CPRD - Tutorial

Exeter Diabetes team

Contents

Chapter 1

Intro

Welcome to the comprehensive guide on accessing CPRD (Clinical Practice Research Datalink) data. This document is designed to assist researchers and analysts in navigating the intricacies of CPRD, offering step-by-step instructions on:

- Accessing the data
  - CPRD data
  - How to request a new download
- Setting up datasets
  - Creating new tables in MySQL
  - Importing data into local RStudio session
  - Other tips
- Variables
  - Codelist
  - Variable definitions

For more information contact the Exeter Diabetes team.

Chapter 2

Accessing CPRD data

The information below is an overview of the process taken to access the data.

2.1 CPRD dataset

The UK routine clinical data is available in the CPRD repository (CPRD; https://cprd.com/research-applications), but restrictions apply to the availability of these data and, therefore, are not publicly available. An application must be made directly to CPRD.

All the stored CPRD tables are the exact tables provided by CPRD.

If you are using CPRD in your analysis, do not forget to add the statement below to the 'Data availability' section:

> The UK routine clinical data analysed during the current study are available in the CPRD repository (CPRD; https://cprd.com/research-applications), but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. For re-using these data, an application must be made directly to CPRD.

2.2 Requesting a new download

The University of Exeter has several key FOB holders that can download a raw version of the CPRD data to answer your questions. Please ask a member of the "CPRD User Group" to direct you towards a key FOB holder member of staff.

2.2.1 Codelist

In order to download the desired cohort, the key FOB holder requires a list of **Medcodes** or **Prodcodes** that can identify the specific patients required. Some codes already defined by the

Exeter Diabetes team can be found in the GitHub repository Exeter-Diabetes/CPRD-Codelists, more information about these codes will later in this tutorial.

2.2.2 CPRD data storage

With a list of **Medcodes** or **Prodcodes**, the key FOB holder can now download a collection of raw tables from CPRD with all of the patients which had the codes in the records. The tables will be zipped (file extension .zip) and will need be to stored in a secure server.

> **Our contract with CPRD states that individual patient-level CPRD data can only be stored in a specific folder on the server, which is an encrypted area of the server, or on MySQL on the server. You should not save individual patient-level data to your local computer, Github (whether a private or public repository), or other areas of the server as this violates our contract. Only summary statistics/aggregate data e.g. averages and counts should be saved elsewhere.**

In the Exeter Diabetes team, the data is stored in a university server which requires you to be connected to the university network (physically or through a VPN) in order to access it.

2.2.3 MySQL setting up the data

This step can be done however best fits your organisation. Below is an overview of how the Exeter Diabetes team sets up the database.

The MySQL database is setup using the R package Exeter-Diabetes/CPRD-analysis-package.

A tutorial on how to get started with the database can be found here:

https://github.com/Exeter-Diabetes/CPRD-analysis-package/blob/master/vignettes/getting-started-redacted.Rmd

Chapter 3

Setting up datasets

This section takes advantage of the R package `aurum`. This R package can be found in the CPRD-analysis-package Github repository: https://github.com/Exeter-Diabetes/CPRD-analysis-package.

3.1 Creating new tables in MySQL

The vignette 'setting-up-an-analysis in the `aurum` R package has examples of most of the functions in the `aurum` package (e.g.: `analysis$cached`).

Link: https://github.com/Exeter-Diabetes/CPRD-analysis-package/blob/full-dataset/vignettes/setting-up-an-analysis.Rmd))

If you want to save (cache) tables on MySQL, you have to first choose the name of your analysis, which will be used as a prefix for all table names:

```
analysis = cprd$analysis("katie_test")
```

Note: caching a table using the same table name as one which already exists will not overwrite it - it will simply create a pointer to the table which already exists. If you want to overwrite a table name you need to delete it first.

When you cache a table (save it on MySQL), you can add indexes to fields of your choosing e.g.:

```
my_table <- my_table %>% analysis$cached("my_table",
                                         indexes=c("drugclass", "dstartdate"),
                                         unique_indexes="patid")
```

Having indexes will speed up future queries if you join or filter based on the indexed fields. Note that RStudio will send the query to create the cached table, and once this query is complete it will send the query to create the indexes, so if you disconnect before the indexing queries are sent then the indexes will not be created.

3.2 Importing data into local RStudio session

> **You should not save individual patient-level data to your local computer or GitHub - this violates our contract with CPRD. Only summary statistics, e.g.: means or medians, should be saved.**

You can import data from a remove MySQL table into your local RStudio.

Use the collect command:

```
local_table <- remote_table %>% collect()
```

When importing data from MySQL into RStudio, sometimes variables are in the `integer64` format which can give errors with some R functions; these need to be converted (I use a function to convert all `integer64` fields to integers):

```
is.integer64 <- function(x){class(x)=="integer64"}
table <- table %>% mutate_if(is.integer64, as.integer)
```

Sometimes this causes `integer overflow` issues for long `patids` - if so, convert these to strings instead:

```
mutate(patid=as.character(patid))
```

3.3 Other tips

- The Exeter-Diabetes/CPRD-Cohort-scripts GitHub repository also has lots of example scripts.
- Functions that do not work in `dbplyr` (the R package that `aurum` uses to translate R into MySQL code) include:
  - `median`
  - `slice`
  - `difftime`: use `datediff` instead
  - `arrange`: use `window_order` instead.

You can add `show_query()` to view the MySQL query that `dbplyr` has constructed from your R code:

```
cprd$tables$observation %>% inner_join(codes$creatinine_blood) %>% show_query()
```

- Note that if a query is in progress when you exit RStudio, it will carry on running until complete.

- Sometimes you need to problem-solve using MySQL directly. Deleting tables is also easier via MySQL.

Note Command-line MySQL:

Using a terminal such as Command Prompt, MobaXTerm or Terminal, connect to the server. You can then use `mysql -u user -p` (replace `user` with your SSO username) to connect to the MySQL server (you will be prompted for your password on running this command). Run `exit` to exit.

Some basic MySQL commands:

- `drop table my_table;` will delete your table. You need to be in the correct database (e.g.: by running `use cprd_analysis_dev;`) or specify the database in the drop command (e.g.: `drop table cprd_analysis_dev.my_table;`)
- `show processlist;` will show you all processes (queries) running under your username. If you are running queries from your local RStudio, these will carry on even if you quit RStudio. `show full processlist;` will show you the full queries running - this is easier to do in MySQL Work bench rather than terminal.

Note MySQL Workbench:

MySQL Workbench is a GUI for MySQL programming; others are available.

Chapter 4

Variables

This section provides variable definitions which the Exeter Diabetes team uses for their datasets.

4.1 Codelist

The Exeter Diabetes codelists are available at Exeter-Diabetes/CPRD-Codelists.

This repository includes **Medcodes** (for use with the Observation table) and **Prodcodes** (for use with the Drug Issue table), as well as **ICD10** and **OPCS4** codelists for use with linked HES APC data. All codelists have been clinically-reviewed except a small subset where review by a non-clinician was deemed acceptable (BMI, weight, height and blood pressure readings) or where PBCL biomarker codes were used (see Codelist generation process: https://github.com/Exeter-Diabetes/CPRD-Codelists/blob/main/readme.md#code-list-generation-process).

We have also included **Read** and **SNOMED** codelists created during medcode list development (see Codelist generation process); where available, these are located in each of the subfolders in the **Medcodes** folder.

All codelists are based on an October 2020 extract of CPRD Aurum; later versions may include extra **Medcodes/Procodes** not included here.

4.2 Variable definitions

The definitions and derivation algorithms for the variables used are available at Exeter-Diabetes/CPRD-Codelists.

This GitHub repository, provides a detailed view on:

1. Biomarker derivation algorithms (https://github.com/Exeter-Diabetes/CPRD-Codelists/tree/main?tab=readme-ov-file#biomarker-algorithms)

We have developed the R package `EHRBiomarkr` (available in the GitHub repository Exeter-Diabetes/EHRBiomarkr), which includes various functions for cleaning and processing biomarkers in EHR, especially in CPRD Aurum. All functions can be used on local data (loaded into R) or data stored in MySQL (by using the `dbplyr` package or another package which uses `dbplyr`).

Two functions for cleaning biomarkers values are included in this package:

- `clean_biomarkers_values`: removes values outside of plausible limits. Run `?biomarkerAcceptableLimits` for details of how these were ascertained and further explanation of variables. More information on the variable limits can be found here: https://github.com/Exeter-Diabetes/CPRD-Codelists/blob/main/Medcodes/Biomarkers/biomarker_acceptable_limits.txt.
- `clean_biomarkers_units`: retains only values with appropriate unit codes (numunitid) or missing unit code in CPRD Aurum. Run `?biomarkerAcceptableUnits` for details of how these were ascertained and further explanation of variables. More information on the variable unit codes can be found here: https://github.com/Exeter-Diabetes/CPRD-Codelists/blob/main/Medcodes/Biomarkers/biomarker_acceptable_units.txt.

These functions can be applied to a large range of biomarkers. See this for more information: https://github.com/Exeter-Diabetes/EHRBiomarkr?tab=readme-ov-file#biomarker-cleaning-functions

The R package also includes functions for calculating more complex variables such as:

- eGFR from creatinine, age and sex: https://github.com/Exeter-Diabetes/EHRBiomarkr?tab=readme-ov-file#egfr-from-creatinine-age-and-sex-using-ckd-epi-creatinine-equation-2021
- cardiovascular risk score functions: https://github.com/Exeter-Diabetes/EHRBiomarkr?tab=readme-ov-file#cardiovascular-risk-score-functions
- kidney risk score functions: https://github.com/Exeter-Diabetes/EHRBiomarkr?tab=readme-ov-file#kidney-risk-score-functions

2. Comorbidity derivation algorithms (https://github.com/Exeter-Diabetes/CPRD-Codelists/tree/main?tab=readme-ov-file#comorbidity-algorithms)
3. Diabetes derivation algorithms (https://github.com/Exeter-Diabetes/CPRD-Codelists/tree/main?tab=readme-ov-file#diabetes-algorithms)
4. Sociodemographics derivation algorithms (https://github.com/Exeter-Diabetes/CPRD-Codelists/tree/main?tab=readme-ov-file#sociodemographics-algorithms)