PhUSE 2017

Paper RW07

Real World Data Surprises

Ewa Śleszyńska-Dopiera, Quanticate, Warsaw, Poland
Jan Rudnik, Quanticate, Warsaw, Poland

ABSTRACT

Our job as Real World Data programmers is to put the whole patient data, usually spread across multiple tables, into one coherent history. This might lead to some surprising discoveries such as:

- Patients with records before birth/after death,
- Patients changing sexes multiple times,
- Finding non-valid codes (procedures/diagnoses).

Such discrepancies usually are pretty easy to classify as data issues and are easy to handle. On the other hand we might encounter real world data events that cannot be as easily classified and need special approaches such as:

- Patient receiving multiple Rx's for the same drug on one day, or receiving a new Rx's before the previous runs out,
- Patients visiting the Doctor's office / Emergency Room during their hospital stays,
- Patients changing wards/level of care multiple times during one hospital visit,
- Finding valid but non-billable codes in insurance claims data

We will present examples and our ways of dealing with them.

INTRODUCTION

We are a part of a multinational programming team working with Real World Data (RWD). We use many different observational longitudinal databases (most of them are commercial, but some are also publicly available). In this paper we share some commonly encountered problems (and our solutions) related to Real World Data analysis.

REAL WORLD DATA

To assess what is happening in the real world, rather than using clinical trials to collect data, the researcher may use data which come directly from the market. Observational longitudinal databases are created to allow for analysis of such data. These contain de-identified medical records for a large number of patients (the biggest one we work with has over 100M patients) over an extended period of time (at least a couple of years) which is a much larger scale than regularly used in clinical trials. This allows for analyses of rare diseases, treatment pattern changes, etc. However, unlike in clinical trials, the researcher will not be able to randomly assign patients to a given therapy nor collect all characteristics that may be of interest. The de-identification process of these data need to follow country specific guidelines for protecting individual health care information[1], such that available data do not include any information that could potentially be used to identify a person.

The world of Real World Data is a long way from being standardised. Some efforts are being made to counteract this but they consistently sacrifice some database specific advantages. What this means for a regular programmer is that each database you work with has its own structure, advantages and limitations and requires an individual approach. The databases that we work with can be grouped into two main categories:

- Administrative Healthcare databases (claims databases) include records (claims) created by medical facilities and sent to an insurance company in order to get paid for the services performed. Note that claims are created for billing purposes, not for research. Administrative Healthcare databases are usually:
  - clean (e.g. low numbers of invalid codes)
  - consistent (they arise from standardised claims documents)
  - restricted in scope (e.g. height/weight information will not be included as this is not something that insurance companies need)
- Electronic Medical Records (EMR/EHR) databases contain patients' medical records from many facilities (sometimes grouped together into networks) aggregated by one entity. Such databases are usually:
  - messy
    - data may come from many different sources and can be hard to shoehorn into a rigid database,
    - different approaches to define certain events may be used by data providers,
    - codes may be invalid,

[1] e.g. in the United States: Final Privacy Rule of the Health Insurance Portability and Accountability Act of 1996 (HIPAA), American Recovery and Reinvestment Act of 2009 (ARRA)

    - records may be incomplete, with missing or illegible values
    - someone may have forgotten to report e.g. an injection; no financial risks involved
  - surprising:
    - complicated algorithms are used to map the data to the database, and these may occasionally give unexpected results
    - different information available for a patient such as: lung volume, pain score etc. (the size of the population with such information is a completely different story but the data is there and could probably support some interesting projects)

Of course even databases of the same type can have many significant differences. As a team working on multiple databases we have to explore standard approaches of solving commonly observed problems. We present some of these (keeping some secrets to ourselves).

RWD PROBLEMS WITH STANDARD SOLUTIONS

There are definitely some problems that are easy to solve. Usually it is enough to be aware of such issues and write a program that handles such cases properly. Even after such measures, the programmer must always review each intermediate step carefully (look for fields with missing values, incorrect dates, etc.) as a content of the database might change over time. The following are some common data-driven problems.

HANDLING DATE OF BIRTH

Due to regulatory authorities most databases mask exact date of birth to avoid identification. Patients above certain age are usually flagged (no age provided). All remaining patients may only have the year of birth recorded (or in special cases year and month of birth). This causes problems with studies that require a more precise age (e.g. newborns).

If date of birth can be found in multiple places in the database then it might happen that a patient has more than one unique date of birth provided. This might mean that some of their records might not be theirs after all. In such cases our approach has been to exclude such patients from the analysis.

Similarly for gender - even though sex changes happen, seeing more than one gender in patient’s records is much more likely to be a result of a data issue than a physical sex change.

PROBLEMS WITH DATE OF DEATH

Since in both types of databases (claims and EMR) we see events taking place mostly within medical facilities, only death that occurred in these facilities can be captured directly. Most of the databases that supply death data for their patients use a different source for this information. This can lead to another source where discrepancies may occur i.e. mismatches of patients’ records and death date can result in death date being assigned to patients that are alive.

To counteract that, we compare all data available in database with date of death. If there are records which are substantially after the date of death we treat these patients as being alive (assuming incorrect assignment of death date).

PROBLEMS WITH CODES

A Real World Data programmer comes across multiple coding systems. Diagnoses, procedures, medications are all provided as codes (e.g. ICD-10, NDC, HCPCS, Readcodes) and looking for a particular diagnosis, for example, one needs to be aware which coding system has been used in the database and what the code value for the diagnosis is (there may be multiple values). Even the same coding system can have database specific format (truncation, dots, dashes, leading zeros etc.). There might also be codes that closely resemble another from the system but may not actually exist (e.g. looking for ICD-9: 250.* (diabetes) and finding 250.07 (non-existing sub-code)). Similarly a given code might belong to more than one coding system so one needs to be sure that the code found in the database is the correct one.

RWD PROBLEMS WITH STORIES

The ease of solving previous problems came mainly from the fact that the origin of a given problem was quite straightforward. Here we present problems that might be results of both data issues and rare patient’s characteristics/treatments. These challenges rarely have fire-and-forget solutions and clinician input is usually needed in order to find the best project-specific approach.

NON-BILLABLE CODES IN INSURANCE CLAIMS DATA

We have already covered the issue of invalid codes and now progress onto more complicated case of billable ICD codes. ICD codes have a hierarchical structure: for example ICD-9 824 is a general code whilst 824.8 is a code more precisely describing given condition..

As the name suggests, only billable – the most specific of the codes – can be used for reimbursement purposes. As a result only such codes should be expected in claims databases. Presence of a non-billable code on a claim might be a data issue. In rare cases however, when the most accurate diagnosis is hard to pick, a physician may not be able to provide a billable code and decide to put only a general (non-billable) code instead. When encountering non-billable codes, the programmer can either treat them all as invalid or use them as correct codes (e.g. for rare diseases).

OUTPATIENT AND EMERGENCY ROOM VISITS

Some of the studies require a programmer to provide Health Care Resource Utilisation which includes visit counts. Unfortunately, most of the databases do not provide a clear grouping of records into a visit. Definition of a single event needs to be agreed upon between the programmers and statisticians.

Consider a patient with a couple of claims (ignoring claims coming from hospitals) on a single day. How should one decide how many visits actually happened that day?

- Usually when a patient wants to see a physician they only go for one visit on a given day
  - Maybe making the assumption of one visit per day is sufficient, regardless of the number of claims present?
- But what if a patient visits both GP and a specialist?
  - Maybe the number of distinct providers (physicians) on that day should be used instead?
- But what if the doctor identifier is not available or masked to only show specialty?

Here is the algorithm we found to be both specific enough and general enough for most projects – if there are multiple claims per date for a patient:

- Group the claims by unique providers,
- If above is not available use specialty/facility instead
- If none of these are available, group such claims into one visit

Now imagine the same situation, but with all claims coming from Emergency Room (ER). This time the assumption of counting visits by unique providers/specialties does not seem right. We found it reasonable to assume at most one ER visit a day, as a patient during one ER visit may end up seeing multiple physicians.

HOSPITAL STAYS

We define inpatient stay (IP) as the patient spending at least one night in a medical facility. For such stays there are two types of data: one summary record containing general information of a given stay (length of stay, date of admission/discharge, primary/principal diagnosis, discharge status etc.) and records for all services provided. Theoretically, using summary record one should be able to easily provide all inpatient related statistics, but unfortunately there are some special cases that must be considered before showing final numbers. First of all one needs to understand correctly how the database groups inpatient records:

- What if a patient changes a ward during the visit,
- What if a patient changes level of care,
- What if a patient is admitted through ER,
- What if a patient is transferred to another facility,
- What if a patient is readmitted on the day after discharge.

The final algorithm has to account for all of this information when analysing hospital stays (counting visits, calculating length of stay etc.). One important note here is that it might (and usually does) happen that grouping performed by vendor (summary records) results in some overlapping visits. The algorithm has to account for these too.

Depending on the purpose of the study:

- To compare inpatient related variables between different databases, inpatient stay definitions need to be “normalised” between the databases (e.g. using OMOP standard),
- To perform an in-depth analysis of history of patient’s hospital stays (readmissions, transfers etc.) one might have to modify vendor’s grouping algorithm so that it handles cases of interest correctly (meaning in line with study documentations),
- If only need basic IP related numbers (e.g. to compare patients on different drugs), vendor’s definition will usually be enough.

DRUG USAGE ANALYSIS

As the world’s largest data-focused CRO, Quanticate is contracted by pharmaceutical companies to provide drug development and pharmaceutical support, which means that most of our analyses are related to drugs. There is a big difference between claims and EHR databases with regards to drug records: the former shows fills (purchased drugs) while the latter shows prescriptions received. There is no guarantee that the patient actually used the medication (in both types of data).

Apart from drugs prescribed, databases also provide information about drugs received in medical facilities. For example injectable drugs, retrieving those might require using procedure codes in addition to drug codes.

Checking how many patients use a given drug is usually straightforward. On the other hand analysing treatment patterns can be pretty sophisticated. The first information needed to perform such an analysis is how many days the prescription covers (usually it is provided as days supplied variable, but sometimes it can only be calculated from the number of pills prescribed and dosage information). Next the approach to administrations has to be decided (assume that it only covers a day of administration, impute some fixed covered days, exclude from analysis). After this part is done one needs to establish how to handle multiple prescriptions on a single day.

- If a patient gets many prescriptions for one compound (might be different products), then one need to decide if that means that the patient is extending days covered or that the patient is increasing their dose. It is crucial to get a clinician input on this as drug regimens can vary widely between medications,

- If a patient gets more than one compound then one needs to decide if it should be treated as a combination therapy or if the two compounds should be treated in a similar fashion as in point above (e.g. analgesics acting in similar way).

After all these considerations, even more complicated items are left:

- Account for overlapping days of supply of given drug and if the algorithm should allow for carrying over remaining pills. There are several reasons as to why a patient might buy a new drug before the previous one is finished:
    - To have a new package ready when the old one runs out,
    - Due to lost/missing pills,
    - To increase the dosage.
- Account for allowable grace period during which it is assumed the treatment is still working but a patient is not actively taking drug.
- Classify a patient initiating competitor medication after the original one. At what point should they be classified as a patient switching treatment versus having discontinued one and starting another.

CONCLUSION

We have covered a wide variety of topics that a Real World Data programmer will encounter across different databases. As you will probably have realised by now, there are a lot of “probably” and “maybe” situations and solutions in this paper. This is all attributed to how vast the area is that we are tackling. You should also be aware that most of the questions (barring only simple ones) in our field do not have clear-cut answers. This does not mean they are not covered in the “database user manual”, but means however that with over 3 years of work experience in this area, many consultations with programmers, statisticians and vendor’s support we still end up with answers that only have a strong maybe at best in them.

ACKNOWLEDGMENTS

We would like to thank Marcin Michalski (of StatFinn, former Quanticate employee) for introducing us to the world of Real Data analysis and bearing with us when we explored every weird and complicated case we encountered.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ewa Śleszyńska-Dopiera
Quanticate
Email: ewa.sleszynska@quanticate.com

Jan Rudnik
Quanticate
Email: jan.rudnik@quanticate.com