Case study in the development of a framework for quality and reproducibility in inner-sourced packages and self-service analytic dashboards to accelerate common study types

James Black, PhD[1], Nayan Chaudhary, MSc[3], Adam J. Forys´, PhD[2], Sriraman Madhavan, MSc[3], **Matthew Secrest, MSc***[3], Kamil Wais, PhD[2]

[1]PD Data Science, Roche; [2]7N Consulting; [3]PD Data Science, Genentech. *Presenting author. All authors contributed equally.

Introduction

When data scientists create interactive dashboards they are likely to approach the dashboard as an interactive version of a traditional static analysis, and as a general rule many data scientists may not have formal training in computer science or be held to, or aware, of the formal approach taken when an application is classed as Software as a Medical Device (FDA, 2018).

To address these gaps, Roche / Genentech data scientists worked with an engineering team to understand how we can improve the quality of our applications in terms of their ability to reliably return insights, ensure interactive results can be easily replicated for static studies, and ensure users are always aware of the limitations of an insight, and the provenance of the data it was formed from.

By learning from common practices applied in software development like loosely coupled architecture (Ganesh, 2005), and assertions (Boehm, 1975), and our own experiences, we propose a set principles that can be immediately implemented by data scientists without a background in computer science, and explain how these principles were applied in a case study.

The manifesto

The following 5 point manifesto are the principles we defined as the base principles we apply to interactive applications to allow us to develop more robust applications, that behave in known ways and produce reproducible results.

Code has value

The use of ad-hoc code within a study should be minimized, and a culture promoted of collaboration on pan-study code in documented packages for reuse. Unit tests are ideally written at function creation, and reviewed and expanded with new use cases.

Democratize access to analytics

Aided by the consolidation of code into packages, less technical users should be given access to well documented packages to promote guided analyses, as well as dashboards considered for fully self-service interaction.

Be verbose and specific on input cohorts

Publish cohort derivation code as user readable markdown vignettes with descriptive statistics and assumption checks.

Assertively limit user inputs in dashboards

Developers should ensure that variables exposed for manipulation in the app can be varied without compromising validation.

Separate logic from the user interface

Scientific logic should be separated from visualization code. This loosely coupled approach improves the robustness of the system by making it easier to isolate, test and document any logic applied to the data.

Case study

By applying these principles to a common study data type, the estimation of duration of treatment in real world settings using routinely collected data, we were able to accelerate time to deliver study results from months to days, while improving robustness and standardization via well documented and tested code.

As shown in Figure 1, in this case study we wrapped repeated study code into a robust and unit tested R package (**Code has value**), which contained all scientific business logic (**Separate logic from the user interface**).

We then created a de-coupled interactive user interface via R-Shiny (**Democratise access to analytics**). Within Shiny we controlled inputs, to ensure the underlying R package was used within a valid scope (**Assertively limit user inputs in dashboards**).

In addition to logic required for the final app, the R package contained vignettes that defined and provided documentation metadata for versioned cohorts. These versioned cohorts, and the accompanying metadata, were then ingested by the app (**Be verbose and specific on input cohorts**).

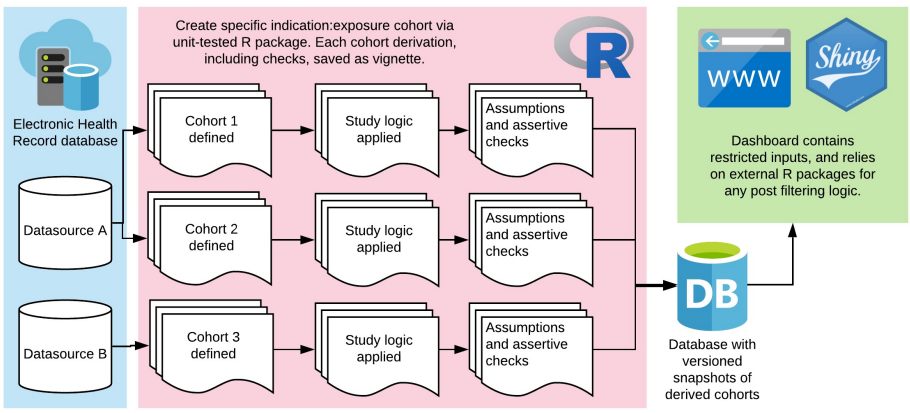

Figure 1. Case study architecture with formal separation of scientific logic from the user interface.

Discussion

These principles provided a mechanism to improve the reliability of our code, and had an important secondary effect of providing a clear framework to communicate with stakeholders and between data scientists steps taken to improve quality and reproducibility of internal study dashboards. In our experience, following all the above steps led to high quality & high fidelity self-service analytics. However there's a cost in terms of time & resources. Hence there should always be constant dialogue on when a certain set of repeated analysis warrants a need to be converted into a dashboard.

We hope to continue to refine, improve and continuously simplify this manifesto as we apply it to all of our internal dashboards.

References

FDA, 2018: https://www.fda.gov/medical-devices/digital-health-center-excellence/software-medical-device-samd

Ganesh, 2005: Web services, enterprise digital dashboards and shared data services: a proposed framework; 10.1109/ECOWS.2005.29

Boehm, 1975: Some experience with automated aids to the design of large-scale reliable software; 10.1145/800027.808430

Email us

SCAN ME