omock: A R package for Mock Data Generation for the Observational Medical Outcomes Partnership Common Data Model

Mike Du[1], **Núria Mercadé-Besora**[1], **Marta Alcalde-Herraiz**[1], **Xihang Chen**[1], **Yuchen Guo**[1], **Kim López-Güell**[1], **Edward Burn**[1], and **Martí Català**[1¶]

1 Health Data Sciences Group, Nuffield Department of Orthopaedics, Rheumatology and Musculoskeletal Sciences, University of Oxford, United Kingdom ¶ Corresponding author

DOI: 10.21105/joss.08178

Software

- Review
- Repository
- Archive

Editor: Julia Romanowska

Reviewers:

- @louisahsmith
- @chitrams

Submitted: 10 February 2025
Published: 12 September 2025

Summary

`omock` is an R package that allows users to create mock patient level data formatted in the Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM) (Overhage et al., 2011). This package provides a flexible and efficient way to create synthetic datasets in OMOP CDM format, facilitating the testing and validation of packages and analytic codes.

Statement of need

Reliable testing is essential in R package development (Vidoni, 2021), especially for packages that run across different server infrastructures. This need is particularly critical for software developed for Common Data Models (CDM) (Yan et al., 2023).

A CDM is a standardised and structured framework that helps define how data is organised and formatted across different databases (Makadia & Ryan, 2014). CDMs provide a standardised vocabulary and schema, making combining, comparing, and analysing data from multiple sources easier. In healthcare settings, CDMs help standardise different datasets such as electronic health records (EHRs), claims data and hospital data, enabling the use of the same analytical code across different data sources. A popular CDM used for medical research is the OMOP CDM ("OMOP Common Data Model," 2024), with over 200 peer-reviewed publications leveraging its standardised data format. More than 800 million patients' health-related data have been mapped to OMOP CDM by over 2,000 collaborators from more than 70 countries, enabling cross-institutional studies and scalable healthcare analytics (Overhage et al., 2011). A diagram of OMOP CDM table structure is shown in figure 1.

Creating robust tests for packages designed for the OMOP CDM is challenging because these packages must be compatible across various database types. Yet, fit-for-purpose datasets for testing are often unavailable due to privacy and ethical constraints. Having to write OMOP CDM datasets to test the different edge cases of the different packages is time consuming and not efficient.

The `omock` R package was developed to address this gap, providing an easy-to-use pipeline for generating mock data tables for the OMOP CDM. `omock` facilitates reliable testing of functions and workflows, enabling developers to validate their packages' compatibility with OMOP CDM standards while preserving data privacy and ethical considerations.

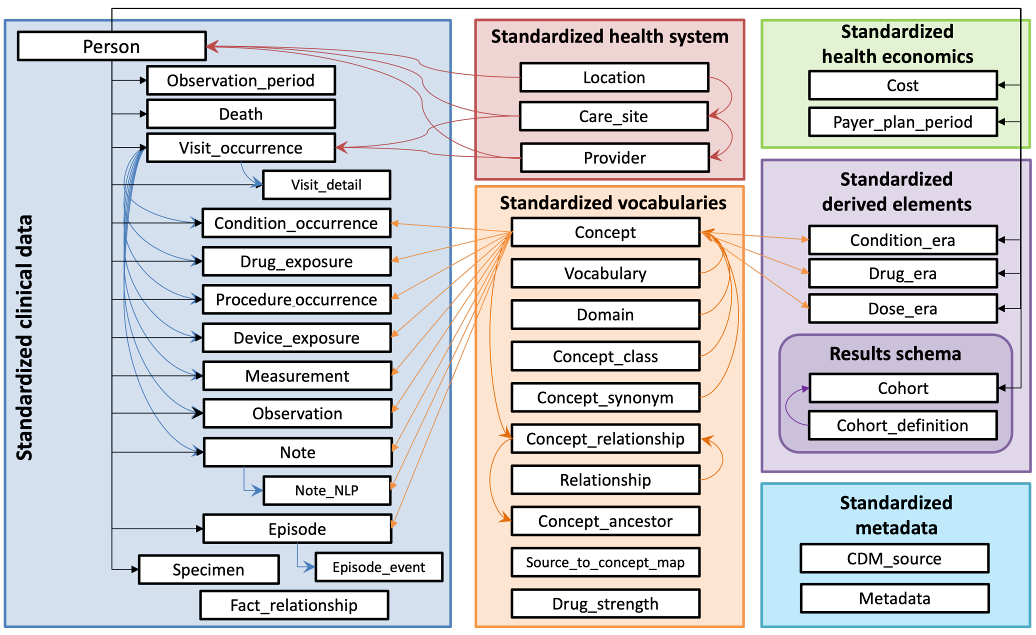

Figure 1: Overview of OMOP CDM standard table.(OHDSI, 2019)

Design principles

The omock R package is built to align with tidyverse design principles (Wickham et al., 2019), ensuring a consistent approach for testing functions within the package. It also relies heavily on tidyverse packages such as `dplyr`, `rlang`, and `purrr` for common data operations, maintaining compatibility with widely used tools in the R ecosystem.

In addition to its alignment with the tidyverse principles, the core dependency of `omock` is the omopgenerics package (Catala & Burn, 2024), which provides essential methods, classes, and basic operations for working with data in the OMOP CDM format. It ensures that the mock data generated by `omock` meets the structure and format requirements of the OMOP CDM.

Overview of the omock R package

The output of `omock` is a local CDM object, which contains all the OMOP CDM tables with mock data in a `cdm_reference` object. The `cdm_reference` object is defined by the `omopgenerics` package and is a named list of tables with classes depending the table name. This allows for dplyr style data analysis pipelines for data exploration. `omock` offers two approaches for creating a mock CDM object.

The first approach allows users to specify population settings, such as the number of patients and the gender composition, to generate a mock CDM object tailored to their specifications.

The second approach enables users to provide bespoke data tables in OMOP CDM format, which are used as a foundation to build a customised mock CDM with them.

These flexible options ensure that `omock` can accommodate various testing scenarios and requirements for developers and researchers working with the OMOP CDM. Both approaches are not mutually exclusive and can be combined.

The `omock` package is available in CRAN version 0.5.0 (Du et al., 2024), and currently there are six packages that depend on it for testing purposes.

Building mock OMOP CDM with population settings

An empty mock CDM can be created using the mockCdmReference function, which includes two key arguments: cdmName and vocabularySet. The cdmName argument allows users to specify the name of the mock OMOP CDM, while the vocabularySet argument lets users define the vocabulary tables to be included. The package contains a mock vocabulary sets. Once the mock CDM is initialised, mock patients and corresponding observation periods can be added using the mockPerson and mockObservation functions.

To expand the mock CDM with additional clinical tables, we can pipe the corresponding functions onto the previously created CDM object. For example, to add a condition occurrence and a drug exposure table to the OMOP CDM we can use mockConditionOccurrence and mockDrugExposure functions, respectively. There is a function for most of the commonly used clinical tables in OMOP CDM.

This modular design, where each table is added via a separate function, provides flexibility in testing and development workflows. Since users can simulate the specific OMOP CDM tables as needed, depending on the scenario being tested, such as testing functionality that only requires drug exposures or condition occurrences. Generating only the necessary tables can be less computationally intensive, which is especially useful when simulating CDM with a large number of patients or when running automated test suites repeatedly. Additionally, table-level control also makes it easier to simulate edge cases and specify population settings, such as the number of patients, gender and number of records within each tables.

Below is an example code snippet demonstrating how to generate a mock CDM with 1,000 patients, valid observation periods, and additional drug exposure and condition occurrence tables:

```
library(omock)

cdm <- mockCdmReference(cdmName = "mock database", vocabularySet = "mock") |>
  mockPerson(nPerson = 1000) |>
  mockObservationPeriod() |>
  mockConditionOccurrence()

print(cdm)
```

```
##

## ── # OMOP CDM reference (local) of mock database ──────────────────────────────

## • omop tables: person, observation_period, cdm_source, concept, vocabulary,
## concept_relationship, concept_synonym, concept_ancestor, drug_strength,
## condition_occurrence

## • cohort tables: -

## • achilles tables: -

## • other tables: -
```

Similarly, the mockCohort function can add a mock cohort table to the mock CDM object. The function contains arguments for the user to customise the mock table, such as its name and size. See the example below.

```
cdm <- cdm |>
  mockCohort(
    name = "omock_example",
    numberCohorts = 1,
    cohortName = c("omock_cohort_1")
  )

print(cdm)
```

```
##

## ── # OMOP CDM reference (local) of mock database ───────────────────────────────

## • omop tables: person, observation_period, cdm_source, concept, vocabulary,
## concept_relationship, concept_synonym, concept_ancestor, drug_strength

## • cohort tables: omock_example

## • achilles tables: -

## • other tables: -
```

Building mock CDM object with bespoke CDM table

To create a mock CDM object from a bespoke OMOP table, we can use the `mockCdmFromTables` function. `mockCdmFromTables` takes the OMOP table in `tibble` format and create a CDM object with valid person and observation period information for those tables. This function is helpful for quickly generating a valid CDM object when one or more clinical tables are already defined, for example, when writing unit tests that require known values in a particular clinical table. For example, if a `condition_occurrence` table is provided, mockCdmFromTables will extract the person_id from the condition_occurrence table and generate a consistent `person` and `observation_period` table to match the records within the condition_occurrence table.

By default, if no base CDM object is passed into `mockCdmFromTables()` explicitly, `mockCdmFromTables()` internally calls `mockCdmReference()` to initialise an empty CDM object. This ensures that all required vocabulary tables are present for creation of a valid CDM object that matches the OMOP CDM standards. This behaviour is particularly useful when users only need to validate code against their bespoke clinical tables, while `mockCdmFromTables` ensures that the resulting CDM object has all the required tables and relationships in place to form a valid CDM object.

By contrast, if an existing CDM object is passed in and contains tables that overlap with the provided ones (e.g., a `person` table), those will be overwritten to add all the extra individuals from the new tables added. A warning will then be issued to notify users that the tables within the CDM object have been overwritten. This design allows for customising an existing CDM object with bespoke tables, while ensuring the CDM object is valid.

Below is an example code snippet on using `mockCdmFromTables` to create a mock CDM object from the `condition_occurrence` table.

```
library(dplyr)
library(omock)

condition_occurrence = tibble(
  person_id = 1:3,
  condition_start_date = as.Date("2020-01-01") + 1:3,
  condition_end_date  = as.Date("2020-01-01") + 4:6,
  condition_concept_id = 1,
  condition_type_concept_id = 1,
  condition_occurrence_id = 1:3
)
```

```
cdm2 <- mockCdmFromTables(
  tables = list(condition_occurrence = condition_occurrence)
)

print(cdm2)

##

## ── # OMOP CDM reference (local) of mock database ───────────────────────────

## • omop tables: condition_occurrence, person, observation_period, cdm_source,
## concept, vocabulary, concept_relationship, concept_synonym, concept_ancestor,
## drug_strength

## • cohort tables: -

## • achilles tables: -

## • other tables: -
```

Testing code with omock

The main motivation for developing omock is to support reliable testing of analytic packages that depend on the OMOP CDM. This section provided two examples on how omock can be use for testing.

Example 1:

Suppose we want to validate a function that counts the number of individuals who have both a person record and an associated observation period. We can generate such testing data as follows:

```
library(dplyr)
library(omock)

# Create a mock CDM with 10 persons and corresponding observation periods

cdm_test <- mockCdmReference() |>
  mockPerson(nPerson = 10) |>
  mockObservationPeriod()

# Simple test function
count_observed_people <- function(cdm) {
  cdm$observation_period |>
    distinct(person_id) |>
    inner_join(cdm$person, by = "person_id") |>
    summarise(n = n()) |>
    pull("n")
}

## Apply the function to validate if the function will provide the correct answer
count_observed_people(cdm_test)

# > [1] 10
```

Example 2:

Suppose we want to validate a function that check if all records within a cohort table have cohort start date within a data range. We can generate such testing data as follows:

```r
library(dplyr)
library(omock)

## Create a mock CDM with 10 persons and corresponding observation periods

cohort_table <- tibble(
    cohort_definition_id = c(1, 1, 1),
    subject_id = c(1, 2,3),
    cohort_start_date = as.Date(c(
        "2020-04-01",
        "2021-06-01",
        "2022-05-22")),
        cohort_end_date = as.Date(c(
        "2020-04-01",
        "2021-06-01",
        "2022-05-22")))

cdm_test2 <- omock::mockCdmFromTables(tables = list(cohort = cohort_table))

# Function to check if all cohort_start_dates fall within a given date range
check_cohort_date_range <- function(cohort_table, start_date, end_date) {
  all(cohort_table$cohort_start_date >= start_date &
      cohort_table$cohort_start_date <= end_date)
}

## Run the check
check_cohort_date_range(cdm_test2$cohort,
                        start_date = as.Date("2020-01-01"),
                        end_date = as.Date("2023-01-01"))
#> [1] TRUE

check_cohort_date_range(cdm_test2$cohort,
                        start_date = as.Date("2020-01-01"),
                        end_date = as.Date("2022-01-01"))

#> [1] FALSE
```

Comparison with existing tools

A commonly used resource for testing OMOP-based tools is the `Eunomia` package (DeFalco et al., 2024), which provides a fixed, small synthetic CDM dataset. `Eunomia` is a valuable tool for exploring the OMOP CDM data structure and for demonstration purposes. It is less suited for testing code that requires custom scenarios, such as examples 1 and 2 shown above. Since the data within the `Eunomia` package is static and cannot be easily modified, this made it challenging to generate custom mock data for unit testing. Whereas `omock` is designed to give users the flexibility for generating mock OMOP CDM data tailored to their testing needs.

Conclusions

Overall, the omock R package allows users to generate mock OMOP CDM datasets tailored to their needs, addressing the need to develop and validate R packages for OMOP CDM data. Hence, the developers can test their tools while adhering to privacy and ethical considerations. Future directions for omock will enhance the realism of the mock data generated.

References

Catala, M., & Burn, E. (2024). *Omopgenerics: Methods and classes for the OMOP common data model*. https://doi.org/10.32614/CRAN.package.omopgenerics

DeFalco, F., Schuemie, M., Sena, A., Adulyanukosol, N., Liu, S., & Black, A. (2024). *Eunomia: Standard dataset manager for observational medical outcomes partnership common data model sample datasets*. https://doi.org/10.32614/CRAN.package.Eunomia

Du, Mike, Catala, Marti, Burn, Edward, Mercade-Besora, Nuria, Chen, & Xihang. (2024). Omock: Creation of mock observational medical outcomes partnership common data model. In *The R Foundation*. https://doi.org/10.32614/CRAN.package.omock

Makadia, R., & Ryan, P. B. (2014). Transforming the premier perspective hospital database into the observational medical outcomes partnership (OMOP) common data model. *EGEMS (Wash., DC)*, *2*(1), 1110.

OHDSI. (2019). *The book of OHDSI: Observational health data sciences and informatics*. OHDSI. ISBN: 9781088855195

OMOP common data model. (2024). In *OMOP Common Data Model*. https://ohdsi.github.io/CommonDataModel/

Overhage, J. M., Ryan, P. B., Reich, C. G., Hartzema, A. G., & Stang, P. E. (2011). Validation of a common data model for active safety surveillance research. *Journal of the American Medical Informatics Association*, *19*(1), 54–60. https://doi.org/10.1136/amiajnl-2011-000376

Vidoni, M. (2021). *Evaluating unit testing practices in r packages*. 1523–1534. https://doi.org/10.1109/ICSE43902.2021.00136

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Yan, C., Nath, S., & Lu, S. (2023). Generating test databases for database-backed applications. *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2048–2059. https://doi.org/10.1109/ICSE48619.2023.00173