omock: An R package for Mock Data Generation for the Observational Medical Outcomes Partnership common data model

06 Jan 2025

**Authors:**  
Mike Du, Núria Mercadé-Besora, Marta Alcalde-Herraiz, Xihang Chen,Yuchen Guo, Kim López-Güell, Edward Burn, Martí Català-Sabate

Affiliation:  
1. Health Data Sciences Group, Nuffield Department of Orthopaedics, Rheumatology and Musculoskeletal Sciences, University of Oxford, UK

# Summary

omock is an R package (Vidoni 2021) that allows users to create mock patient data in the standardised format of the Observational Medical Outcomes Partnership (OMOP) common data models (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, Nath, and Lu 2023).

A CDM is a standardised and structured framework that helps define how data is organised and formatted across different databases (Makadia and 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 Observational Medical Outcomes Partnership (OMOP) CDM, 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 diagram 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.

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.

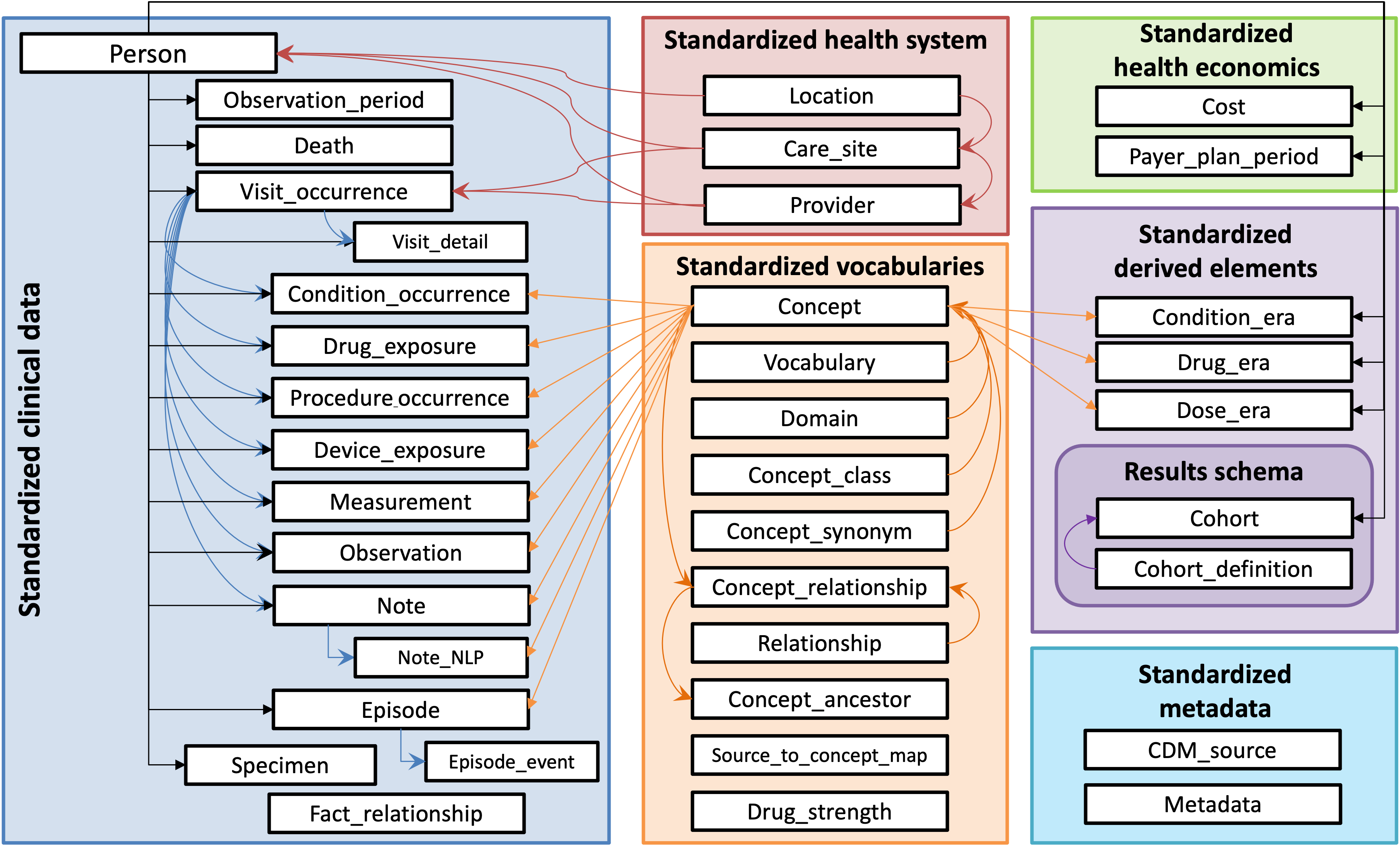


Figure 1: Overview of all standard tables in the OMOP CDM.(OHDSI 2019)

# 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 and 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 OMOP CDM. Both approaches are not mutually exclusive and can be combined.

The omock package is available in CRAN version 0.3.1 (Du et al. 2024), and six packages that depend on it for testing currently.

## 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. 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.

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.

library(omock)  
  
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. 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  
)  
  
  
cdm <-  
 mockCdmFromTables(tables = list(condition\_occurrence = condition\_occurrence))  
  
print(cdm)

##  
  
## ── # 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: -

This functionality is particularly useful for unit tests, as we can create a mock cdm\_reference that contains the specific OMOP CDM data we need to test the code’s performance as expected.

# 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, Marti, and Edward Burn. 2024. *Omopgenerics: Methods and Classes for the OMOP Common Data Model*. <https://darwin-eu-dev.github.io/omopgenerics/>.

Du, Mike, Martí Català, Edward Burn, Nuria Mercade-Besora, and Xihang Chen. 2024. *Omock: Creation of Mock Observational Medical Outcomes Partnership Common Data Model*. <https://oxford-pharmacoepi.github.io/omock/>.

Makadia, Rupa, and Patrick B Ryan. 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. <https://books.google.co.uk/books?id=JxpnzQEACAAJ>.

Overhage, J Marc, Patrick B Ryan, Christian G Reich, Abraham G Hartzema, and Paul E Stang. 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, Melina. 2021. “Evaluating Unit Testing Practices in r Packages.” In, 1523–34. <https://doi.org/10.1109/ICSE43902.2021.00136>.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the tidyverse.” *Journal of Open Source Software* 4 (43): 1686. <https://doi.org/10.21105/joss.01686>.

Yan, Cong, Suman Nath, and Shan Lu. 2023. “Generating Test Databases for Database-Backed Applications.” In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2048–59. <https://doi.org/10.1109/ICSE48619.2023.00173>.