Book Reviews

ELIZABETH ZWICKY AND MARK LAMOURINE

Data Science for Business

Foster Provost and Tom Fawcett
O'Reilly Media, 2013. 366 pp.
ISBN 978-1-449-36132-7

Reviewed by Elizabeth Zwicky

Data Science for Business is an introduction to data science as it is applied to business. The book includes enough information to tell you what you can do, how you can do it, and whether or not your data scientists are crazy and/or making things up to impress you. If you have exposure to machine learning already, this book is enough to give you a start on how you can expect the real world issues to go. You'll need another book if you want to actually implement these ideas, but those books are easy to come by.

The authors clearly have real-world experience, both with the kinds of misery that occur in real data sets and with the common problems of academic data analysts encountering them. Many beautiful theories do badly when faced by the messiness of real data and questions, and worse yet, they do badly in subtle ways. Producing apparently good results that are in fact pointless or actively bad is easy, which makes for a lot of tension between data scientists and the business groups they are working with.

This is highly technical stuff, and explaining it to people who are not mathematicians or computer scientists without oversimplifying it is hard. The authors do a nice job of simplifying it just far enough. You still must be willing to think about abstract concepts with numbers in them, and not to be intimidated by mathematical symbols, but understanding the topic doesn't require higher math. (On the flip side, if you know all the math already, you're going to need a tolerance for simplified notation.)

I'd recommend this book to people on either side of the business group/data scientist relationship, or to people trying to understand what data science can realistically do for their organization. Somebody who understood this book would meet my group's criteria for "not a loser" about data science or machine learning.

You'd need a bit more to get to "useful in the fraud space," including an understanding of why you could have an overfitted model and still need more features. In case you're curious, overfit is caused by the number of features you use, not the number you have, and is worsened by having poor quality features, plus it is heavily dependent on sample size. On big data, overfit is often not reached until >100 features, which the authors do point out, but they fail to mention that you may need twice that many to get enough good ones. Naive data scientists with first-order understanding are almost always too afraid of overfit and not focused enough on features. Still, somebody worried about overfit is easier to deal with than somebody who happily reports that they have an excellent model, because it performs perfectly on the data it was built on. People really do this, to my amazement. They also really build models based on the feature they're trying to predict. This book explains how not to.

Hiring the Best Knowledge Workers, Techies & Nerds

Johanna Rothman
Dorset House, 2004. 330 pp.
ISBN 978-0-932633-59-0

Reviewed by Elizabeth Zwicky

Hiring the Best Knowledge Workers, Techies & Nerds lays out a humane and effective hiring process, including a way of developing job descriptions that actually results in something practical for all players. This is a recent digital release, which is how it caught my eye. The publication date is mostly irrelevant, although occasionally it is noticeable. Do job seekers still consult newspapers? They must somewhere, but not in Silicon Valley.

This title would be most useful for somebody in a small company because it is geared toward people with relatively little assistance in the hiring process. But the sections on writing job descriptions, tailoring them for specific places they may be used, evaluating resumes, and training junior people to interview are useful to most people, regardless of their environment. The author's suggestions are nicely balanced, and she provides good stories to illustrate why you want to treat job seekers with generosity, not rejecting people blindly for typos and mistakes. She does not mention the contentious topic of thank you notes at all.

I found it a little slow-starting, and worried at the beginning that it would all be too HR-speak, but I warmed up to it. I would have liked more discussion of cases in which you have a ton of hiring to do and may not be interested in tying yourself tightly to a specific job description, but I think the book will still be useful to most people who need to hire technical people.

Why We Fail: Learning from Experience Design Failures

Victor Lombardi
Rosenfeld Media, 2013. 214 pp.
ISBN 978-1-933820-17-0

Reviewed by Elizabeth Zwicky

Other people's disasters are always amusing, and *Why We Fail* presents a fine collection of products that failed even with plenty of starting advantages. The author ties them together as failures to delight the customer with the experience, and traces down the reasons why the experience came out lacking. He believes that better testing of the experience and the assumptions being made will fix the problem; I'm not so sure, as several of these companies seem to have made explicit decisions to favor other factors. Possibly more testing would have helped designers better understand the cost they were paying, but it's also possible that they would have made the same bets anyway.

Still, whether you agree with the author's conclusions or not, there's a lot to think about in this book. If nothing else, you should come away convinced that technical excellence, being first to market, being well funded, and being the existing market leader are not enough to save you from disaster, whereas engaging with and delighting the customer might. You will probably also be convinced that if you want to delight the customer, you should find one, give her the experience, and believe what she tells you about it.

And, if your next project fails, you'll at least be able to console yourself with the idea that you are in excellent company.

Adrenaline Junkies and Template Zombies: Understanding Patterns of Project Behavior

Tom DeMarco, Peter Hruschka, Tim Lister, Steve McMenamin, James Robertson, and Suzanne Robertson
Dorset House, 2008. 234 pp.
ISBN 978-0-932633-67-5

Reviewed by Elizabeth Zwicky

Adrenaline Junkies and Template Zombies is an excellent book for somebody who is starting to realize that project management involves a lot of interesting stuff that is not covered in normal project management books. Somebody beginning to ask questions such as "Is it me, or is this somehow a train wreck in progress?" and "Why do some teams just work better than others? Can it actually involve chocolate and foam weaponry?" and "Can a really great team still be the problem?" (Probably it's a train wreck, yes the frivolity is causally related to the excellence, and yes, teams can be misplaced.)

For me, this book is amusing, but not transformative. I recognize many of the patterns and anti-patterns, I appreciate the authors' willingness to acknowledge that issues of fit mean that something can be a positive or a negative depending on its surroundings, and I got the satisfying feeling of having experiences fall into patterns and seeing them with new eyes.

Tableau Data Visualization Cookbook

Ashutosh Nandeshwar
Packt Publishing, 2013. 152 pp.
ISBN 978-1-84968-978-6

Reviewed by Elizabeth Zwicky

Tableau Data Visualization Cookbook is not a cookbook; rather, this book is a manual arranged by concept. If it were about food, this book would have "recipes" with titles like "Dicing" and "Sautéing." (To be fair, your average cookbook in computing would have recipes like "Egg-based sauce" with a brief example about Hollandaise and a passing mention of Eggs Benedict as an example use, with a cross reference to a recipe for toasting things and possibly one for poaching eggs. Nobody wants the computer equivalent of a cookbook, really.)

A cookbook approach would make sense if the manuals were terrible, but they aren't. They are more oriented to Tableau's view of the world (you have to look up graphing things under "Building Views"), but they are better illustrated, and I prefer their format. The cookbook layout involves bars that unfortunately highlight the unchanging section headers "Getting Ready" and "How to do it . . ."

The book is not terrible, and if the manuals don't meet your needs it may help you out. *Tableau Data Visualization Cookbook* has some interesting tricks, but on the whole I was disappointed. In a cookbook format I expect either a lot of information about specific issues (like the *Regular Expression Cookbook*) or an approach that illuminates the peculiarities of the software by using tasks (like the *R Cookbook*). *Tableau Data Visualization Cookbook* is closer to the latter, but doesn't carry its examples through with enough detail or independence to get there.

Vagrant: Up and Running

Mitchell Hashimoto
O'Reilly Media, 2013. 156 pp.
ISBN 10:1-4493-3583-7

Reviewed by Mark Lamourine

Desktop virtualization has been around for about a decade now. Until recently the desktop VM systems have focused on how to create a VM and then how to power it on and off. Little details like OS installation and network configuration were generally left to the system administrator to manage just as they would

have been for physical hosts. Vagrant lets users ignore or automate those tasks as they wish.

In eight brief chapters, Hashimoto fulfills the promise of the title. The first chapter is a dusting of background, theory, and terminology, closing with a walk-through of the installation process for Vagrant. By the second page of the second chapter he has you running your first VM, a stock Ubuntu image.

The process takes longer than that sounds because of the way Vagrant simplifies OS installation by providing a set of pre-built minimal installation images called "Boxes." When you first create a new VM, you will need a network connection so that the initialization process can download the initial box. Although the boxes are small for OS images, they are nonetheless complete bootable disk images for the target operating system.

The next three chapters cover provisioning a real system on top of the base image (using scripts, Puppet, or Chef), network variations, and creating complex system simulations with multiple virtual machines. The coverage is brief but clear and sufficient. Hashimoto doesn't try to fill in every detail. He indicates that the reader will need to be comfortable with an editor or with basic Ruby syntax and moves on. The examples are clear, not too cluttered, and, remarkably, not at all contrived.

The last two chapters introduce the two major ways of extending Vagrant. The first is by creating custom "Boxes." These are just VM images (see above) with a little metadata to help Vagrant manage them, but boxes provide a means for a developer or tester to throw away a polluted environment and restore to a well-known start state quickly and reliably. New boxes can be pulled from remote Web sites and are cached locally.

Hashimoto discusses plugins last. In writing Vagrant, he has used the standard mechanisms offered by Ruby to provide hooks for extending the existing behavior, both adding new commands and altering the behavior of existing ones.

Often "up and running" style books move so fast in an attempt to show everything that a tool can do that they leave the reader without a real working knowledge of the main task. Hashimoto has written about a tool with a specific purpose and has focused on showing the new user how to do that job easily. He also manages to provide hints for the possibilities for extensions.

Vagrant is a tool that helps manage desktop virtual machines. If you're a software developer for complex Web services, you're going to want to at least look at Vagrant, and *Vagrant: Up and Running* is a really good place to start. Mitchell Hashimoto is himself the creator of Vagrant; not all software developers can also write for humans, but Hashimoto is an effective advocate and instructor for his tool.

Git Pocket Guide: A Working Introduction

Richard E. Silverman
O'Reilly Media, 2013. 215 pp.
ISBN-10: 1449325866

Reviewed by Mark Lamourine

I've been a fan of the O'Reilly pocket references for quite a while. Especially when learning some new programming language or tool, I find that the pocket reference is a quick way to get what I need without the narrative of a tutorial or the depth of a traditional user guide or reference text.

The Git Pocket Guide uses the same physical format as the pocket references, but the format is task-oriented instead of the more typical feature list. This makes sense for Git as its purpose is to help manage the process of collaborative software development.

I like the way Silverman interlaces usage explanation, examples, and theory. Git takes a significantly different approach to implementing source code revision control than previous tools, such as CVS and Subversion (though more similar to Mercurial and Bazaar). Each chapter in the pocket guide provides a brief view into how Git works when performing the task under discussion. Unlike many other tools, understanding how Git works helps a lot in understanding how to use it.

The physical and structural formats create a portable and utilitarian book that should be enough for most people who are already familiar with the fundamentals of source code control to get started and do most tasks with Git.

Puppet Types and Providers: Extending Puppet with Ruby

Dan Bode and Nan Liu
O'Reilly Media, 2013. 80 pp.
ISBN: 978-1-4493-3932-6

Reviewed by Mark Lamourine

O'Reilly has recently begun publishing a series of thin short-topic books, and Puppet Types and Providers is one of that line. The authors focus on a single advanced aspect of working with Puppet: extending Puppet by adding new managed resources. This book was timely for me, and I just wish there was documentation this good for the rest of Puppet.

At 80 pages, the book does not have much room for introduction, but Bode and Liu do manage to cover enough of the Puppet internals so that the purpose and use of the Types and Providers mechanisms is clear. Then they get right to the meat.

Briefly, Puppet models target systems by allowing the user to define a set of resources that should exist on those systems. The Types mechanism provides means for Puppet developers to create abstract definitions for new resources. Providers are the concrete implementation backend for the abstract Type definitions. Especially interesting to me is the explanation of the "suitability" mechanism that Puppet uses to decide which provider to use to execute the resource requirements.

The book only has four chapters. Following an introduction to Puppet's resource model, the exposition of Puppet types and providers each takes one of the remaining chapters. The final chapter touches on several advanced features of Puppet resource management using the type/provider mechanism. Before finding this book , I looked for quite a while for documentation that clearly demonstrates how to define new Puppet resources. I'm done looking.

Statement of Ownership, Management, and Circulation, 10/1/13

Title: ;login: Pub. No. 0008-334. Frequency: Bimonthly. Number of issues published annually: 6. Subscription price $90.
Office of publication: USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.
Headquarters of General Business Office of Publisher: Same. Publisher: Same.
Editor: Rik Farrow; Managing Editor: Rikki Endsley, located at office of publication.
Owner: USENIX Association. Mailing address: As above.
Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: None.
The purpose, function, and nonprofit status of this organization and the exempt status for federal income tax purposes have not changed during the preceding 12 months.

Extent and Nature of Circulation			Average No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue (August 2013) Published Nearest to Filing Date
a. Total Number of Copies			3479	3100
b. Paid Circulation	(1)	Outside-County Mail Subscriptions	1723	1667
	(2)	In-County Subscriptions	0	0
	(3)	Other Non-USPS Paid Distribution	1010	987
	(4)	Other Classes	0	0
c. Total Paid Circulation			2733	2654
d. Free Distribution By Mail	(1)	Outside-County	0	0
	(2)	In-County	0	0
	(3)	Other Classes Mailed Through the USPS	64	75
	(4)	Free Distribution Outside the Mail	490	219
e. Total Free Distribution			554	294
f. Total Distribution			3287	2948
g. Copies not distributed			192	152
h. Total			3479	3100
i. Percent Paid			83%	90%
Paid Electronic Copies			417	432
Total Paid Print Copies			3150	3086
Total Print Distribution			3703	3380
Percent Paid (Both Print and Electronic Copies)			85%	91%

I certify that the statements made by me above are correct and complete.
Anne Dickison, Co-Executive Director 10/1/13