Effective Data Science Notebooks

Justus Perlwitz

2018-01-06

Contents

1 1. Introduction

I believe that being a data scientist is about communication first, and data science second. This means that two abilities are necessary:

- The ability to gain insight and knowledge using data
- The ability to successfully communicate what I learn to stakeholders

I have always wanted to share some of my ideas for improving those abilities and have finally learned enough to know what to write for the first time. Therefore, I have written down some of the most important lessons that I have learned on my path to becoming a better data science communicator.

Since Jupyter, IPython and Pandas are my main tools, the article is written from the perspective of a user of these tools. There are three themes that will be covered in this article.

1. **Communication with Purpose**
2. **Notebook structure**
3. **Clean Processing**

We will discuss how to **communicate with purpose**. We will see how to justify **why** a data science document, i.e. a notebook, exists, and **what** question it answers. A notebook would not be complete without answering that question and providing tangible **results**.

Notebook structure plays a big role when discussing notebooks with your client. I will share my experience with you and we will learn how to structure notebooks effectively.

Clean processing can mean the difference between a good notebook, and a **great** notebook. It is not only about the right method call to parse a CSV column, it is also about clearly documenting how you found the data, and what you did with it. Again, we will see what to do and what to avoid.

2 2. Communicate with Purpose

2.1 2.1. What question are you answering?

Writing happens for a reason. This includes writing IPython notebooks, and data science writing in general. Think about the last time you wrote a notebook. Why did you write it?

Let me name a few reasons why I create Data Science notebooks:

- A client approaches me with their data and a question. I answer their question using their data and guide them on a business decision making process.
- I develop a machine learning model and evaluate its performance. I want to explain how the model works and how it is evaluated, and how confident I am in the result.
- I analyze data quality and want to summarize my findings. This allows me to better estimate the quality of subsequent data analysis.
- There is a new data science concept that I want to understand and teach. It could be an addition to the Pandas API, it could be a new idea on Data Engineering, or it could be an interesting machine learning model.

I am sure there are many more reasons for using notebooks to share data science ideas and results. Knowing the purpose of a document allows me to quickly decide what to include and what not to include.

For example, you can add an infinite amount of subsegments when doing a segment analysis on user data. At the same time, looking at more data than necessary does not really help answer the original question of a document. Realizing that I am not working on solving the goal prompts me to get back to answering the core question in a document.

Perhaps I am analyzing how conversion rate of new and returning users are different, but get sidetracked by some interesting subsegment of returning users. That may be valuable by itself, but it also deserves its own independent treatment. In that instance, I like mentioning interesting findings to the people I work with and I suggest that these findings are worth being analysed further in a new document. This is always received with enthusiasm. Working closely with data will always give you an edge at identifying further interesting topics to research.

Furthermore, I believe that focusing on one main purpose maximizes the impact of the notebook's core result. Just providing one main chart that demonstrates an important finding has more value than a notebook containing several interesting but weaker findings. The presence of many results dilutes the impact of one result.

Consider the opposite: A document could completely lack interesting findings and never get close to answering any question. In that case, returning to answering the core question and dedicating all effort to effectively sharing results should have the highest priority.

To sum it up, a reader should always know what was asked and what was answered in a notebook.

2.2 2.2. How well could you answer the question?

A notebook is an answer to a question. The way a notebook answers this question comes in the form of a result. This result may be a chart, a table, or even a simple yes or no. What all those results have in common is something intangible: Quality. Not only does the result itself matter, but also the quality of that result. While I feel unable I would like to show a few examples of quality in notebooks. I will also show some examples where quality can be improved.

Delivering a **histogram** clearly demonstrating an insight about user behavior has a strong quality. Furthermore, if the data is cleaned thoroughly, the split-apply-combine process is reproducible, the charts are well-formatted, and the explanation and summary are easy to understand, the document will be truly valuable for the business.

Delivering a weak **yes** to the question *"Do our users exhibit behavior A"* lacks quality. Maybe the data source contained format errors, maybe the event data could only partially be joined with the user data, and maybe the explanation and summary do not clearly describe the process. All these factors together will make the document not useful.

And yes, problems with data source quality can not always be avoided. Maybe the client changed the data storage format in the middle of the year, maybe a metric changed and it wasn't properly documented.

There is something that can be done about this: **Clearly document data quality issues** with the data at hand. Make the client aware that there are problems. But most importantly, present your results in such a way that quality issues and potential caveats are transparent.

We conclude that **strong results** are a product of

1. Clean data processing,

2. Reproducible and transparent computation, and
3. Clear result communication.

That answers one half of *how well could you answer the question?*, the question of **result quality**.

2.3 2.3. Can It Be Answered?

Let us move on to the other half of answering *how well could you answer the question?*, the question of **feasibility**. Before we start working on answering a question, we need to ask ourselves whether

1. the question can be solved using the tools and skills that are available, and
2. whether the question is answerable at all.

There is a difference between questions that can be answered, and questions that can't be answered. It is not always possible to know whether a question can be answered when you start working on it. At the same time, you should always question yourself how confident you are given what you have learned so far about the problem.

Here are some questions that a client may ask. Whether they can be answered needs to be critically examined.

> **Question**: We have been collecting purchase data for 2 weeks now, and we would like to examine whether our users like chocolate or hard candy.

Answer: This is hard to answer since 2 weeks of data does not account for seasonality and sales campaigns.

> **Question**: What do our users feel when they put an item in their shopping basket?

Answer: We do not yet possess the technology to infer feelings from event logs.

Furthermore, there are questions that are unlikely to be answered successfully. This may not be clear from the outset.

> **Question**: Can you predict sales date for Quarter 1/2018, given sales data for the the past 10 Quarter 1 numbers?

Answer: It depends on whether an appropriate and robust model can be found. It also depends on whether the future can be predicted.

> **Question**: We have had a data loss in August and September, can you still perform an A/B test analysis using partial data for October?

Answer: It depends on whether the amount of data is enough to get to a statistically significant result.

So what is our takeaway here?

- If a result could not be found: **Be upfront about it**
- If a result was found: **Be transparent about result quality**
- If you achieve a great result: **Celebrate!**

Giving the client feedback when you have good results encourages them to improve their data quality even further. It also lets them know what kind of questions you can answer. This will create a positive feedback cycle between you and your client and will ensure a long and fruitful collaboration.

3 3. Structure your Notebook

3.1 3.1. Separate Setup and Analysis

Almost every notebook that I write can be divided in two big parts: the **setup** and the **analysis**.

The **setup** is concerned with

- reading in data,
- selecting rows by date ranges,
- joining user records,
- performing QA sampling of the data,

- counting the total amount of data that was read,
- setting up distributed workers,
- performing group bys,
- and so on.

To summarize: it is everything that a good data scientist requires to become productive with data. What it should not contain is the actual **analysis**.

In my experience, someone who is just interested in results will not immediately spot the difference between an IPython cell that produces an insightful result, and a line that just imports Matplotlib. This has nothing to do with a lack of tech skills. It has everything to do with the fact that a reader is interested in seeing the results first, and reading through code second. This is why I strongly believe that separating those two concerns in a document is important.

Where the **setup** step uses code to retrieve and prepare data, the **analysis** step uses data and code to generate insight. Here the density of code should be as low as possible. If you can restrict yourself to 1 line of code per insight that you generate, you will produce a lean and easy to read notebook.

As a guideline, I recommend refactoring commonly used data transformations into **setup** subsections. So for example, if you catch yourself writing

Analysis 1

```
df.query('purchase_value > 100').grouby('user_id').hist()
```

and soon thereafter

Analysis 2

```
df.query('purchase_value > 50').groupby('user_id').hist()
```

and so on, it will be wise to refactor as follows:

Setup

```
def purchase_segment_hist(df, value):
    df.query(
        'purchase_value > {}'.format(value)
    ).groupby('user_id').hist()
```

Analysis 1

```
purchase_segment_hist(df, 100)
```

Analysis 2

```
purchase_segment_hist(df, 50)
```

What you'll gain is additional clarity in the **analysis** phase for the small price of four additional lines of code in the **setup** phase.

I use the following rule of thumb: If you can, put all your code into the **setup** phase and all your insight into the **analysis** phase.

While no **analysis** will be completely free of code, minimizing the amount of it will increase readability of results. Having all critical code in the **setup** phase will ease maintenance and understandability of data processing code.

3.2 3.2. Be generous with sections

Imagine this: You're on the phone with a client, it is 15 minutes until their next board meeting. They are in a hurry and are sifting through the notebook that you have provided for them. Just yesterday they have seen an important number and now they really need it.

On the phone, you start hearing your client flipping through a paper document and realize that they are looking at a printed version of your notebook. So it would be useless to tell them to *scroll down to the middle*. What do you do? If you add sections and a table of content, nothing will be easier than that. You can simply tell them that the important number can be found in section **2.3.4**. And this is exactly what happened to me countless times and was always quickly resolved because I have made sure to structure my notebooks. To make everyone's lives easier, it is important to add the following structural elements to your data science notebook

1. Table of Contents (TOC)
2. Section Header for every major part of the notebook
3. Subsection, or Sub-subsection for every result and other important insight or step

This has the following advantages:

- It is easy to walk someone through a notebook when everyone can navigate it with ease
- It makes sharing a document and referring to parts of it easy
- Navigating a notebook on different media becomes easy (paper vs. HTML vs. Jupyter IPython)
- It allows you and your client to compare notebook versions (*"Where did histogram X move?" "It's in section 3.2 now"*)

As always, clear and easy to follow communication makes everyone happy.

4 4. Clean Processing

One important step in creating transparent and easy to evaluate notebooks is taking extra care when loading and processing data. Throughout my work I have identified two important qualities that can be found in a clean notebook.

- **Reproducibility**: Data is read in as cleanly as possible data and results are calculated deterministically.
- **Transparency**: Parse errors, imprecisions and compounding errors are transparently communicated

If our whole processing pipeline creates **reproducible** results, we have a system that we can trust and count on even when we are in a hurry. We can easily create additional notebooks that use the same data, and we can reliably compare results from different notebooks using the same data foundation. It also allows us to rerun notebooks, for example after adding a new chart, and trust that the results will look the same.

While **reproducibility** is concerned with how results are computed, **transparency** creates trust between a data scientist and a notebook's reader. **Transparency** allows both parties to understand how results come to be and how to evaluate them. In a **transparent** document we explain what could be processed and what could not be processed.

We will see how these three qualities can be implemented in the following sections.

4.1 4.1. What data was used? What period was observed?

Perhaps the most important factor in cleanly and **transparently** processing your data is to exactly document:

1. The start and end date of the data you analyse.
2. How you filter data by date ranges.
3. File paths, S3 paths, and similar, that you import data from.
4. The amount of data rows you read.

Providing this information allows you to compare notebook results to other notebooks that have been created with the same data.

I will note at this point that one subtle way results might change is if dates are filtered inconsistently. There is a difference between

```
df[start_date <= df]
```

and

```
df[start_date < df]
```

The first one looks at rows starting with and including the `start_date`, the second only considers rows with an index larger than the `start_date`. While both can be used, it is imperative to communicate this clearly. This allows for independent verification of results.

Depending on confidentiality requirements, it is not always possible to include file paths in notebooks. Documenting the exact source of data is useful. It allows the reader of a notebook to immediately notice if outdated data was used. Perhaps the company has loaded new data into the data warehouse and did not yet have a chance to inform you. If your results deviate from what is expected, it becomes an easy fix.

Finally, a really simple marker, a sort of data fingerprint, is to write down the amount of rows that have been read into a Pandas DataFrame. There is nothing easier than writing a quick

```
len(df)
```

in your notebook. Should the number change from run to run, you know that something is up. It is similar to canaries used in mining and will immediately let you know that there is trouble ahead. This also plays a role in merging and joining data, as you will see below. Ideally, you will be printing row numbers after every important processing step.

4.2 4.2. How was the data cleaned?

Data cleaning needs to be **reproducible**. It is crucial to be as precise and intentional as possible when selecting and processing data. A common process for data analysis is to pull CSV, JSON or similar data from a Data Warehouse, pre-process it to a binary format like parquet, and then use it in the actual data analysis notebook. This is done for the reason that this intermediate result can then be reused in multiple notebooks.

These are the critical points in this process:

Pre-Processing could subtly change data that was analysed. If further data is added for a particular month without your knowledge, then rerunning the preprocessing step could potentially add data that changes the result in your data analysis notebooks. It is therefore important to clearly version your data and indicate which version you use.

Parsing columns can make a big difference. Always be aware of the data format each column is supposed to have. Let your client know of any issues that you encounter while parsing data. Document how you parse columns and be explicit about number formats. Never leave parsing integers or floating point numbers to chance, not even once.

Dropping rows with n/a values or similar needs special attention and care. Sometimes you might accidentally drop too many rows because there was a parse error. Instead of dropping those rows, stop and investigate why the rows are incorrectly parsed. Other times it is perfectly acceptable to drop rows, for example when only rows with a non-empty URL should be considered. In either case, document row counts before and after.

Clean data is a continuous effort and never perfect. My job as a data scientist is to not only process data cleanly, but also communicate about data clearly.

4.3 4.3. How was data joined?

In many cases you will read user data from an event log and join it with user rows from a database. We want to ensure that this process is **reproducible**.

It is vital to record the following things:

1. Which columns were used to join?
2. How many rows on each side could be joined?
3. How many rows on each side could not be joined?

Sometimes there is an issue with user data being incomplete. If the full user data set becomes available, the amount of user data that can be associated with event rows will suddenly rise. This can have an effect on results, and therefore properly documenting the amount of joined rows is important.

4.4 4.4. What is the quality of data?

Not only do we need to get a feel for the data important in order to understand and analyze it, it is also a perfect opportunity evaluate the quality of the underlying input data. We want to be as **transparent** as possible in our evaluation.

Even assuming that all data could be parsed without any errors, we might still encounter the following problems:

1. A column containing category data might only contain one category because of a data source problem.
2. A numerical column might contain test data. Sometimes a developer accidentally leaves in a `0xBEEF`.
3. A URL columns might accidentally contain User Agent strings.
4. A string column might contain test values like `qux`, `foo`, and `bar`.

None of these problems will be obvious in the beginning. But I can tell you that they will sneak in subtle errors that will be responsible for a big cumulative error in the end. Therefore, I recommend you take the following precautions:

1. Look at category value distributions using a simple Pandas `.values_count()`. If you see only one value, you know that something is wrong.
2. For numerical data, create a histogram and understand their distribution. If you know how the data is supposed to be distributed, you can immediately spot errors.
3. Try to parse string columns that are supposed to follow some syntax. In the case of URLs, `urllib.parse` in Python 3 can quickly help out there. This is helpful even if you only work on a small uniformly sampled subset of the data you are working with.
4. Test values in string columns are difficult to handle. Manually sampling 100 to 200 rows is often necessary in this case. Again, it is important to take a uniform sample.

I am confident that there are many more useful techniques for quality evaluation. It is always good to keep a lookout for problems with your data. Never take results for granted and be ready to justify every single step. This will make your data processing pipeline as robust as possible.

5 5. Conclusion

How do you write notebooks and what strategies do you use? What do you use to clearly communicate your intent and your results in your notebooks? I would be excited to learn more.