Rapid Regression Detection in Software Deployments through Sequential Testing

Michael Lindon, Chris Sanden, Vaché Shirikian

mlindon@netflix.com | csanden@netflix.com | vache@netflix.com

Introduction

To minimize the risk of deploying defective software, engineering practices have evolved towards regression-driven experimentation, namely canary testing. This practice involves exposing a change to a small group of users or traffic and studying its effects under production workloads.

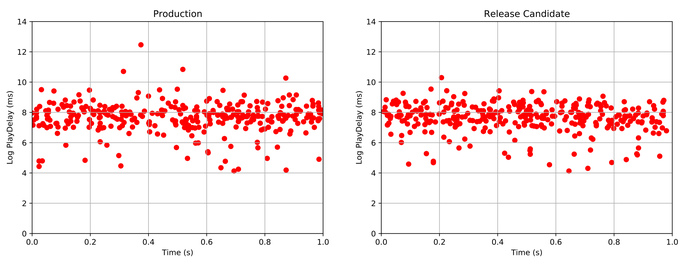
One common approach to canary testing is to treat it as a controlled experiment. In this approach, a small portion of users are randomly assigned to one of two variants: control or treatment. The treatment represents the release candidate and contains the change to be tested while the control is a copy of the previous version. This approach is related to an A/B test design found in online controlled experiments.

A major shortcoming of existing approaches is the reliance on *fixed-n* tests. These are statistical tests that provide guarantees, such as Type-I (false detection) error control, when used *strictly once*. In practice, it can be tempting to apply these tests for continuous monitoring in an attempt to quickly detect regressions. However, this can result in an unacceptable number of false positives.

We present a statistical framework for rapidly detecting regressions in software deployments. Our approach is based on sequential tests of stochastic order and of equality in distribution. This enables canary tests to be continuously monitored, permitting regressions to be rapidly detected while strictly controlling the false detection probability throughout. The utility of this approach is demonstrated based on case studies at Netflix.

Can You Spot The Difference?

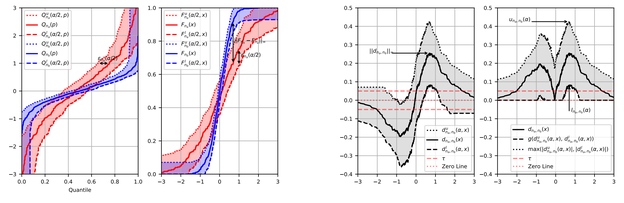
Consider two event streams of *PlayDelay* - the time taken for a title to start streaming. The Netflix client application measures this delay and sends it to Netflix every time the user begins to stream. Each red dot corresponds to a measurement of *PlayDelay* from a user starting a stream.



New versions of the client application must not ”increase” *PlayDelay*. We formulate the notion of an ”increase” in terms of stochastic ordering. Under the null hypothesis, measurements from the release candidate are stochastically less than or equal to those in the production version of the app. This captures a broader class of changes in the distribution than just differences in the mean.

Sequential Tests of Stochastic Dominance

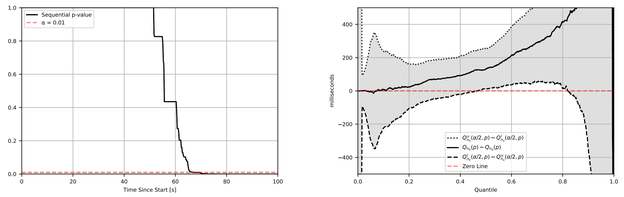
We perform sequential inference on the entire distribution and quantile functions to get a complete understanding of the changes caused by the release candidate. Leveraging the results of Howard [1], we construct *time uniform confidence sequences* on the distribution and quantile functions (1). These provide coverage guarantees across time, tighten with incoming measurements, and are guaranteed to reveal any difference eventually. The canary test can be concluded as soon as these fail to intersect, or when the confidence bands become so tight that any difference, if it exists, is not practically meaningful.



$$\begin{aligned}\mathbb{P}[F_n^l(\alpha,x) \le F(x) \le F_n^u(\alpha,x)\,\forall x \in \mathcal{X}\,\forall n \in \mathbb{N}] \ge 1-\alpha,\\ \mathbb{P}[Q_n^l(\alpha,p) \le Q(p) \le Q_n^u(\alpha,p)\,\forall p \in [0,1]\,\forall n \in \mathbb{N}] \ge 1-\alpha,\end{aligned} \tag{1}$$

Example: Increase in PlayDelay

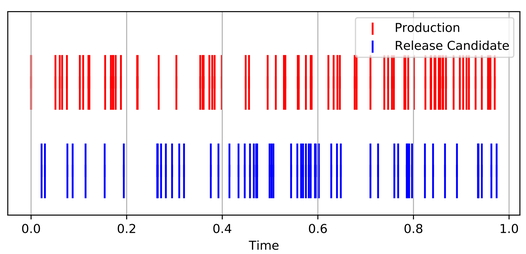
In practice this procedure detects even small performance regressions very quickly. The following example shows a canary tests where there was a small increase in *PlayDelay* for the release candidate which was detected at the 0.01 level in approximately 1 minute.



The median value of *PlayDelay* increased by at least 11 and at most 255 milliseconds in the release candidate, while the 75-percentile increased by at least 51 and at most 635 milliseconds (based on a 0.99 confidence interval).

What About Count Metrics?

Consider two event streams of *Successful Play Starts (SPS)*. When playback begins successfully, an *SPS* event is sent to Netflix by the client application. A significant drop in *SPS* events between software versions therefore implies a significant increase in playback failures, and is an important metric for detecting software regressions.



The raw data is simply a point process in time and we seek a sequential test to determine if the point processes generated by the production version is different from that generated by the release candidate. The following approach is also used to monitor the rates of errors produced by each application version.

Sequential Tests of Renewal Processes

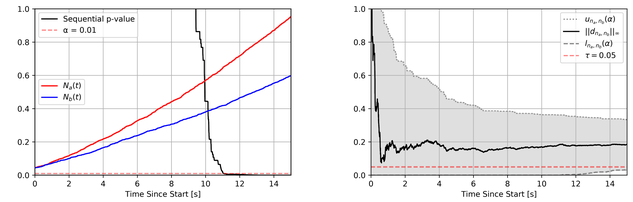
A renewal process is a stochastic process in time with waiting times between successive timestamps (epochs) drawn i.i.d. from a *renewal distribution*. Consider a sequence of timepoints $t_1, t_2, \dots t_n$ observed in an interval $[0, \mathcal{T}]$. Let g and G be the renewal density and distribution function of the process respectively. The likelihood for G given the observed times is given by

$$p((t_1, t_2, \dots, t_n)|G) = \prod_{i=1}^{n-1} g(t_{i+1} - t_i) g(t_1)(1 - G(\mathcal{T} - t_n), \tag{2}$$

We adopt the same approach for sequentially testing differences in the renewal distribution that governs the point processes in both versions of the client application. This amounts to learning the renewal distribution from the epochs $\Delta_2 := (t_2 - t_1), \Delta_3 := (t_3 - t_2)$ etc.

Example: Decrease in Successful Play Starts

The following example is taken from a canary test in which the release candidate contained a serious bug, preventing users from streaming. The decrease in successful play starts for the release candidate was detected within 11 seconds at the 0.01 level.



Conclusion

We present an approach to canary testing that has successfully detected performance regressions and bugs in software deployments at Netflix. The novel contributions of this approach are the formulation of regressions in terms of stochastic orderings and the sequential testing framework. Testing hypotheses of stochastic order enables developers to assign an undesirable direction to changes in a metric distribution beyond simply comparing the means.

References

[1] Steven R. Howard and Aaditya Ramdas. 2022. Sequential estimation of quantiles with applications to A/B testing and best-arm identification. Bernoulli 28, 3 (2022), 1704 – 1728. https://doi.org/10.3150/21-BEJ1388

[2] Michael Lindon, Chris Sanden, and Vaché Shirikian. 2022. Rapid Regression Detection in Software Deployments through Sequential Testing. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (Washington, DC, USA) (KDD ’22). ACM, New York, NY, USA