Audit Analytics with R

Jonathan Lin

2020-06-06

Contents

Welcome

This is the website for Audit Analytics in R. This audience of this book is for:

- Leaders who are looking to design their environment to encourage code sharing and data products,
- Data analytics practitioners, who are looking to leverage R in their data analytics tasks.

You will learn what tools and technologies are well suited for a modern audit analytics toolkit, as well as learn skills with R to perform data analytics tasks. Consider this book to be your roadmap of practical items to implement and follow.

If you are brand new to R, it is encouraged you to read https://rstudio-education.github.io/hopr/ and https://r4ds.had.co.nz. While some foundations will be covered in this book, this book is focused on an applied view of R to the financial auditor practice.

About the author

Chapter 1

Introduction

Within the accounting and audit profession, analytics has been around for several decades, under the concept of Computer Aided Auditing Techniques (CAATs), where the software of choice was led by ACL Analytics. ACL Analytics was a significant audit enabler at the time, as it allowed direct access to analyze mainframe information that was otherwise inaccessible by mainstream software on the market. It enabled audit teams to obtain transparency in analysis, a rigorous audit trail, and even automation of scripts.

As computers, data analytic technology and accessibility of coding in the Accounting practice has become mainstream, there is far more tools that enable auditors to become far more powerful and self sufficient than ever before. Tools that are typically reserved for software engineers and statisticians have empowered financial auditors to expand their breadth and scope.

A traditional internal audit team would consider themselves to be consumers of information, limited by flat files sent by emails from their stakeholders. While most internal auditors have considered data analytics in one way or another, the realm of possibilities and challenges have outpaced audit shop capabilities. The expectation now is for auditors to be fully integrated into the business, and contribute directly to the management of the financial and IT risks the company faces on a regular basis.

The most effective way to meet this new standard is to implement a current data analytics architecture and empower your team to leverage modern data analytics techniques.

1.1 Make the right thing easy to do

The right set of technologies enables your team to propel forward, and should amplify your teams efforts. By thoughtfully choosing your tools and encouraging

sustainable processes, your team will create a positive cycle of development, learning and deployment.

1.1.1 Code-based development

Consider the creation and auditing of a spreadsheet, where every row and column has the potential to be manipulated. Following the motto, 'trust but verify', an auditor would need to examine the values, the formulas, the relationships, and keep a sharp eye out for manual adjustments. Once you consider that it becomes a manual process to also verify the source of data in the spreadsheet, and there is no built-in tracability to how the spreadsheet is used after it has been created, it contributes to the madness that is the spreadsheet ecosystem.

While it is easier to superficially consume information from a spreadsheet, to understand the inner workings is a tedious task into itself.

Contrast this to a code-based environment. To achieve anything in code, you need to be explicit and specific on the mechanisms taken to reach the end state. Each line will tell the program what inputs it needs, how it is processed, and the collection of lines tells you what is achieved. The beauty of this is that it also tells the reader exactly what was executed to achieve the end result. A code based environment is inherently self documenting.

Once a baseline code has been established, finding ongoing changes becomes trivial. Similar to black-line functionality in Microsoft Word or Track Changes in Google Docs, its far easier to detect how code (and the process it supports) has changed.

Perhaps with some merit, code can be difficult to read at times, as it is still a completely different language. **Notebooks** address this problem. Notebooks are interactive renderings of code, containing not only the code, but can also include sections of commentary as to why something was done, and can include data visualizations and interactivity. It is the ultimate form of auditability.

1.1.2 Automate relentlessly

By writing your routines in code, then the next logical step is to automate. Automation not only frees up your time from performing a task, but it also frees up your mental critical thinking and creative processing power. Every professional has a cognitive load - don't waste it on routine tasks.

A common use case for automation in audit is the creation of a data mart that is relevant to auditors. Now imagine that instead of needing to email HR for the latest employee list, an always up-to-date database makes research a snap for internal auditors. Instead of asking the vendor management team on how much money was spent per vendor, having this information in an audit mart

already available means you can spend more time thinking about how to assess these vendors for risk.

Gone are the days where you only audit a topic once a year. A byproduct of every audit is a monitoring mechanism - how do you know something has gone off the rails? Traditional remediation paths include a follow-up in the future - after that, there are few mechanisms to faithfully maintain confidence that the process is still operating reasonably.

If you've coded your audit, then you have already done the hard work of structuring how to extract your data data, finding exceptions and distributing actionable results for your stakeholders. Automation takes that one step further, and allows you to ensure the steps are done repeatedly. The value you derived initially from your audit can now be done continuously.

Once you've audited the identification of issues and the metrics around them, then you can focus on the automation the handling and resolution of them. With a true audit results database, you can the results of automated monitoring into it, enabling on-demand availability, visibility and transparency… and sets up the next step, where where reporting can now be part of that automation process.

Its a self reinforcing, positive cycle.

1.1.3 Share everything

> Tribal knowledge is information or knowledge that is known within a tribe but often unknown outside of it. A tribe, in this sense, may be a group or subgroup of people that share such a common knowledge. From a corporate perspective, "Tribal Knowledge or know-how is the collective wisdom of the organization. It is the sum of all the knowledge and capabilities of all the people". (**?**)

When you code the procedures, not only are you creating letters to yourself in the future, but you're also writing letters to everyone else on your team, even those who haven't joined. These fully contained notebooks serve as a guide to those on your team who are learning what you have developed.

The only left to do is to put them into a central source where everyone on your team can access them. While you can opt for files and folders on a network drive, more practical technologies exist. **Code repositories**, such as git and subversion, serve a purpose in your data analytics environment to track changes to code and notebooks. You can encourage your team to upload notebooks to these repositories to access the latest and greatest code that your team has developed for problems already solved. New team members can go into a repository to learn how code has solved prior problems, and become inspired to solve problems of their own.

As your team gains more experience and consistency, it may be more practical to write repeatable processes and functions instead of copy and pasting code between notebooks. Code **packages** enable you to share your code and functions to solve specific problems, templates to encourage consistency amongst team members, and are easily distributed to your team for quick installation.

1.1.4 Don't be a hero

Not everything needs to be fixed by you, in code. In a well-functioning organization where hundreds of people are supporting different software, application or databases, sometimes the best way to solve the problems found isn't directly in the work you do, but how you get others to fix systemic issues in the source itself. Key indicators of this are when you have to start hard-coding compensating controls in your code because the upstream data isn't standardized.

For example, in a typical company, employees are issued unique ID numbers. This is preferably at the source of truth - the Human Resources department. This identifier is generally considered reliable and stable, especially if is the source of truth. Consider now that you're now trying to join the data to a credit card system, where employees are issued credit cards based on their first and last name. If you take an assumption to join this dataset using an individuals first and last name, that would be a reasonable first attempt. However, last names can change over time, or individuals may prefer to go with their middle names, or even simple spelling mistakes can occur. Instead of compensating by adding different ways of 'joining' information, its more effective to work with the application's data owner to see if they would be willing to adopt the unique employee ID instead as a field in their system.

1.1.5 "The opportunity cost"

> The idea of opportunity costs presumes the fungibility of human experience: all our activities are equivalent or interchangeable once they are reduced to the abstract currency of clock time, and its wage correlate. (Crawford, 2009)

An internal auditors' largest limitation is time. The biggest barrier to the adoption and execution of data analytic talents is the fact there is 'more' perceived value in doing something else. A common excuse is that a task will only be performed once (or even worse, at an irregular basis) - auditors may instead opt to perform a manual task in an unsustainable way. Instead of learning a new skill, our opportunity cost is projected against how much time can be used have to perform this task at hand.

If your existence was highly limited by the lifespan of a fruit fly, it would be hard to argue against such a position. However, your career is long (hopefully),

fruitful and full of exciting and interesting work. Truly engaging work should challenging and rewarding, and applying coding to audit lends itself to a skill worth mastering. Using these skills, no matter how immature, will continue to pay off dividends in future implementations, no matter how insignificant or incremental. Implementing repeatable processes means not only does the end consumer receive their product quicker, but the data auditor is enabled to continuously refine or tackle another code-related problem. And that is worth the opportunity cost, today.

Chapter 2

Architecture

2.1 R and RStudio

2.2 Code Sharing

2.2.1 Git

2.2.2 Packages

2.3 Data Products

2.3.1 Galvanize Highbond

2.3.2 RStudio Connect

2.4 Data sources

2.4.1 Databases

2.4.2 External sources

Chapter 3

Setup

3.1 RStudio

3.2 Github

Chapter 4

Audit analytics

4.1 Import data

4.2 Explore

4.3 Manipulate

4.4 Report

Chapter 5

Applied audit analytics

5.1 Package creation

5.2 Continious Monitoring

5.3 Controls automation

5.4 Audit Data Mart

5.5 All-in-one toolkits

Chapter 6

Other practices to follow

6.1 Passwords

6.2

Chapter 7

Audit data products

Bibliography

Crawford, M. B. (2009). *Shop Class As Soulcraft: An Inquiry into the Value of Work.*