people tend to approach problems.

As we discussed in Section 2.2, statistics is already an accepted language in sports and, as such, the casual fan is more likely to have exposure to the relationship of batting averages to overall success than they are to how programming constructs can calculate these. Even for niche baseball and sports analytics discussions online, there's a lack of "collective discussion specifically about tooling" (P6) or "what's going on under the hood because...the appeal is to a much more broad audience" (P3) which does not assume programming familiarity.

When these skill-based discussions do occur, several of our participants emphasize that it is not simply what resources a community shares, but the attitude in how they share it. A supportive community in their skill development is one of the primary ways they learn and spread ideas:

[9] https://www.fangraphs.com/
[10] https://www.retrosheet.org/

> (P6) I think people who make programming languages or people who make frameworks or libraries are starting to understand is that, like, your ability to make newcomers feel excited about what your doing and about what they could do with your thing, almost like a marketing effort, is instrumental to the ongoing success.

A skill- or tool-focused community that is "incredibly nice and incredibly empathetic and more than willing to step out of their way to help" (P7) compared to the alternative can be a critical point for attracting new members.

6 FINDINGS FOR RQ2: ANALYST SOFTWARE SKILL PRACTICE

Here, we examine how members of the community produce software as an end in itself while trying to acknowledge, first, the variability in software engineering skills that our sample possesses, and second, the variety of project scopes that can be undertaken. There are existing, mixed-discipline processes for analysis and development [2, 37], but in our sample, most analysts use only their own ideas and experience to organize their work. Without subdividing our sample, our survey asked respondents (n=71) whether they used any data science processes: 59 participants (83.1%) said they had no consistent process, compared to 6 (8.5%) which used CRISP-DM and 2 (2.8%) who used KDD. For this reason, we do not give preference to any pre-defined process.

6.1 Requirements & Design: Clarifying Needs

The SWEBok defines requirements as "the needs and constraints placed on a software product that contribute to the solution of some real-world problem" [5], and design as "the process of defining the architecture, components, interfaces, and other characteristics of a system or component" [1] to make the resources comply with the requirements Our data describe factors that distinguish useful projects from poor ones, the communities that influence project direction, and constraints on the overall scope of projects.

6.1.1 What real-world problem does a project solve? In an analyst mindset, the central problem is not the software product in itself but the improvement of the game itself. However, there are many elements that can be addressed. Many analyses orient around modifying team strategy for a competitive advantage: "How many more runs is this finding going to give you over your competition?" (P3). This includes optimizing biomechanical performance for individual players, such as using sensor data to "classify pitches...based on release speed and how much the pitch breaks in both directions" (P2) to diagnose the weaknesses in a pitcher's repertoire. It also includes general team strategy: whom to buy or sell, and where to put them. On the other hand, other analyses are more concerned with the cultural conversation about the game, or about how entertaining or historically rich baseball can be. These include analyses about "the main variables influencing the length of games" (P8) or the comparative assessment of Hall of Fame candidates.

Therefore, many of our interview participants frame projects with questions and hypotheses, which constitute high-level functional requirements for the software that supports an answer. The requirements discovery is an exploratory process that values *flexibility* in discovering what that question should be. From experience, they find that "the question you start off with it will change once you start doing some exploration" (P8), or greater familiarity with the tools and data reveals that the original question is unfeasible. In consultancy positions, it can also be an intentional risk-mitigation strategy when negotiating an outcome:

> (P1) The end goal is that you will have additional tools to develop your pitchers and intentionally left a little vague and not specific because I don't, I don't want to give them something they'll look at and say, well this isn't exactly what I wanted, and then they're disappointed with it.

However, unclear requirements can be a barrier when it takes the form of "formulating a half-question [and] doing some work that ends up being not important" (P5). As far as independent

analyses go, then, many sabermetricians must maintain a posture that fosters well-formed questions but can update the inquiry for new interesting directions.

From a hypothesis-driven angle, the sabermetricians engage in creating several forms of analytic reports: summary reports of their work, comprising statistical models explained for a non-analyst or mixed-expertise audience, visualizations of data, and other mixed-media evidence of the overall point. However, as we have remarked in Table 6, larger sabermetric enterprises encourage specialization; though questions are the goal, there are new tools and infrastructures that expedite the question-answering process and improve information sharing. For data curatorial work, sources must be identified, crosswalks between existing databases built, and discrepancies or reconciled reconciled. This can be a substantial responsibility, as for P10, yet can be explored by hobbyists, as P6 describes the common sabermetric project of a game log scraper or parser for their fantasy league:

> (P6) We were basically doing all the stuff you would do on Yahoo, ESPN or CBS on their fantasy websites, but in our website, Justin Verlander threw six innings, allowed two runs, he gave up a homer. That's a stat in our league... [Major League Baseball]'s data is pretty good but it's not always super clean...so some statistical events had to be extracted from text parsing which was fun, and by fun I mean not fun.

In terms of software development, those who collaborate with teams, such as P2, P4, and P9, may build large-scale projects like "infrastructure for getting the data imported [to and from SQL databases] and...infrastructure for building the web app that displays the data to the users" (P9). These infrastructures may be undertaken agnostically of any particular question but nevertheless "streamline the process of being an analyst" (P9). For many of the independent sabermetricians, exclusive specialization is not possible, so they take on a each role themselves to varying extents.

6.1.2 Who is involved in the process? The context of the sabermetrician—be it a team, a consultancy, or independent work—influences the kinds of collaborations they must navigate. Hobbyist workers, like P6, are of course free to involve as many or as few friends as they want, and academic researchers (such as P3, P8, P10), in our interviews, do not involve many other researchers and perhaps some publishers and reviewers of finished work. On the other hand, team employees and consultants coordinate directly with small groups of analytic peers but be responsible for supporting dozens of others in the office. For example, we spoke about collaborative experiences with both P1 and P2: P1 worked alone, whereas P2 worked in a group of about 12 university students with a few mentors. Likewise, P9's experience as an employee years before the interview described a small analytics group of one or two, growing over time, whereas P4's experience was in a small analytic enclave of 7 enmeshed in the broader organization of the team's front office.

In team-oriented situations, the analysts we interviewed are not working with the players of the game but instead with "coaches or player development people" (P4) or other members of a team's front office. It is, first, partly a matter of physical limitation: "there's no way I can have a conversation and go through the report with 150 players in the course of a day or two" (P1). Two, the analytic result in itself is not an answer to be incorporated totally and immediately but evidence in a negotiation that will go on between other stakeholders in the team.

> (P1) I look at a pitcher's arsenal and say, 'Okay, these are pitch sequences and the locations that he's going to be really effective throwing.' And then you take that to the pitching coach and the pitching coach says, 'Okay, actually, I've got a one page two page philosophy that I share with my pitchers and here it is.' ...So then it's a matter of well, we'll have to go back and let me sync up with what I've got with what you've got, and then I can put together pitch sequences that match your philosophy and what you tell your guys and syncs up better with what you want.

In negotiating tasks, baseball analysts have to navigate a tradeoff in preserving freedom and pursuing valuable tasks. Many sabermetricians, industry or not, retain a fair amount of intellectual freedom on deciding how to accomplish the work.

> (P9) There was a fair amount of that, of management coming to you with, we want to see this, can you create this. But then there was also a fair amount of self-directed stuff where it was like, oh, I just read this article online that has this interesting new statistic like, let me go add that to the website.

Though this may be true in certain contexts, one of the biggest professional barriers reported by others is that "there are way more projects than there are people to work on those projects" and the lack of human resources forces an analyst "to prioritize and pick the projects that are going to give you the biggest bang for your buck" (P3).

6.1.3 Are there software design artifacts? In our interviews, references to specific, non-code design artifacts were rarely codified into anything formal. Some generic solutions were provided in terms of diagrams, such as a "box representing a data frame, an arrow pointing to a new box" (P1) or pseudocode or diagrams of an interface:

> (P4) If there's like a couple other people in the office who are users then I would try to talk to those people...And then I would tell them what I think the program is, and then I would ask them questions about all of the features...I would draw out what that report would look like, or what the application would look like, so it's like a mock-up of what the end-product would look like.

Table 8. Survey: "Before you begin programming, do you spend any time making designs for your analytics scripts?" Multiple choices permitted, but the last two are exclusive.

	Total		Baseball	
	#	%	#	%
Total	80	100.0	46	100.0
I scribble informal notes.	41	51.3	18	39.1
I describe it in conversation.	36	45.0	20	43.5
I draw diagrams.	26	32.5	10	21.7
I write pseudocode.	21	26.3	12	26.1
I describe it in textual documents.	13	16.3	8	17.4
Thought only.	15	18.8	12	26.1
No design.	8	10.0	5	10.9

Our survey results, shown in Table 8, corroborates a trend towards informality. For the baseball analyst sample, over a third say they do not make concrete designs before beginning the development process. Of specific design artifacts, more tend towards informal notes or conversational explanation, fewer towards persisting diagrams or documents.

Whether or not designs are recorded, our interview participants were generally indicate independent a general design phase "before ever jumping in and ever sitting in front of a computer actually writing any code" (P1), but coding is quickly introduced. Our participants take advantage of process flexibility to not lock themselves into one approach; this is often accomplished by uniting requirements, design, and some construction under the single step of exploratory coding as distinct from "real" coding:

> (P2) A lot of the thinking also has *light* coding that's in it, so like taking 10 observations and just making sure that something fits or something like that...but the *actual* coding where I have what I want to do, an end goal, and a method that I'm going to fit, I would say that that's less than 50% of the time. [emphases ours]

This phenomenon, which has been explored by other researchers [28], underscores a non-linear progression through phases of software development. We also distinguish it from full-featured prototyping, which some of the participants used to test feasibility in paths for development "basically to see if it's even possible" (P6). Both exploratory coding and prototyping are practiced as a way not only to allow analysts to become more familiar with the data and the tools, but also to test the feasibility of research and development paths. At least half of the 10 interview participants describe their design approach as experimenting with tenuous code.

In this way, explicit design documents are secondary to active trial and error and iteration for figuring out what works best. In some ways, trial and error not only describes learning the right path but the creation of design artifacts in retrospect:

> (P5) There's a lot of trial and error in that, and so just adding comments in certain lines for like why I did that, or like what didn't work, it's almost just like talking to myself productively.

The products of exploratory coding, although not necessarily intended to be a final product, can serve as a design story; one participant clarifies that it is worth keeping even incorrect approaches, as the trail of work "is almost like documenting your thought process." (P5)

6.2 Construction & Maintenance: Making Intentional Software

By construction, we refer to the behaviors that directly create executable software. Maintenance includes the activities to "modify existing software while preserving its integrity" [5]. Given that our discussions of design and requirements highlight the place for preliminary and exploratory coding, distinctions between phases did not emerge cleanly throughout our data. Primary modes of work remain flexible approaches like trial and error. To some extent, this is true of professional software engineering too, as the SWEBoK notes that "[a]lthough some detailed design may be performed prior to construction, much design work is performed during the construction activity." [5] This difficulty might also stem from the differences in frames supplied by software literature and those from the interview participants:

> (P3) I feel like the coding is the secondary concern for me. I'm more concerned with what statistical methodology I'm going to be using.

Nevertheless, since we selected participants on the criteria of programming, construction played a part in every conversation.

What are common priorities? One theme in building software was to prefer reuse, when possible, over writing from scratch. Code reuse takes many forms: invoking shared code libraries to "leverage the code that other people have written" to "write as little code as possible" (P9); building off the generalizable elements on analyses you wrote before; or using learning resources like recipe books, pedagogical blogs, or Q&A sites to see what functional code looks like for a given purpose and using that as a reference point. Reuse is also encouraged in that many analyses culminate in visualizations of salient baseball concepts with minor modifications.

> (P3) If I'm like graphing the strike zone, I might have like a function that does and does that. But for you know like a left-handed hitter or a right-handed hitter, all have like a specific strike zone that gets plotted or whatever. So, if you take a look at my [*project name*] repository on GitHub, you'll probably see like the same functions being reused over and over again to plot the strike zone.

If the analysis is not contributing to a long-lasting platform for more analyses, some nonfunctional traits can be deprioritized. As an example of a expendable trait, concerns of efficiency were largely secondary in individual analyses (not necessarily infrastructures) because many typical analyses are not computationally intensive, where "even with inefficient code, the runtimes are at the most seconds, never end up making it to minutes, hours or days, so it doesn't end up being super important" (P2). However, traits that enhance program comprehension, such as readability and structure, allow developers to overcome barries of complexity especially when coming from a nonprofessional software background. They can also encourage other positive attributes, as P6 discusses, "I knew working alone the only way to have reliable code was to make sure it was readable."

What are their tools? In alignment with other data analyst surveys [25], our survey and interviews corroborate the dominance of specific software tools, including programming languages and data manipulation technologies. Among 42 baseball analysts responding to this survey question, 31 (73.8%) make use of Excel spreadsheets and 27 (64.3%) of SQL, the highest response rates of all tools in our sample. The dominant programming languages are R (25, 59.5%) and Python (21, 50.0%); 11 (26.2%) use both and only 7 (16.7%) use neither. Others languages like Java, JavaScript, and MATLAB garnered 3 (7.1%) responses each. A few visualization platforms also figure into analyst workflows, predominantly Tableau (9, 21.4%) and SAS (7, 16.7%). For version control, remote

versioning interfaces like GitHub are the most popular general solution for saving scripts (18, 42.9%), iteratively saving under different filenames (14, 33.3%) or local versioning system, like git, on a personal computer (7, 16.7%). Only 5 (11.9%) say they do not version control at all, maintaining one ongoing file.

Our interviews agree with the survey: the dominant languages that our participants explicitly discussed are R (used by at least 8 of 10 interview participants), Python (6/10) and SQL (6/10). Other languages like JavaScript (2/10) occasionally came up, often in the context of platform or visualization design. Participants cited specific R or Python libraries, such as Shiny, ggplot, Jupyter notebooks, and R markdown (2/10 each). Spreadsheet programs like Microsoft Excel were often recognized but had conflicting evaluations. Though three of our interview participants brought it up as a useful tool for viewing raw data, many contextualized it as a tool they used to use but had migrated away from in their growth as analysts:

> (P1) As you get into a little bit more interesting things, like spray charts...or the heat map of the strike zone, those sort of things are pretty easy to do in R. I'm not exactly sure how you do it in Excel simply, so it's natural to kind of migrate to R...Once you move into kind of like the next step with these teams, anything you're going to do is going to have to be in R or python or any other programming, but something beyond Excel.

Evolution in tool adoption was a frequently repeated theme of gains with experience that analysts naturally undergo. Other examples were analysts alternating between R and Python depending on which had more relevant libraries or choosing the right operating system for quantitative work.

As data analysts, data technologies and sources also play a significant role in enabling work. Those situated on teams can receive data produced by sensor networks such as TrackMan. As discussed in Section 5.4, much of the highly specific data is not released generally; instead, anyone can consult open databases like Retrosheet or the Lehman Database for bulk downloads of structured data. However, while these may suffice for historical analyses, many analysts want more recent data for games just played. Therefore, scraping public sports media constitute both a popular tool and beginner's project for new sabermetricians, transforming box scores and verbal descriptions of games from websites like ESPN into structured data.

At least half of the participants discussed contributing to, publishing, or maintaining their development work on open-source platforms. For statistical development, websites like CRAN and GitHub are popular platforms for sharing useful packages. Deployment of a software library, when applicable, comes with its own choices about the rigor of review. For the developers who use R, some praise the work that the CRAN community has done in "ensuring that all of the packages on CRAN are actually somewhat working" (P7) and guaranteeing a base level of quality. Others acknowledge this as a community benefit but an individual burden to test and maintain it indefinitely, also noting that it is not the only option they have for deployment: "For my packages in R...I find it more convenient to host them on GitHub because I can make changes quickly and make it publicly available" (P8).

What challenges do they face? Participants expressed many barriers they encounter throughout their analytical work, and Table 9 contains an assessment of the most frequent responses from the survey. Tool limitations were a repeated problem and a drive for the analyst's toolset to evolve, as P2 recounts, "I started using SAS and got annoyed with it quickly, and so I moved to R quickly." For programming languages, many of the analyses are enacted through libraries uniquely designed to manipulate data and calculate statistics, and many participants express dissatisfaction with insufficient documentation:

> (P7) And there's even cases where there's R packages where there's been a paper published alongside the package. The paper might not even be readable, they might be talking about the theory of the statistical

> methodology of what their package tries to address, the core sort of implementation, but then they assume that you can sort of go from that theory and then immediately know how to use their software.

However, many of the barriers that participants discuss facing are not in software management but in the management of data, including tracking down elusive data and or managing defects or complexity in data. This is corroborated in Table 9, where issues surrounding finding, understanding, and verifying data constitute some of the most often identified barriers in analyst practice.

Table 9. Survey: "Which of these barriers, if mitigated, would save you the most frustration? Choose up to 3." Top 10 selections.

Barrier	Total Pop.		Baseball	
	#	%	#	%
Total	71	100.0	39	100.0
Detecting, handling erroneous data	25	35.2	11	28.2
Overcoming a lack of experience	24	33.8	15	38.5
Not enough time	19	26.8	15	38.5
Finding and scraping complex data	13	18.3	7	17.9
Writing documentation	11	15.5	7	17.9
Understanding complex data	10	14.1	6	15.4
Working with less experienced people	9	12.7	6	15.4
Lack of material resources (e.g. money)	9	12.7	5	12.8
Unreliable software documentation	7	9.9	5	12.8
Lack of confidence in your results	7	9.9	2	5.1

> (P4) I think the obstacles are, a lot of the new data is really big, and it has flaws in it, and it's just really hard to read it because you need like specialized software and skills. You need certain people to be able to understand it, and there are issues with it, and it's hard to contextualize because sports data can be really structured too.

In conversation, many of developers downplayed the impact of coding errors and syntactic bugs in their code as inconveniences whereas conceptual misunderstandings in the statistics or goals—"not accounting for certain things, or there's some issue with the data, or the users want something different" (P4)—are more insidious.

An undercurrent in many of our discussions is not in a material barrier itself, but most participants express doubts in experience both in themselves and in their close collaborators in managing all of the required skills at once:

> (P10) And I don't know that those types of skills are something that's really accessible to people, because people do ask me, how do I get started with this? And the best answer I can give them is, well, go take a social science degree where somebody will teach you a bit of software engineering or practical coding alongside, come back to me in four years and we can talk. That is, of course, not a very sensible answer, is it? But I can't say, well you go to this, you know, take this course here, take that course here, take that Coursera. There seems to be, there's not a one-stop shop.

This, as well as the barrier of a lack of time (which was explicitly mentioned by every participant), is not a barrier which one encounters then solves in the course of one project. It is an underlying state of affairs that modifies the severity and frequency by which other barriers occur.

6.3 Testing: Corroborating Correctness

The SWEBoK clarifies software testing as the "*dynamic* verification that a program provides *expected* behaviors on a *finite* set of test cases, suitably *selected* from the usually infinite execution domain [emphases from source]" [5]. However, when we asked our participants how they earn confidence in their own work, we noted that testing in practice conflates forms of testing beyond the definition: the data and underlying statistical methodology can be tested along with the software estimating those statistics. The different modes of testing are not mutually exclusive, as some techniques indicate errors without indicating the source.

The statistical methodology informs the design of the software and can be tested by software, but good software design will not make unsound statistics meaningful. An analyst has a number of concerns in testing statistical methods, furnishing techniques from backgrounds in foundational

statistics and data-mining like cross-validating training and testing sets. For example, one concern is testing whether the chosen method is appropriate for calculating the intended statistic; issues like these can be detected through comparing empirical distributions and outliers against a statistical test's assumptions or "[estimating] parameter values to see if you can get values similar to what you start off with" (P8) in known datasets. At the highest, most informal level, however, the importance of intuition based on solid domain knowledge emerged as the first defense of the work. Consider P2's description of a classification problem, wherein they attempt to assess a pitcher's repertoire:

> (P2) So for the pitch clustering one, I was then able to, once it effectively clustered the pitches, I would do a series of like 3 to 5 plots that were the different variables and how those variables interacted. ...And then we talked to the team, and they go, he throws three pitches. Oh, okay, well we have these three pitches and they look like this. And they go, yup, that looks good, and so now you can do some analysis on the pitches.

The "base validity" (P8) or "economic sense" (P10) of the results here can be assessed by people who view the software even as a black box, though it is especially helped by visualization. As long as the viewer has domain knowledge, they can say whether there seems to be sense in the results or not. Sometimes, however, the problem cannot easily be traced to either the statistics or the program.

Testing as a programmatic concept is acknowledged throughout our sample but unevenly applied. Whether testing is performed depends on the scope of the research project being built, since projects that are primarily one-off investigations might not warrant the extra effort. It also depends on whether they're acting in a professional capacity or out of individual curiosity, as P6 contrasts, "At work I write tests on everything, but I say like on my personal work, I write tests when I think it might be important later." Analyst-developers in team and professional contexts in our sample do acknowledge the use of testing infrastructure in their general work, including sandboxing interfaces on separate devices before deployment and unit tests with a care for edge cases.

Practice does not always include it as its own phase for a few reasons. Some participants acknowledge being content with only lightweight or opportunistic programmatic testing for the limited resources and time with which they work.

> (P4) We don't necessarily have that much extensive unit-testing-like stuff ...Just, like, all the data stays intact and functionality works as expected, and that edge cases that are always like, trying and trying to break the software. I think we could do more with testing stuff, but I think it works for us right now in what we're doing.

For those who express that they believe they should test more, some are not sure where to go to find testing infrastructure for their data and statistical analyses. Part of this lies in the invisibility of their use within the communities they participate in:

> (P3) I want to learn how to, and I need to get better doing unit testing, and something that's to get more into the statistics world, but no one I know uses unit tests.

Even in cases where some testing frameworks are known, another participant notes the "current infrastructure...could be greatly improved" (P7). Therefore, in this subject particularly, the concepts of software testing have limited penetration, leading to a lack of accessible, popular testing frameworks from the perspective of our participants.

Our survey asked the extent to which participants "test the validity of your results (more than half the time)" at a high-level, focusing less on the specific objectives of testing—unit testing versus acceptance or regression testing, and so forth—and more on the formality with which some testing process occurs. Of 42 baseball analysts, nearly all (39, 92.9%) used personal intuition as a gauge of correctness, though over half still consulted with other people's intuition (25, 59.5%) or informal testing like print statements (24, 57.1%) to develop confidence. These, however, largely refer to programmatic testing, whereas another response from statistical literature, cross-validation of training and testing sets, nevertheless garnered a majority positive response (23, 54.8%). Only 10

(23.8%) of the respondents in this sample claimed to use formal testing methods like test suites in their typical projects, and no one in our sample responded that they did no testing whatsoever.

7 DISCUSSION

The previous sections have presented data that occupy two themes: one, conceptual backgrounds and resources that contextualize analyst work, and two, software practices that emerge from these contexts. In this section, we reflect on our observations for these themes, fitting our data into what other researchers have found.

7.1 Types of Sabermetric Work

Within a community, we can raise the question: How do members of the community complement each other's work? The roles we present in Table 6 are consonant with some of the other models proposed in data analytic literature. For example, Kandel's classification—Hackers, Scripters, and Application Users—uses participant toolsets and the intensity of their programmatic solutions [25], whereas Kim and colleagues generate 9 clusters by the distribution of workday activities [30]. Harris and colleague's work perhaps most closely resembles our results: they identified 4 clusters for self-identification—Developer, Researcher, Creative, and Businessperson—equipped to varying degrees with 5 sets of skills—Business, Big Data, Math, Programming, and Statistics. As with ours, all of the clusters participate in each of the skills that constitute good data analytic practice, but certain backgrounds privilege an expertise in one area over others.

A focus on skills allows us, as researchers, to reflect on our classifications to see potential cases for us for reinterpreting data. For example, Harris and colleagues' inclusion of Business as a cluster underlies an opportunity to focus on, for example, the consultancies of P1 or P10 as skillsets or unique contexts to explore in the scope of data-oriented work. Additionally, Kim and colleagues' Data Evangelist cluster reflects on the presence of journalism background or media contexts in our own data, perhaps as a latent, public-facing role about the dissemination of sabermetric insight to a more general audience on websites or television.

A skill-based focus also enables researchers to explore communities by the different schools of thought that frame and inform community evolution over time. In sabermetrics, statistics and data curation have deep roots with the sport's origin long preceding software engineering, and our results suggests other fields of study, such as economics, which offer competing frames. Nevertheless, for our focus on software, we note how software skills have been integrated and necessary if one wants to move from reading box scores to comparing players with data. For people looking to learn or to teach this form of analytics, then, learning resources must be designed with the interplay of skills in mind. There exist some resources which explicitly address bringing schools of practice together, such as in textbooks or recipe books that explore statistical concepts in executable code, but there remains work in making people aware of these resources and possibilities.

7.2 Sabermetrics as a Community

Our operational definition of baseball analyst—anyone who seeks insight in baseball data—leaves room for a lot of variance within the population. Entrance into this community is not barred by proficiency, status, access, or location: we had participants from official Major League employees to hobbyists or independent researchers with no official connections whatsoever. While the permeability of boundaries allows many different schools of practice to coexist, as noted in Section 7.1, our data suggests that there is still work for synthesizing them into something more coherent. Other researchers have identified baseball analytics' fragmented history, noting that, for the sabermetrics community, "the term community can be problematic as these networked individuals chaotically played in the wake of [Bill] James' writings" [7].

Therefore, in some parts of the community, the labor may manifest more as a community of interest than one of practice [21], albeit with a long history and potential to coalesce into something longer-term. We can see this fragmentation, for example. in how participants express that the discourse emphasizes finished statistics at a neglect of process or tooling discussions to achieve them. Although we observed some efforts to better unite methodological discussion with tooling discussion on media platforms for baseball discussion, there remains resistance among some that those topics are too technical or niche. For the baseball case, this may be true for a large portion of the audience: the community is oriented around an entertainment product, and other subcommunities of baseball fans may find too deep an analytic approach at odds with the fun of the game. This is perhaps a distinction of this community compared to others in prior research, where barriers to entry based on experience select for more homogenous expectations among their populations.

Nevertheless, for community participants interested in spreading these discussions, fostering welcoming spaces and attitudes where insight can be presented alongside suitable tools seems to be more promising, especially since peer interaction is one of the most important learning resources that surveyees selected. Notable approaches from our data include leveraging social media as platforms for teaching, and co-located, temporary events like hackathons could bring peers into closer interaction, albeit our data suggested these voluntary events do not have as wide an impact.

7.3 Software in an Analyst Community

If we frame sabermetric analysis as a community of end-user software engineering, then we might expect, as Ko and colleagues identify, practices defined by "implicit requirements, implicit specifications, unplanned reuse, overconfident testing and verification, and opportunistic debugging" [32]. For the data that we have, we found that this is generally true for much analytic work: for example, requirements and specifications are less often expressed within formal designs and more with hypotheses bolstered by exploratory coding. However, when accounting for range of behaviors and roles we observed, we do not see this as a uniform description of all work within the sabermetric space. Building reliable analytic systems in a team context, as some of our interview participants do for teams or consultancies, invite participation into more rigorous or professional software development processes as the individuals or teams see fit to include.

Additionally, this community's approach to certain processes makes it consonant with research in other fields. For example, in our data, although we found barriers that frustrated testing, a few participants admit that a rigorous suite of software tests might be inappropriately burdensome for many cases, instead privileging intuition (within reason) anchored by a deep domain knowledge. This preference resonates with a central tension that Passi and Jackson perceived in corporate data science, wherein "leveraging intuition [is] an informal yet significant means to ratify results and processes." [46]. Furthermore, we find that the nature of sabermetric work requires not merely a competency with managing data in one's own software but also a working knowledge of the tools and contexts in baseball that produce it. This echoes, for example, multilayered vision in Paine and Lee's study of plots in scientific software and visualization, where testing is a creative act that "requires remembering and grasping deeply embedded elements of multiple infrastructures as unexpected, weird issues emerge" [45]. Therefore, many of the implications for the software practice in sabermetrics also resonate through the broader data analytic population.

8 LIMITATIONS

The data collection methods we used, interviews and surveys, have inherent weaknesses that constrain the way participants responded. For example, although we were transparent about the subject of the interview, we did not provide them with the questions or topics beforehand. Therefore,

all interview participants spoke from memory, not from immediate or thoroughly articulated experience, and are subject to recall bias. In the survey, we limited the ways that participants could respond into predetermined options, and this choice yields a few threats to validity. First, though we tried to choose our options based on literature and answers we received in the interviews, the options do not exhaustively represent all possible responses. We mitigated this by providing self-filled "Other" options where applicable; however, like the interviews, these depend on memory and the willingness of participants to reflect on every possible answer in their own experience. Second, our survey included instructions midway through about what projects—baseball-related versus general data analytic projects—they should be thinking about when answering our questions. We cannot guarantee that these directives were read and followed, so we cannot guarantee that the information they submitted accurately reflects that context we wanted.

For both of our samples, we cannot be sure that the vocabulary we used to describe analyst and software engineering practice were understood in the same way. This is especially significant given that, according to our data, our participants come from a variety of academic backgrounds: the way that a statistics professor understands the difference between software development and engineering may not resonate with a self-trained software developer in the industry. We mitigated this by offering several definitions for important terms in our questions (see Table 2), but there are more relevant concepts than we could define in our available space, especially when attempting to bridge different fields.

Given the grounded theory approach we took for our data interpretation, our results are naturally limited by the theoretically sensitivity and creativity of the researchers participating directly in that process. The coding process involved collecting the evidence into a more cohesive picture of practices in our sample, omitting information that did not cleanly fit even if it was significant to an individual's experience. As such, the concepts described in this paper can be complemented with other perspectives, with similar data or different, in order to collaboratively extend its qualitative usefulness among the research community. Furthermore, although many approaches to grounded theory limit the exposure to relevant literature, we did use our knowledge existing software engineering literature to formulate interview questions. Therefore, our interpretations were primed to include influences from current software engineering knowledge, whether or not those concepts would come up otherwise, although we did limit early exposure to analyst or sabermetric processes.

9 CONCLUSION

In this study, we explored a community of data analysts and their work at the intersection of analytics and software development. We chose baseball analytics as our subject because of its unique positioning as a cultural, historical, and academic field: with more than a century of public interest and development, it offers extensive datasets with expansive and passionate conversation around it. In order to supplement theories of what development work looks like, we conducted interviews and surveys that inquired about participants' individual practices in developing software to fulfill their goals.

Our results emphasize the diverse nature of a community defined more by a shared interest than a uniform approach to analytic or development work. Though the responsibilities of a sabermetrician support general analysis by definition, it constitutes a number of specializations—in development and data curation, for example—that allow for different academic backgrounds and interests to apply new frames for work. Whatever the case, the differences between individuals is mitigated by what sabermetricians share: a passion for the sport of baseball and its statistical language, extensive historical datasets, and physical and virtual communities where hobbyists and professionals alike can support or debate research. These resources make it possible for analysts, for baseball and in general, to practice skills productively through personal projects and to find supportive peers or

communities. We also find that the analytic posture has definite effects on the software development cycle: analysts must balance flexibility in finding interesting questions while nevertheless being rigorous with the scientific method in responsibly carrying through from data to insight. Still, there remains work in overcoming barriers in construction and testing that result, in part, from a lack of shared community practice. Therefore, as researchers and practitioners continue to develop better interdisciplinary resources, we acknowledge the potential in this community to capitalize on its particular position between science and entertainment as it continues to evolve.

ACKNOWLEDGMENTS

This material is based upon work supported in whole or in part with funding from the National Science Foundation under Grant No. 1645136 and from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government. The authors offers many thanks to the anonymous reviewers for their feedback, and to Danny Berlin, Souti Chattopadhyay, Ciera Jaspan, Ash Kumar, and Omar Sheikh for their suggestions.

REFERENCES

[1] IEEE Standards Association et al. 2010. Systems and software engineering - Vocabulary ISO/IEC/IEEE 24765: 2010. *Iso/Iec/Ieee* 24765 (2010), 1–418.
[2] Ana Isabel Rojão Lourenço Azevedo and Manuel Filipe Santos. 2008. KDD, SEMMA and CRISP-DM: a parallel overview, In Proceedings of the IADIS European Conference on Data Mining. *IADS-DM*, 182–185.
[3] Andrew Begel and Thomas Zimmermann. 2014. Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 12–23.
[4] Christine L Borgman, Jillian C Wallis, and Matthew S Mayernik. 2012. Who's got the data? Interdependencies in science and technology collaborations. *Computer Supported Cooperative Work (CSCW)* 21, 6 (2012), 485–523.
[5] Pierre Bourque, Richard E Fairley, et al. 2014. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press.
[6] danah boyd and Kate Crawford. 2011. Six provocations for big data. In *A decade in internet time: Symposium on the dynamics of the internet and society*, Vol. 21. Oxford Internet Institute Oxford, UK.
[7] Benjamin Burroughs. 2018. Statistics and baseball fandom: Sabermetric infrastructure of expertise. *Games and Culture* (2018), 1555412018783319.
[8] Longbing Cao. 2017. Data science: a comprehensive overview. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 43.
[9] Parmit K Chilana, Carole L Palmer, and Amy Ko. 2009. Comparing bioinformatics software development by computer scientists and biologists: An exploratory study. In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*. IEEE Computer Society, 72–79.
[10] George Chin Jr, Olga A Kuchar, and Katherine E Wolf. 2009. Exploring the analytical processes of intelligence analysts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 11–20.
[11] Joohee Choi and Yla Tausczik. 2017. Characteristics of collaboration in the emerging practice of open Data analysis. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, 835–846.
[12] Thomas H Davenport and DJ Patil. 2012. Data scientist. *Harvard business review* 90, 5 (2012), 70–76.
[13] Sašo Džeroski. 2006. Towards a general framework for data mining. In *International Workshop on Knowledge Discovery in Inductive Databases*. Springer, 259–300.
[14] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. 1996. The KDD process for extracting useful knowledge from volumes of data. *Commun. ACM* 39, 11 (1996), 27–34.
[15] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, et al. 1996. Knowledge Discovery and Data Mining: Towards a Unifying Framework.. In *KDD*, Vol. 96. 82–88.
[16] Danyel Fisher, Rob DeLine, Mary Czerwinski, and Steven Drucker. 2012. Interactions with big data analytics. *interactions* 19, 3 (2012), 50–59.
[17] Marc Fisher II, Gregg Rothermel, Darren Brown, Mingming Cao, Curtis Cook, and Margaret Burnett. 2006. Integrating automated test generation into the WYSIWYT spreadsheet testing methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 15, 2 (2006), 150–194.

[18] R Stuart Geiger, Charlotte Mazel-Cabasse, Chihoko Y Cullens, Laura Noren, Brittany Fiore-Gartland, Diya Das, and Henry Brady. 2018. Career Paths and Prospects in Academic Data Science: Report of the Moore-Sloan Data Science Environments Survey. (2018).

[19] Erica Rosenfeld Halverson and Richard Halverson. 2008. Fantasy baseball: The case for competitive fandom. *Games and Culture* 3, 3-4 (2008), 286–308.

[20] Harlan Harris, Sean Murphy, and Marck Vaisman. 2013. *Analyzing the analyzers: an introspective survey of data scientists and their work.* " O'Reilly Media, Inc.".

[21] France Henri and Béatrice Pudelko. 2003. Understanding and analysing activity and learning in virtual communities. *Journal of Computer Assisted Learning* 19, 4 (2003), 474–487.

[22] Felienne Hermans. 2013. Improving spreadsheet test practices. In *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*. IBM Corp., 56–69.

[23] Brett Hutchins. 2016. Tales of the digital sublime: Tracing the relationship between big data and professional sport. *Convergence* 22, 5 (2016), 494–509.

[24] Bill James. 1984. *The Bill James Baseball Abstract, 1984.* Ballantine Books New York.

[25] Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. 2012. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2917–2926.

[26] Youn-ah Kang and John Stasko. 2011. Characterizing the intelligence analysis process: Informing visual analytics design through a longitudinal field study. In *2011 IEEE conference on visual analytics science and technology (VAST)*. IEEE, 21–30.

[27] Helena Karasti and Karen S Baker. 2004. Infrastructuring for the long-term: Ecological information management. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*. IEEE, 10–pp.

[28] Mary Beth Kery and Brad A Myers. 2017. Exploring exploratory programming. In *Visual Languages and Human-Centric Computing (VL/HCC), 2017 IEEE Symposium on*. IEEE, 25–29.

[29] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The emerging role of data scientists on software development teams. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 96–107.

[30] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2018. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering* 44, 11 (2018), 1024–1038.

[31] Gary Klein, Jennifer K Phillips, Erica L Rall, and Deborah A Peluso. 2007. A data–frame theory of sensemaking. In *Expertise out of context*. Psychology Press, 118–160.

[32] Amy Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, et al. 2011. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)* 43, 3 (2011), 21.

[33] Andhy Koesnandar, Sebastian Elbaum, Gregg Rothermel, Lorin Hochstein, Christopher Scaffidi, and Kathryn T. Stolee. 2008. Using assertions to help end-user programmers create dependable web macros. In *Symposium on Foundations of Software Engineering*.

[34] Lukasz A Kurgan and Petr Musilek. 2006. A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review* 21, 1 (2006), 1–24.

[35] Sandeep K Kuttal, Anita Sarma, and Gregg Rothermel. 2014. On the benefits of providing versioning support for end users: an empirical study. *ACM Transactions on Computer-Human Interaction (TOCHI)* 21, 2 (2014), 9.

[36] Paul Luo Li, Amy J Ko, and Andrew Begel. 2017. Cross-disciplinary perspectives on collaborations with software engineers. In *Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press, 2–8.

[37] Oscar Marbán, Javier Segovia, Ernestina Menasalvas, and Covadonga Fernández-Baizán. 2009. Toward data mining engineering: A software engineering approach. *Information systems* 34, 1 (2009), 87–107.

[38] Max Marchi and Jim Albert. 2013. *Analyzing baseball data with R.* CRC Press.

[39] Gonzalo Mariscal, Oscar Marban, and Covadonga Fernandez. 2010. A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review* 25, 2 (2010), 137–166.

[40] Tim Menzies and Thomas Zimmermann. 2013. Software analytics: so what? *IEEE Software* 30, 4 (2013), 31–37.

[41] Brad Millington and Rob Millington. 2015. âĂŸThe datafication of everythingâĂŹ: Toward a sociology of sport and big data. *Sociology of Sport Journal* 32, 2 (2015), 140–160.

[42] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 126.

[43] Bonnie A Nardi. 1993. *A small matter of programming: perspectives on end user computing.*

[44] Gina Neff, Anissa Tanweer, Brittany Fiore-Gartland, and Laura Osburn. 2017. Critique and contribute: A practice-based framework for improving critical data studies and data science. *Big data* 5, 2 (2017), 85–97.

[45] Drew Paine and Charlotte P Lee. 2017. Who has plots?: Contextualizing scientific software, practice, and visualizations. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (2017), 85.
[46] Samir Passi and Steven J Jackson. 2018. Trust in Data Science: Collaboration, Translation, and Accountability in Corporate Data Science Projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 136.
[47] Emily S Patterson, Emilie M Roth, and David D Woods. 2001. Predicting vulnerabilities in computer-supported inferential analysis under data overload. *Cognition, Technology & Work* 3, 4 (2001), 224–237.
[48] Gregory Piatetsky. 2014. CRISP-DM, still the top methodology for analytics, data mining, or data science projects. *KDD News* (2014).
[49] Peter Pirolli and Stuart Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, Vol. 5. McLean, VA, USA, 2–4.
[50] Gearan Paul Rexer, Karl and Heather Allen. 2015. 2015 Rexer Analytics Data Science Survey. (2015).
[51] David Ribes and Thomas A Finholt. 2007. Tensions across the scales: planning infrastructure for the long-term. In *Proceedings of the 2007 international ACM conference on Supporting group work*. ACM, 229–238.
[52] Leah Riungu-Kalliosaari, Marjo Kauppinen, and Tomi Männistö. 2017. What Can Be Learnt from Experienced Data Scientists? A Case Study. In *International Conference on Product-Focused Software Process Improvement*. Springer, 55–70.
[53] Martin P Robillard and Robert Deline. 2011. A field study of API learning obstacles. *Empirical Software Engineering* 16, 6 (2011), 703–732.
[54] Betsy Rolland and Charlotte P Lee. 2013. Beyond trust and reliability: reusing data in collaborative cancer epidemiology research. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 435–444.
[55] Gregg Rothermel, Margaret Burnett, Lixin Li, Christopher Dupuis, and Andrei Sheretov. 2001. A methodology for testing spreadsheets. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 10, 1 (2001), 110–147.
[56] Judith Segal. 2007. Some problems of professional end user developers. In *Visual Languages and Human-Centric Computing*. IEEE, 111–118.
[57] Colin Shearer. 2000. The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing* 5, 4 (2000), 13–22.
[58] Susan Leigh Star and Karen Ruhleder. 1996. Steps toward an ecology of infrastructure: Design and access for large information spaces. *Information systems research* 7, 1 (1996), 111–134.
[59] Kathryn T. Stolee and Sebastian Elbaum. 2011. Refactoring pipe-like mashups for end-user programmers. In *International Conference on Software Engineering* (Waikiki, Honolulu, HI, USA). 10.
[60] Kathryn T. Stolee and Sebastian Elbaum. 2013. Identification, Impact, and Refactoring of Smells in Pipe-Like Web Mashups. *IEEE Trans. Softw. Eng.* 39, 12 (Dec. 2013), 1654–1679. https://doi.org/10.1109/TSE.2013.42
[61] Anselm Strauss and Juliet Corbin. 1994. Grounded theory methodology. *Handbook of qualitative research* 17 (1994), 273–85.
[62] Erik H Trainer, Chalalai Chaihirunkarn, Arun Kalyanasundaram, and James D Herbsleb. 2015. From personal tool to community resource: What's the extra work and who will do it?. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 417–430.
[63] Medha Umarji, Mark Pohl, Carolyn Seaman, A Güne_ Koru, and Hongfang Liu. 2008. Teaching software engineering to end-users. In *Proceedings of the 4th international workshop on End-user software engineering*. ACM, 40–42.
[64] Ricardo Valerdi. 2017. Why software is like baseball. *IEEE Software* 34, 5 (2017), 7–9.
[65] Dhawal Verma, Jon Gesell, Harvey Siy, and Mansour Zand. 2013. Lack of software engineering practices in the development of bioinformatics software. *ICCGI* 2013 (2013), 57–62.

Received October 2019; revised January 2020; accepted March 2020