Abstract (50-75 words, do not include in 1-page submission)

Through a worked case study of an R package and Shiny dashboard for self-service health authority submission quality duration estimates, we lay a framework for robust efficiency. We make recommendations on 1) transparent and robust cohort definitions, 2) defensive restriction of user inputs to maintain results integrity, 3) the importance of separating scientific logic into a documented and tested independent package, 4) and a strategy for clear communication of validation scope and assumptions.

After the poster session, the learner should be better able to:

- Understand the importance of isolating, testing and documenting scientific logic as independent of graphical user interface code
- Have an understanding of unit testing,

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 A. Black, PhD[1], Sriraman Madhavan[3], Adam J. Foryś, PhD[2], Nayan Chaudhary[3], Kamil Wais[2]
[1]Personalised Healthcare Analytics, Roche, Switzerland; [2]7N Consulting, Poland; [3]Personalised Healthcare Analytics, Genentech, United States

Introduction and Principles

Replicability of a research question and reproducibility of individual studies are hampered in studies due to the diversity in the code that sits between a statistical analysis plan and a result. We propose a set principles that can be immediately implemented to boost efficiency.

Code has value: The use of ad-hoc code within a study should be minimised, 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.

Democratise access to analytics: Aided by the consolidation of 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 are limited to variance that

Separate logic from the user interface: As shown in Figure 1, scientific logic should be separated from visualisation code. This loosely coupled approach improves the general robustness of the system by making it easier to isolate, test and document any logic applied to the data.

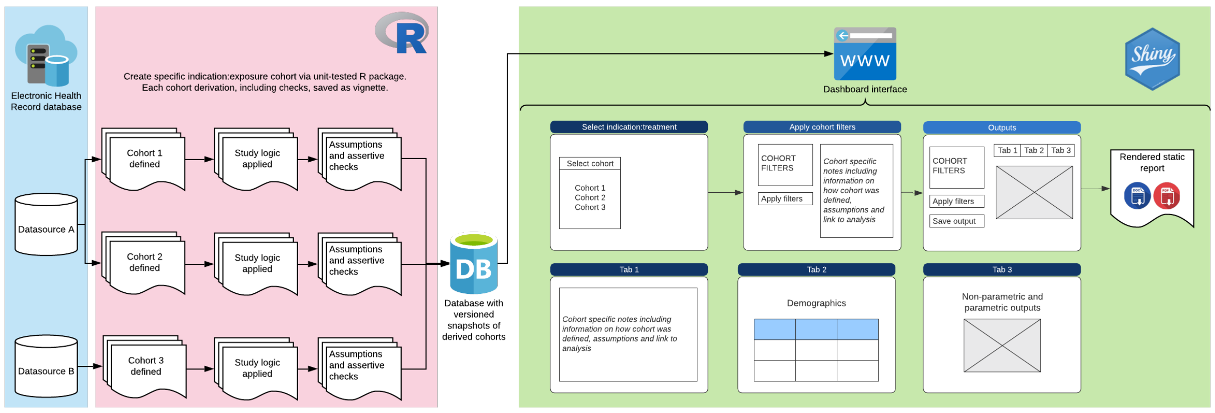
Electronic Health Record database
Datasource A
Datasource B
Create specific indication:exposure cohort via unit-tested R package. Each cohort derivation, including checks, saved as vignette.
Cohort 1 defined
Study logic applied
Assumptions and assertive checks
Cohort 2 defined
Study logic applied
Assumptions and assertive checks
Cohort 3 defined
Study logic applied
Assumptions and assertive checks
DB
Database with versioned snapshots of derived cohorts
WWW
Dashboard interface
Shiny
Select indication:treatment
Select cohort
Cohort 1
Cohort 2
Cohort 3
Apply cohort filters
COHORT FILTERS
Apply filters
Cohort specific notes including information on how cohort was defined, assumptions and link to analysis
Outputs
COHORT FILTERS
Apply filters
Save output
Tab 1
Tab 2
Tab 3
Rendered static report
Tab 1
Cohort specific notes including information on how cohort was defined, assumptions and link to analysis
Tab 2
Demographics
Tab 3
Non-parametric and parametric outputs

Figure 1: Framework for efficiency in running routine study types while maintaining robust and validated results

Conclusion

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 standardisation via well documented and tested code.