Journal of Data Science xx (xx), 1–13 DOI: xx.xxxx/xxxx_xxxx
July 2023

My wonderful paper

Abstract

Unexpected results in data analysis can prompt researchers to examine data quality and analysis steps. While some researchers can typically diagnose issues effectively with a few checks, others may struggle to identify appropriate checks to diagnose the problem. [an example of check] These checks are often informal and difficult to trace and discuss in publications, resulting in others questioning the trustworthiness of the analysis. To address this, we formalize the informal checks into an *analysis plan* that encompasses the analysis steps and (the unit tests): one test for whether the result meets expectations and multiple tests for checking the analysis. We then present a procedure to assess the quality of these unit tests based on their accuracy and redundancy on simulated versions of the original data. The accuracy is assessed using binary classification metrics, *i.e.*, precision and recall, while redundancy is calculated using mutual information. This procedure can be used to conduct a sensitivity analysis, compare different analysis plans, and to identify the optimal cutoff point for the unit tests.

Table of contents

1 Introduction

- emphasize trustworthy data science since it is the theme of the special issue

In a data analysis, an experienced researcher can often quickly assess whether a result meets their expectation, while others may be unsure of what to anticipate from the analysis or how to recognize when results diverge form expected norms. (These expectations are often based on prior knowledge, domain expertise, or common sense and they can often be framed into a Boolean question that can be answered by a unit test. For example, in a linear regression model, one may expect the p-value of the coefficient to be less than 0.05. On top of the expectation on the final results, researchers may also have expectations on the intermediate steps of the analysis process. For example, one may expect the data doesn't contain outliers, etc. These expectations are often checked early on to prevent an unexpected propagate through the analysis process.)

This expectation-setting process is often implicit and rarely documented or discussed in publications. (There are most publication requires open source code for publication, however, it is no instruction or guidance on checking …). Yet, making them explicit is crucial for 1) helping junior researchers interpret results and diagnose the unexpected, 2) providing checkpoints that support independent replication or application of methods to new data, 3) aligning assumption across researchers from different backgrounds.

In this paper, we formalize these informal checks into an *analysis plan* comprising the analysis steps along with the associated unit tests. This formalization allows us to examine assumptions made about the data during analysis and to compare different unit tests in diagnosing unexpected outcomes. We then introduce a procedure to assess the quality of these unit tests based on their accuracy and redundancy across simulated version of the original data. The accuracy is assessed using binary classification metrics – precision and recall – while redundancy is measured using mutual information. The workflow provides

numerical guarantee of the consistency of the expectation and the data for the analysis plan. This means that given the assumption of the data generating mechanism, the analysis will provide the outcome we expect. In data analysis, analysts are not necessarily collecting the data and they can only make reasonable assumption about the quality of the data. This method we propose provides a numerical guarantee, based on simulation, that the analysis will produce the outcome we expect, given the assumption of the data generating mechanism. This provides transparency of the analysis and build trustworthy data analysis.

The rest of the paper is organized as follows: Section 3 describes the concept of analysis plan in detail. Section 5 provides examples of analysis plan [more details]. (need another section here or before examples?) Section 7 concludes the paper.

2 Literature review

2.1 Diagnosing unexpected outcomes in data analysis

(Peng et al., 2021) describes three pedagogical exercises of introducing diagnosing unexpected outcome into a classroom setting.

TODO: what if the expectation is "wrong"

2.2 Data analysis checks in statistical software

Currently, little has been done on how to computationally incorporate this diagnostic approach into data analysis workflow or software. Most of the diagnostic tools focus on defining user-specified rules, such as data quality checks or producing metrics to summarize model performance, as in model diagnostics. For example, the `assertr` package (Fischetti, 2023) and the `pointblank` package (Iannone et al., 2024) provide data validation tools to allow users to define data quality checks. In contrast, packages provide model checks tools like `performance` (Lüdecke et al., 2021) and `yardstick` (Kuhn et al., 2024), from the `tidymodels`

(Kuhn and Wickham, 2020) ecosystem, offers goodness-of-fit, residual analysis, and model performance metrics.

These packages provide the tools to conduct diagnostics but still don't reflect the mental process of how data analysts diagnose the unexpected output. For example, when an unexpected output occurs, an analyst may check on whether a column in the data frame is between two values for data quality. However, it is not documented what motivates the analyst to conduct this check – whether it also applies to other researchers analyzing new data in the same contexts, whether it is a common practice in the field, or whether it is a reaction to this particular data or scenario. Currently, most of these assumptions are not discussed in the publication or captured by tools themselves. While one might be able to infer some of the mental process of the analysts from external sources, such as talking to them or watching screencast videos produced by the analysts e.g. TidyTuesday screencast videos, these insights are not systematically documented or made machine-readable. This gap highlights the need for tools that provide higher level documentation of reasoning behind the checks, facilitating a more transparent and interpretable analysis process.

2.3 Unit tests

Why develop unit tests in software engineering? Because it is difficult to predict the outcome of a program, especially when the program is complex. Because the loss associated with a program failure is costly.

Existing tools for checks in data analysis including those check for data quality and model diagnostics. In software development,

- the `testthat` package (Wickham, 2011) provides the infrastructure for testing R code. It allows users to write unit tests for functions and packages.
- The `assertthat` package (Wickham, 2019) helps to write assertion statements that are

supposed to be true at a certain point in the code for defensive programming.

3 Analysis plan

3.1 Framing checks into unit tests

Q: Whether we should formulate these concept with math notation?? A: only if it helps

An analysis plan is a set of analysis steps combined with expectations. Expectations represent our belief about certain aspects of the analysis, independent of the analysis itself. It can be divided into two types: *outcome expectation* and *plan expectation*. Outcome expectation refers to what we anticipate from the main result of the analysis based on prior knowledge. They shape how we interpret the results and assess whether they are consistent with existing knowledge or indicate the need for updates (Grolemund and Wickham, 2014). For example, in public health, prior research shows the average increase in mortality rate per unit increase in PM10 is about 0.50% (Liu et al., 2019). This serves as an expectation for similar future studies. Plan expectations concern the intermediate steps within the analysis rather than the final outcome. They serve as checkpoints to detect deflection in the analysis process For example, we may expect temporal data to be ordered with no gaps and duplicates, or expect that temperature will be a significant covariate in the linear regression model of PM10 on mortality.

Experienced analysts often have implicit expectation about the outcome and rely on a few "directional signs" to check when the outcome deviate from those expectation. However, these expectations are rarely made explicit within the analysis workflow. This makes it challenging for consumers of the analysis to evaluate the results, since it becomes difficult to disentangle whether discrepancies arise from differing expectations or from the use of statistical technique, without running the analysis themselves. Non-expert analysts, lacking prior knowledge or instinct, may not have clear expectations of the results. This can lead

to reduced confidence of the analysis and makes it more difficult and time-consuming to diagnose the cause of the deviation when the results don't align with expectations. By explicitly formulating these expectations, an analysis plan can guide the analysis process, facilitate the communication and evaluate the validity of the results.

3.2 A toy example

Consider a 30-day step count goal. Suppose you resolve to walk at least 8,000 steps each day, using an app to record your daily step count. Your target average is 9,000 steps per day, with some "lows" day, where you walk around 4,000 steps and "high" days where you reach about 12,000 steps. After 30 days, you check how many times your step count fall below 8,000 and aim for no more than five days under this threshold.

To simulation this data, three normal distributions with different means are used for the daily step counts: $\mathcal{N}(4000, 200)$ for low days, $\mathcal{N}(12000, 200)$ for high days, and $\mathcal{N}(9000, 300)$ for typical days. The number of low and high days can be simulated from a Poisson distribution with $\lambda = 4$. Figure 1 displays the number of days with fewer than 8,000 steps across 300 simulated 30-day periods.

In this scenario, the outcome expectation is that the number of days with a step count below 8,000 will be no more than five. To diagnose potential reasons why this outcome expectation might fail, we can establish a few plan expectations. For example, if the average step count is too low, this may suggest there are too many low days, potentially lead to an unexpected outcome. Similarly, we can also check the quantile of the step count, if more than a third of the days fall below 8,000, this could indicate an excess of low-count days. Additionally, we may may expect the standard deviation of the step count not to be overly large.

These considerations yield the following three unit tests as plan expectations:

- test1: the test fails if the mean step count is below 8,200

- test2: the test fails if the 30th percentile of the step counts is below 8,200
- test3: the test fails if the standard deviation of the step countsexceeds 2,500.

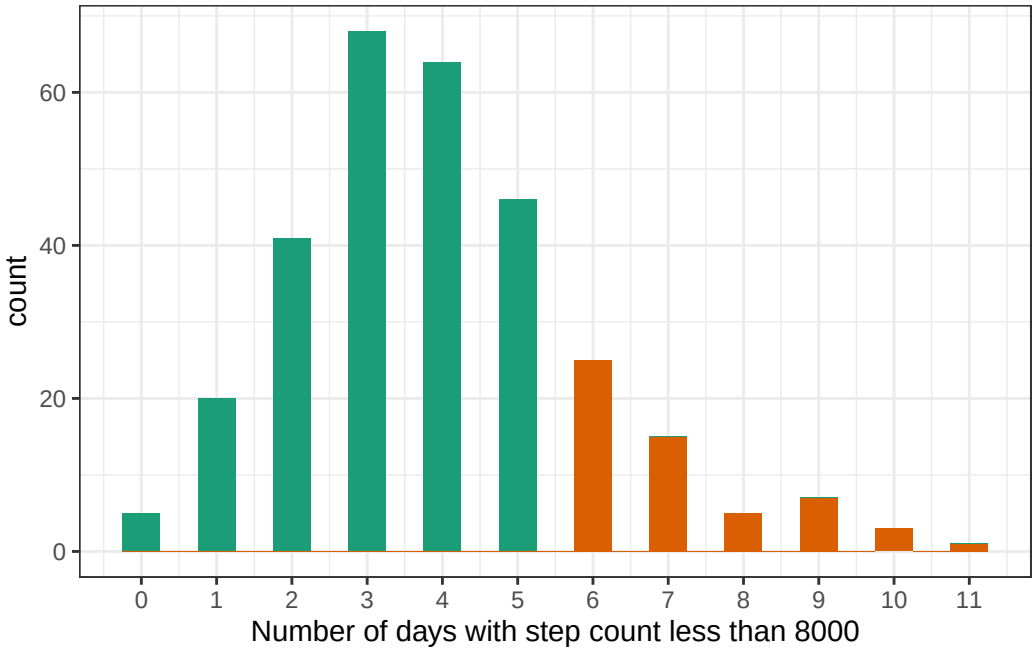

Figure 1: Number of days with fewer than 8,000 steps across 300 simulated 30-day periods. The orange bars indicate instances where the count exeeds five days, representing an unexpected outcome in this scenario.

Source: Article Notebook

4 Method

4.1 A workflow to assess the quality of unit tests

In real-world applications, it is rare to create a set of unit tests can fully guarantee expected results. On one hand, there is the cost of effort involved in manually developing all these tests; on the other, there is the inherent complexity of the problem. (can we - if we're thriving for detecting 95% of the cause?). However, the quality of unit tests can be evaluated by simulated data. One set of tests is considered better than another if a small set of tests

can reliably detect unexpected outcomes, which brings two criteria in the evaluation metric: accuracy and parsimony.

Accuracy refers to a set of tests' ability to accurately detect unexpected outcomes while minimizing false positives and false negatives. A false positive can indicate (caution or skepticism on checking the data), whereas a false negative suggests the tests may lack sensitivity to unexpected outcomes. Since both plan and outcome expectations can be framed as unit tests with binary outcomes, one approach is to predict the outcome expectation based on the plan expectation. Performance of the tests can then be evaluated using precision and recall metrics from the confusion matrix:

- precision: the proportion of correctly identified unexpected results (true positives) out of all the predicted unexpected results (true positives + false positives)
- recall: the proportion of correctly identified unexpected results (true positives) out of all the actual unexpected results (true positives + false negatives)

The second criteria is parsimony in the tests. While tests may score high on accuracy, they may be less effective at explaining the reasons behind unexpected results. This could happen if a set of tests are all tangentially related to the cause of the unexpected results, but none addressing the root cause. It may also occur if the tests are correlated with one another, leading to redundancy.

- explain mutual information
- explain logic regression
- explain combining precision, recall, and independence together through "means"
- A logic regression (ref) is used to model the relationship between the plan and outcome expectations. (justify the use of Logic regression)

4.2 Toy example revisited

- provide interpretation at different scenarios:
  1) one test is flagged, the prediction is as expected:
  2) multiple tests are flagged, the prediction is unexpected,
  3) no test is flagged, the prediction is unexpected,
  4) no test is flagged, the prediction is as expected

5 Applications

Three examples are presented to illustrate how the concept of analysis plan can be applied to data analysis. [toy example]. Section 5.1 illustrates how constructing the result universe in a linear regression model of PM10 on mortality can help understand the impact of sample size, model specification, and variable correlation structure on data analysis. [example three]

5.1 Linear regression

Consider a linear regression model to study the effect of PM10 on mortality (provide context of using PM10 to study mortality). Analysts may expect a significant (p-value ≤ 0.05) PM10 coefficient in the linear model from the literature. This is the *outcome expectation*. There are multiple factors that can affect the outcome expectation of linear regression, which here is called *plan expectation*, for example, 1) sample size, 2) model specification, and 3) correlation structure between variables. Adequate sample size is required to achieve the desired power to detect the significance of PM10 on mortality. Temperature is often an important confounder to consider in such study (add reference). From some domain knowledge, an analyst may expect that the significance of PM10 coefficient can be attained by adding temperature to the model. Analysts may also expect certain correlation structure between PM10, temperature, and mortality, and the distribution of each variable.

- add a paragraph to describe the simulation process

Source: Article Notebook

6 Discussion

- how to systematically simulate data is still unknown, sensitivity of the simulation to the results
- plotting is a critical way to check data and they can still be frame into a unit test. it is a open problem to how to encode the visualization into the unit tests. Maybe a procedure like confirm plot (this looks alright to you) and then press the button to continue
- currently no automated way to generate unit tests. It is interesting to see the automation of generating unit tests, although it requires the inputs from experts across a wide array of common scenarios.
- There are cost and benefit on setting expectation on different granularity. At the lowest level, one may have a plan for each data entry and every data handling steps. This requires more work from the analysts and may not be practical in practice. For more complex analyses, analysts may divide the analysis into sections and set expectations for each. They can then focus on the specific sections flagged by the tests and sub-divide the sections to set expectation and diagnose the analysis in a hierarchical manner.
- (only mention it) Software testing relies on "oracles" to provide the expected output necessary for verifying test results. For example, to test whether the program correctly calculates 1 + 1, one need to supply the correct answer, 2. However, establishing this "correct" output can sometimes be challenging, where obtaining a solution may be difficult without the program itself. This situation leads to the oracle problem (Barr et al., 2014). In data analysis, the similar oracle problem can happen, as the "truth" of an outcome, the expectation, depends on the underlying theory or interpretation. For example, in a linear regression model, the significance of a coefficient may be expected or unexpected

based on the theory, making it challenging for researcher with a different theory to assess the results and the analysis.

7 Conclusion

References

Earl T Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing: A survey. *IEEE transactions on software engineering*, 41(5): 507–525, 2014.

Tony Fischetti. *assertr: Assertive Programming for R Analysis Pipelines*, 2023. URL https://CRAN.R-project.org/package=assertr. R package version 3.0.1.

Garrett Grolemund and Hadley Wickham. A Cognitive Interpretation of Data Analysis. *International Statistical Review*, 82(2):184–204, 08 2014. doi: 10.1111/insr.12028. URL https://onlinelibrary.wiley.com/doi/10.1111/insr.12028.

Richard Iannone, Mauricio Vargas, and June Choe. *pointblank: Data Validation and Organization of Metadata for Local and Remote Tables*, 2024. URL https://CRAN.R-project.org/package=pointblank. R package version 0.12.2.

Max Kuhn and Hadley Wickham. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.*, 2020. URL https://www.tidymodels.org.

Max Kuhn, Davis Vaughan, and Emil Hvitfeldt. *yardstick: Tidy Characterizations of Model Performance*, 2024. URL https://CRAN.R-project.org/package=yardstick. R package version 1.3.1.

Cong Liu, Renjie Chen, Francesco Sera, Ana M Vicedo-Cabrera, Yuming Guo, Shilu Tong, Micheline SZS Coelho, Paulo HN Saldiva, Eric Lavigne, Patricia Matus, et al. Ambient particulate air pollution and daily mortality in 652 cities. *New England Journal of Medicine*, 381(8):705–715, 2019.

Daniel Lüdecke, Mattan S. Ben-Shachar, Indrajeet Patil, Philip Waggoner, and Dominique Makowski. performance: An R package for assessment, comparison and testing of statistical models. *Journal of Open Source Software*, 6(60):3139, 2021. doi: 10.21105/joss.03139.

Roger D. Peng, Athena Chen, Eric Bridgeford, Jeffrey T. Leek, and Stephanie C. Hicks. Diagnosing Data Analytic Problems in the Classroom. *Journal of Statistics and Data Science Education*, 29(3):267–276, September 2021. doi: 10.1080/26939169.2021.1971586. URL https://doi.org/10.1080/26939169.2021.1971586.

Hadley Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

Hadley Wickham. *assertthat: Easy Pre and Post Assertions*, 2019. URL https://CRAN.R-project.org/package=assertthat. R package version 0.2.1.