Automated Sample Ratio Mismatch (SRM) Detection and Analysis

Vermeer, Lukas; Anderson, Kevin; Acebal, Mauricio

DOI
10.1145/3530019.3534982

Publication date
2022

Document Version
Accepted author manuscript

Published in
Proceedings of the ACM International Conference on Evaluation and Assessment in Software Engineering, EASE 2022

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable). Please check the document version above.
Automated Sample Ratio Mismatch (SRM) detection and analysis

Lukas Vermeer  
Vista  
Delft, Netherlands  
lukas@lukasvermeer.nl

Kevin Anderson  
Delft University of Technology / Vista  
Utrecht, Netherlands  
k.s.anderson@tudelft.nl

Mauricio Acebal  
Vista  
Barcelona, Spain  
mauricio.acebal@vista.com

ABSTRACT

Background: Sample Ratio Mismatch (SRM) checks can help detect data quality issues in online experimentation [3]. Not all experimentation platforms provide these checks as part of their solution. Users of these platforms must therefore manually check for SRM, or rely on additional processes—such as checklists [2]—or automation.

Objective: To ensure reliable and early detection of SRM, we wanted to automate the detection and analysis of SRM in experiments running on third-party experimentation platforms.

Method: A set of Looker dashboards were built to facilitate self-serve SRM detection and root cause analysis. In addition, we added email and chat based alerting to pro-actively inform experimenters of SRM and guide them towards these dashboards when needed.

Results: Several cases of SRM have been detected and experimenters have been warned. Bad decisions based on flawed data were avoided. We provide one such example as an illustration.

Conclusions: SRM checks are relatively straightforward to automate and can be useful for data quality monitoring even for companies who rely on third-party experimentation platforms. Pro-active alerting—rather than passive reporting—can reduce time to detection and help non-experts avoid making decisions based on biased data.

CCS CONCEPTS

• General and reference → Experimentation.

KEYWORDS

A/B Testing, Online Controlled Experimentation, Sample Ratio Mismatch, SRM, Infrastructure, Trustworthiness, Data Quality

ACM Reference Format:


1 INTRODUCTION

At Vista, we want to run (online) controlled experiments at a large scale [6] comparable to other well known online companies [4]. We want to make our experimentation flywheel [1] spin faster, so that more people can get more value from experiments. This involves many non-experimentation-experts setting up and executing these experiments in a self-service manner.

We also want our experiments to be trustworthy, without creating knowledge or process bottlenecks. Although we will partially rely on checklists such as those suggested by Fabijan et al. [2] as well as a structured education curriculum, we are also investing in automation and infrastructure to improve consistency and reliability of our data quality checks.

One such data quality check is the Sample Ratio Mismatch test [3]. While this test requires only summary statistics from experiments and is relatively straightforward to perform, it can help uncover a wide variety of data quality issues. This makes it a good candidate for one of the first data quality mechanisms to automate.

2 METHOD

Before embarking on this project, several of the authors were using the SRM Checker Chrome plugin [8] which flags SRM issues in one of the third-party experimentation platforms we are using. This approach had several downsides compared with our automation:

- It can only warn experimenters who have the plugin installed. Since not all experimenters have the plugin installed, data quality monitoring was inconsistent between teams.
- It has limited support for platforms. Since not all experimentation platforms in use at Vista are supported, data quality monitoring is inconsistent between platforms.
- It can only warn the user about issues when they actively check the experiment results. In some cases this led to SRM issues going unnoticed for weeks, because nobody with the plugin installed was actively checking the results.

To ensure consistent and reliable detection of SRM issues—and to reduce opportunity cost as a result of late detection of SRM issues—we decided to automate SRM checks with the aim of pro-actively notifying experimenters within a day when they occur. Our approach makes use of several (third-party) infrastructure components which were already in place.

- An events pipeline (Segment) which was used to collect triggering events when a user enters an experiment.
- A data lake (Snowflake) which was used to store these triggering events—as well as experiment metadata from the experimentation platform—and allows us to create views and tables on top of this data.
- A reporting tool (Looker) which could be used to build reports and compute summary statistics from the data lake.

Using these components, we built four things:

- An events pipeline (Segment) which was used to collect triggering events when a user enters an experiment.
- A data lake (Snowflake) which was used to store these triggering events—as well as experiment metadata from the experimentation platform—and allows us to create views and tables on top of this data.
- A reporting tool (Looker) which could be used to build reports and compute summary statistics from the data lake.

Using these components, we built four things:

- A Snowflake table which exposes how many visitors were triggered into each variation of each experiment.
Although we are getting immediate value from the current implementation by detecting and reducing the impact of SRM issue, our current approach to detection and analysis is rather basic. We see several potential areas of improvement to these checks.

4 CONCLUSION

Although we are getting immediate value from the current implementation by detecting and reducing the impact of SRM issue, our current approach to detection and analysis is rather basic. We see several potential areas of improvement to these checks.

We should guard against this, either through process controls or additional tooling around the third-party reporting.

In addition, this project has proven the feasibility and usefulness of building data quality checks on top of existing third-party experimentation platforms when those platforms do not already perform such checks. We could extend this idea beyond SRM checks and implement additional monitoring capabilities, for example taking guidance from Fabijan et al. [2] or Perrin [7].

ACKNOWLEDGMENTS

We would like to thank the other members of the Vista Experimentation Hub for their support in implementing the features outlined in this paper. We would also like to thank the wider experimentation community for the experiment which served as an example.

REFERENCES