Barista: Brewing A New Methodology for Governing Study Execution

Ajit Londhe, Martin Lavallee, Katy Sadowski, Carmen Ng, Casey Tilton
Boehringer Ingelheim

Background

In studies leveraging OHDSI tools, it can be challenging to collect, maintain, and correctly leverage study assets, such as concept sets and cohort definitions, at scale. Tools like Atlas[1] and Capr[2] are useful for designing those study assets, but the identification and lifecycle of each asset at scale across studies is not covered by these tools. CohortGenerator[3] adds robust cohort generation capabilities such as cohort organization, subsets, and definition hashing, but the complexity of a study repository can grow as a result. Ulysses[4] establishes a framework for organized study execution but is limited in its governance and enforcement of a standardized approach. We sought to address these challenges with the new R package Barista[5].

Methods

Our team designs studies using Ulysses, where study metadata is captured first, all study cohorts are generated next, ordered analysis tasks to obtain raw results are executed after cohort generation, and lastly, migration steps are utilized to obtain the tables needed for downstream dissemination. To support the collection of cohorts and concept sets, we also adopted a de facto standard of requiring manifests for these assets to avoid brittle file system-based conventions. Manifests, in this context, are simply CSV files tracking study asset ids, names, provenance, file paths, and optional metadata.

For the design of Barista, we considered common study patterns used by our team and in OHDSI. We observed that even with Ulysses and manifest files, we encountered challenges with manual edits to the manifest CSV, identifier management, handling dependencies of cohorts and subsets, constraining analyses to use only the appropriate assets, and maintaining metadata in a structured manner. Consequently, we formalized a few principles to govern Barista:

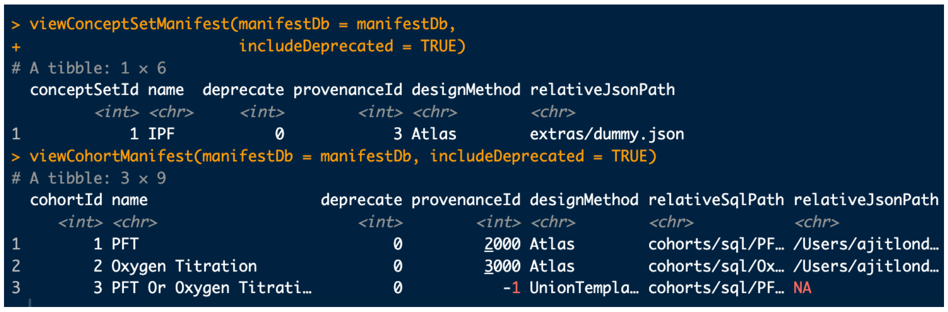
1. Cohort and concept set ids are meaningless; they simply need to be unique and consistent within a study repository.
2. Accurate manifest management is only possible with a strongly governed system.
3. Ulysses is the basis for how the study repository is organized and executed.
4. A study asset can be added to a manifest but can never be modified; it can be deprecated but never deleted.
5. CohortGenerator is the exclusive method for generating cohorts.
6. Metadata on each study asset is critical to transparency and quality control.

We then developed Barista as an R package leveraging R6 classes for the purpose of strict adherence to the above principles. User-facing functions are exported as the exclusive pathways for interacting with the Barista database and its manifest tables.

Results

For the Barista R package, we built the foundation atop the first two principles. We found the open-endedness of CSV files risky and decided to use SQLite[6] to store manifests, leveraging its built-in auto-incremented primary keys and table constraints. Within a Barista database, a manifest is represented by a table. There are manifest tables for Concept Sets, Cohorts, and File Paths within the study repository;

and, in keeping with the Ulysses principle, we have manifest tables for Analysis steps and Migration steps. To support the dependencies and metadata of the other manifests, we introduced a Dependency manifest and Tag manifest, respectively.



```
> viewConceptSetManifest(manifestDb = manifestDb,
+                        includeDeprecated = TRUE)
# A tibble: 1 × 6
  conceptSetId name  deprecate provenanceId designMethod relativeJsonPath
         <int> <chr>     <int>        <int> <chr>        <chr>
1            1 IPF           0            3 Atlas        extras/dummy.json
> viewCohortManifest(manifestDb = manifestDb, includeDeprecated = TRUE)
# A tibble: 3 × 9
  cohortId name                    deprecate provenanceId designMethod relativeSqlPath relativeJsonPath
     <int> <chr>                       <int>        <int> <chr>        <chr>           <chr>
1        1 PFT                             0         2000 Atlas        cohorts/sql/PF… /Users/ajitlond…
2        2 Oxygen Titration                0         3000 Atlas        cohorts/sql/Ox… /Users/ajitlond…
3        3 PFT Or Oxygen Titrati…          0           -1 UnionTempla… cohorts/sql/PF… NA
```

Figure 1: Examples of Concept Sets and Cohort Definitions maintained within SQLite manifest tables in Barista. Each manifest type follows assumptions to ensure quality and transparency of all study assets within a study repository.

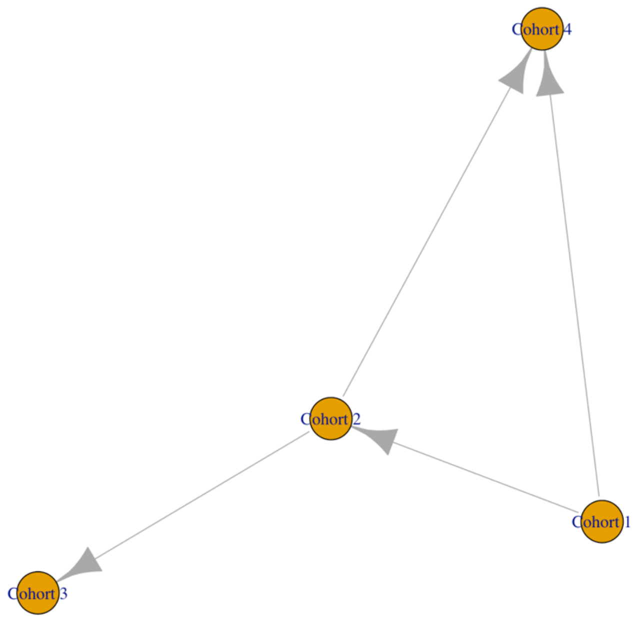


Figure 2: Dependencies are handled via the Dependency Manifest, with graphs used to not only illustrate the cohort execution plan to the user but also to govern the full impact of an asset being deprecated.

For the fourth principle, we observed that edits to existing definitions within a study repository often led to unintended and unwanted consequences. For Barista, any item added to a manifest is permanent but can be deprecated. This ensures that no false assumptions about study asset content or generation are made during the study lifecycle; instead, all actions taken with a manifest are explicit and intentional.

For the CohortGenerator principle, we designed an active binding within the Cohort manifest's R6 class that produces a CohortGenerator-compliant cohort definition set object. This endpoint is the result of detailed and highly controlled Cohort Manifest management. User-facing functions handle addition of new cohorts based on design type, be it an independent cohort that can be executed without a specific ordering, or a dependent cohort that needs to be generated at the right time due to dependencies on other cohorts.

Lastly, to address the metadata principle, we implemented tagging of assets with a one-to-many approach in the Tag manifest. Here, any manifest item can be tagged with a name-value pair that can be used for filtering other manifests. With the first principle rendering ids meaningless, it is important to provide efficiencies for identifying study assets during a study to provide transparency and avoid mistakes.

Conclusion

Built atop the foundations of Ulysses and CohortGenerator, Barista provides a new paradigm and a practical implementation for maintaining study assets and delivering results from an OHDSI study repository. The Barista database consists of a comprehensive set of manifest tables spanning the various phases of a study repository's lifecycle.

Iterative study design is a constant, so even the most seasoned study developer is challenged with ensuring accuracy of design and execution. Though still in a nascent form, we believe Barista will offer immediate value to organizations looking to improve the control and reliability of their study repositories.

References

1. Knoll, Christopher, Anthony Sena, and OHDSI et al. 2024. "Atlas: An Open Source Software Tool for Researchers to Conduct Scientific Analyses on Standardized Observational Data." Available at: https://github.com/OHDSI/Atlas
2. Lavallee M, Black A (2025). Capr: Cohort Definition Application Programming. R package version 2.1.0, https://github.com/OHDSI/Capr/, https://ohdsi.github.io/Capr/.
3. Sena A, Gilbert J, Rao G, Schuemie M (2024). CohortGenerator: Cohort Generation for the OMOP Common Data Model. https://ohdsi.github.io/CohortGenerator/, https://github.com/OHDSI/CohortGenerator.
4. Lavallee M (2024). Ulysses: Automate OHDSI Study Setup. R package version 0.0.7.
5. Londhe A, Lavallee M, Sadwoski K, Ng C, Tilton C (2025). Barista: Brewing A Resource Integration Standard Transforming Analyses. R package version 0.0.1, https://github.com/OHDSI/Barista
6. SQLite Consortium (n.d.). SQLite. https://www.sqlite.org/.