An introduction to high-resolution patient record data for observational health research

Darren Wilkinson

darrenjw.github.io

Introduction

Background

Modern hospitals within “digitally mature” NHS Trusts have very sophisticated electronic health record (EHR) systems for tracking patient information:

- Basic demographic information
- Medical background
- Hospital admittance information (and departure, and death)
- Movements within the hospital (eg. moves between wards, and in and out of ICU)
- Vital signs monitoring (temperature, heart rate, blood pressure, respiratory rate, . . .)
- Every sample and measurement taken, observation made
- Every treatment and drug administered
- Every intervention (eg. ventilation, ventilation settings)

All of this information is recorded, and it has enormous potential for understanding patient trajectories and improving patient care.

The need for standardised data

- Hospitals are full of custom and proprietary devices and IT systems with no common standard
- This makes it extremely difficult to directly use the data recorded by any given hospital IT system
- Different systems will:
  - Use different names/codes for the same thing
  - Use the same name/code for different things
  - Measure things in different units
  - Store things in different ways in different (proprietary) data formats
- Ultimately, it would be nice if all hospital IT systems stored data in the same way, but that isn't going to happen any time soon, and in any case is not necessary to facilitate better utilisation of EHR data
- All we need is that each hospital is able to output the data from its own internal representation into a standardised format for subsequent analysis

The OMOP Common Data Model

The OMOP Common Data Model

- The Observational Medical Outcomes Partnership (OMOP) was a major NIH-funded US public-private partnership created to address the need to common data formats and IT infrastructure for patient data and their EHR
- This led to the creation of the OMOP Common Data Model (CDM), which is now maintained by the Observational Health Data Sciences and Informatics (OHDSI, pronounced "Odyssey") community
- The OMOP CDM is essentially just a standardised set of relational database tables for EHRs, described precisely in a Data Definition Language (DDL), along with a standardised vocabulary (ontology) of clinical terms and concepts
- Software tools and algorithms designed to work with OMOP data can work more-or-less uniformly across any data from any patient population anywhere in the world, so long as the data can be transformed into the OMOP CDM

Standard tables

- The OMOP CDM is based on the standard relational model, and assumes data stored in a relational database management system (RDBMS), to be queried using the Structured Query Language (SQL)
- Standard tables include `person`, `visit_occurrence`, `visit_detail`, `measurement`, `observation`, `specimen`, `concept`, `concept_ancestor`, ..., each of which contains standardised fields
- eg. the `measurement` table includes fields `measurement_id`, `person_id`, `measurement_concept_id`, `measurement_datetime`, `value_as_number`, `unit_concept_id`, ...

Querying and joining

- Connect to the RDBMS and issue queries via SQL

```
psql -h localhost -d synpuf
```

- Information is extracted from the RDBMS using SQL queries

```
select * from person
  where (year_of_birth < 1960) and
(gender_concept_id = 8507);
```

- Information in one table can be linked against information in another by joining on the linking fields

```
select v.*, p.*
  from visit_occurrence v, person p
  where v.person_id = p.person_id;
```

Standard vocabulary and the concept tables

- Most of the CDM tables contain fields with names of the form `XXX_concept_id`
- These fields contain integers representing a concept contained within a standard vocabulary of clinical terms
- The CDM `concept` tables contain the information necessary to decode these IDs into meaningful concepts, including human-readable `concept_names`

```
select m.*, c.concept_name
  from measurement m
  left join concept c on
m.measurement_concept_id = c.concept_id;
```

- The databases of clinical terms can also be searched online using the *OHDSI Athena* database

Querying from R

```r
library(DBI)
con <- DBI::dbConnect(RPostgres::Postgres(),
       host = "dell13.home", dbname = "synpuf")

df = dbGetQuery(con, "select v.*, p.*
  from visit_occurrence v, person p
  where v.person_id = p.person_id;")

df[1:5, 1:3]
```

```
##   visit_occurrence_id person_id visit_concept_id
## 1                   6         1                0
## 2                   2         1             9202
## 3                   5         1                0
## 4                   4         1                0
## 5                   3         1                0
```

Resolving concepts

```r
df = dbGetQuery(con, "select m.*, c.concept_name
  from measurement m
  left join concept c on
    m.measurement_concept_id = c.concept_id;")

df[1:5, 1:3]
```

```
##   measurement_id person_id measurement_concept_id
## 1             35         4                      0
## 2            131         7                      0
## 3            342        14                      0
## 4            776        23                      0
## 5            932        28                      0
```

Python

- There is a strong correspondence between the results of an SQL `select` query and data frame objects in languages such as R and Python
- In Python, Pandas is the standard library providing data frame functionality similar to that found in R
- We can send SQL queries and get results back into a Pandas data frame similarly to how we have done it in R

```
import pyodbc
import pandas as pd
import numpy as np
# ... set connection info here...
cnxn = pyodbc.connect(f"DRIVER={{ODBC Driver ...

df = pd.read_sql("""select v.*, p.*
  from visit_occurrence v, person p
  where v.person_id = p.person_id;""", cnxn)
```

Querying and wrangling with tidyverse tools

- Statisticians and ML researchers typically want all of their data arranged into a big rectangular data frame object for input into their algorithms
- Getting data into the form expected by typical prediction models requires very large and complex SQL queries
- Statisticians and data scientists would typically prefer to wrangle data using tools such as those provided by the R `tidyverse`
- OMOP databases are often very large, so it is a bad idea to pull big tables into R and then start joining them using R code (and similar for Python)
- Want to do as much work as possible *inside* the RDBMS, since that is designed for working with large datasets
- Actually possible to get the best of both worlds, even using R, using packages such as `dbplyr` (note the `b`)

dbplyr

- dbplyr connects SQL RDBMSs to the tidyverse by means of a "lazy tibble", which can be computed with in R much like any other dplyr tibble, but is really just a tibble-like wrapper around an SQL query

```
library(tidyverse)
tbl(con, "visit_occurrence") %>%
  inner_join(tbl(con, "person"), by = "person_id")
```

```
## # Source:   lazy query [?? x 34]
## # Database: postgres [ndjw1@dell13.home:5432/synpuf]
##    visit_occurrenc~ person_id visit_concept_id visit_sta
##               <int>     <int>            <int> <date>
##  1                6         1                0 2009-07-2
##  2                2         1             9202 2008-09-0
##  3                5         1                0 2010-11-0
##  4                4         1                0 2010-04-0
##  5                3         1                0 2009-10-1
```

omopr

- I've written a little R package, omopr (on CRAN), to facilitate working with OMOP CDM data using a tidyverse approach
- As well as initialising a connection by creating a list of CDM table references, the main function in the package resolves concepts in a lazy tibble by left-joining each concept_id column against the concept table

```
library(omopr)
tRefs = omopr_init(con)
vpc = tRefs[["visit_occurrence"]] %>%
  inner_join(tRefs[["person"]], by = "person_id") %>%
  concept_names()
vpc
```

```
## # Source:   lazy query [?? x 45]
## # Database: postgres [ndjw1@dell13.home:5432/synpuf]
##    visit_occurrenc~ person_id visit_concept_id visit_sta
##               <int>     <int>            <int> <date>
```

Executing the query

When you've finished wrangling your lazy tibbles, you can force execution of the query, and extraction of data into a strict `dplyr` tibble, for in-memory analysis using R, by calling `collect()`

```r
vpc %>% collect()
```

```
## # A tibble: 55,261 x 45
##    visit_occurrenc~ person_id visit_concept_id visit_sta
##               <int>     <int>            <int> <date>
##  1               54         4                0 2009-11-2
##  2               55         4                0 2009-09-2
##  3             5062        96                0 2008-03-0
##  4             5046        96                0 2008-06-0
##  5             5047        96                0 2010-01-0
##  6             5048        96                0 2010-05-2
##  7             5049        96                0 2009-10-2
##  8             5050        96                0 2008-10-3
##  9             5051        96                0 2010-01-2
```

Statistical modelling of ICU data

Prediction models

- The querying and joining examples we have seen so far are incredibly simple, but in practice, rather complex wrangling code is needed to coerce the OMOP data into the form required by typical statistical analysis routines
- Prediction models will often attempt to compute a risk score for a patient on the basis of the available information at the time of admission, or transfer to ICU, or within 24 hours of transfer to ICU
- The computation of illness severity scores typically requires non-trivial wrangling of OMOP data
- Most existing scores were developed prior to the current coronavirus pandemic - assessing the utility of existing severity scores, and improving on them, for patients exhibiting COVID-19 symptoms, is a topic of current interest
- Wynants et al (2020) Prediction models for diagnosis and prognosis of COVID-19: systematic review and critical appraisal, *BMJ* - "living" systematic review

Causal inference from observational data

- In addition to making predictions about likely outcome, we might also be interested in identifying effective treatment interventions
- Carefully designed large randomised clinical trials are the gold standard for checking the effectiveness of a proposed new treatment, as they provide some protection against confounding variables confusing association and causation
- When making inferences using observational data, great care must be taken to measure and adjust for as many potential confounders as possible, and not to over-interpret findings
- In recent years, increasing use has been made of techniques from the causal inference literature, using large observational data sets to emulate a "target trial" you would have liked to have done
- Hernán & Robins (2016) Using big data to emulate a target trial when a randomized trial is not available, *American Journal of Epidemiology*.

Conclusions

Conclusions

- High-resolution EHR data from ICU patients has very significant potential for generating insight that could improve patient outcomes
- Data-standardisation is necessary to allow pooling of data across multiple hospitals and Trusts
- The OHDSI OMOP CDM format is the closest thing we have to a proper data standard for EHR data
- Wrangling hospital data from internal proprietary systems into the OMOP CDM is a non-trivial undertaking
- Wrangling OMOP data into rectangular data frame formats that statisticians and other analysts are most comfortable with is also challenging
- The potential long-term payoffs of this standardised approach to EHR data is huge

Links for further reading

- https://www.ohdsi.org/
    - https://www.ohdsi.org/data-standardization/the-common-data-model/
- https://github.com/OHDSI/
    - https://github.com/OHDSI/CommonDataModel/
- https://athena.ohdsi.org/
- https://cran.r-project.org/web/packages/dbplyr/vignettes/dbplyr.html
- https://cran.r-project.org/web/packages/omopr/
- Wynants et al: https://www.bmj.com/content/369/bmj.m1328
- Hernán & Robins: https://doi.org/10.1093/aje/kwv254
- https://www.hsph.harvard.edu/miguel-hernan/causal-inference-book/
- https://decovid.org/