# The Ulysses Standard Repository Structure

## Contents



## 1 Introduction

Ulysses implements a standard repository structure for an RWE study. A major part of the Ulysses philosophy is setting up a repo structure that can be tracked with git. This turns an RWE study/pipeline into any other software project. As RWE studies have taken on more and more software elements, it is vital to institute consistent standards to ensure that elements of the study are easily communicated for collaboration on code and self-contained for consistent study execution and versioning. In this vignette we describe the standard Ulysses structure.

## 2 Ulysses Repo Outline

A Ulysses repo will contain the following structure:

- analysis: folder containing functions that run the study; performing an aggregation on a study population
- inputs: folder containing input files needed to specify study populations and clinical assumptions such as medical codes
- dissemination: folder organizing outputs that should be committed to the git repository
- exec: folder containing the raw output results and logs from the study execution. Should not be commited to git
- extras: folder containing extra files necessary for the study
- config.yml: a settings file that describes the database and schema parameters used for a study execution
- README.md: the cover-page of the repo describing aspects of the study
- NEWS.md: the change log of the repo describing changes to the study over time
- .Rproj: an R administrative file that identifies the folder as an R projects and simplifies folder routing
- .gitignore: a file specifying which files should not be tracked by git

Next we will get into more details about each element in the structure.

# 3 Elements of the Ulysses Repo

## 3.1 Vital Files

### 3.1.1 .Rproj

The .Rproj file is an administrative file designed for the RStudio IDE that encapsulates the contents (code, settings, and docs) required for a coding project in R. The biggest benefits of the .RProj file is that it ensures that file paths are relative to the project root. A very common break point in custom coded RWE study is the use of absolute locations, that are based on the file structure of the developer's machine. Absolute paths allow repos to be more portable and collaborative. In general the .RProj encourages reproducibility of code and workspace isolation to avoid interfering settings options.

### 3.1.2 .gitignore

The .gitignore file specifies which files should not be tracked by a git enabled repository. We do not want to commit all files to a git remote, because it would be available for all to see. User profiles, patient level data, and credential files should never be committed to a git repository. .RData and .RHistory are other misc files that should not be committed due to potentital leakage of credentials and sensitive information. Ulysses sets the gitignore file to automatically ignore: .Rproj.user, .RData, .RHistory and exec/results.

### 3.1.3 README

The README is a markdown file that serves as the cover-page of the repository. It should communicate the essential elements of the RWE study such as a description of the study, key personnel and study status. The README file holds important project tracking/management information that can be leveraged within an organizations git host setup. The README also stores tags and links. Tags serve as keywords to search similar stuides within an organization. Links provide access to vital html widgets or cloud hosted documents linked to the study. On launch, Ulysses initializes the README with a template but it is highly suggested that users update the README according to the expectations of their organization and ease the communication of the project.

### 3.1.4 NEWS

The NEWS is a markdown file that serves as the change-log for the repository. It is common practice in software projects that developers release notes describing changes from the previous release of the software. In an RWE study it is very common that a request evolves. For example a cohort definition changes or an analysis assumption is removed based on preliminary data. It is essential that these major changes are documented over the life-cycle of the RWE study. While git tracks changes to code through commits, the NEWS file summarizes the series of commits from version to version. Typically we use semantic versioning for Ulysses studies, incrementing releases based on either the level of impact it has on the pipeline or level of dissemeniation. The NEWS file should be brief, with a bulleted list of changes from the previous version.

### 3.1.5 config.yml

This is crucial settings file specifies the parameters needed to establish a connection to the database hosting RWD using `DatabaseConnector`. The config file uses Yaml a simple, readable data-serialization language used to provide settings or configurations for applications. The config file has two types of sections: default and block headers. The default block holds universal settings while the block header specifies details when the user has specified the block. When we source a block header in a task file, it will run the task based only on the block. This allows us to run pipelines easily on multiple databases. Credentials needed to run a study are as follows:

- **dbms**: the name of the dbms used to host the OMOP data
- **user**: your user name needed to access the data in the dbms
- **password**: your password needed to access the data in the dbms

- **connectionString**: this string sets the connection to the database. It will typically look something like: `jdbc:<dbms>://<server url>:<dbms port>/<database>`. This string will vary from site to site. Contact your system administrator to get the information you need for this credential
- **databaseName**: the name of the database used for reference. This should be stored in a snakecase format including the name of the database and snapshot date
- **databaseLabel**: the pretty label of the database used for final formats
- **cdmDatabaseSchema**: a name that defines where the cdm tables sit for a particular database in the dbms. `Database` defines the name of the database where the omop data sits and the `Schema` defines where the cdm is within that database. In databases like sql server or snowflake this variable is often separated by a "." i.e. <database>.<schema>
- **vocabDatabaseSchema**: a name that defines where the vocabulary tables sit for a particular database in the dbms. The vocabDatabaseSchema is typically the same as the cdmDatabaseSchema, although same sites split the vocabulary into a different area.
- **workDatabaseSchema**: a name that defines where the work tables sit for a particular database in the dbms. The work section (referred to synonymously with scratch or write) is an area where the user is given read and write access in the database. The cdmDatabaseSchema is typically only a read-only schema, so users do not corrupt information. The workDatabaseSchema is a dedicated area where researchers can build cohort tables or other intermediary tables needed for studies. Contact your database administrator to ensure you have a dedicated workDatabaseSchema. Note this should be separate from the database schema dedicated to ATLAS.
- **tempEmulationSchema** (Optional): this is a schema used to evaluate temp tables in certain databases like snowflake and oracle.
- **cohortTable**: the name of the cohort table to create in the work schema. By default this will take the repo name and database name in snakecase.

Below is an example of a config file:

```
# Config File for my_ohdsi_study

default:
  projectName: my_ohdsi_study
  version: v0.0.1

# Config block for example

example:
  dbms: snowflake
  user: !expr Sys.getenv('dbUser')
  password: !expr Sys.getenv('dbPassword')
  connectionString: !expr Sys.getenv('dbConnectionString')
  databaseName: my_cdm
  databaseLabel: My CDM
  cdmDatabaseSchema: cdm_schema
  vocabDatabaseSchema: cdm_schema
  workDatabaseSchema: work_schema
  tempEmulationSchema: temp_schema
  cohortTable: o99996_example
```

If we wanted to add another credential in this block we can add new line beneath `cohortTable`. Be sure to keep the indentation and add an extra space after.

## 3.2 Analysis Folder

The analysis folder contains all R code that executes the RWE study. There are two sub-sections: tasks and src (pronounced source). Tasks are files that implement the RWE pipeline when we call an execute command. **src** internal files maintain code that helps run these functions.

The task folder maintains a numbered list of files that run an element of the pipeline. Task files follow this general cadence: a) list parameters and assumptions b) connect to the database, c) extract aggregations from database based on specifications [this may include additional steps using R], and d) store output to the correct results folder. In each task file we specify a configBlock. As stated in the config.yml section, the config block specifies the block header of the data we want to use in the connection. This allows us to reuse the same task code to run on multiple databases.

## 3.3 Inputs Folder

The inputs folder is meant to organize files that are used to generate cohort definitions and concept sets in the database. When using OHDSI tools, cohort definitions and concept sets are specified in a `json` file format following the specifications of `circe-be`. We suggest placing cohorts used in the analysis in the inputs folder in order for the study to be self-contained. Keeping track of cohorts can be a bit tricky. Ulysses provides manifest functions to maintain the organization of these inputs, described in another vignette. The inputs folder has the following sub-folders:

- cohorts: location of objects describing the logic of study population for analysis
  - /json: in OHDSI studies the json folder holds circe-be definitions that have standard serialization to sql
  - /sql: some studies require custom sql to implement a cohort. Custom sql should be stored here
- conceptSets: storage of code lists used to describe clinical entities used in the analysis -/json: in OHDSI studies concept sets are described in circe-be. This allows for more efficient use of vocabulary hierachy.

When manifests are enables, there will be csv files tracking the life-cycle of study input assets in this folder as well.

## 3.4 Dissemination Folder

Every RWE Study needs to disseminate content explaining the assumptions and results of the study. All of these files are stored in the dissemination folder within Ulysses. There are some sub-folders:

- documents: stores important documentation for the study. Possible complimentary literature or a word doc of the study protocol would be stored here
- export: stores data that is ready for export and may be stored in git
  - /merge: holds csv files merging different cohorts and data sources run in the pipeline. This data format is long.
  - /pretty: holds xlsx files storing pivoted results to send to stakeholders. This data format is custom based on the stakeholder request.
- quarto: holds files needed to build and host the study hub. Details to follow.

### 3.4.1 Study Hub

The StudyHub is the website of the RWE study, a series of linked html files that describe the studies, outline assumptions and communicate results. A StudyHub uses quarto websites to weave together multiple html files in a single linked website. Each study should have a hub communicating final results in a sleak html format. This is analgous to a package website built using `pkgdown` for sites communicating the ideas of an R package. A study hub has the following elements:

- _quarto.yml: a settings file dictating how to knit the quarto website
- egp.qmd: A quarto file communicating the layman assumptions used in the RWE study. More details about the EGP are left for another vignette.
- images: optional folder to add images
- index.qmd: A quarto file copying the README.md file serving as the cover-page of the study hub for html
- news.qmd: A quarto file copying the NEWS.md file for html
- R: A folder holding R scripts that may be referenced in the study hub for rendering tables or figures
- report: A folder holder qmd files to use in the report

- style.css: a file that dictates universal html stylings such as color schemes
- _site: a folder created after the first build of the StudyHub holding rendered html files

### 3.5 Extras Folder

Sometimes studies have files that don't have a natural location from those specified above. The extras folder is a location to store any additional files needed to run or support the execution of the study.