16
Scaling Experiment Analyses

If you want to increase your success rate, double your failure rate
– *Thomas J. Watson*

***Why you care:** For a company to move to the later phases of experimentation maturity ("Run" or "Fly"), incorporating data analysis pipelines as part of the experimentation platform can ensure that the methodology is solid, consistent, and scientifically founded, and that the implementation is trustworthy. It also helps save teams from needing to do time-consuming ad hoc analysis. If moving in this direction, understanding the common infrastructure steps for data processing, computation, and visualization can be useful.*

Data Processing

To get the raw instrumented data into a state suitable for computation, we need to *cook* the data. Cooking data typically involves the following steps.

1. **Sort and group the data**. As information about a user request may be logged by multiple systems, including both client- and server-side logs, we start by sorting and joining multiple logs (see Chapter 13). We can sort by both the user ID and timestamp. This allows joining events to create sessions or visits, and to group all activity by a specified time window. You may not need to materialize this join, as a virtual join as a step during processing and computation may suffice. Materialization is useful if the output is used not just for experimentation analysis, but also for debugging, hypothesis generation, and more.
2. **Clean the data**. Having the data sorted and grouped makes it easier to clean the data. We can use heuristics to remove sessions that are unlikely to be

real users (e.g., bots or fraud, see Chapter 3). Some useful heuristics on sessions are whether the sessions have too much or too little activity, too little time between events, too many clicks on a page, users who engage with the site in ways that defy the laws of physics, and so on. We can also fix instrumentation issues, such as duplicate event detection or incorrect timestamp handling. Data cleansing cannot fix missing events, which may be a result of lossiness in the underlying data collection. Click logging, for example, is inherently a tradeoff between fidelity and speed (Kohavi, Longbotham and Walker 2010). Some filtering may unintentionally remove more events from one variant than another, potentially causing a sample ratio mismatch (SRM) (see Chapter 3).

3. **Enrich the data**. Some data can be parsed and enriched to provide useful dimensions or useful measures. For example, we often add browser family and version by parsing a user agent raw string. Day of week may be extracted from dates. Enrichments can happen at a per-event, per-session or per-user levels, such as marking an event as a duplicate or computing event duration, adding the total number of events during the session or the total session duration. Specific to experiments, you may want to annotate whether to include this session in the computation of the experiment results. These annotations are pieces of business logic that are often added during enrichment for performance reasons. Other experimentation specific annotations to consider include experiment transitions information (e.g., starting an experiment, ramping up an experiment, changing the version number) to help determine whether to include this session in the computation of the experiment results. These annotations are pieces of business logic that are often added for performance reasons.

Data Computation

Given the processed data, you can now compute the segments and metrics, and aggregate the results to get summary statistics for each experiment, including both the estimated Treatment effect itself (e.g., delta of mean or percentiles) and the statistical significance information (p-value, confidence interval, etc.). Additional information, such as which segments are interesting can also occur within the data computation step (Fabijan, Dmitriev and McFarland et al. 2018).

There are many options for how to architect the data computation. We describe two common approaches. Without loss of generality, we assume the experimental unit is user.

1. The first approach is to materialize per-user statistics (i.e., for every user, count the number of pageviews, impressions, and clicks) and join that to a table that maps users to experiments. The advantage of this approach is that you can use the per-user statistics for overall business reporting, not just experiments. To effectively utilize compute resources, you may also consider a flexible way to compute metrics or segments that are only needed for one or a small set of experiments.
2. An alternative architecture is to fully integrate the computation of per-user metrics with the experiment analysis, where per-user metrics are computed along the way as needed without being materialized separately. Typically, in this architecture, there is some way to share the definitions of metrics and segments to ensure consistency among the different pipelines, such as the experiments data computation pipeline and the overall business reporting computation pipeline. This architecture allows more flexibility per-experiment (which may also save machine and storage resources) but requires additional work to ensure consistency across multiple pipelines.

Speed and efficiency increase in importance as experimentation scales across an organization. Bing, LinkedIn, and Google all process terabytes of experiment data daily (Kohavi et al. 2013). As the number of segments and metrics increase, the computation can be quite resource-intensive. Moreover, any delay in experiment scorecard generation can add delay to decision making, which can be costly as experimentation becomes more common and integral to the innovation cycle. In the early days of the experimentation platform, Bing, Google, and LinkedIn generated experiment scorecards daily with a ~24-hour delay (e.g., Monday's data shows up by end-of-day Wednesday). Today, we all have near real-time (NRT) paths. The NRT path has simpler metrics and computations (i.e., sums and counts, no spam filtering, minimal statistical tests) and is used to monitor for egregious problems (such as a misconfigured or buggy experiment), and often operates directly on raw logs without the data processing discussed above (except for some real-time spam processing). The NRT path can then trigger alerts and automatic experiment shut-off. The batch processing pipeline handles intra-day computation and updates to the data processing and computation to ensure that trustworthy experiment results are available in a timely manner.

To ensure speed and efficiency as well as correctness and trustworthiness, we recommend that every experimentation platform:

- Have a way to define common metrics and definitions so that everyone shares a standard vocabulary, everyone builds the same data intuition, and you can discuss the interesting product questions rather than re-litigating definitions and investigating surprising deltas between similar-looking metrics produced by different systems.

- Ensure consistency in the implementation of those definitions, be it a common implementation or some testing or ongoing comparison mechanism.
- Think through change management. As discussed in the experimentation maturity model (see Chapter 4), the metrics, OEC, and segments will all evolve, so specifying and propagating changes is a recurring process. Changing the definition of an existing metric is often more challenging than additions or deletions. For example, do you backfill the data (e.g., do you propagate the changes historically), and if so, for how long?

Results Summary and Visualization

Ultimately, the goal is to visually summarize and highlight key metrics and segments to guide decision makers. In your summary and visualizations:

- Highlight key tests, such as SRM, to clearly indicate whether the results are trustworthy. For example, Microsoft's experimentation platform (ExP) hides the scorecard if key tests fail.
- Highlight the OEC and critical metrics, but also show the many other metrics, including guardrails, quality, and so on.
- Present metrics as a relative change, with clear indications as to whether the results are statistically significant. Use color-coding and enable filters, so that significant changes are salient.

Segment drill-downs, including automatically highlighting interesting segments, can help ensure that decisions are correct and help determine whether there are ways to improve the product for poorly behaving segments (Wager and Athey 2018, Fabijan, Dmitriev and McFarland et al. 2018). If an experiment has triggering conditions, it is important to include the overall impact in addition to the impact on the triggered population (for more details, see Chapter 20).

Beyond the visualization itself, to truly scale experimentation, scorecard visualizations should be *accessible* to people with various technical backgrounds, from Marketers to Data Scientists and Engineers to Product Managers. This requires ensuring that not just experimenters, but executives and other decision makers see and understand the dashboard. This may also mean hiding some metrics, such as the debugging metrics from the less technical audience, to reduce confusion. Information accessibility helps establish a common language for definitions and a culture of transparency and curiosity, encouraging employees to run experiments and learn about how changes impact the business or how Finance can tie A/B test results to business outlook.

The visualization tool is not just for per-experiment results but is also useful for pivoting to **per-metric results** across experiments. While innovation tends to be decentralized and evaluated through experimentation, the global health of key metrics is usually closely monitored by stakeholders. Stakeholders should have visibility into the top experiments impacting the metrics they care about. If an experiment is hurting their metrics above some threshold, they may want to be involved in making the launch decision. A centralized experimentation platform can unify views of both the experiments and the metrics. Two optional features the platform can provide to cultivate a healthy decision process are:

1. Allow individuals to subscribe to metrics they care about and get an email digest with the top experiments impacting these metrics.
2. If an experiment has a negative impact, the platform can initiate an approval process, where it forces the experiment owner to start a conversation with the metrics owners before the experiment can be ramped up. Not only does this drive transparency regarding experiment launch decisions, it encourages discussion, which increases the overall knowledge of experimentation in the company.

The visualization tools can also be a gateway to accessing **institutional memory** (see Chapter 8).

Finally, as an organization moves into the Run and Fly phases of experimentation maturity, how many metrics your organization uses will continue to grow, even into the thousands, at which point we suggest using these features:

- Categorizing metrics into **different groups**, either by tier or function. For instance, LinkedIn categorizes metrics into three tiers: 1) Companywide 2) Product Specific 3) Feature Specific (Xu et al. 2015). Microsoft groups metrics into 1) Data quality 2) OEC 3) Guardrail 3) Local features/diagnostic (Dmitriev et al. 2017). Google uses categories similar to LinkedIn. The visualization tool provides controls to dig into different groups of metrics.
- **Multiple testing** (Romano et al. 2016) becomes more important as the number of metrics grow, with one common question from experimenters: Why did this metric move significantly when it seems irrelevant? While education helps, one simple yet effective option is using p-value thresholds

smaller than the standard value of 0.05, as it allows experimenters to quickly filter down to the most significant metrics. See Chapter 17 for more discussion on well-studied approaches such as the Benjamini-Hochberg procedure to address multiple testing concerns.

- **Metrics of interest**. When an experimenter goes through the experiment results, they likely already have a set of metrics in mind to review. However, there are always unexpected movements in other metrics that are worth examining. The platform can automatically identify these metrics by combining multiple factors, such as the importance of these metrics for the company, statistical significance, and false positive adjustment.
- **Related metrics.** A metric's movement or lack of movement can often be explained by other related metrics. For example, when click-through rate (CTR) is up, is it because clicks are up or because page views are down? The reason for the movement may lead to different launch decisions. Another example is metrics with high variance such as revenue. Having a more sensitive, lower variance version such as trimmed revenue or other indicators, allows more informed decisions.