# The Book of OHDSI

*Observational Health Data Science and Informatics*

*2019-07-12*

# Contents

# Preface

This is a book about OHDSI, and is currently very much under development.

The book is written in RMarkdown with bookdown. It is automatically rebuilt from source by travis.

## Goals of this book

This book aims to be a central knowledge repository for OHDSI, and focuses on describing the OHDSI community, data standards, and tools. It is intended both for those new to OHDSI and veterans alike, and aims to be practical, providing the necessary theory and subsequent instructions on how to do things. After reading this book you will understand what OHDSI is, and how you can join the journey. You will learn what the common data model and standard vocabularies are, and how they can be used to standard an observational healthcare database. You will learn there are three main uses cases for these data: characterization, population-level estimation, and patient-level prediction, and that all three activities are supported by OHDSI's open source tools, and how to use them. You will learn how to establish the quality of the generated evidence through data quality, clinical validity, software validity, and method validity. Lastly, you will learn how these tools can be used to execute these studies in a distributed research network.

## Structure of the book

This book is organized in five major sections:

I) The OHDSI Community
II) Uniform data representation
III) Data Analytics
IV) Evidence Quality
V) OHDSI Studies

Each section has multiple chapters, and each chapter aims to follow the following main outline: Introduction, Theory, Practice, Excercises.

## Contributors

TODO: make list of contributors complete

Each chapter lists one or more chapter leads. These are the people who lead the writing of the chapters. However, there are many others that have contributed to the book, whom we would like to acknowledge here:

| | | |
|---|---|---|
| Hamed Abedtash | Mustafa Ascha | Mark Beno |
| Clair Blacketer | Brian Christian | Gino Cloft |
| Sara Dempster | Jon Duke | Sergio Eslava |
| Clark Evans | Thomas Falconer | George Hripcsak |
| Mark Khayter | Greg Klebanov | Kristin Kostka |
| Bob Lanese | Wanda Lattimore | Chun Li |
| David Madigan | Sindhoosha Malay | Harry Menegay |
| Akihiko Nishimura | Ellen Palmer | Nirav Patil |
| Jose Posada | Dani Prieto-Alhambra | Christian Reich |
| Jenna Reps | Peter Rijnbeek | Patrick Ryan |
| Craig Sachson | Izzy Saridakis | Paula Saroufim |
| Martijn Schuemie | Sarah Seager | Chan Seng You |
| Anthony Senna | Sunah Song | Matt Spotnitz |
| Marc Suchard | Joel Swerdel | Devin Tian |
| Don Torok | Kees van Bochove | Mui Van Zandt |
| Kristin Waite | Mike Warfe | Jamie Weaver |
| James Wiggins | Andrew Williams | Chan You Seng |

# Part I

# The OHDSI Community

# Chapter 1

# Mission, vision, values

*Chapter lead: George Hripcsak*

## 1.1 Our Mission

To improve health by empowering a community to collaboratively generate the evidence that promotes better health decisions and better care.

## 1.2 Our Vision

A world in which observational research produces a comprehensive understanding of health and disease.

## 1.3 Our Objectives

- **Innovation**: Observational research is a field which will benefit greatly from disruptive thinking. We actively seek and encourage fresh methodological approaches in our work.
- **Reproducibility**: Accurate, reproducible, and well-calibrated evidence is necessary for health improvement.
- **Community**: Everyone is welcome to actively participate in OHDSI, whether you are a patient, a health professional, a researcher, or someone who simply believes in our cause.
- **Collaboration**: We work collectively to prioritize and address the real world needs of our community's participants.
- **Openness**: We strive to make all our community's proceeds open and publicly accessible, including the methods, tools and the evidence that we generate.

- **Beneficence**: We seek to protect the rights of individuals and organizations within our community at all times.

# Chapter 2

# Collaborators

*Chapter lead: Patrick Ryan*

History of OHDSI

Map of collaborators Forums Wiki Workgroups and chapters Symposia and hack-a-thons

Governance at local sites

# Chapter 3

# Open Science

*Chapter lead: Kees van Bochove*

From the inception of the OHDSI community, the goal was to establish an international collaborative by building on open science values, such as the use of open source software, public availability of all conference proceedings and materials, and transparent, open access publication of generated medical evidence. But what exactly is open science? And how could OHDSI build an open science or open data strategy around medical data, which is very privacy sensitive and typically not open at all for good reasons? Why is it so important to have reproducibility of analysis, and how does the OHDSI community aim to achieve this? These are some of the questions that we touch on in this chapter.

## 3.1 Open Science

The term 'open science' has been used since the nineties, but really gained traction in the 2010s, during the same period OHDSI was born. Wikipedia (Wikipedia, 2019a) defines it as "the movement to make scientific research (including publications, data, physical samples, and software) and its dissemination accessible to all levels of an inquiring society, amateur or professional", and goes on to state that it is typically developed through collaborative networks. Although the OHDSI community never positioned itself explicitly as an 'open science' collective or network, the term is frequently used to explain the driving concepts and principles behind OHDSI. For example, in 2015, Jon Duke presented OHDSI as "An Open Science Approach to Medical Evidence Generation"[1], and in 2019, the EHDEN projects' introductory webinar hailed the OHDSI network approach as "21st Century Real World Open Science"[2]. Indeed, as we shall see in this chapter, many of the practices of open science can be found in today's OHDSI community. One could argue that the OHDSI community is a grassroots open science collective driven by a shared desire for improving the transparency and reliability of medical evidence generation.

Open science or "Science 2.0" (Wikipedia, 2019b) approaches mean to address a number of perceived

[1] https://www.ohdsi.org/wp-content/uploads/2014/07/ARM-OHDSI_Duke.pdf
[2] https://www.ehden.eu/webinars/

problems within the current scientific practice. Information technology has led to an explosion of data generation and analysis methods, and for individual researchers, it is very hard to keep up with all literature published in their area of expertise. This holds even more true for medical doctors who have a practice to run as day job, but still need to keep abreast of the latest medical evidence. In addition, there is growing concern that many experiments may suffer from poor statistical designs, publication bias, p-hacking and similar statistical problems, and are hard to reproduce. The traditional method of correcting these problems, peer review of published articles, often fails to identify and tackle these problems. The special 2018 Nature edition on "Challenges in irreproducible research"[3] includes several examples of this. A group of authors attempting to apply systematic peer review on the articles in their field found that, for various reasons, it was very hard to get the errors they identified rectified. They encountered common statistical problems such as poor randomization designs leading to false conclusions about statistical significance, miscalculations in meta-analyses, and inappropriate baseline comparisons. (Allison et al., 2016) Another paper from the same collection, taking experiences from physics as an example, argues that it is critical to not only provide access to the underlying data, but also to publish openly the data processing and analysis scripts to achieve reproducibility.

The OHDSI community addresses these challenges in its own way, and puts significant emphasis on the importance of generating medical evidence at scale. As stated in Schuemie et al. (2018b), while the current paradigm "centres on generating one estimate at a time using a unique study design with unknown reliability and publishing (or not) one estimate at a time", the OHDSI community "advocates for high-throughput observational studies using consistent and standardized methods, allowing evaluation, calibration and unbiased dissemination to generate a more reliable and complete evidence base." This is achieved by a combination of a network of medical data sources that map their data to the OMOP common data model, open source analytics code that can be used and verified by all, and large-scale baseline data such as the condition occurences published at howoften.org. In the following paragraphs, concrete examples are provided and the open science approach of OHDSI is detailed further using the triad of Open Standards, Open Source and Open Data as a guide. The chapter is concluded with a brief reference to the FAIR principles and outlook for OHDSI from an open science perspective.

## 3.2 Open Science in Action: the Study-a-thon

A recent development in the community is the emergence of 'study-a-thons': short, concentrated face to face gatherings of a multidisciplinary group of scientists aimed at answering an important clinically relevant research question using the OMOP data model and the OHDSI tools. A nice example is the 2018 Oxford study-a-thon, which is explained in an EHDEN webinar (https://youtu.be/X5yuoJoL6xs) which provides a walkthrough of the process and also highlights the openly available results. In the period leading up to the study-a-thon, the participants propose medically relevant research questions to study, and one or more research questions are selected to study during the study-a-thon itself. Data is provided through participants that have access to patient-level data in OMOP format and are able to run queries on these data sources. Much of the actual study-a-thon

[3] https://www.nature.com/collections/prbfkwmwvz

time is devoted to discussing the statistical approach (see also the next chapter), the suitability of the data sources, the results which are interactively produced and the follow-up questions that are inevitably raised by these results. In the case of the Oxford study-a-thon, the questions centered around studying adverse post-surgical effects of different knee replacement methods, and the results were published interactively during the study-a-thon using the OHDSI forums and tools. [4]

## 3.3 Open Standards

A very significant community resource that is maintained in the OHDSI community is the OMOP Common Data Model and associated Standardized Vocabularies. The model itself is scoped to capture observational healthcare data, and is specifically meant to analyze associations between exposures such as drugs, procedures, devices etc. and outcomes such as conditions and measurements (see also chapter 5.1). However, harmonizing healthcare data worldwide from a wide variety of coding systems, healthcare paradigm and different types of healthcare sources requires a massive amount of 'mappings' between source codes and their closest standardized counterparts. The OMOP Standardized Vocabulary is further described in chapter 6 and includes mappings from hundreds of medical coding systems that are in used worldwide, and is browseable through the OHDSI Athena tool. By providing these vocabularies and mappings as a freely available community resource, OMOP and the OHDSI community make a significant contribution to healthcare data analytics and is by several accounts the most comprehensive model for this purpose, representing approximately 1.2 billion healthcare records worldwide (Garza et al., 2016) [5].

## 3.4 Open Source

Another key resource the OHDSI community provides are open source programs. These can be divided in several categories, such as the helper tools to map data to OMOP, the OHDSI Methods Library which contain a powerful suite of commonly used statistical methods, and ATLAS, Athena and other infrastructure-related software which underpins the OHDSI ecosystem. See chapter 9 for a detailed overview. From an open science perspective, one of the most important resources is the Methods Library, which ensures a consistent re-use of statistical methods across analytical use cases, and which can be inspected, reviewed and contributed to via GitHub.

## 3.5 Open Data

Because of the privacy-sensitive nature of healthcare data, fully open comprehensive patient-level datasets are typically not available. However, the OHDSI community provides simulated datasets

[4] https://forums.ohdsi.org/t/ohdsi-oxford-study-a-thon-any-data-partners-want-to-participate-this-week-in-two-studies-on-knee-
5650

[5] https://www.ema.europa.eu/en/events/common-data-model-europe-why-which-how

such as SynPUF for testing and development purposes, and the OHDSI Research Network (see chapter 21) can be leveraged to run studies in a network of available datasources that have mapped their data to OMOP.

## 3.6 OHDSI and the FAIR Guiding Principles

This last paragraph of the chapter takes a look at the current state of the OHDSI community and tooling, using the 15 FAIR Data Guilding Principles published in Wilkinson et al. (2016).

### 3.6.1 Findability

Any healthcare database that is mapped to OMOP and used for analytics, should from a scientific perspective be persisted for future reference and reproducibility. The use of persistent identifiers for OMOP databases is not yet widespread, partly because these databases are often contained behind firewalls and on internal networks and not necessarily connected to the internet. However, it is of course entirely possible to publish summaries of the databases as a descriptor record that can be referenced for e.g. citation purposes. This method is followed in for example the EMIF catalog [6], which provides a comprehensive record of the database in terms of data gathering purpose, sources, vocabularies and terms, access control mechanisms, license, consents etc. (Oliveira et al., 2019) This approach is further developed in the IMI EHDEN project.

### 3.6.2 Accessibility

Accessibility of OMOP mapped data through an open protocol is typically achieved through the SQL interface, which combined with the OMOP CDM provides a standardized and well-documented method for accessing OMOP data. However, as discussed above, OMOP sources are often not directly available over the internet for security reasons. Creating a secure worldwide healthcare data network that is accessible for researchers is an active research topic and operational goal of projects like IMI EHDEN. However, what can be openly published are results of analyses in multiple OMOP databases, as shown through OHDSI initiatives such as LEGEND and howfoften.org.

### 3.6.3 Interoperability

Interoperability is arguably the strong suit of the OMOP data model and OHDSI tooling. In order to build a strong network of medical data sources worldwide which can be leveraged for evidence generation, achieving interoperability between healthcare data sources is key, and this is achieved through the OMOP model and Standardized Vocabularies. However, by sharing cohort definitions and statistical approaches, the OHDSI community goes beyond code mapping and also provides a platform to build an interoperable understanding of the analysis methods for healthcare data. Since

[6] https://emif-catalogue.eu

healthcare systems such as hospitals are often the source of record for OMOP data, the interoperability of the OHDSI approach could be further enhanced by alignment with operational healthcare interoperability standards such as HL7 FHIR, HL7 CIMI, openEHR. The same goes for alignment with clinical interoperability standards such as CDISC and biomedical ontologies. Especially in areas such as oncology, this is an important topic, and the Oncology Working Group and Clinica Trials Working Group in the OHDSI community provide good examples of forums where these issues are actively discussed. In terms of references to other data and specifically ontology terms, OHDSI Athena is an important tool as it allows the exploration of the OMOP Standardized Vocabularies in the context of other available medical coding systems.

### 3.6.4 Reusability

The FAIR principles around reusability focus on important issues such as the data license, provenance and the link to relevant community standards. The data provenance of OMOP databases is a very interesting topic, as there are potential improvements for making these available in an automated way, provided the ETL and mapping tools would persist metadata about for example the used CDM version, Standardized Vocabularies release, custom code lists etc. The OHDSI ETL tools do not currently produce this information automatically, but working groups such as the Data Quality Working Group and Metadata Working Group actively work on these. Another important aspect is the provenance of the underlying databases itself, for example it is important to know if a hospital or GP information system was replaced or changed, and when known data omissions or other data issues ocurred historically. Exploring ways to attach this metadata systematically in the OMOP CDM is the domain of the Metadata Working Group.

## 3.7 Conclusions

To conclude, the OHDSI community itself can be seen as an open science community that is actively pursuing the interoperability and reproducibility of medical evidence generation. It also advocates a paradigm shift from single study and single estimate medical research to large-scale systematic evidence generation, where facts such as baseline occurence are known and the evidence focuses on statistically estimating the effects of interventions and treatments from real world healthcare sources.

# Chapter 4

# Where to begin

*Chapter leads: Hamed Abedtash and Krista Kostka*

> "A journey of a thousand miles begins with a single step." - Lao Tzu

The OHDSI community represents a mosaic of stakeholders across academia, industry and government-entities. Our work benefits a range of individuals and organizations, including patients, providers, and researchers, as well as health care systems, industry, and government agencies. This benefit is achieved by improving both the quality of healthcare data analytics as well as the usefulness of healthcare data to these stakeholders. We believe observational research is a field which benefits greatly from disruptive thinking. We actively seek and encourage fresh methodological approaches in our work.

## 4.1 Joining the Journey

Everyone is welcome to actively participate in OHDSI, whether you are a patient, a health professional, a researcher, or someone who simply believes in our cause. OHDSI maintains an inclusive membership model. To become an OHDSI collaborator requires no membership fee. Collaboration is as simple as raising a hand to be included in the yearly OHDSI membership count. Involvement is entirely at-will. A collaborator can have any level of contribution within the community, ranging from someone who attends weekly community calls to leading network studies or OHDSI working groups. Collaborators do not have to be data holders to be considered active members of the community. The OHDSI community aims to serve data holders, researchers, health care providers and patients & consumers alike. A record of collaborator profiles are maintained and periodically updated on the OHDSI website. Membership is fostered via OHDSI community calls, workgroups and regional chapters.

### 4.1.1 OHDSI Community Calls

OHDSI Community Calls are a weekly forum to spotlight ongoing activity within the OHDSI community. Held every Tuesday from 12-1pm ET, these teleconferences are a time for the OHDSI community to come together to share recent developments and recognize the accomplishments of individual collaborators, working groups and the community as a whole. Each week's meeting is recorded, and presentations are archived in the OHDSI website resources.

All OHDSI Collaborators are welcome to participate in this weekly teleconference and encouraged to propose topics for community discussion. OHDSI Community Calls can be a forum to share research findings, present and seek feedback for active works-in-progress, demonstrate open-source software tools under development, debate community best practices for data modeling and analytics, and brainstorm future collaborative opportunities for grants/publications/conference workshops. If you are a Collaborator with a topic for an upcoming OHDSI Collaborator meeting, you are invited to post your thoughts on the OHDSI Forums.

As a newcomer to the OHDSI community, it's highly encouraged to add this call to your calendar to get acquainted with what's happening across the OHDSI network. Newcomers are invited to introduce themselves on their first call and tell the community about themselves, their background and what brought them to OHDSI. If you'd like to join an OHDSI call, please contact Maura Beaton (beaton@ohdsi.org) for the latest dial-in details or consult the OHDSI wiki (https://www.ohdsi.org/web/wiki/doku.php?id=projects:ohdsi_community). Community call topics vary from week-to-week. Consult the OHDSI Weekly Digest on the OHDSI forum for more information on weekly presentation topics.

### 4.1.2 OHDSI Workgroups

OHDSI has a variety of ongoing projects lead by workgroup teams. Each workgroup has its own leadership team which determine the project's objectives, goals and artefacts to be contributed to the community. Workgroup participation is open to all who have an interest in contributing to the project objectives and goals. Workgroups may be long-standing, strategic objectives or short-term projects to accomplish a specific need in the community. Workgroup meeting cadence is determined by the project leadership and will vary from group to group. A list of the active workgroups is maintained on the OHDSI Wiki (https://www.ohdsi.org/web/wiki/doku.php?id=projects:overview).

```
**May update to include a graphic of the workgroups by use case**
```

### 4.1.3 OHDSI Regional Chapters

An OHDSI regional chapter represents a group of OHDSI collaborators located in a geographic area who wish to hold local networking events and meetings to address problems specific to their geographic location. Today, OHDSI regional chapters include OHDSI in Europe (https://www.ohdsi-europe.org/), OHDSI in South Korea (http://forums.ohdsi.org/c/For-collaborators-wishing-to-communicate-in-Korean) and OHDSI in China (https://ohdsichina.org/). If you would like to set-up an OHDSI regional chapter in your region,

you may do so by following the OHDSI regional chapter process outlined on the OHDSI website (https://www.ohdsi.org/who-we-are/regional-chapters).

### 4.1.4 OHDSI Research Network

Many OHDSI collaborators are interested in converting their data into the OMOP Common Data Model. The OHDSI research network represents a diverse, global community of observational databases that have undergone [[ETL]] processes to become OMOP compliant. If your journey in the OHDSI community includes transforming data, there are numerous community resources available to aid you in your journey including [[tutorials]] on the OMOP CDM and Vocabularies, freely available tools to assist with conversion [[ETL chapter reference]] and workgroups targeting specific domains or types of data conversions. OHDSI collaborators are encouraged to utilize the OHDSI forum to discuss and troubleshoot challenges that arise during CDM conversions.

## 4.2 Navigating Your Odyssey in OHDSI

As discussed in the previous section, there are many ways to begin your journey in the OHDSI community. Collaborators often find the journey from initial interest in OHDSI to actively contributing to be as circuitous as Homer's Odyssey. For those interested in running OHDSI research studies, the simplest way to navigate your path forward is to learn how to speak in "OHDSI" terms.

### 4.2.1 How to Translate Your Research Question into an OHDSI Framework

The OHDSI community has a wide range of standardized analytic tools depending on the type of question you are formulating. In the following chapters, we will discuss the intricacies of these frameworks as well as the open source tools and code available to conduct these analyses. In this section, we will briefly discuss how to take your question and reframe it in OHDSI-speak, what analytical methods and tools are appropriate for data analysis, and where you can find the resources within the OHDSI community.

#### 4.2.1.1 Step 1: Identify the proper framework

Before formulating your research question for execution on OHDSI platform, it is important to understand which OHDSI framework suits the objectives of research question, whether it is a "clinical characterization", "population-level estimation", or "patient-level prediction". Clinical characterization provides answers to "What happened to them" questions, population-level estimation responses to "What are the causal effects" question, and population-level estimation answers "What will happen to me" question. To learn more about the OHDSI use cases, please refer to Chapter 8.

Once you understand the relationship between OHDSI framework and different study types you will be using, you can then further refine your question into OHDSI-speak. THere are examples of study categories that correspond to each OHDSI framework:

- **Clinical characterization**
  - Disease natural history
  - Incidence rate
  - Prevalence
  - Treatment utilization
  - Treatment pathway
  - Quality improvement
- **Population-level effect estimation**
  - Safety surveillance
  - Effect estimation
  - Comparative effectiveness
- **Patient-level prediction**
  - Precision Medicine
  - Disease onset and progression
  - Treatment choice
  - Disease interception
  - Treatment response
  - Treatment safety
  - Treatment adherence

Many people start with a simple characterization (e.g., how many people have angioedema? How often does a patient receive ACE inhibitors?). Even if you are thinking about an estimation or prediction question, you'll probably want to start with preliminary characterization analyses to understand your target and outcome cohorts. In fact, estimation and prediction studies produce characterization results as part of their standardized outputs.

The tool below also provides a crosswalk of the type of question with a desired output you may be formulating to what OHDSI framework may be most appropriate for that question.

- If your question is:
  - How many patients…?
  - How often does…?
  - What proportion ofpatients…?
  - What is the distribution of values for lab…?
  - What are the HbA1c levels for patients with…?
  - What are the [lab values] for patients…?
  - What is the median length of exposure for patients on….?
  - What are the trends over time in…?
  - What are other drugs that these patients are using?
  - What are concomitant therapies?
  - Do we have enough cases of…?
  - Would it be feasible to study X…?
  - What are the demographics of…?
  - What are the risk factors of…? (if identifying a specific risk factor, maybe estimation, not prediction)
  - What are the predictors of…?

- And the desired output is:
  - Count or percentage
  - Averages
  - Descriptive statistics
  - Incidence rate
  - Prevalence
  - Cohort
  - Rule-based phenotype
  - Drug utilization
  - Disease natural history
  - Adherence
  - Comorbidity profile
  - Treatment pathways
  - Line of therapy
- Then you're probably asking for:
  - **Clinical characterization**
- If your question is:
  - What is the effect of…?
  - What if I do [intervention]…?
  - Which treatment works better?
  - What is the risk of X on Y?
  - What is the time-to-event of…?
- And the desired output is:
  - Relative risk
  - Hazards ratio
  - Odds ratio
  - Average treatment effect
  - Causal effect
  - Association
  - Correlation
  - Safety surveillance
  - Comparative effectiveness
- Then you're probably asking for:
  - **Population-level effect estimation**
- If your question is:
  - What is the chance that this patient will…?
  - Who are candidates for…?
- And the desired output is:
  - Probability for an individual
  - Prediction model
  - High/low risk groups
  - Probabilistic phenotype
- Then you're probably asking for:
  - **Patient-level prediction**

#### 4.2.1.2 Step 2: Frame research question in OHDSI-speak

You now know from the previous step what OHDSI framework best suits your research question. Depending on the study objectives, an OHDSI-compliant research question should be structured in a way that explicitly describes the target population, the output(s) of interest, and analysis method (if applicable).

The list below provides OHDSI-speak template and example questions for different study categories as the best practice to formulate research questions. The subsequent chapters will explain the analytical methods to run your study.

| Study Category | Template Question | Example |
|---|---|---|
| Disease onset and progression | Amongst patients who are newly diagnosed with **[insert the disease of interest]**, which patients will go on to have **[another disease or related complication]** within **[time horizon from diagnosis]**? | Among newly diagnosed **AFib** patients, which patients will go onto to have **ischemic stroke** in next **3 years**?<br>Among newly diagnosed **Melonoma**, which patients will go onto to have **brain cancer** in next **6 months**? |
| Treatment choice | Amongst patients with **[indicated disease]** who are treated with either **[treatment 1]** or **[treatment 2]**, which patients were treated with **[treatment 1] (on day 0)**? | Among **AFib** patients who took either **warfarin** or **dabigatran**, which patients got warfarin? (as defined for propensity score model) |
| Treatment response | Amongst patients who are new users of **[insert the chronically-used drug of interest]**, which patients will **[insert desired effect]** in **[time window]**? | Which patients with **T2DM** who start **metformin** stay on **metformin** after **3 years**? |
| Treatment safety | Amongst patients who are new users of **[insert the drug of interest]**, which patients will experience **[insert your favorite known adverse event from the drug profile]** within **[time horizon following exposure start]**? | Among new users of **warfarin**, which patients will have **GI bleeding** in **1 year**? |

| Study Category | Template Question | Example |
|---|---|---|
| Treatment adherence | Amongst patients who are new users of **[insert the chronically-used drug of interest]**, which patients will achieve **[adherence metric threshold]** at **[time horizon]**? | Which patients with **T2DM** who start on **metformin** will achieve **>=80% proportion of days covered** at **1 year**? |
| Comparative effectiveness | To compare the risk of **[Insert the outcome of interest]** between **[Insert the target exposure]** and **[Insert the comparator cohort]**, we will estimate the population-level effect of exposure on the **[Insert the metric of analysis model here: hazards for Cox/ odds for logistic / rate ratio for Poisson]** of the outcome during the period from **[Insert the time-at-risk start: e.g. 1 day after exposure start]** to **[Insert the time-at-risk end: e.g. 30 days after exposure end]**. | To compare the risk of **angioedema** between new users of **levetiracetam** and new users of **phenytoin**, we will estimate the population-level effect of exposure on the **hazards** of the outcome during the period from **1 day after exposure start** to **0 days after exposure end**. |

### 4.2.2 Example of a Study in OHDSI-speak

You're a researcher interested in studying the effects of ACE inhibitor monotherapy vs. thiazide diuretic monotherapy on the outcomes of acute myocardial infarction and angioedema as first-line treatment for hypertension. You understand that based on the OHDSI literature, you are asking a population-level effect estimation question but first, you need to do some homework on how to characterize this particular treatment of interest.

#### 4.2.2.1 Characterization Questions

Acute myocardial infarction is a cardiovascular complication that can occur in patients with high blood pressure, so effective treatment for hypertension should reduce the risk. Angioedema is a known side effect of ACE inhibitors, which is rare but potentially serious. You start by creating [[cohorts]] for the exposures of interest (new users of ACE inhibitors and new users of thiazide diuretics). You perform a [[characterization analysis]] to summarize baseline characteristics of these exposure populations, including demographics, comorbid conditions, and concomitant medications.

You perform another characterization analysis to estimate the incidence of selected outcomes within these exposure populations. Here, you ask 'how often does 1) acute myocardial infarction and 2) angioedema occur during the period of exposure to ACE inhibitors and thiazide diuretics?' These characterizations allow us to assess the feasibility of conducting a [[population-level effect estimation]], to evaluate whether the two treatment groups are comparable, and to identify 'risk factors' that might predict which treatment choice that patients made.

#### 4.2.2.2 Population-Level Estimation Question

The population-level effect estimation study estimates the relative risk of ACE inhibitor vs, thiazide use for the outcomes of AMI and angioedema. Here, you further evaluate through study diagnostics and negative controls whether we can produce a reliable estimate of the average treatment effect.

Independent of whether there is a causal effect of the exposures, you are also interested in trying to determine which patients are at highest risk of the outcomes. (This is a patient-level prediction problem). Here, you develop a prediction model that evaluates: amongst the patients who are new users of ACE inhibitors, which patients are at highest risk of developing acute myocardial infarction during the 1 year after starting treatment. The model allows us to predict, for a patient who has just been prescribed ACE for the first time, based on events observed from their medical history, what is the chance that they will experience AMI in the next 1 year.

### 4.2.3 More real example questions

In this section, we provide more real examples of questions as they have been submitted to the community. We have also reframed them to the OHDSI-speak format (if needed) and mapped them to OHDSI analytic frameworks:

| Unframed question we've heard from potential researchers | Reframed Question in OHDSI-speak *(italics denote additions to the original question for clarification)* | OHDSI Framework |
|---|---|---|
| Among patients addicted to opioids, what is the proportion of patients taking benzos concurrently? | Amongst opioid-addicted patients, how many patients did concurrently use benzodiazepines *any time over the last 5 years of data*? | Clinical characterization |
| Among patients newly diagnosed with cancer, how many received guideline-concordant care? | Amongst newly diagnosed cancer patients, how many patients did receive guideline-concordant care *over the last 5 years of data*? | Clinical characterization |

| Unframed question we've heard from potential researchers | Reframed Question in OHDSI-speak *(italics denote additions to the original question for clarification)* | OHDSI Framework |
|---|---|---|
| Among patients diagnosed with pneumonia, who develops ocular retinopathy within 2 years? | Amongst patients who are *newly* diagnosed with pneumonia, which patients will develop ocular retinopathy after 2 years? | Patient-level prediction |
| Among Patients with an ICD9/10 for psoriasis over the last 5 years, how many presented with major adverse cardiovascular events (e.g., heart attack, stroke, MI, atrial fibrillation)? What were the red cell distribution width (RDW) values for these patients? | Amongst diagnosed patients with psoriasis over the last 5 years *of data*, how many patients experienced major cardiovascular adverse events *any time during the study period*? Amongst diagnosed patients with psoriasis over the last 5 years of data, what were *max-min range, median, IQR, mean, and SD* of red cell distribution width (RDW) values *any time during the study period*? | Clinical characterization |
| How many patients had a shoulder arthroscopy in the last year? | How many patients did undergo arthroscopy *procedure* in the last year? | Clinical characterization |
| Among patients with neuro-degenerative Parkinson's disease, onset age 65 or older, how many subsequently suffered from brain stroke or dementia? | Amongst patients diagnosed with neurodegenerative Parkinson's disease who were 65 years or older at onset, how many patients experienced brain stroke or dementia *after diagnosis*? | Clinical characterization |
| What are the rates of chemotherapy-induced neutropenia and subsequent chemotherapy withdrawal in patients taking cisplatin? | Amongst patients who take cisplatin, what is the rate of developing chemotherapy-induced neutropenia *per 1,000 patient per year over the last 2 years of data*? | Clinical characterization |

| Unframed question we've heard from potential researchers | Reframed Question in OHDSI-speak *(italics denote additions to the original question for clarification)* | OHDSI Framework |
|---|---|---|
| Among outpatients, how many presented with ADHD were taking Ritalin for the last 6 months? | Amongst admitted patients in ambulatory setting *over the last year of data*, how many ADHD patients did use Ritalin within the last 6 months *prior to last visit*? | Clinical characterization |

## 4.3 Summary

```
**add if needed**
```

# Part II

# Uniform Data Representation

# Chapter 5

# The Common Data Model

*Chapter lead: Clair Blacketer*

No single observational data source provides a comprehensive view of the clinical data a patient accumulates while receiving healthcare, and therefore none can be sufficient to meet all expected outcome analysis needs. This explains the need for assessing and analyzing multiple data sources concurrently using a common data standard. This standard is provided by the OMOP Common Data Model (CDM). In this chapter we provide an overview of the data model itself, design, conventions, and discussion of select tables.

The CDM is designed to support the conduct of research to identify and evaluate associations between interventions (drug exposure, procedures, healthcare policy changes etc.) and outcomes caused by these interventions (condition occurrences, procedures, drug exposure etc.). Outcomes can be efficacious (benefit) or adverse (safety risk). Often times, specific patient cohorts (e.g., those taking a certain drug or suffering from a certain disease) may be defined for treatments or outcomes, using clinical events (diagnoses, observations, procedures, etc.) that occur in predefined temporal relationships to each other. The CDM, combined with its standardized content (via the Standardized Vocabularies), will ensure that research methods can be systematically applied to produce meaningfully comparable and reproducible results.

An overview of all the tables in the CDM is provided in Figure 5.1.

## 5.1 Design Principles

The CDM is designed to include all observational health data elements (experiences of the patient receiving health care) that are relevant for analysis use cases to support the generation of reliable scientific evidence about disease natural history, healthcare delivery, effects of medical interventions, the identification of demographic information, health care interventions and outcomes.

Therefore, the CDM is designed to store observational data to allow for research, under the following principles:

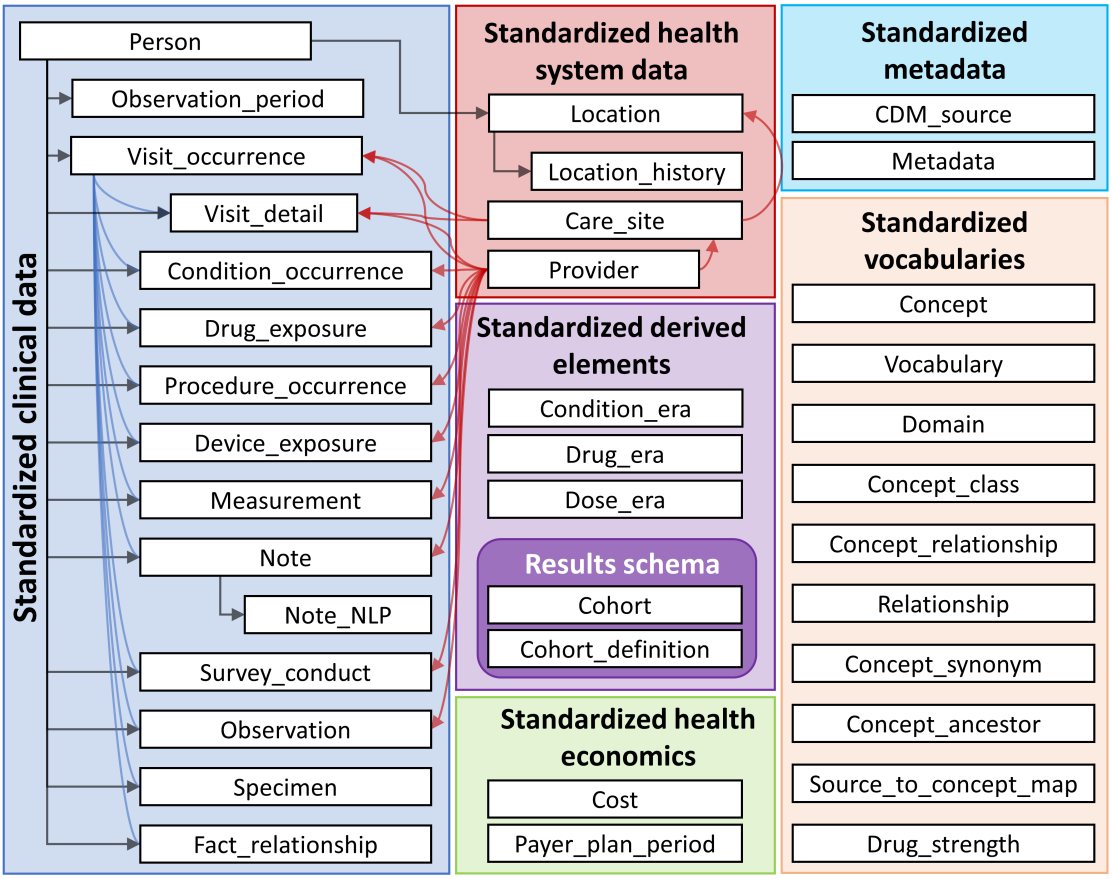



Figure 5.1: Overview of all tables in the CDM version 6.0. Note that not all relationships between tables are shown.

- **Suitability for purpose**: The CDM aims to provide data organized in a way optimal for analysis, rather than for the purpose of addressing the operational needs of health care providers or payers.
- **Data protection**: All data that might jeopardize the identity and protection of patients, such as names, precise birthdays etc. are limited. Exceptions are possible where the research expressly requires more detailed information, such as precise birth dates for the study of infants.
- **Design of domains**: The domains are modeled in a person-centric relational data model, where for each record the identity of the person and a date is captured as a minimum. Here, a relational data model is one where the data is represented as a collection of tables linked by primary and foreign keys.
- **Rationale for domains**: Domains are identified and separately defined in an entity-relationship model if they have an analysis use case (conditions, for example) and the domain has specific attributes that are not otherwise applicable. All other data can be preserved as an observation in the observation table in an entity-attribute-value structure.
- **Standardized Vocabularies**: To standardize the content of those records, the CDM relies on the Standardized Vocabularies containing all necessary and appropriate corresponding standard healthcare concepts.
- **Reuse of existing vocabularies**: If possible, these concepts are leveraged from national or industry standardization or vocabulary definition organizations or initiatives, such as the National Library of Medicine, the Department of Veterans' Affairs, the Center of Disease Control and Prevention, etc.
- **Maintaining source codes**: Even though all codes are mapped to the Standardized Vocabularies, the model also stores the original source code to ensure no information is lost.
- **Technology neutrality**: The CDM does not require a specific technology. It can be realized in any relational database, such as Oracle, SQL Server etc., or as SAS analytical datasets.
- **Scalability**: The CDM is optimized for data processing and computational analysis to accommodate data sources that vary in size, including databases with up to hundreds of millions of persons and billions of clinical observations.
- **Backwards compatibility**: All changes from previous CDMs are clearly delineated in the github repository (https://github.com/OHDSI/CommonDataModel). Older versions of the CDM can be easily created from the current version, and no information is lost that was present previously.

## 5.2 Data Model Conventions

There are a number of implicit and explicit conventions that have been adopted in the CDM. Developers of methods that run against the CDM need to understand these conventions.

### 5.2.1 General conventions of the model

The OMOP CDM is considered a "person-centric" model, meaning that the people (or patients) drive the event and observation tables. At a minimum, the tables have a foreign key into the PERSON table

and a date. This allows for a longitudinal view on all healthcare-relevant events by person. The exceptions from this rule are the standardized health system data tables, which are linked directly to events of the various domains.

### 5.2.2 General conventions of schemas

New to CDM v6.0 is the concept of schemas. This allows for more separation between read-only and writeable tables. The clinical data, event, and vocabulary tables are in the "CDM" schema and are considered read-only to the end user. This means that the tables can be queried but no information can be accidentally removed or written over except by the database administrator. Tables that need to be manipulated by web-based tools or end users have moved to the "Results" schema. Currently the only two tables in the "Results" schema are COHORT and COHORT_DEFINITON, though likely more will be added over the course of v6.0 point releases. The COHORT and COHORT_DEFINITION tables are meant to describe groups of interest that the user might define, as detailed in chapter 11. These tables can be written to, meaning that a cohort created in ATLAS or by a user can be stored in the COHORT table and accessed at a later date. This does mean that cohorts in the COHORT table can be manipulated by anyone so it is always recommended that the SQL code used to create the cohort be saved along with the project or analysis in the event it needs to be regenerated.

### 5.2.3 General conventions of data tables

The CDM is platform-independent. Data types are defined generically using ANSI SQL data types (VARCHAR, INTEGER, FLOAT, DATE, DATETIME, CLOB). Precision is provided only for VARCHAR. It reflects the minimal required string length and can be expanded within a CDM instantiation. The CDM does not prescribe the date and datetime format. Standard queries against CDM may vary for local instantiations and date/datetime configurations.

In most cases, the first field in each table ends in "_id", containing a record identifier that can be used as a foreign key in another table. For example, the CONDITION_OCCURRENCE table contains the field visit_occurrence_id which is a foreign key to the VISIT_OCCURRENCE table where visit_occurrence_id is the primary key.

*Note*: While the data model itself is platform independent, many of the tools that have been built to work with it require certain specifications. For more about this please see chapter 9

### 5.2.4 General conventions of domains

One of the ways in which the CDM is standardized is by the use of concept domains. Domains refer to the nature of a clinical entity and define the event table or field in an event table where a data record should be stored. This idea is covered fully in chapter 6 but it is important to the understanding of the data model that we touch on them here. All standard concepts in the OMOP vocabularies are organized into 30 domains, as shown in table 5.1. Domains, like concepts, are a major reason why every researcher who uses the OMOP CDM is considered to be "speaking the same language". Once

source codes from a native database are mapped to standard concepts (see section 5.2.7) the concepts themselves dictate which table they belong in by use of domains. In this way, a researcher using the CDM always knows where a record should be located rather than having to guess.

Table 5.1: Number of standard concepts belonging to each domain.

| Concept Count | Domain_Id |
|---|---|
| 1731378 | Drug |
| 477597 | Device |
| 257000 | Procedure |
| 163807 | Condition |
| 145898 | Observation |
| 89645 | Measurement |
| 33759 | Spec Anatomic Site |
| 17302 | Meas Value |
| 1799 | Specimen |
| 1215 | Provider Specialty |
| 1046 | Unit |
| 944 | Metadata |
| 538 | Revenue Code |
| 336 | Type Concept |
| 194 | Relationship |
| 183 | Route |
| 180 | Currency |
| 158 | Payer |
| 123 | Visit |
| 51 | Cost |
| 50 | Race |
| 13 | Plan Stop Reason |
| 11 | Plan |
| 6 | Episode |
| 6 | Sponsor |
| 5 | Meas Value Operator |
| 3 | Spec Disease Status |
| 2 | Gender |
| 2 | Ethnicity |
| 1 | Observation Type |

### 5.2.5 General conventions of fields

Variable names across all tables follow one convention:

Table 5.2: Field name conventions.

| Notation | Description |
|---|---|
| [entity]_id | Unique identifiers for key entities, which can serve as foreign keys to establish relationships across entities. For example, person_id uniquely identifies each individual. visit_occurrence_id uniquely identifies a PERSON encounter at a point of care. |
| [entity]_source_value | Verbatim information from the source data, typically used in ETL to map to concept_id, and not to be used by any standard analytics. For example, condition_source_value = '787.02' was the ICD-9 code captured as a diagnosis from the administrative claim. |
| [entity]_concept_id | Foreign key into the Standardized Vocabularies (i.e. the standard concept attribute for the corresponding term is true), which serves as the primary basis for all standardized analytics. For example, condition_concept_id = 31967 contains the reference value for the SNOMED concept of "Nausea". |
| [entity]_source_concept_id | Foreign key into the Standardized Vocabularies representing the concept and terminology used in the source data, when applicable. For example, condition_source_concept_id = 45431665 denotes the concept of "Nausea" in the Read terminology; the analogous condition_concept_id is 31967, since SNOMED-CT is the Standardized Vocabulary for most clinical diagnoses and findings. |
| [entity]_type_concept_id | Delineates the origin of the source information, standardized within the Standardized Vocabularies. For example, drug_type_concept_id can allow analysts to discriminate between "Pharmacy dispensing" and "Prescription written" |

### 5.2.6 Representation of content through Concepts

In CDM data tables the content of each record is represented using Concepts. Concepts are stored in event tables with their concept IDs as foreign keys to the CONCEPT table, which contains concepts necessary to describe the healthcare experience of a patient. If a Standard Concept does not exist or cannot be identified, the concept ID 0 is used, representing a non-existing concept or un-mappable source value.

Records in the CONCEPT table contain detailed information about each concept (name, domain, class etc.). Concepts, Concept Relationships, Concept Ancestors and other information relating to Concepts is contained in the tables of the Standardized Vocabularies.

### 5.2.7 Difference between Concept IDs and Source Values

Many tables contain equivalent information in multiple places: As a Source Value, a Source Concept and as a Standard Concept.

- **Source Values** contain the codes from public code systems such as ICD-9-CM, NDC, CPT-4, READ etc. or locally controlled vocabularies (such as F for female and M for male) copied from the source data. Source Values are stored in the [entity]_source_value fields in the data tables.
- **Concepts** are CDM-specific entities that represent the meaning of a clinical fact. Most concepts are based on code systems used in healthcare (called Source Concepts), while others were created de-novo (concept_code = "OMOP generated"). Concepts have unique IDs across all domains.
- **Source Concepts** are the concepts that represent the code used in the source. Source Concepts are only used for common healthcare code systems, not for OMOP-generated Concepts. Source Concepts are stored in the [entity]_source_concept_id field in the data tables.
- **Standard Concepts** are those concepts that are used to define the unique meaning of a clinical entity. For each entity there is one Standard Concept. Standard Concepts are typically drawn from existing public vocabulary sources. Concepts that have the equivalent meaning to a Standard Concept are mapped to the Standard Concept. Standard Concepts are referred to in the [entity]_concept_id field of the data tables.

Source Values are only provided for convenience and quality assurance (QA) purposes. Source Values and Source Concepts are optional, while **Standard Concepts are mandatory**. Source Values may contain information that is only meaningful in the context of a specific data source. This mandatory use of Standard Concepts is what allows all OHDSI collaborators to speak the same language. For example, let's look at the condition "Pulmonary Tuberculosis" (TB). Figure 5.2 shows that the ICD9CM code for TB is 011.

Without the use of a standard way to represent TB the code 011 could be interpreted as "Hospital Inpatient (Including Medicare Part A)" in the UB04 vocabulary, or as "Nervous System Neoplasms without Complications, Comorbidities" in the DRG vocabulary. This is where Concept IDs, both Source and Standard, are valuable. The Concept ID that represents the 011 ICD9CM code is 44828631. This differentiates the ICD9CM from the UBO4 and from the DRG. The Standard Concept that ICD9CM code maps to is 253954 as shown in figure 5.3 by the relationship "Non-standard to Standard map (OMOP)". This same mapping relationship exists between Read, ICD10, CIEL, and MeSH codes, among others, so that any research that references the standard SNOMED concept is sure to include all supported source codes.

An example of how the standard concept-source code relationship is depicted in the tables is shown in Table 5.7.

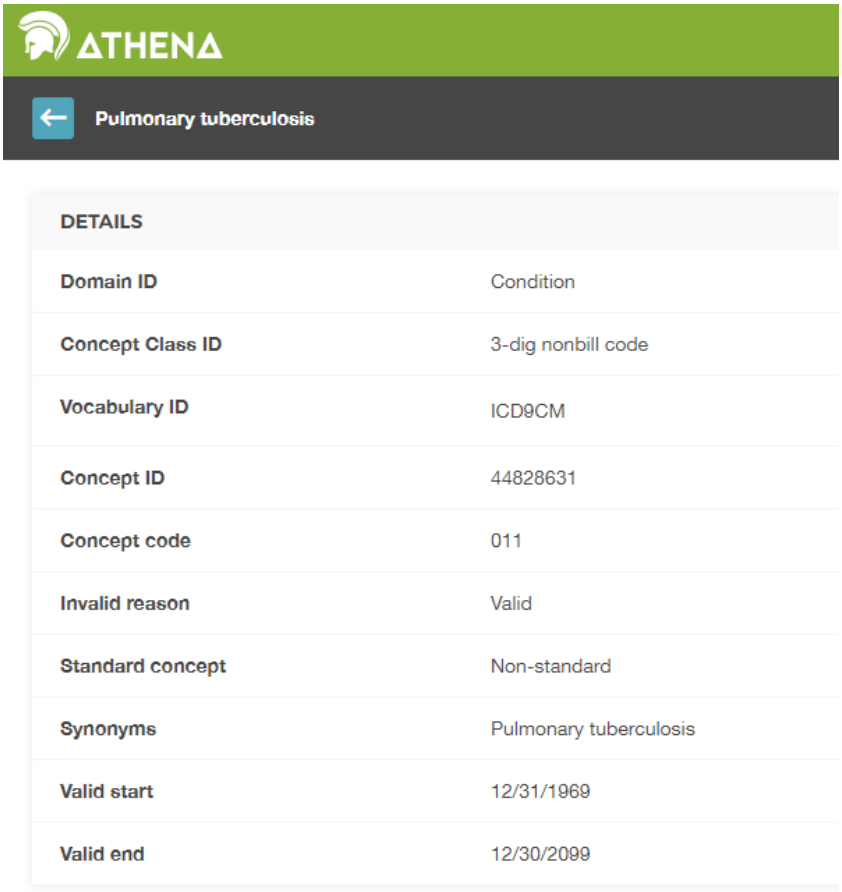



Figure 5.2: ICD9CM code for Pulmonary Tuberculosis

TERM CONNECTIONS (82)

| RELATIONSHIP | RELATES TO | CONCEPT ID | VOCABULARY |
|---|---|---|---|
| ICD-9-CM to MedDRA (MSSO) | Pulmonary tuberculosis | 36110777 | MedDRA |
| Non-standard to Standard map (OMOP) | Pulmonary tuberculosis | 253954 | SNOMED |
| Subsumes | Other specified pulmonary tuberculosis | 44830894 | ICD9CM |
| | Other specified pulmonary tuberculosis, bacteriological or histological examination not done | 44836741 | ICD9CM |
| | Other specified pulmonary tuberculosis, bacteriological or histological examination unknown (at present) | 44836742 | ICD9CM |
| | Other specified pulmonary tuberculosis, tubercle bacilli found (in sputum) by microscopy | 44821641 | ICD9CM |
| | Other specified pulmonary tuberculosis, tubercle bacilli not found (in sputum) by microscopy, but found by bacterial culture | 44833188 | ICD9CM |

Figure 5.3: SNOMED code for Pulmonary Tuberculosis

## 5.3 OMOP CDM Standardized Tables

The OMOP CDM contains 16 Clinical data tables, 10 Vocabulary tables, 2 Metadata tables, 4 Health System data tables, 2 Health Economics data tables, 3 standardized derived elements, and 2 results schema tables. These tables are fully specified in the CDM Wiki[1].

To illustrate how these tables are used in practice the data of one person will be used as a common thread throughout the rest of the chapter. While part of the CDM the Vocabulary tables are not covered here, rather, they are detailed in depth in Chapter 6.

### 5.3.1 Running Example: Endometriosis

Endometriosis is a painful condition whereby cells normally found in the lining of a woman's uterus occur elsewhere in the body. Severe cases can lead to infertility, bowel, and bladder problems. The following sections will detail one patient's experience with this disease and how her clinical experience might be represented in the Common Data Model.

[1] https://github.com/OHDSI/CommonDataModel/wiki

Every step of this painfull journey I had to convince everyone how much pain I was in.

Lauren had been experiencing endometriosis symptoms for many year; however, it took a ruptured cyst in her ovary before she was diagnosed. You can read more about Lauren at https://www.endometriosis-uk.org/laurens-story.

### 5.3.2 PERSON table

As the Common Data Model is a person-centric model (see section 5.2.1) let's start with how she would be represented in the PERSON table.

**What do we know about Lauren?**

- She is a 36-year-old woman
- Her birthday is 12-March-1982
- She is white
- She is english

With that in mind, her PERSON table might look something like this:

Table 5.3: The PERSON table.

| Column name | Value | Explanation |
|---|---|---|
| person_id | 1 | person_id should be an integer, either directly from the source or generated as part of the build process. |
| gender_concept_id | 8532 | The concept ID referring to female gender is 8532. |
| year_of_birth | 1982 | |
| month_of_birth | 3 | |
| day_of_birth | 12 | |
| birth_datetime | 1982-03-12 00:00:00 | When the time is not known midnight is used. |
| death_datetime | | |
| race_concept_id | 8527 | The concept ID referring to white race is 8527. |

| Column name | Value | Explanation |
|---|---|---|
| ethnicity_concept_ id | 38003564 | Typically hispanic status is stored for ethnicity. The concept ID 38003564 refers to "Not hispanic". |
| location_id | | Her address is not known. |
| provider_id | | Her primary care provider is not known. |
| care_site_id | | Her primary care site is not known. |
| person_source_value | 1 | Typically this would be her identifier in the source data, though often is it the same as the person_id. |
| gender_source_value | F | The gender value as it appears in the source is stored here. |
| gender_source_ concept_id | 0 | If the gender value in the source was coded using a vocabulary recognized by OHDSI, that concept ID would go here. For example, if her gender was "Sex-F" in the source and it was stated to be in the PCORNet vocabulary concept ID 44814665 would go in this field. |
| race_source_value | white | The race value as it appears in the source is stored here. |
| race_source_concept_ id | 0 | Same principle as gender_source_concept_id. |
| ethnicity_source_ value | english | The ethnicity value as it appears in the source is stored here. |
| ethnicity_source_ concept_id | 0 | Same principle as gender_source_concept_id. |

### 5.3.3 OBSERVATION_PERIOD table

The OBSERVATION_PERIOD table is designed to define the amount of time for which a patient's clinical events are recorded in the source system. For US healthcare insurance claims this is typically the enrollment period of the patient. When working with data from electronic health records (EHR) often the first record in the system is considered the observation_period_start_date and the latest record is considered the observation_period_end_date with the understanding that only the clinical events that happened within that particular system were recorded.

**How can we determine Lauren's observation period?**

Lauren's information as shown in Table 5.4 is most similar to EHR data in that we only have records of her encounters from which to determine her observation period.

Table 5.4: Lauren's healthcare encounters.

| Encounter ID | Start date | Stop date | Type |
|---|---|---|---|
| 70 | 2010-01-06 | 2010-01-06 | outpatient |

| Encounter ID | Start date | Stop date | Type |
|---|---|---|---|
| 80 | 2011-01-06 | 2011-01-06 | outpatient |
| 90 | 2012-01-06 | 2012-01-06 | outpatient |
| 100 | 2013-01-07 | 2013-01-07 | outpatient |
| 101 | 2013-01-14 | 2013-01-14 | ambulatory |
| 102 | 2013-01-17 | 2013-01-24 | inpatient |

Based on the encounter records her OBSERVATION_PERIOD table might look something like this:

Table 5.5: The OBSERVATION_PERIOD table.

| Column name | Value | Explanation |
|---|---|---|
| observation_period_ id | 1 | This is typically an autogenerated field that creates a unique ID number for each record in the table. |
| person_id | 1 | This comes from the PERSON table and links PERSON and OBSERVATION_PERIOD. |
| observation_period_ start_date | 2010-01-06 | This is the start date of her earliest encounter on record. |
| observation_period_ end_date | 2013-01-24 | This is the end date of her latest encounter on record. |
| period_type_ concept_id | 44814725 | The best option in the Vocabulary with the concept class “Obs Period Type” is 44814724, which stands for “Period covering healthcare encounters”. |

### 5.3.4 VISIT_OCCURRENCE

The VISIT_OCCURRENCE table houses information about a patient’s encounters with the health care system. Within the OHDSI vernacular these are referred to as visits and are considered to be discreet events. There are 12 categories of visits though the most common are inpatient, outpatient, emergency and long term care.

**How do we represent Lauren’s encounters as visits?**

As an example let’s represent the inpatient encounter in Table 5.4 as a record in the VISIT_OCCURRENCE table.

Table 5.6: The VISIT_OCCURRENCE table.

| Column name | Value | Explanation |
|---|---|---|
| visit_occurrence_id | 514 | This is typically an autogenerated field that creates a unique ID number for each visit on the person's record in the converted CDM database. |
| person_id | 1 | This comes from the PERSON table and links PERSON and VISIT_OCCURRENCE. |
| visit_concept_id | 9201 | The concept ID referring to an inpatient visit is 9201. |
| visit_start_date | 2013-01-17 | The start date of the visit. |
| visit_start_datetime | 2013-01-17 00:00:00 | The date and time of the visit started. When time is unknown midnight is used. |
| visit_end_date | 2013-01-24 | The end date of the visit. If this is a one-day visit the end date should match the start date. |
| visit_end_datetime | 2013-01-24 00:00:00 | The date and time of the visit end. If time is unknown midnight is used. |
| visit_type_concept_ id | 32034 | This column is intended to provide information about the provenance of the visit record, i.e. does it come from an insurance claim, hospital billing record, EHR record, etc. For this example the concept ID 32035 ("Visit derived from EHR encounter record") is used as the encounters are similar to electronic health records |
| provider_id* | NULL | If the encounter record has a provider associated, the ID for that provider goes in this field. This should be the provider_id from the PROVIDER table that represents the provider on the encounter. |
| care_site_id | NULL | If the encounter record has a care site associated, the ID for that care site goes in this field. This should be the care_site_id from the CARE_SITE table that codes for the care site on the encounter. |
| visit_source_value | inpatient | The visit value as it appears in the source goes here. In this context "visit" means outpatient, inpatient, emergency, etc. |
| visit_source_ concept_id | 0 | If the visit value from the source is coded using a vocabulary that is recognized by OHDSI, the concept ID that represents the visit source value would go here. |

| Column name | Value | Explanation |
|---|---|---|
| admitted_from_ concept_id | 0 | If known, this is the concept ID that represents where the patient was admitted from. This concept should have the concept class "Place of Service" and the domain "Visit". For example, if a patient was admitted to the hospital from home, the concept ID would be 8536 ("Home"). |
| admitted_from_ source_ value | NULL | This is the value from the source that represents where the patient was admitted from. Using the above example, this would be "home". |
| discharge_to_ concept_id | 0 | If known, this is the concept ID that represents where the patient was discharged to. This concept should have the concept class "Place of Service" and the domain "Visit". For example, if a patient was released to an assisted living facility, the concept ID would be 8615 ("Assisted Living Facility"). |
| discharge_to_source_ value | 0 | This is the value from the source that represents where the patient was discharged to. Using the above example, this would be "assisted living facility". |
| preceding_visit_ occurrence_id | NULL | The visit_occurrence_id for the visit immediately preceding the current one in time for the patient. |

- A patient may interact with multiple health care providers during one visit, as is often the case with inpatient stays. These interactions can be recorded in the VISIT_DETAIL table. While not covered in depth in this chapter, you can read more about the VISIT_DETAIL table in the CDM wiki.

### 5.3.5 CONDITION_OCCURRENCE

Records in the CONDITION_OCCURRENCE table are diagnoses, signs, or symptoms of a condition either observed by a Provider or reported by the patient.

**What are Lauren's conditions?**

Revisiting her account she says:

> About 3 years ago I noticed my periods, which had also been painful, were getting increasingly more painful. I started becoming aware of a sharp jabbing pain right by my colon and feeling tender and bloated around my tailbone and lower pelvis area. My periods had become so painful that I was missing 1-2 days of work a month. Painkillers sometimes dulled the pain, but usually they didn't do much.

The SNOMED code for painful menstruation cramps, otherwise known as dysmenorrhea, is 266599000. Table 5.7 shows how that would be represented in the CONDITION_OCCURRENCE table:

Table 5.7: The CONDITION_OCCURRENCE table.

| Column name | Value | Explanation |
|---|---|---|
| condition_ occurrence_id | 964 | This is typically an autogenerated field that creates a unique ID number for each condition on the person's record in the converted CDM database. |
| person_id | 1 | This comes from the PERSON table and links PERSON and CONDITION_OCCURRENCE. |
| condition_concept_id | 194696 | The concept ID that represents the SNOMED code 266599000 is 194696. |
| condition_start_date | 2010-01-06 | The date when the instance of the Condition is recorded. |
| condition_start_ datetime | 2010-01-06 00:00:00 | The date and time when the instance of the Condition is recorded. Midnight is used when the time is unknown |
| condition_end_date | NULL | If known, this is the date when the instance of the Condition is considered to have ended. |
| condition_end_ datetime | NULL | If known, this is the date and time when the instance of the Condition is considered to have ended. |
| condition_type_ concept_id | 32020 | This column is intended to provide information about the provenance of the condition, i.e. does it come from an insurance claim, hospital billing record, EHR record, etc. For this example the concept ID 32020 ("EHR encounter diagnosis") is used as the encounters are similar to electronic health records. Concept IDs in this field should be in the "Condition Type" vocabulary. |
| condition_status_ concept_id | 0 | If known, the this represents when and/or how the condition was diagnosed. For example, a condition could be an admitting diagnosis, in which case the concept ID 4203942 would be used. |
| stop_reason | NULL | If known, the reason that the Condition was no longer present, as indicated in the source data. |
| provider_id | NULL | If the condition record has a diagnosing provider listed, the ID for that provider goes in this field. This should be the provider_id from the PROVIDER table that represents the provider on the encounter. |

| Column name | Value | Explanation |
|---|---|---|
| visit_occurrence_id | 509 | If known, this is the visit (represented as visit_occurrence_id taken from the VISIT_OCCURRENCE table) during which the condition was diagnosed. |
| visit_detail_id | NULL | If known, this is the visit detail encounter (represented as VISIT_DETAIL_ID from the VISIT_DETAIL table) during which the condition was diagnosed. |
| condition_source_ value | 266599000 | This is the value from the source that represents the condition. In Lauren's case of dysmenorrhea the SNOMED code for that condition is stored here and the standard concept ID mapped from that code is stored in condition_concept_id. |
| condition_source_ concept_id | 194696 | If the condition value from the source is coded using a vocabulary that is recognized by OHDSI, the concept ID that represents that value would go here. In the example of dysmennorhea the source value is a SNOMED code so the concept ID that represents that code is 194696. In this case it is the same as the condition_concept_id since the SNOMED vocabulary is the standard condition vocabulary. |
| condition_status_ source_ value | 0 | If the condition status value from the source is coded using a vocabulary that is recognized by OHDSI, the concept ID that represents that source value would go here. |

### 5.3.6 DRUG_EXPOSURE

The DRUG_EXPOSURE table captures records about the utilization of a drug when ingested or otherwise introduced into the body. Drugs include prescription and over-the-counter medicines, vaccines, and large-molecule biologic therapies. Radiological devices ingested or applied locally do not count as Drugs.

Drug exposures are inferred from clinical events associated with orders, prescriptions written, pharmacy dispensings, procedural administrations, and other patient-reported information.

**What are Lauren's drug exposures?**

We know that Lauren was given 60 acetaminophen 325mg oral tablets for 30 days (NDC code 69842087651) at her visit on 2010-01-06 to help with her dysmenorrhea pain. Here's how that might look in the DRUG_EXPOSURE table:

Table 5.8: The DRUG_EXPOSURE table.

| Column name | Value | Explanation |
|---|---|---|
| drug_exposure_id | 1001 | This is typically an autogenerated field that creates a unique ID number for each drug exposure on the person's record in the converted CDM database. |
| person_id | 1 | This comes from the PERSON table and links PERSON and DRUG_EXPOSURE. |
| drug_concept_id | 1127433 | The NDC code for acetaminophen maps to the RxNorm code 313782 which is represented by the concept ID 1127433. |
| drug_exposure_start_ date | 2010-01-06 | The start date of the drug exposure |
| drug_exposure_start_ datetime | 2010-01-06 00:00:00 | The start date and time of the drug exposure. Midnight is used when the time is not known. |
| drug_exposure_end_ date | 2010-02-05 | The end date of the drug exposure. Depending on different sources, it could be a known or an inferred date and denotes the last day at which the patient was still exposed to the drug. In this case the end is inferred since we know Lauren had a 30 days supply. |
| drug_exposure_end_ datetime | 2010-02-05 00:00:00 | The end date and time of the drug exposure. Similar rules apply as to drug_exposure_end_date. Midnight is used when time is unknown |
| verbatim_end_date | NULL | If the source provides an end date rather than just days supply that date goes here. |
| drug_type_concept_id | 38000177 | This column is intended to provide information about the provenance of the drug, i.e. does it come from an insurance claim, prescription record, etc. For this example the concept ID 38000177 ("Prescription written") is used as the drug record is from a written prescription. Concept IDs in this field should be in the "Drug Type" vocabulary. |
| stop_reason | NULL | The reason the drug was stopped. Reasons include regimen completed, changed, removed, etc. |
| refills | NULL | The number of refills after the initial prescription. The initial prescription is not counted, values start with null. In the case of Lauren's acetaminophen she did not have any refills so the value is NULL. |

| Column name | Value | Explanation |
|---|---|---|
| quantity | 60 | The quantity of drug as recorded in the original prescription or dispensing record. |
| days_supply | 30 | The number of days of supply of the medication as prescribed. |
| sig | NULL | The directions (‘signetur’) on the Drug prescription as recorded in the original prescription (and printed on the container) or dispensing record. |
| route_concept_id | 4132161 | This concept is meant to represent the route of the drug the patient was was exposed to. Lauren took her acetaminophen orally so the concept ID 4132161 (“Oral”) is used. |
| lot_number | NULL | An identifier assigned to a particular quantity or lot of drug product from the manufacturer. |
| provider_id | NULL | If the drug record has a prescribing provider listed, the ID for that provider goes in this field. This should be the PROVIDER_ID from the PROVIDER table that represents the provider on the encounter. |
| visit_occurrence_id | 509 | If known, this is the visit (represented as visit_occurrence_id taken from the VISIT_OCCURRENCE table) during which the drug was prescribed. |
| visit_detail_id | NULL | If known, this is the visit detail (represented as visit_detail_id taken from the VISIT_DETAIL table) during which the drug was prescribed. |
| drug_source_value | 69842087651 | This is the source code for the drug as it appears in the source data. In Lauren’s case she was prescribed acetaminophen and the NDC code is stored here. |
| drug_source_concept_ id | 750264 | This is the concept ID that represents the drug source value. In this example the concept ID is 750264, the NDC code for “Acetaminophen 325 MG Oral Tablet”. |
| route_source_value | NULL | The information about the route of administration as detailed in the source. |
| dose_unit_source_ value | NULL | The information about the dose unit as detailed in the source. |

### 5.3.7 PROCEDURE_OCCURRENCE

The PROCEDURE_OCCURRENCE table contains records of activities or processes ordered by, or carried out by, a healthcare provider on the patient to have a diagnostic or therapeutic purpose. Procedures are present in various data sources in different forms with varying levels of standardization. For example:

- Medical Claims include procedure codes that are submitted as part of a claim for health services rendered, including procedures performed.
- Electronic Health Records that capture procedures as orders.

**What procedures did Lauren have?** From her description we know she had a ultrasound of her left ovary on 2013-01-14 that showed a 4x5cm cyst. Here's how that would look in the PROCEDURE_OCCURRENCE table:

Table 5.9: The PROCEDURE_OCCURRENCE table.

| Column name | Value | Explanation |
|---|---|---|
| procedure_ occurrence_id | 1277 | This is typically an autogenerated field that creates a unique ID number for each procedure occurrence on the person's record in the converted CDM database. |
| person_id | 1 | This comes from the PERSON table and links PERSON and PROCEDURE_OCCURRENCE |
| procedure_concept_id | 4127451 | The SNOMED procedure code for a pelvic ultrasound is 304435002 which is represented by the concept ID 4127451. |
| procedure_date | 2013-01-14 | The date on which the procedure was performed. |
| procedure_datetime | 2013-01-14 00:00:00 | The date and time on which the procedure was performed. Midnight is used when time is unknown. |
| procedure_type_ concept_id | 38000275 | This column is intended to provide information about the provenance of the procedure, i.e. does it come from an insurance claim, EHR order, etc. For this example the concept ID 38000275 ("EHR order list entry") is used as the procedure record is from an EHR record. Concept IDs in this field should be in the "Procedure Type" vocabulary. |
| modifier_concept_id | 0 | This is meant for a concept ID representing the modifier on the procedure. For example, if the record indicated that a CPT4 procedure was performed bilaterally then the concept ID 42739579 ("Bilateral procedure") would be used. |

| Column name | Value | Explanation |
|---|---|---|
| quantity | 0 | The quantity of procedures ordered or administered. |
| provider_id | NULL | If the procedure record has a provider listed, the ID for that provider goes in this field. This should be the provider_id from the PROVIDER table that represents the provider on the encounter. |
| visit_occurrence_id | 740 | If known, this is the visit (represented as visit_occurrence_id taken from the VISIT_OCCURRENCE table) during which the procedure was performed. |
| visit_detail_id | NULL | If known, this is the visit detail (represented as visit_detail_id taken from the VISIT_DETAIL table) during which the procedure was performed. |
| procedure_source_ value | 304435002 | The source code for the procedure as it appears in the source data. This code is mapped to a standard procedure Concept in the Standardized Vocabularies and the original code is, stored here for reference. |
| procedure_source_ concept_id | 4127451 | This is the concept ID that represents the procedure source value. |
| modifier_source_ value | NULL | The source code for the modifier as it appears in the source data. |

## 5.4 Additional Information

This chapter covers only a portion of the tables available in the OMOP CDM as examples of how data is represented. You are encouraged to visit the wiki site https://github.com/OHDSI/CommonDataModel/wiki for more information.

## 5.5 Summary

- The OMOP CDM is designed to support the conduct of research to identify and evaluate associations between interventions and outcomes
- The OMOP CDM is a “person-centric” model
- Source codes are represented as standard concept ids

## 5.6 Exercises

TODO

# Chapter 6

# Standardized Vocabularies

The OMOP Standardized Vocabulary: Christian's (almost) finished paper + http://www.ohdsi.org/web/wiki/doku.php?id=documentation:vocabulary

# Chapter 7

# Extract Transform Load

*Chapter lead: Clair Blacketer*

## 7.1 Introduction

In order to get from the native/raw data to the OMOP CDM an extract, transform, and load (ETL) process has been designed and developed. This process consists of four major steps:

1. Data experts and CDM experts together design the ETL
2. People with medical knowledge create the code mappings
3. A technical person implements the ETL
4. All are involved in quality control

There are tools available that have been developed by the community and this chapter will cover the steps in the process and the three tools available to facilitate those steps.

**Chapter Objectives:**

- Examine best practices around designing an ETL specification
- Introduce community tools available for facilitating the ETL process
- Discuss CDM and ETL maintenance

## 7.2 ETL Step 1 - Data experts and CDM experts together design the ETL

### 7.2.1 White Rabbit

**Description**

To initiate an ETL process on a database you need to understand your data, including the tables, fields, and content. This is where the White Rabbit tool comes in; it scans your data and creates a report containing all the information necessary to begin writing ETL logic.

**Scope and Purpose**

WhiteRabbit's main function is to perform a scan of the source data, providing detailed information on the tables, fields, and values that appear in a field. The source data can be in comma-separated text files, or in a database (MySQL, SQL Server, Oracle, PostgreSQL, Microsoft APS, Microsoft Access, Amazon RedShift). The resulting scan will generate a report that can be used as a reference when designing the ETL, for instance by using it in conjunction with the Rabbit-In-a-Hat tool. White Rabbit differs from standard data profiling tools in that it attempts to prevent the display of personally identifiable information (PII) data values in the generated output data file.

**Process Overview**

The typical sequence for using the software to scan source data in preparation of developing an ETL into an OMOP CDM:

1. Set working folder, the location on the local desktop computer where results will be exported.
2. Connect to the source database or CSV text file and test connection.
3. Select the tables of interest for the scan and scan the tables.
4. WhiteRabbit creates an export of information about the source data.

**Installation and support**

All source code and installation instructions available on GitHub: https://github.com/OHDSI/WhiteRabbit

Additional information available on the OHDSI wiki: http://www.ohdsi.org/web/wiki/doku.php?id=documentation:software:whiterabbit

**Setting a Working Folder**

After downloading and installing the WhiteRabbit application, the first thing you need to do is set a working folder. Any files that WhiteRabbit creates will be exported to this local folder. Use the "Pick Folder" button to navigate in your local environment where you would like the scan document to go.

**Connection to a Database**

WhiteRabbit supports delimited text files, Oracle, Sql Server, MySQL, and PostgreSQL. More detailed information for how to connect can be found on the wiki.

**Scanning the Tables in a Database**

After connecting to a database, you can scan the tables contained therein. A scan generates a report containing information on the source data that can be used to help design the ETL. Using the Scan tab in WhiteRabbit you can either select individual tables in the selected source database by clicking on 'Add' (Ctrl + mouse click), or automatically select all tables in the database by clicking on 'Add all in DB'.

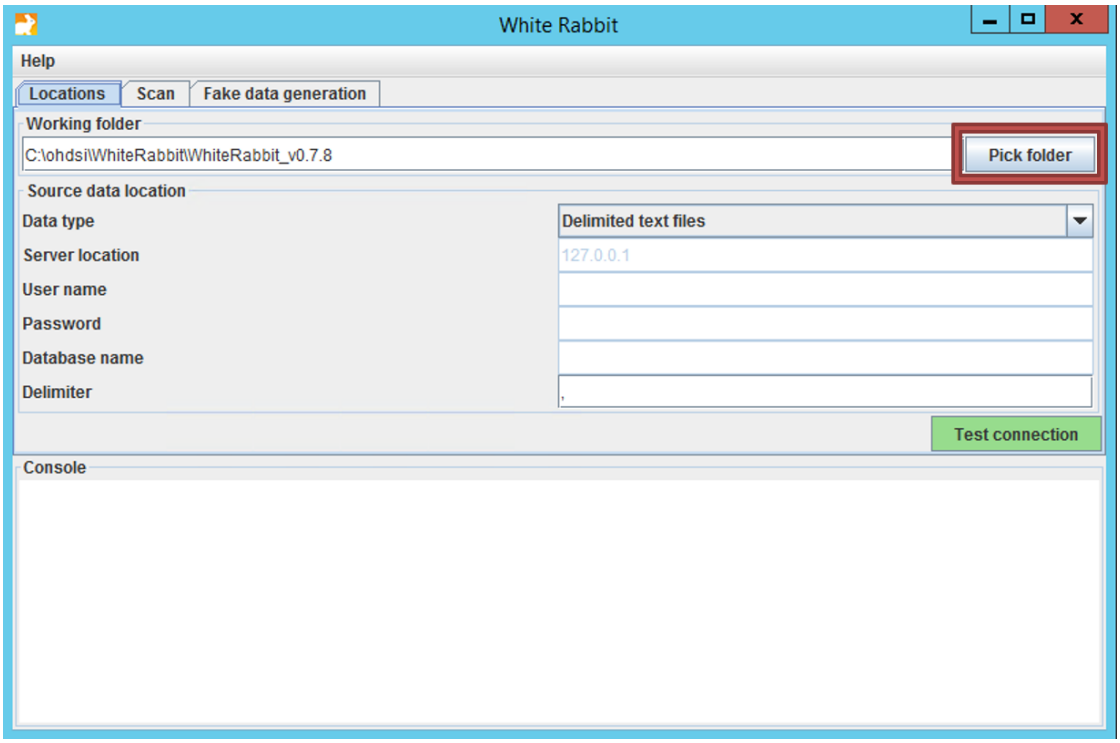
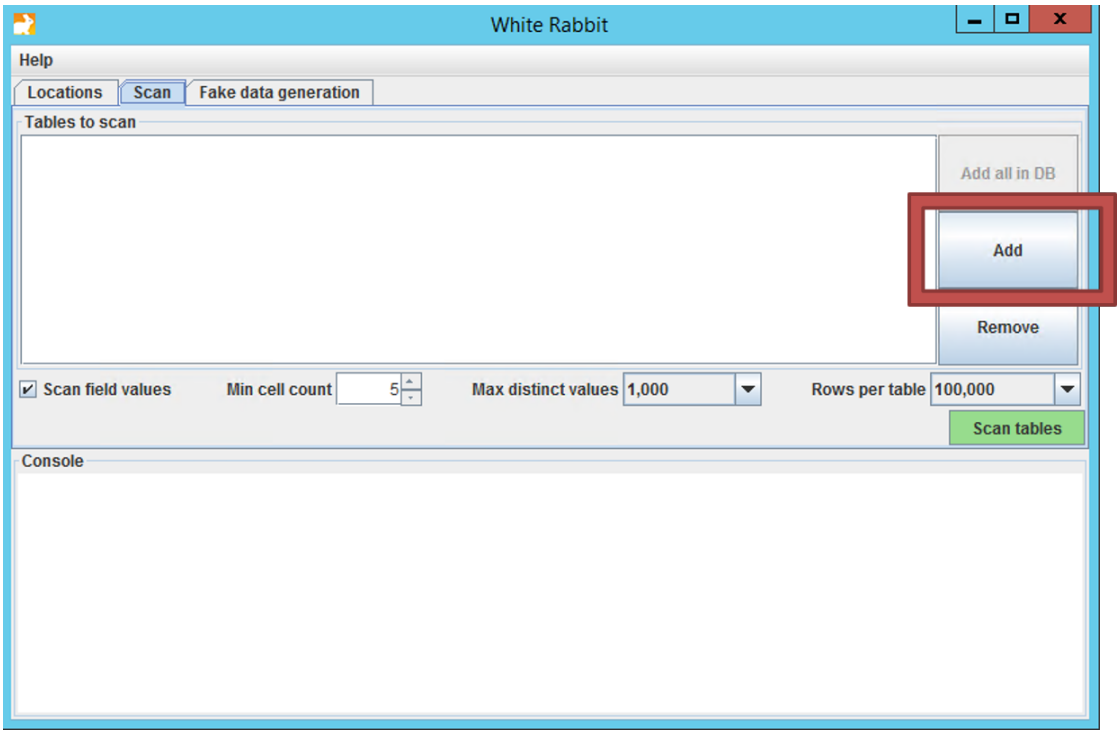



Figure 7.1: The ”Pick Folder” button allows the specification of a working folder for the WhiteRabbit application




There are a few setting options as well with the scan:

- Checking the 'Scan field values' tells WhiteRabbit that you would like to investigate raw data items within tables selected for a scan (i.e. if you select Table A, WhiteRabbit will review the contents in each column in Table A).
- 'Min cell count' is an option when scanning field values. By default this is set to 5, meaning values in the source data that appear less than 5 times will not appear in the report.
- 'Rows per table' is an option when scanning field values. By default, WhiteRabbit will random 1 million rows in the table. There are other options to review 100,000 or all rows within the table.
- Unchecking the 'Scan field values' tells WhiteRabbit to not review or report on any of the raw data items.
- Once all settings are completed, press the "Scan tables" button. After the scan is completed the report will be written to the working folder.

**Interpreting the Scan Report**

Once the scan is complete, an excel file is generated in the selected folder with one tab present for each table scanned as well as an overview tab. The overview tab lists all tables scanned, each field in each table, the data type of each field, the maximum length of the field, the number of rows in the table, the number of rows scanned, and how often each field was found to be empty.

The tabs for each of the tables, for example the conditions table in the raw_synthea database, show each field, the values in each field, and the frequency of each value.

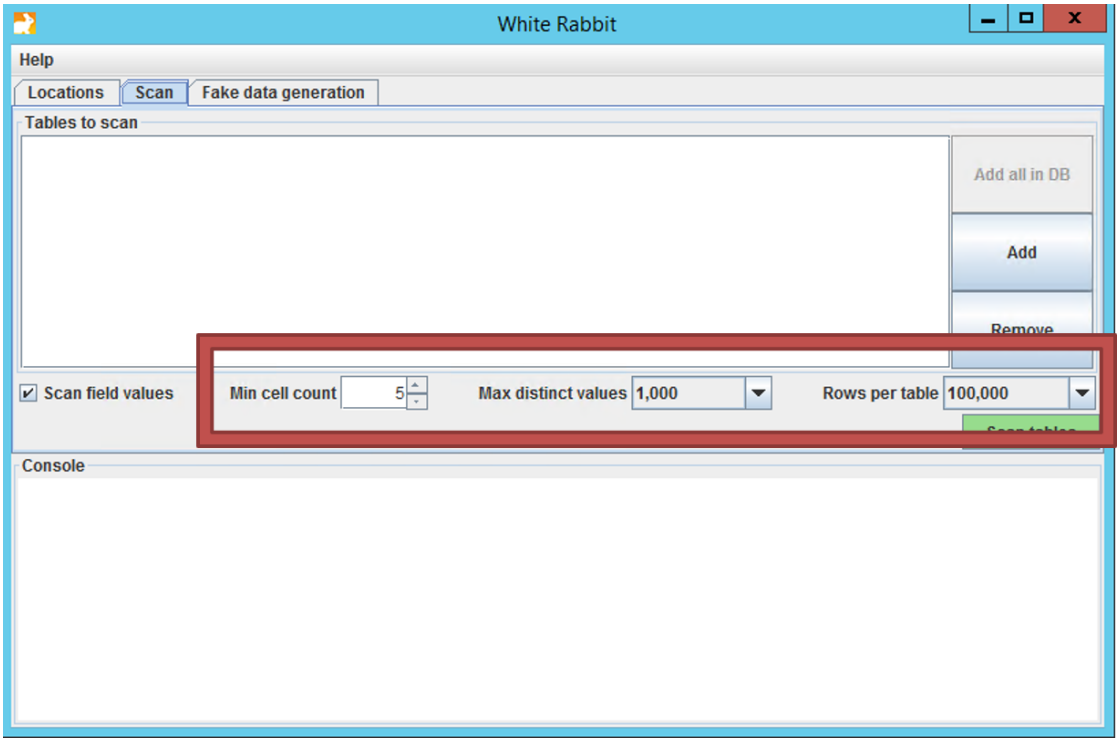



Figure 7.2: WhiteRabbit scan settings

### 7.2.2 Rabbit-In-a-Hat

Rabbit-In-a-Hat comes with WhiteRabbit and is designed to read and display a WhiteRabbit scan document. WhiteRabbit generates information about the source data while Rabbit-In-a-Hat uses that information and through a graphical user interface to allow a user to connect source data to tables and columns within the CDM. Rabbit-In-a-Hat generates documentation for the ETL process it does not generate code to create an ETL.

Similar to WhiteRabbit, installation information can be found on the github[1] and information about the different options available in the application can be found on the wiki[2].

**Writing ETL Logic**

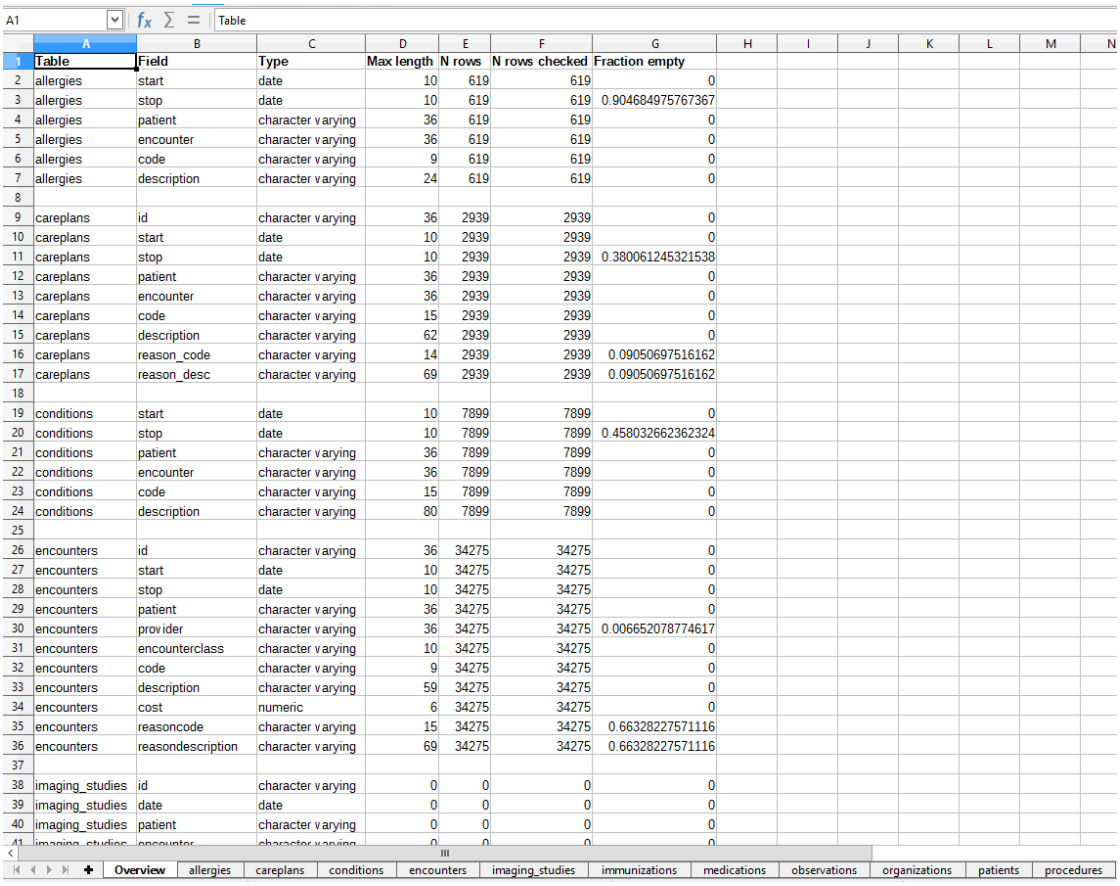
Once you have opened your WhiteRabbit scan report in Rabbit-In-a-Hat you are ready to begin designing and writing the logic for how to convert the source data to the OMOP CDM. As an example, the next few sections will depict how some of the tables in the Synthea$^{TM}$[3] database might look during conversion.

**General Flow of an ETL**

[1] https://github.com/OHDSI/WhiteRabbit

[2] http://www.ohdsi.org/web/wiki/doku.php?id=documentation:software:whiterabbit

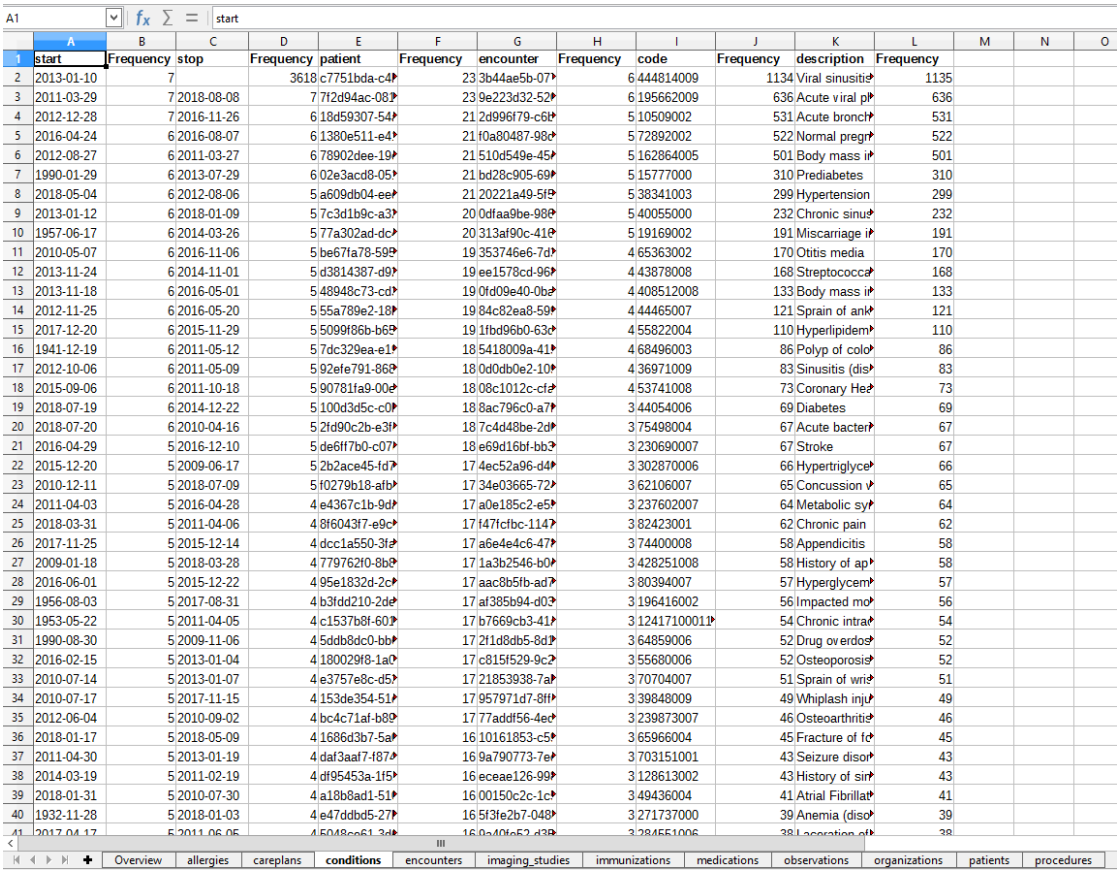
[3] Synthea$^{T}$M is a patient generator that aims to model real patients. Data are created based on parameters passed to the application.The structure of the data can be found here: https://github.com/syntheticahealth/synthea/wiki.



| Table | Field | Type | Max length | N rows | N rows checked | Fraction empty |
|---|---|---|---|---|---|---|
| allergies | start | date | 10 | 619 | 619 | 0 |
| allergies | stop | date | 10 | 619 | 619 | 0.904684975767367 |
| allergies | patient | character varying | 36 | 619 | 619 | 0 |
| allergies | encounter | character varying | 36 | 619 | 619 | 0 |
| allergies | code | character varying | 9 | 619 | 619 | 0 |
| allergies | description | character varying | 24 | 619 | 619 | 0 |
| | | | | | | |
| careplans | id | character varying | 36 | 2939 | 2939 | 0 |
| careplans | start | date | 10 | 2939 | 2939 | 0 |
| careplans | stop | date | 10 | 2939 | 2939 | 0.380061245321538 |
| careplans | patient | character varying | 36 | 2939 | 2939 | 0 |
| careplans | encounter | character varying | 36 | 2939 | 2939 | 0 |
| careplans | code | character varying | 15 | 2939 | 2939 | 0 |
| careplans | description | character varying | 62 | 2939 | 2939 | 0 |
| careplans | reason_code | character varying | 14 | 2939 | 2939 | 0.09050697516162 |
| careplans | reason_desc | character varying | 69 | 2939 | 2939 | 0.09050697516162 |
| | | | | | | |
| conditions | start | date | 10 | 7899 | 7899 | 0 |
| conditions | stop | date | 10 | 7899 | 7899 | 0.458032662362324 |
| conditions | patient | character varying | 36 | 7899 | 7899 | 0 |
| conditions | encounter | character varying | 36 | 7899 | 7899 | 0 |
| conditions | code | character varying | 15 | 7899 | 7899 | 0 |
| conditions | description | character varying | 80 | 7899 | 7899 | 0 |
| | | | | | | |
| encounters | id | character varying | 36 | 34275 | 34275 | 0 |
| encounters | start | date | 10 | 34275 | 34275 | 0 |
| encounters | stop | date | 10 | 34275 | 34275 | 0 |
| encounters | patient | character varying | 36 | 34275 | 34275 | 0 |
| encounters | provider | character varying | 36 | 34275 | 34275 | 0.006652078774617 |
| encounters | encounterclass | character varying | 10 | 34275 | 34275 | 0 |
| encounters | code | character varying | 9 | 34275 | 34275 | 0 |
| encounters | description | character varying | 59 | 34275 | 34275 | 0 |
| encounters | cost | numeric | 6 | 34275 | 34275 | 0 |
| encounters | reasoncode | character varying | 15 | 34275 | 34275 | 0.66328227571116 |
| encounters | reasondescription | character varying | 69 | 34275 | 34275 | 0.66328227571116 |
| | | | | | | |
| imaging_studies | id | character varying | 0 | 0 | 0 | 0 |
| imaging_studies | date | date | 0 | 0 | 0 | 0 |
| imaging_studies | patient | character varying | 0 | 0 | 0 | 0 |
| imaging_studies | encounter | character varying | 0 | 0 | 0 | 0 |

Overview | allergies | careplans | conditions | encounters | imaging_studies | immunizations | medications | observations | organizations | patients | procedures

Figure 7.3: Example overview tab from a scan report



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | start | Frequency | stop | Frequency | patient | Frequency | encounter | Frequency | code | Frequency | description | Frequency | | | |
| 2 | 2013-01-10 | 7 | | 3618 | c7751bda-c4 | 23 | 3b44ae5b-07 | 6 | 444814009 | 1134 | Viral sinusitis | 1135 | | | |
| 3 | 2011-03-29 | 7 | 2018-08-08 | 7 | 7f2d94ac-08 | 23 | 9e223d32-52 | 6 | 195662009 | 636 | Acute viral ph | 636 | | | |
| 4 | 2012-12-28 | 7 | 2016-11-26 | 6 | 18d59307-54 | 21 | 2d996f79-c6 | 5 | 10509002 | 531 | Acute bronch | 531 | | | |
| 5 | 2016-04-24 | 6 | 2016-08-07 | 6 | 1380e511-e4 | 21 | f0a80487-98 | 5 | 72892002 | 522 | Normal pregn | 522 | | | |
| 6 | 2012-08-27 | 6 | 2011-03-27 | 6 | 78902dee-19 | 21 | 510d549e-45 | 5 | 162864005 | 501 | Body mass in | 501 | | | |
| 7 | 1990-01-29 | 6 | 2013-07-29 | 6 | 02e3acd8-05 | 21 | bd28c905-69 | 5 | 15777000 | 310 | Prediabetes | 310 | | | |
| 8 | 2018-05-04 | 6 | 2012-08-06 | 5 | a609db04-ee | 21 | 20221a49-5f | 5 | 38341003 | 299 | Hypertension | 299 | | | |
| 9 | 2013-01-12 | 6 | 2018-01-09 | 5 | 7c3d1b9c-a3 | 20 | 0dfaa9be-98 | 5 | 40055000 | 232 | Chronic sinus | 232 | | | |
| 10 | 1957-06-17 | 6 | 2014-03-26 | 5 | 77a302ad-dc | 20 | 313af90c-41 | 5 | 19169002 | 191 | Miscarriage in | 191 | | | |
| 11 | 2010-05-07 | 6 | 2016-11-06 | 5 | be67fa78-59 | 19 | 353746e6-7d | 4 | 65363002 | 170 | Otitis media | 170 | | | |
| 12 | 2013-11-24 | 6 | 2014-11-01 | 5 | d3814387-d9 | 19 | ee1578cd-96 | 4 | 43878008 | 168 | Streptococca | 168 | | | |
| 13 | 2013-11-18 | 6 | 2016-05-01 | 5 | 48948c73-cd | 19 | 0fd09e40-0b | 4 | 408512008 | 133 | Body mass in | 133 | | | |
| 14 | 2012-11-25 | 6 | 2016-05-20 | 5 | 55a789e2-18 | 19 | 84c82ea8-59 | 4 | 44465007 | 121 | Sprain of ank | 121 | | | |
| 15 | 2017-12-20 | 6 | 2015-11-29 | 5 | 5099f86b-b6 | 19 | 1fbd96b0-63 | 4 | 55822004 | 110 | Hyperlipidem | 110 | | | |
| 16 | 1941-12-19 | 6 | 2011-05-12 | 5 | 7dc329ea-e1 | 18 | 5418009a-41 | 4 | 68496003 | 86 | Polyp of colo | 86 | | | |
| 17 | 2012-10-06 | 6 | 2011-05-09 | 5 | 92efe791-86 | 18 | 0d0db0e2-10 | 4 | 36971009 | 83 | Sinusitis (dis | 83 | | | |
| 18 | 2015-09-06 | 6 | 2011-10-18 | 5 | 90781fa9-00 | 18 | 08c1012c-cf | 4 | 53741008 | 73 | Coronary He | 73 | | | |
| 19 | 2018-07-19 | 6 | 2014-12-22 | 5 | 100d3d5c-c0 | 18 | 8ac796c0-a7 | 3 | 44054006 | 69 | Diabetes | 69 | | | |
| 20 | 2018-07-20 | 6 | 2010-04-16 | 5 | 2fd90c2b-e3 | 18 | 7c4d48be-2d | 3 | 75498004 | 67 | Acute bacter | 67 | | | |
| 21 | 2016-04-29 | 5 | 2016-12-10 | 5 | de6ff7b0-c07 | 18 | e69d16bf-bb | 3 | 230690007 | 67 | Stroke | 67 | | | |
| 22 | 2015-12-20 | 5 | 2009-06-17 | 5 | 2b2ace45-fd | 17 | 4ec52a96-d4 | 3 | 302870006 | 66 | Hypertriglyce | 66 | | | |
| 23 | 2010-12-11 | 5 | 2018-07-09 | 5 | f0279b18-afb | 17 | 34e03665-72 | 3 | 62106007 | 65 | Concussion w | 65 | | | |
| 24 | 2011-04-03 | 5 | 2016-04-28 | 4 | e4367c1b-9d | 17 | a0e185c2-e5 | 3 | 237602007 | 64 | Metabolic sy | 64 | | | |
| 25 | 2018-03-31 | 5 | 2011-04-06 | 4 | 8f6043f7-e9c | 17 | f47fcfbc-114 | 3 | 82423001 | 62 | Chronic pain | 62 | | | |
| 26 | 2017-11-25 | 5 | 2015-12-14 | 4 | dcc1a550-3f | 17 | a6e4e4c6-47 | 3 | 74400008 | 58 | Appendicitis | 58 | | | |
| 27 | 2009-01-18 | 5 | 2018-03-28 | 4 | 7797620-8b | 17 | 1a3b2546-b0 | 3 | 428251008 | 58 | History of ap | 58 | | | |
| 28 | 2016-06-01 | 5 | 2015-12-22 | 4 | 95e1832d-2c | 17 | aac8b5fb-ad | 3 | 80394007 | 57 | Hyperglycem | 57 | | | |
| 29 | 1956-08-03 | 5 | 2017-08-31 | 4 | b3fdd210-2d | 17 | af385b94-d0 | 3 | 196416002 | 56 | Impacted mo | 56 | | | |
| 30 | 1953-05-22 | 5 | 2011-04-05 | 4 | c1537b8f-60 | 17 | b7669cb3-41 | 3 | 12417100011 | 54 | Chronic intra | 54 | | | |
| 31 | 1990-08-30 | 5 | 2009-11-06 | 4 | 5ddb8dc0-bb | 17 | 2f1d8db5-8d | 3 | 64859006 | 52 | Drug overdos | 52 | | | |
| 32 | 2016-02-15 | 5 | 2013-01-04 | 4 | 180029f8-1a | 17 | c815f529-9c | 3 | 55680006 | 52 | Osteoporosis | 52 | | | |
| 33 | 2010-07-14 | 5 | 2013-01-07 | 4 | e3757e8c-d5 | 17 | 21853938-7a | 3 | 70704007 | 51 | Sprain of wris | 51 | | | |
| 34 | 2010-07-17 | 5 | 2017-11-15 | 4 | 153de354-51 | 17 | 957971d7-8f | 3 | 39848009 | 49 | Whiplash inju | 49 | | | |
| 35 | 2012-06-04 | 5 | 2010-09-02 | 4 | bc4c71af-b8 | 17 | 77addf56-4e | 3 | 239873007 | 46 | Osteoarthritis | 46 | | | |
| 36 | 2018-01-17 | 5 | 2018-05-09 | 4 | 1686d3b7-5a | 16 | 10161853-c5 | 3 | 65966004 | 45 | Fracture of fe | 45 | | | |
| 37 | 2011-04-30 | 5 | 2013-01-19 | 4 | daf3aaf7-f87 | 16 | 9a790773-7e | 3 | 703151001 | 43 | Seizure disor | 43 | | | |
| 38 | 2014-03-19 | 5 | 2011-02-19 | 4 | df95453a-1f5 | 16 | eceae126-99 | 3 | 128613002 | 43 | History of sir | 43 | | | |
| 39 | 2018-01-31 | 5 | 2010-07-30 | 4 | a18b8ad1-51 | 16 | 00150c2c-1c | 3 | 49436004 | 41 | Atrial Fibrillat | 41 | | | |
| 40 | 1932-11-28 | 5 | 2018-01-03 | 4 | e47ddbd5-27 | 16 | 5f3fe2b7-048 | 3 | 271737000 | 39 | Anemia (diso | 39 | | | |
| 41 | 2017-04-17 | 5 | 2011-06-05 | 4 | 5048ce61-3d | 16 | 0a40fe52-d3 | 3 | 284551006 | 38 | Laceration of | 38 | | | |

Overview | allergies | careplans | **conditions** | encounters | imaging_studies | immunizations | medications | observations | organizations | patients | procedures

Figure 7.4: Example tab from a scan report

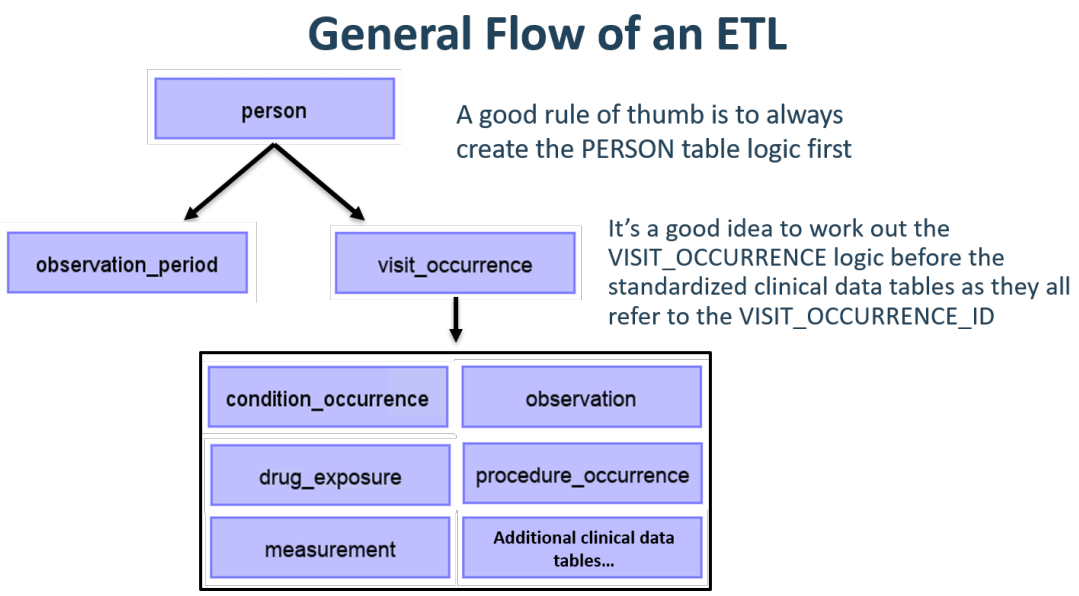



Figure 7.5: General flow of an ETL and which tables to map first

Since the OMOP CDM is a person-centric model it is always a good idea to start mapping the PERSON table first. Every clinical event table (CONDITION_OCCURRENCE, DRUG_EXPOSURE, PROCEDURE_OCCURRENCE, etc.) refers back to the PERSON table by way of the person_id so working out the logic for the PERSON table first makes it easier later on. After the PERSON table a good rule of thumb is to convert the OBSERVATION_PERIOD table next. Each person in a CDM database should have at least one OBSERVATION_PERIOD and, generally, most events for a person fall within this timeframe. Once the PERSON and OBSERVATION_PERIOD tables are done the dimensional tables like PROVIDER, CARE_SITE, and LOCATION are typically next. The final table logic that should be worked out prior to the clincal tables is VISIT_OCCURRENCE. Often this is the most complicated logic in the entire ETL and it is some of the most crucial since most events that occur during the course of a person's patient journey will happen during visits. Once those tables are finished it is your choice which CDM tables to map and in which order.

*Note*

It is often the case that, during CDM conversion, you will need to make provisions for intermediate tables. This could be for assigning the correct visit_occurrence_ids to events, or for mapping source codes to standard concepts (doing this step on the fly is often very slow). This is 100% allowed and encouraged. What is discouraged is the persistence and reliance on these tables once the conversion is complete.

#### 7.2.2.1 Mapping Example: Person table

The Synthea data structure contains 20 columns in the patients table (https://github.com/synthetichealth/synthea/wiki/CSV-File-Data-Dictionary#patients) but not all were needed to

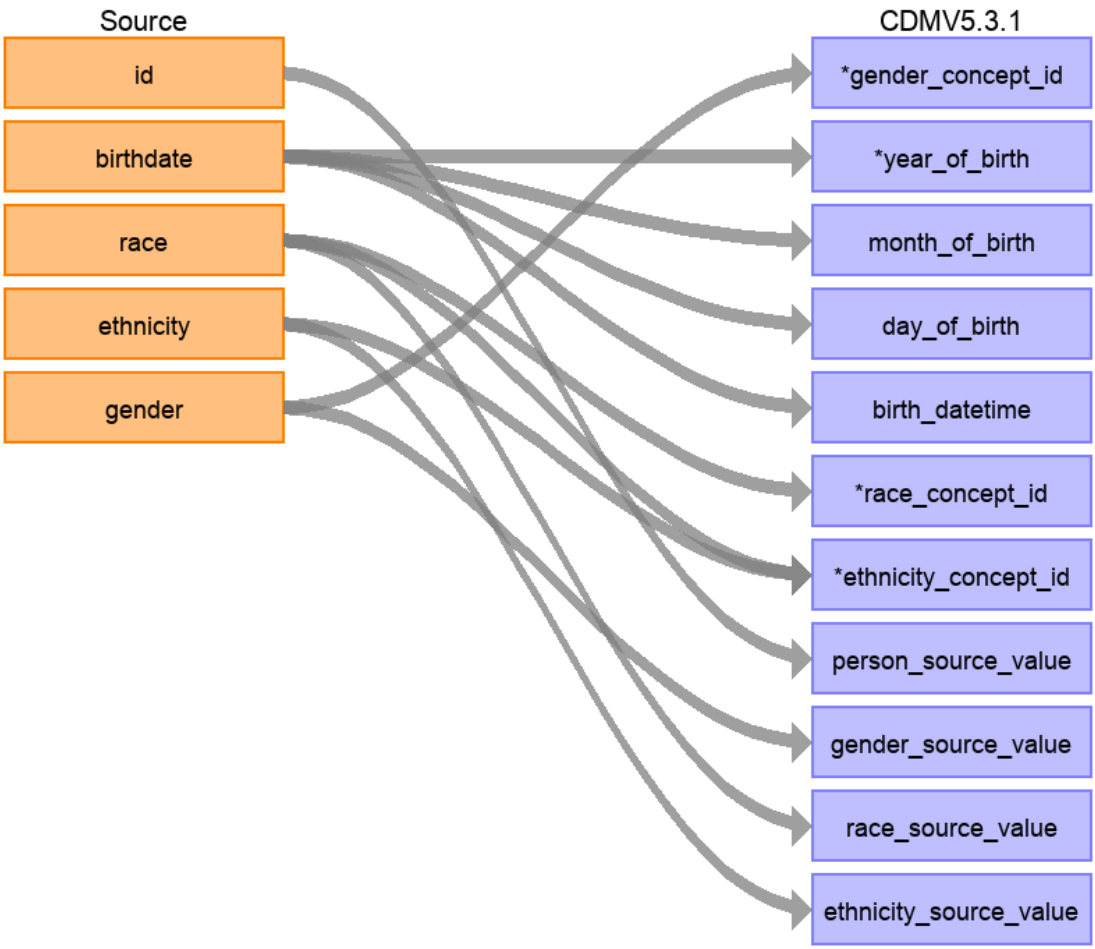



Figure 7.6: Mapping of Synthea Patients table to CDM PERSON table.

populate the PERSON table, as seen in figure 7.6. This is very common and should not be cause for alarm. In this example many of the data points in the Synthea patients table that were not used in the CDM PERSON table were additional identifiers like patient name, driver's license number, and passport number.

The table below shows the logic that was imposed on the Synthea patients table to convert it to the CDM PERSON table. The 'Comment field' column gives explanations for why the logic was chosen.

Table 7.1: ETL logic to convert the Synthea Patients table to CDM PERSON table.

| Destination Field | Source field | Logic | Comment field |
|---|---|---|---|
| person_id | | Autogenerate | The person_id will be generated at the time of implementation. This is because the id value from the source is a varchar value while the person_id is an integer. The id field from the source is set as the person_source_value to preserve that value and allow for error-checking if necessary. |
| gender_concept_id | gender | When gender = 'M' then set gender_concept_id to 8507, when gender = 'F' then set to 8532. Drop any rows with missing/unknown gender. | These two concepts were chosen as they are the only two standard concepts in the gender domain[a]. The choice to drop patients with unknown genders tends to be site-based, though it is recommended they are removed as people without a gender are excluded from analyses. |
| year_of_birth | birthdate | Take year from birthdate | |
| month_of_birth | birthdate | Take month from birthdate | |
| day_of_birth | birthdate | Take day from birthdate | |
| birth_datetime | birthdate | With midnight as time 00:00:00 | Here, the source did not supply a time of birth so the choice was made to set it at midnight. |

[a] http://athena.ohdsi.org/search-terms/terms?domain=Gender&standardConcept=Standard&page=1&pageSize=15&query=

| Destination Field | Source field | Logic | Comment field |
|---|---|---|---|
| race_concept_id | race | When race = 'WHITE' then set as 8527, when race = 'BLACK' then set as 8516, when race = 'ASIAN' then set as 8515, otherwise set as 0 | These concepts were chosen because they are the standard concepts belonging to the race domain that most closely align with the race categories in the source[a]. [a] http://athena.ohdsi.org/search-terms/terms?domain=Race&standardConcept=Standard&page=1&pageSize=15&query= |
| ethnicity_concept_id | race ethnicity | When race = 'HISPANIC', or when ethnicity in ('CENTRAL_AMERICAN', 'DOMINICAN', 'MEXICAN', 'PUERTO_RICAN', 'SOUTH_AMERICAN') then set as 38003563, otherwise set as 0 | This is a good example of how multiple source columns can contribute to one CDM column. In the CDM ethnicity is represented as either hispanic or not hispanice so values from both the source column race and source column ethnicity will determine this value. |
| location_id | | | |
| provider_id | | | |
| care_site_id | | | |
| person_source_value | id | | |
| gender_source_value | gender | | |
| gender_source_concept_id | | | |
| race_source_value | race | | |
| race_source_concept_id | | | |
| ethnicity_source_value | ethnicity | | In this case the ethnicity_source_value will have more granularity than the ethnicity_concept_id. |

| Destination Field | Source field | Logic | Comment field |
|---|---|---|---|
| ethnicity_source_concept_id | | | |

## 7.3 ETL Step 2 - People with medical knowledge create the code mappings

## 7.4 ETL Step 3 - A technical person implements the ETL

## 7.5 ETL Step 4 - All are involved in quality control

Business Rules and Conventions: From the CDM Wiki + Themis

Conversion to OMOP CDM (ETL - Extract, Transform, Load): http://www.ohdsi.org/web/wiki/doku.php?id=documentation:etl_best_practices

- WhiteRabbit and Rabbit-in-a-Hat: http://www.ohdsi.org/web/wiki/doku.php?id=documentation:software:whiterabbit
- Usagi: http://www.ohdsi.org/web/wiki/doku.php?id=documentation:software:usagi
- Achilles: http://www.ohdsi.org/web/wiki/doku.php?id=documentation:software:achilles
- Athena: http://www.ohdsi.org/web/wiki/doku.php?id=documentation:vocabulary_etl

Mapping and QA of codes to Standard Concepts

- Mapping codes locally versus through the OHDSI Standard Vocabularies
- Usagi
- Systematic mapping of Drug codes
- Systematic mapping of Condition codes
- Systematic mapping of Procedure codes
- Systematic mapping of other codes

# Part III

# Data Analytics

# Chapter 8

# Data Analytics Use Cases

*Chapter lead: David Madigan*

The OHDSI collaboration focuses on generating reliable evidence from real-world healthcare data, typically in the form of claims databases or electronic health record databases. The use cases that OHDSI focuses on fall into three major categories:

- Characterization
- Population-level estimation
- Patient-level prediction

We describe these in detail below. Note, for all the use cases, the evidence we generate inherits the limitations of the data; we discuss these limitations at length in the book section on Evidence Quality (Chapters 15 - 19)

## 8.1 Characterization

Characterization attempts to answer the question

> What happened to them?

We can use the data to provide answers to questions about the characteristics of the persons in a cohort or the entire database, the practice of healthcare, and study how these things change over time.

The data can provide answers to questions like:

- For patients newly diagnosed with atrial fibrillation, how many receive a prescription for warfarin?
- What is the average age of patients who undergo hip arthroplasty?
- What is the incidence rate of pneumonia in patients over 65 years old?

## 8.2 Population-level estimation

To a limited extent, the data can support causal inferences about the effects of healthcare interventions, answering the question

> What are the causal effects?

We would like to understand causal effects to understand consequences of actions. For example, if we decide to take some treatment, how does that change what happens to us in the future?

The data can provide answers to questions like:

- For patients newly diagnosed with atrial fibrillation, in the first year after therapy initiation, does warfarin cause more major bleeds than dabigatran?
- Does the causal effect of metformin on diarrhea vary by age?

## 8.3 Patient-Level prediction

Based on the collected patient health histories in the database, we can make patient-level predictions about future health events, answering the question

> What will happen to me?

The data can provide answers to questions like:

- For a specific patient newly diagnosed with major depressive disorder, what is the probability the patient will attempt suicide in the first year following diagnosis?
- For a specific patient newly diagnosed with atrial fibrillation, in the first year after therapy initiation with warfarin, what is the probability the patient suffers an ischemic stroke?

Population-level estimation and patient-level prediction overlap to a certain extent. For example, an important use case for prediction is to predict an outcome for a specific patient had drug A been prescribed and also predict the same outcome had drug B been prescribed. Let's assume that in reality only one of these drugs is prescribed (say drug A) so we get to see whether the outcome following treatment with A actually occurs. Since drug B was not prescribed, the outcome following treatment B, while predictable, is "counterfactual" since it is not ever observed. Each of these prediction tasks falls under patient-level prediction. However, the difference between (or ratio of) the two outcomes is a unit-level *causal* effect, and should be estimated using causal effect estimation methods instead.

People have a natural tendency to erroneously interpret predictive models as if they are causal models. But a predictive model can only show correlation, never causation. For example, diabetic drug use might be a strong predictor for myocardial infarction (MI) because diabetes is a strong risk factor for MI. However, that does not mean that stopping the diabetic drugs will prevent MI!

## 8.4 Limitations of observational research

There are many important healthcare questions for which OHDSI databases cannot provide answers. These include:

- Causal effects of interventions compared to placebo. Sometimes it is possible to consider the causal effect of a treatment as compared with non-treatment but not placebo treatment.
- Anything related to over-the-counter medications.
- Many outcomes and other variables are sparsely recorded if at all. These include mortality, behavioral outcomes, lifestyle, and socioeconomic status.
- Since patients tend to encounter the healthcare system only when they are unwell, measurement of the benefits of treatments can prove elusive.

### 8.4.1 Missing data

Missingness in OHDSI databases presents subtle challenges. A health event (e.g., prescription, laboratory value, etc.) that should be recorded in a database, but isn't, is "missing." The statistics literature distinguishes between types of missingness such as "missing completely at random," "missing at random", and "missing not at random" and methods of increasing complexity attempt to address these types. Perkins et al. (2017) provide a useful introduction to this topic.

## 8.5 Summary

- In observational research we distinguish three large categories of uses cases.
- **Characterization aims** to answer the questions "What happened to them?"
- **Population-level estimation** attempts to answer the question "What are the causal effects?"
- **Patient-level prediction** tries to answer "What will happen to me?"
- Prediction models are not causal models; There is no reason to believe that intervening on a strong predictor will impact the outcome.

# Chapter 9

# OHDSI Analytics Tools

*Chapter leads: Martijn Schuemie & Frank DeFalco*

OHDSI offers a wide range of open source tools to support the various data-analytics use cases. What these tools have in common is that they can all interaction with one or more databases using the Commond Data Model (CDM). Furthermore, these tools standardize the analytics for various use cases; Rather than having to start from scratch, an analysis can be implemented by filling in standard templates. This makes performign analysis easier, and also improves reproducibility and transparancy. For example, there appear to be a near-infinte number of ways to compute an incidence rate, but these can be specified in the OHDSI tools with a few choices, and anyone making those same choices will compute incidence rates the same way.

In this chapter we first describe various ways in which we can choose to implement an analysis, and what strategies the analysis can employ. We then review the various OHDSI tools and how they fit the various use cases.

## 9.1 Analysis implementation

Figure 9.1 shows the various ways in which we can choose to implement a study against a database using the CDM.

We may choose to write our analysis as custom code, and not make use of any of the tools OHDSI has to offer. One could write a de novo analysis in R, SAS, or any other language. This provides the maximum flexibility, and may in fact be the only option if the specific analysis is not supported by any of our tools. However, this path requires a lot of technical skill, time, and effort, and as the analysis increases in complexity it becomes harder to avoid errors in the code.

An alternative is to develop the analysis in R, and make use of the packages in the OHDSI Methods Library. At a minimum, one could use the SqlRender and DatabaseConnector packages described in more detail in Chapter 10 that allow the same code to be executed on various database platforms, such as PostgreSQL, SQL Server, and Oracle. Other packages such as CohortMethod and PatientLevel-

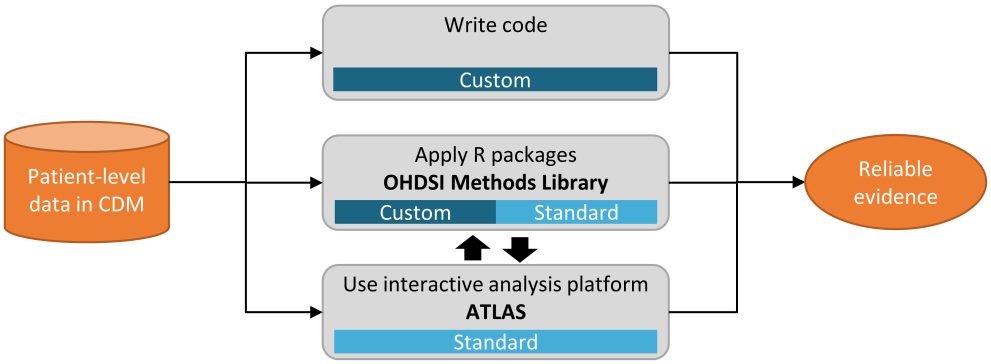



Figure 9.1: Different ways to implement an analysis against data in the CDM.

Prediction offer R functions for advanced analytics against the CDM that can be called on in one's code. This still requires a lot of tecnhical expertise, but by re-using the validated components of the Methods Library we can be more efficient and error-free than when using completely custom code.

The third approach relies on our interactive analysis platform ATLAS, a web-based tool that allows non-programmers to perform a wide range of analyses efficiently. The downside is that some options may not be available.

ATLAS and the Methods Library are not independent. Some of the more complicated analytics that can be invoked in ATLAS are executed through calls to the packages in the Methods Library. Similarly, cohorts used in the Methods Library are often designed in ATLAS.

## 9.2 Analysis strategy

More or less independently of how we choose to implement our analysis is the strategy that our analytics takes in answering specific questions. Figure 9.2 highlights three strategies that are employed in OHDSI.

The first strategy views every analysis as a single individual study. The analysis must be pre-specified in a protocol, implemented as code, executed against the data, after which the result can be compiled and interpreted. For every question, all steps must be repeated. An example of such an analysis is the OHDSI study into the risk of angioedema associated with levetiracetam compared with phenytoin. (Duke et al., 2017) Here, a protocol was first written, analysis code using the OHDSI Methods Library was developed and executed across the OHDSI network, and results were compiled and disseminated in a journal publication.

The second strategy develops some app that allows users to answer a specific class of questions in real time or near-real time. Once the app has been developed, users can interactively define queries, submit them, and view the results. An example is the cohort definition and generation tool in ATLAS. This tool allows users to specify cohort definitions of arbitrary complexity, and execute the definition against a database to see how many people meet the various inclusion and exclusion criteria.

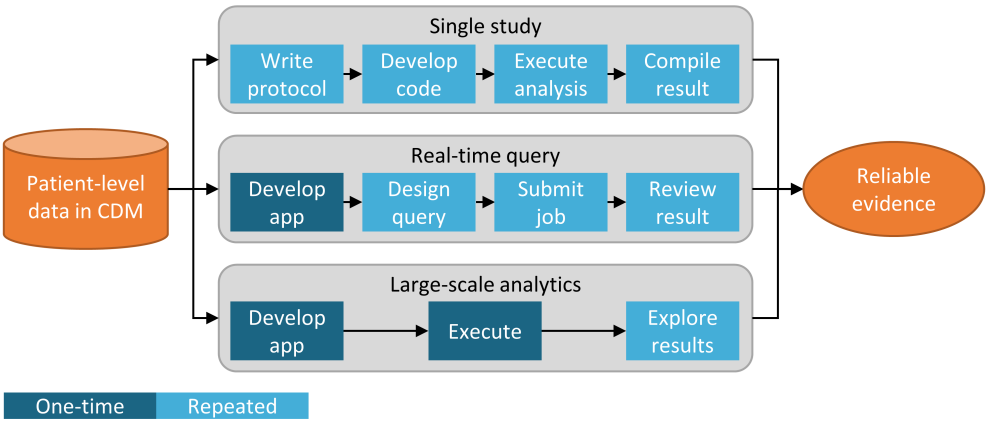



Figure 9.2: Strategies for generating evidence for (clinical) questions.

The third strategy similarly focuses on a class of questions, but then attempts to exhaustively generate all the evidence for the questions within the class. Users can then explore the evidence as needed, usually through some viewer app. One example is the OHDSI study into the effects of depression treatments (Schuemie et al., 2018b). In this study all depression treatments are compared for a large set of outcomes of interest across four large observational databases. The full set of results, including 17,718 empirically calibrated hazard ratios along with extensive study diagnostics, is available in an interactive web app [1].

## 9.3 ATLAS

ATLAS is a web-based tool that must run on a server with access to the patient-level data in the CDM. To directly run the analyses against the data, ATLAS must therefore be installed behind your organization's firewall. However, there is also a public ATLAS [2], and although this ATLAS instance only has access to a small simulated dataset, it can still be used for many purposes. For example, it is possible to fully define an effect estimation of prediction study in the public ATLAS, and automatically generate the R code for executing the study.

A screenshot of ATLAS is provided in Figure 9.3. On the left is a navigation bar showing the various functions provided by ATLAS:

**Data Sources** Data sources provides the capability review descriptive, standardized reporting for each of the data sources that you have configured within your Atlas platform. This feature uses the large-scale analytics strategy: all descriptives have been pre-computed. Data sources is discussed in Chapter 12.

**Vocabulary Search** Atlas provides the ability to search and explore the OMOP standardized vocabulary to understand what concepts exist within those vocabularies and how to apply those

[1] http://data.ohdsi.org/SystematicEvidence/
[2] http://www.ohdsi.org/web/atlas

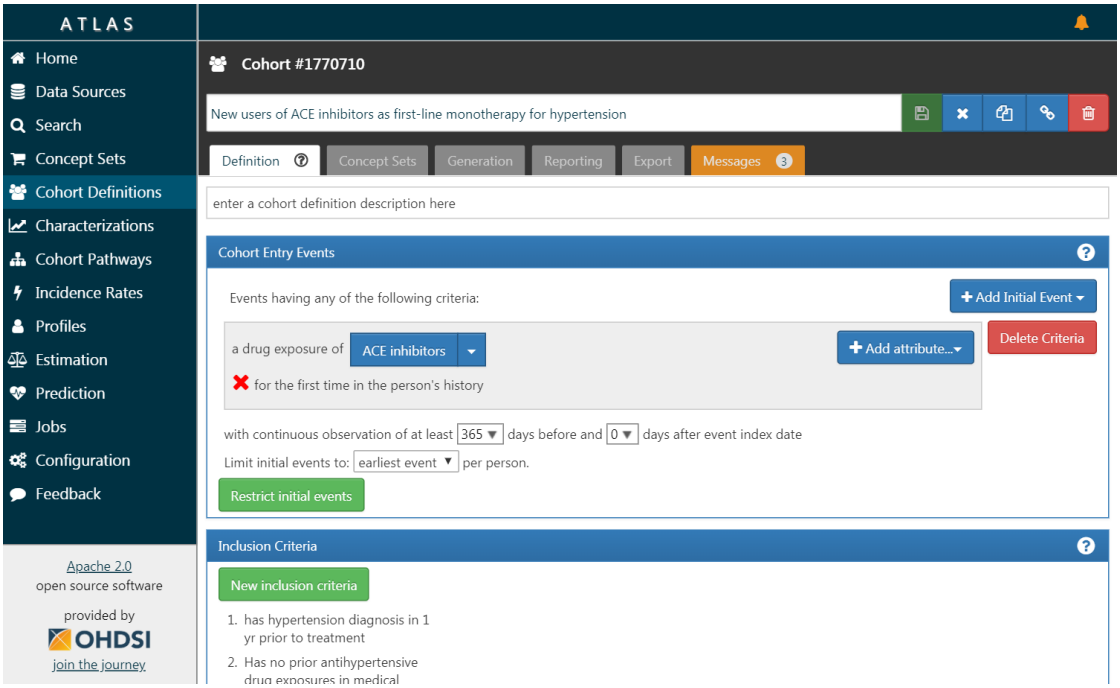



Figure 9.3: ATLAS user interface.

concepts in your standardized analysis against your data sources. This feature is discussed in Chapter 6.

**Concept Sets** Concept sets is the ability to create your own lists of codes that you are going to use throughout your standardized analyses so by searching the vocabulary and identifying the sets of terms that you're interested in you can save those and reuse them in all of your analyses.

**Cohort Definitions** Cohort definitions is the ability to construct a set of persons who satisfy one or more criteria for a duration of time and these cohorts can then serve as the basis of inputs for all of your subsequent analyses. This feature is discussed in Chapter 11.

**Characterizations** Characterisations is an analytic capability that allows you to look at one or more cohorts that you've defined and to summarize characteristics about those patient populations. This feature uses the real-time query strategy, and is discussed in Chapter 12.

**Cohort Pathways** Cohort pathways is an analytic tool that allows you to look at the sequence of clinical events that occur within one or more populations. This feature uses the real-time query strategy, and is discussed in Chapter 12.

**Incidence Rates** Incidence rates is a tool that allows you to estimate the incidence of outcomes within target populations of interest. This feature uses the real-time query strategy, and is discussed in Chapter 12.

**Profiles** Profiles is a tool that allows you to explore an individual patients longitudinal observational data to summarize what is going on within a given individual. This feature uses the real-time query strategy.

**Population Level Estimation** Estimation is a capability to allow you to conduct population level effect estimation studies using a comparative cohort design whereby comparisons between

one or more target and comparator cohorts can be explored for a series of outcomes. This feature can be said to implement the real-time query strategy, as no coding is required, and is discussed in Chapter 13.

**Patient Level Prediction** Prediction is a capability to allow you to apply machine learning algorithms to conduct patient level prediction analyses whereby you can predict an outcome within any given target exposures. This feature can be said to implement the real-time query strategy, as no coding is required, and is discussed in Chapter 14.

**Jobs** Select the "jobs" menu item to explore jobs that are running in the background for long running processes such as generating a cohort or computing cohort reports.

**Configuration** Select the "configuration" menu item to review the data sources that have been configured in the source configuration section.

**Feedback** This will take you to the issue log for Atlas so that you can log a new issue or to search through existing issues. If you have ideas for new features or enhancements, this is also a place note these for the development community.

### 9.3.1 Security

### 9.3.2 Documentation

### 9.3.3 System requirements

### 9.3.4 How to install

## 9.4 Methods Library

The OHDSI Methods Library is the collection of open source R packages show in Figure 9.4.

The packages offer R functions that together can be used to perform an observation study from data to estimates and supporting statistics, figures, and tables. The packages interact directly with observational data in the CDM, and can be used simply to provide cross-platform compatibility to completely custom analyses as described in Chapter 10, or can provide advanced standardized analytics for population characterization (Chapter 12), population-level causal effect estimation (Chapter 13), and patient-level prediction (Chapter 14). The Methods Library supports best practices for use of observational data as learned from previous and ongoing research, such as transparency, reproducibility, as well as measuring of the operating characteristics of methods in a particular context and subsequent empirical calibration of estimates produced by the methods.

The Methods Library has already been used in many published clinical studies (Boland et al., 2017; Duke et al., 2017; Ramcharran et al., 2017; Weinstein et al., 2017; Wang et al., 2017; Ryan et al., 2017, 2018; Vashisht et al., 2018; Yuan et al., 2018; Johnston et al., 2019), as well as methodological studies (Schuemie et al., 2014, 2016; Reps et al., 2018; Tian et al., 2018; Schuemie et al., 2018a,b; Reps et al., 2019). Great care is taken to ensure the validity of the Methods Library, as described in Chapter 18.


Figure 9.4: Packages in the OHDSI Methods Library.

### 9.4.1 Support for large-scale analytics

One key feature incorporated in all packages is the ability to efficiently run many analyses. For example, when performing population-level estimation, the CohortMethod package allows for computing effect-size estimates for many exposures and outcomes, using various analysis settings, and the package will automatically choose the optimal path to compute all the required artifacts. Steps that can be re-used, such as extraction of covariates, or fitting a propensity model, will be executed only once. Where possible, computations will take place in parallel to maximuze the use of computational resources.

This feature allows for large-scale analytics, answering many questions at once, and is also essential for including control hypotheses (e.g. negative controls) to measure the operating characteristics of our methods, and perform empirical calibration as described in Chapter 19.

### 9.4.2 Support for big data

The Methods Library is also designed to run against very large databases and be able to perform computations involving large amounts of data. This achieved in three ways:

1. Most data manipulation is performed on the database server. An analysis usually only requires a small fraction of the entire data in the database, and the Methods Library, through the SqlRender and DatabaseConnector packages, allows for advanced operations to be performed on the server to preprocess and extract the relevant data.
2. Large local data objects are stored in a memory-efficient manner. For the data that is downloaded to the local machine, the Methods Library uses the ff package to store and work with large data objects. This allows us to work with data much larger than fits in memory.
3. High-performance computing is applied where needed. For example, the Cyclops package implements a highly efficient regression engine that is used throughout the Methods Library to perform large-scale regressions (large number of variables, large number of observations) that would not be possible to fit otherwise.

### 9.4.3 Documentation

R provides a standard way of documenting package. Each package has a *package manual* that documents every function and data set in the package. All package manuals are available online through the Methods Library website [3], through the package GitHub repositories, and for those packages available through CRAN they can be found in CRAN. Furtermore, from within R the package manual can be consulted by using the question mark. For example, after loading the DatabaseConnector package, typing the command `?connect` brings up the documentation on the "connect" function.

In addition to the package manual, many packages provide *vignettes*. Vignettes are long-form documentation that describe how a package can be used to perform certain tasks. For example, one

[3] https://ohdsi.github.io/MethodsLibrary

vignette [4] describes how to perform multiple analyses efficiently using the CohortMethod package. Vignettes can also be found through the Methods Library website , through the package GitHub repositories, and for those packages available through CRAN they can be found in CRAN.

### 9.4.4 System requirements

Two computing environments are relevant when discussing the system requirements: The database server, and the analytics workstation.

The database server must hold the observational healthcare data in CDM format. The Methods Library supports a wide array of database management systems including traditional database systems (PostgreSQL, Microsoft SQL Server, and Oracle), parallel data warehouses (Microsoft APS, IBM Netezza, and Amazon RedShift), as well as Big Data platforms (Hadoop through Impala, and Google BigQuery).

The analytics workstation is where the Methods Library is installed and run. This can either be a local machine, such as someone's laptop, or a remote server running RStudio Server. In all cases the requiments are that R is installed, preferrably together with RStudio. The Methods Library also requires that Java is installed. The analytics workstation should also be able to connect to the database server, specifically, any firewall between them should have the database server access ports opened the the workstation. Some of the analytics can be computationally intensive, so having multiple processing cores and ample memory can help speed up the analyses. We recommend having at least four cores and 16 gigabytes of memory.

### 9.4.5 How to install

Here are the steps for installing the required environment to run the OHDSI R packages. Four things needs to be installed:

1. **R** is a statistical computing environment. It comes with a basic user interface that is primarily a command-line interface.
2. **RTools** is a set of programs that is required on Windows to build R packages from source.
3. **RStudio** is an IDE (Integrated Development Environment) that makes R easier to use. It includes a code editor, debugging and visualization tools. Please use it to obtain a nice R experience.
4. **Java** is a computing environment that is needed to run some of the components in the OHDSI R packages, for example those needed to connect to a database.

Below we describe how to install each of these in a Windows environment.

In Windows, both R and Java come in 32-bit and 64-bits architectures. If you install R in both architectures, you **must** also install Java in both architectures. It is recommended to only install the 64-bit version of R.

[4] https://ohdsi.github.io/CohortMethod/articles/MultipleAnalyses.html


Figure 9.5: Downloading R from CRAN.

**Installing R**

1. Go to https://cran.r-project.org/, click on "Download R for Windows", then "base", then click the Download link indicated in Figure 9.5.

2. After the download has completed, run the installer. Use the default options everywhere, with two exceptions: First, it is better not to install into program files. Instead, just make R a subfolder of your C drive as shown in Figure 9.6. Second, to avoid problems due to differing architectures between R and Java, disable the 32-bit architecture as shown in Figure 9.7.

Once completed, you should be able to select R from your Start Menu.

**Installing RTools**

1. Go to https://cran.r-project.org/, click on "Download R for Windows", then "Rtools", and select the very latest version of RTools to download.

2. After downloading has completed run the installer. Select the default options everywhere.

**Installing RStudio**

1. Go to https://www.rstudio.com/, select "Download RStudio" (or the "Download" button under "RStudio"), opt for the free version, and download the installer for Windows as shown in Figure 9.8.

2. After downloading, start the installer, and use the default options everywhere.

## 9.5 Installing Java

1. Go to https://java.com/en/download/manual.jsp, and select the Windows 64-bit installer as shown in Figure 9.9. If you also installed the 32-bit version of R, you *must* also install the other (32-bit) version of Java.

2. After downloading just run the installer.

**Verifying the installation**

You should now be ready to go, but we should make sure. Start RStudio, and type

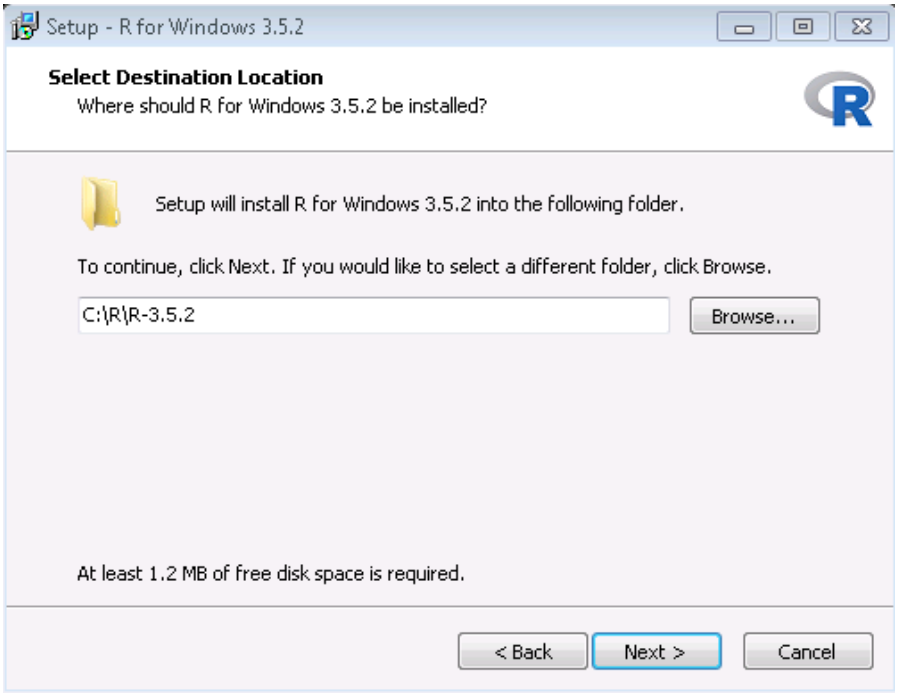



Figure 9.6: Settings the destination folder for R.

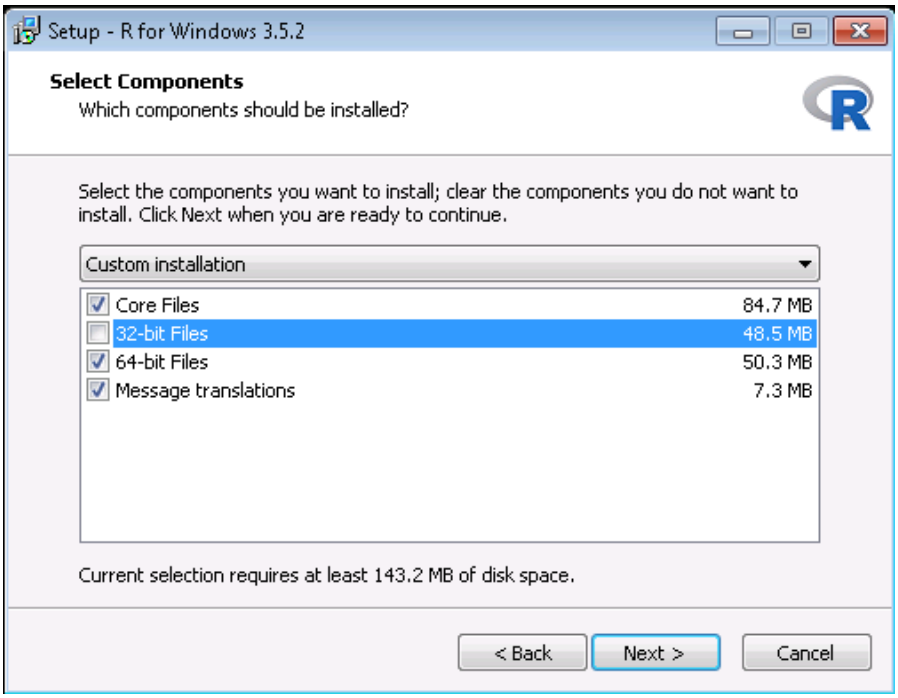



Figure 9.7: Disabling the 32-bit version of R.

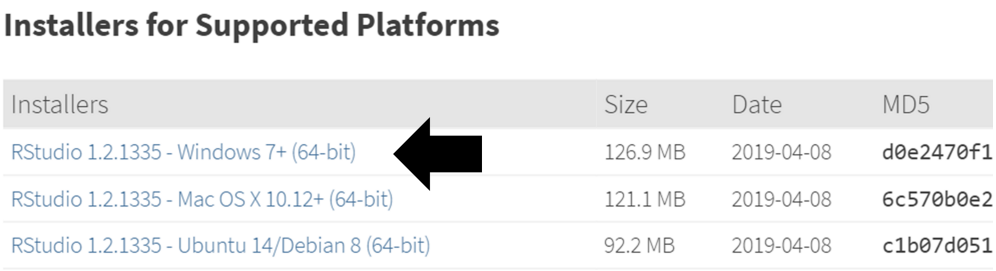



Figure 9.8: Downloading RStudio.

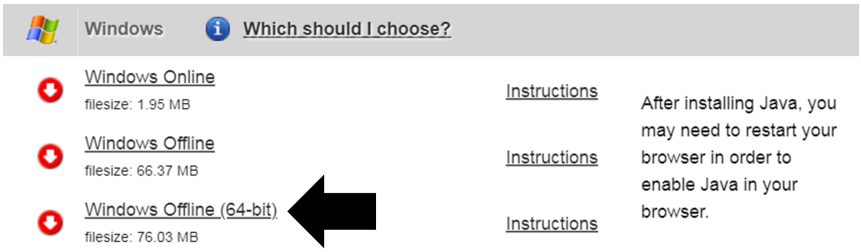



Figure 9.9: Downloading Java.

```
install.packages("SqlRender")
library(SqlRender)
translate("SELECT TOP 10 * FROM person;", "postgresql")
```

```
## [1] "SELECT  * FROM person LIMIT 10;"
```

This function uses Java, so if all goes well we know both R and Java have been installed correctly!

Another test is to see if source packages can be built. Run the following R code to install the `CohortMethod` package from the OHDSI GitHub repository:

```
install.packages("drat")
drat::addRepo("OHDSI")
install.packages("CohortMethod")
```

## 9.6 Deployment strategies

Deploying the entire OHDSI tool stack, including ATLAS and the Methods Library, in an organization is a daunting task. There are many components with dependencies that have to be considered, and configurations to set. For this reason, two initiatives have developed integrated deployment strategies that allow the entire stack to be installed as one package, using some forms of virtualization: Broadsea and Amazon Web Services (AWS).

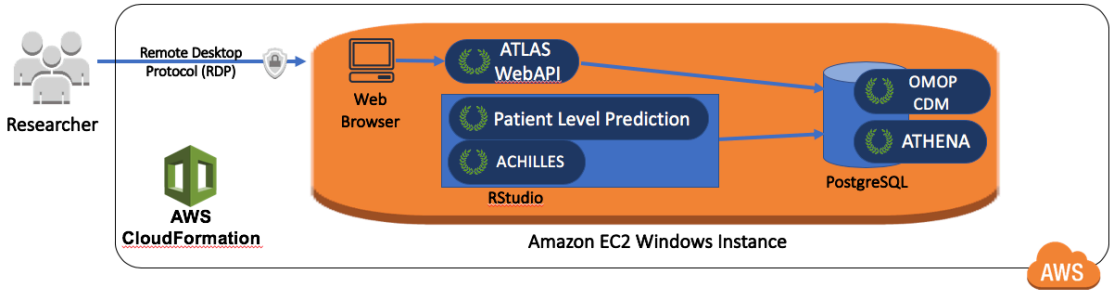



Figure 9.10: The Amazon Web Services architecure for OHDSI-in-a-Box.

### 9.6.1 Broadsea

BroadSea[5] uses Docker container technology[6]. The OHDSI tools are packaged along with dependencies into a single portable binary file called a Docker Image. This image can then be run on a Docker engine service, creating a virtual machine with all the software installed and ready to run. Docker engines are available for most operating systems, including Microsoft Windows, MacOS, and Linux. The Broadsea Docker image contains the main OHDSI tools, including the Methods Library and ATLAS.

### 9.6.2 Amazon AWS

Amazon has prepared two environments that can be instantiated in the AWS cloud computing environment with a click of the button: OHDSI-in-a-Box[7] and OHDSIonAWS[8].

OHDSI-in-a-Box is specifically created as a learning environment, and is used in most of the tutorials provided by the OHDSI community. It includes many OHDSI tools, sample data sets, RStudio and other supporting software in a single, low cost Windows virtual machine. A PostgreSQL database is used to store the CDM and also to store the intermediary results from ATLAS. The OMOP CDM data mapping and ETL tools are also included in OHDSI-in-a-Box. The architecture for OHDSI-in-a-Box is depicted in Figure 9.10.

OHDSIonAWS is a reference architecture for enterprise class, multi-user, scalable and fault tolerant OHDSI environments that can be used by organizations to perform their data analytics. It includes several sample datasets and can also automatically load your organization's real healthcare data. The data is placed in the Amazon Redshift database platform, which is supported by the OHDSI tools. Intermediary results of ATLAS are stored in a PostgreSQL database. On the front end, users have access to ATLAS and to RStudio through a web interface (leveraging RStudio Server). In RStudio the OHDSI Methods Library has already been installed, and can be used to connect to the databases. The automation to deploy OHDSIonAWS is open-source, and can be customized to include your

[5] https://github.com/OHDSI/Broadsea
[6] https://www.docker.com/
[7] https://github.com/OHDSI/OHDSI-in-a-Box
[8] https://github.com/OHDSI/OHDSIonAWS

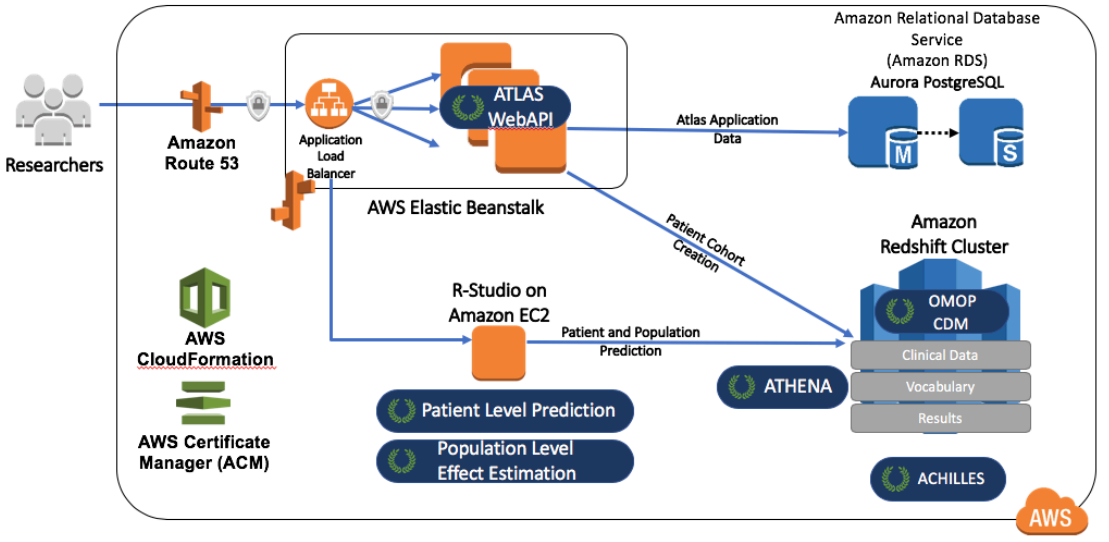



Figure 9.11: The Amazon Web Services architecure for OHDSIonAWS.

organization's management tools and best practices. The architecture for OHDSIonAWS is depicted in Figure 9.11.

## 9.7 Summary

- TODO: add

## 9.8 Exercises

Todo

# Chapter 10

# SQL and R

*Chapter leads: Martijn Schuemie & Peter Rijnbeek*

The Common Data Model (CDM) is a relational database model (all data is represented as records in tables that have fields), which means that the data will typically be stored in a relational database using a software platform like PostgreSQL, Oracle, or Microsoft SQL Server. The various OHDSI tools such as ATLAS and the Methods Library work by querying the database behind the scene, but we can also query the database directly ourselves if we have appropriate access rights. The main reason to do this is to perform analyses that currently are not supported by any existing tool. However, directly querying the database also comes with greater risk of making mistakes, as the OHDSI tools are often designed to help guide the user to appropriate analysis of the data, and direct queries do not provide such guidance.

The standard language for querying relational databases is SQL (Structured Query Language), which can be used both to query the database as well as to make changes to the data. Although the basic commands in SQL are indeed standard, meaning the same across software platforms, each platform has its own dialect, with subtle changes. For example, to retrieve the top 10 rows of the PERSON table on SQL Server one would type:

```
SELECT TOP 10 * FROM person;
```

Whereas the same query on PostgreSQL would be:

```
SELECT * FROM person LIMIT 10;
```

In OHDSI, we would like to be agnostic to the specific dialect a platform uses; We would like to 'speak' the same SQL language across all OHDSI databases. For this reason OHDSI developed the SqlRender package, an R package that can translate from one standard dialect to any of the supported dialects that will be discussed later in this chapter. This standard dialect - **OHDSI SQL** - is mainly a subset of the SQL Server SQL dialect. The example SQL statements provided throughout this chapter will all use OHDSI SQL.

Each database platform also comes with its own software tools for querying the database using SQL. In OHDSI we developed the DatabaseConnector package, one R package that can connect to many database platforms. DatabaseConnector will also be discussed later in this chapter.

So although one can query a database that conforms to the CDM without using any OHDSI tools, the recommended path is to use the DatabaseConnector and SqlRender packages. This allows queries that are developed at one site to be used at any other site without modification. R itself also immediately provides features to further analyse the data extracted from the database, such as performing statistical analyses and generating (interactive) plots.

In this chapter we assume the reader has a basic understanding of SQL. We first review how to use SqlRender and DatabaseConnector. If the reader does not intend to use these packages these sections can be skipped. In Section 10.3 we discuss how to use SQL (in this case OHDSI SQL) to query the CDM. The following section highlight how to use the OHDSI Standardized Vocabulary when querying the CDM. We highlight the QueryLibrary, a collection of commonly-used queries against the CDM that is publicly available. We close this chapter with an example study estimating incidence rates, and implement this study using SqlRender and DatabaseConnector.

## 10.1 SqlRender

The SqlRender package is available on CRAN (the Comprehensive R Archive Network), and can therefore be installed using:

```
install.packages("SqlRender")
```

SqlRender supports a wide array of technical platforms including traditional database systems (PostgreSQL, Microsoft SQL Server, SQLite, and Oracle), parallel data warehouses (Microsoft APS, IBM Netezza, and Amazon RedShift), as well as Big Data platforms (Hadoop through Impala, and Google BigQuery). The R package comes with a package manual and a vignette that explores the full functionality. Here we describer some of the main features.

### 10.1.1 SQL parameterization

One of the functions of the package is to support parameterization of SQL. Often, small variations of SQL need to be generated based on some parameters. SqlRender offers a simple markup syntax inside the SQL code to allow parameterization. Rendering the SQL based on parameter values is done using the `render()` function.

#### Substituting parameter values

The `@` character can be used to indicate parameter names that need to be exchange for actual parameter values when rendering. In the following example, a variable called `a` is mentioned in the SQL. In the call to the render function the value of this parameter is defined:

```
sql <- "SELECT * FROM concept WHERE concept_id = @a;"
render(sql, a = 123)
```

```
## [1] "SELECT * FROM concept WHERE concept_id = 123;"
```

Note that, unlike the parameterization offered by most database management systems, it is just as easy to parameterize table or field names as values:

```
sql <- "SELECT * FROM @x WHERE person_id = @a;"
render(sql, x = "observation", a = 123)
```

```
## [1] "SELECT * FROM observation WHERE person_id = 123;"
```

The parameter values can be numbers, strings, booleans, as well as vectors, which are converted to comma-delimited lists:

```
sql <- "SELECT * FROM concept WHERE concept_id IN (@a);"
render(sql, a = c(123, 234, 345))
```

```
## [1] "SELECT * FROM concept WHERE concept_id IN (123,234,345);"
```

**If-then-else**

Sometimes blocks of codes need to be turned on or off based on the values of one or more parameters. This is done using the `{Condition} ? {if true} : {if false}` syntax. If the *condition* evaluates to true or 1, the *if true* block is used, else the *if false* block is shown (if present).

```
sql <- "SELECT * FROM cohort {@x} ? {WHERE subject_id = 1}"
render(sql, x = FALSE)
```

```
## [1] "SELECT * FROM cohort "
```

```
render(sql, x = TRUE)
```

```
## [1] "SELECT * FROM cohort WHERE subject_id = 1"
```

Simple comparisons are also supported:

```
sql <- "SELECT * FROM cohort {@x == 1} ? {WHERE subject_id = 1};"
render(sql, x = 1)
```

```
## [1] "SELECT * FROM cohort WHERE subject_id = 1;"
```

```
render(sql, x = 2)
```

```
## [1] "SELECT * FROM cohort ;"
```

As well as the `IN` operator:

```
sql <- "SELECT * FROM cohort {@x IN (1,2,3)} ? {WHERE subject_id = 1};"
render(sql, x = 2)
```

```
## [1] "SELECT * FROM cohort WHERE subject_id = 1;"
```

### 10.1.2 Translation to other SQL dialects

Another function of the SqlRender package is to translate from OHDSI SQL to other SQL dialects. For example:

```
sql <- "SELECT TOP 10 * FROM person;"
translate(sql, targetDialect = "postgresql")
```

```
## [1] "SELECT  * FROM person LIMIT 10;"
```

The `targetDialect` parameter can have the following values: "oracle", "postgresql", "pdw", "redshift", "impala", "netezza", "bigquery", "sqlite", and "sql server".

There are limits to what SQL functions and constructs can be translated properly, both because only a limited set of translation rules have been implemented in the package, but also some SQL features do not have an equivalent in all dialects. This is the primary reason why OHDSI SQL was developed as its own, new SQL dialect. However, whenever possible we have kept to the SQL Server syntax to avoid reinventing the wheel.

Despite our best effords, there are quite a few things to consider when writing OHDSI SQL that will run without error on all supported platforms. In what follows we discuss these considerations in detail.

**Functions and structures supported by translate**

These SQL Server functions have been tested and were found to be translated correctly to the various dialects:

Table 10.1: Functions supported by translate.

| Function | Function | Function |
|---|---|---|
| ABS | EXP | RAND |
| ACOS | FLOOR | RANK |

| Function | Function | Function |
|---|---|---|
| ASIN | GETDATE | RIGHT |
| ATAN | HASHBYTES* | ROUND |
| AVG | ISNULL | ROW_NUMBER |
| CAST | ISNUMERIC | RTRIM |
| CEILING | LEFT | SIN |
| CHARINDEX | LEN | SQRT |
| CONCAT | LOG | SQUARE |
| COS | LOG10 | STDEV |
| COUNT | LOWER | SUM |
| COUNT_BIG | LTRIM | TAN |
| DATEADD | MAX | UPPER |
| DATEDIFF | MIN | VAR |
| DATEFROMPARTS | MONTH | YEAR |
| DATETIMEFROMPARTS | NEWID | |
| DAY | PI | |
| EOMONTH | POWER | |

* Requires special privileges on Oracle. Has no equivalent on SQLite.

Similarly, many SQL syntax structures are supported. Here is a non-exhaustive lists of expressions that we know will translate well:

```
-- Simple selects:
SELECT * FROM table;

-- Selects with joins:
SELECT * FROM table_1 INNER JOIN table_2 ON a = b;

-- Nested queries:
SELECT * FROM (SELECT * FROM table_1) tmp WHERE a = b;

-- Limiting to top rows:
SELECT TOP 10 * FROM table;

-- Selecting into a new table:
SELECT * INTO new_table FROM table;

-- Creating tables:
CREATE TABLE table (field INT);

-- Inserting verbatim values:
INSERT INTO other_table (field_1) VALUES (1);

-- Inserting from SELECT:
INSERT INTO other_table (field_1) SELECT value FROM table;
```

```
-- Simple drop commands:
DROP TABLE table;

-- Drop table if it exists:
IF OBJECT_ID('ACHILLES_analysis', 'U') IS NOT NULL
  DROP TABLE ACHILLES_analysis;

-- Drop temp table if it exists:
IF OBJECT_ID('tempdb..#cohorts', 'U') IS NOT NULL
  DROP TABLE #cohorts;

-- Common table expressions:
WITH cte AS (SELECT * FROM table) SELECT * FROM cte;

-- OVER clauses:
SELECT ROW_NUMBER() OVER (PARTITION BY a ORDER BY b)
  AS "Row Number" FROM table;

-- CASE WHEN clauses:
SELECT CASE WHEN a=1 THEN a ELSE 0 END AS value FROM table;

-- UNIONs:
SELECT * FROM a UNION SELECT * FROM b;

-- INTERSECTIONs:
SELECT * FROM a INTERSECT SELECT * FROM b;

-- EXCEPT:
SELECT * FROM a EXCEPT SELECT * FROM b;
```

**String concatenation**

String concatenation is one area where SQL Server is less specific than other dialects. In SQL Server, one would write `SELECT first_name + ' ' + last_name AS full_name FROM table`, but this should be `SELECT first_name || ' ' || last_name AS full_name FROM table` in PostgreSQL and Oracle. SqlRender tries to guess when values that are being concatenated are strings. In the example above, because we have an explicit string (the space surrounded by single quotation marks), the translation will be correct. However, if the query had been `SELECT first_name + last_name AS full_name FROM table`, SqlRender would have had no clue the two fields were strings, and would incorrectly leave the plus sign. Another clue that a value is a string is an explicit cast to VARCHAR, so `SELECT last_name + CAST(age AS VARCHAR(3)) AS full_name FROM table` would also be translated correctly. To avoid ambiguity altogether, it is probable best to use the `CONCAT()` function to concatenate two or more strings.

**Table aliases and the AS keyword**

Many SQL dialects allow the use of the `AS` keyword when defining a table alias, but will also work

fine without the keyword. For example, both these SQL statements are fine for SQL Server, PostgreSQL, RedShift, etc.:

```
-- Using AS keyword
SELECT *
FROM my_table AS table_1
INNER JOIN (
  SELECT * FROM other_table
) AS table_2
ON table_1.person_id = table_2.person_id;

-- Not using AS keyword
SELECT *
FROM my_table table_1
INNER JOIN (
  SELECT * FROM other_table
) table_2
ON table_1.person_id = table_2.person_id;
```

However, Oracle will throw an error when the `AS` keyword is used. In the above example, the first query will fail. It is therefore recommended to not use the `AS` keyword when aliasing tables. (Note: we can't make SqlRender handle this, because it can't easily distinguish between table aliases where Oracle doesn't allow `AS` to be used, and field aliases, where Oracle requires `AS` to be used.)

**Temp tables**

Temp tables can be very useful to store intermediate results, and when used correctly can be used to dramatically improve performance of queries. On most database platforms temp tables have very nice properties: they're only visible to the current user, are automatically dropped when the session ends, and can be created even when the user has no write access. Unfortunately, in Oracle temp tables are basically permanent tables, with the only difference that the data inside the table is only visible to the current user. This is why, in Oracle, SqlRender will try to emulate temp tables by

1. Adding a random string to the table name so tables from different users will not conflict.
2. Allowing the user to specify the schema where the temp tables will be created.

For example:

```
sql <- "SELECT * FROM #children;"
translate(sql, targetDialect = "oracle", oracleTempSchema = "temp_schema")
```

```
## [1] "SELECT * FROM temp_schema.rkq8xmxnchildren ;"
```

Note that the user will need to have write privileges on `temp_schema`.

Also note that because Oracle has a limit on table names of 30 characters, **temp table names are only allowed to be at most 22 characters long** because else the name will become too long after appending the session ID.

Furthermore, remember that temp tables are not automatically dropped on Oracle, so you will need to explicitly `TRUNCATE` and `DROP` all temp tables once you're done with them to prevent orphan tables accumulating in the Oracle temp schema.

**Implicit casts**

One of the few points where SQL Server is less explicit than other dialects is that it allows implicit casts. For example, this code will work on SQL Server:

```
CREATE TABLE #temp (txt VARCHAR);

INSERT INTO #temp
SELECT '1';

SELECT * FROM #temp WHERE txt = 1;
```

Even though `txt` is a VARCHAR field and we are comparing it with an integer, SQL Server will automatically cast one of the two to the correct type to allow the comparison. In contrast, other dialects such as PostgreSQL will throw an error when trying to compare a VARCHAR with an INT.

You should therefore always make casts explicit. In the above example, the last statement should be replaced with either

```
SELECT * FROM #temp WHERE txt = CAST(1 AS VARCHAR);
```

or

```
SELECT * FROM #temp WHERE CAST(txt AS INT) = 1;
```

**Case sensitivity in string comparisons**

Some DBMS platforms such as SQL Server always perform string comparisons in a case-insensitive way, while others such as PostgreSQL are always case sensitive. It is therefore recommended to always assume case-sensitive comparisons, and to explicitly make comparisons case-insensitive when unsure about the case. For example, instead of

```
SELECT * FROM concept WHERE concep_class_id = 'Clinical Finding'
```

it is preferred to use

```
SELECT * FROM concept WHERE LOWER(concep_class_id) = 'clinical finding'
```

**Schemas and databases**

In SQL Server, tables are located in a schema, and schemas reside in a database. For example, `cdm_data.dbo.person` refers to the `person` table in the `dbo` schema in the `cdm_data` database.

In other dialects, even though a similar hierarchy often exists they are used very differently. In SQL Server, there is typically one schema per database (often called `dbo`), and users can easily use data in different databases. On other platforms, for example in PostgreSQL, it is not possible to use data across databases in a single session, but there are often many schemas in a database. In PostgreSQL one could say that the equivalent of SQL Server's database is the schema.

We therefore recommend concatenating SQL Server's database and schema into a single parameter, which we typically call `@databaseSchema`. For example, we could have the parameterized SQL

```
SELECT * FROM @databaseSchema.person
```

where on SQL Server we can include both database and schema names in the value: `databaseSchema = "cdm_data.dbo"`. On other platforms, we can use the same code, but now only specify the schema as the parameter value: `databaseSchema = "cdm_data"`.

The one situation where this will fail is the `USE` command, since `USE cdm_data.dbo;` will throw an error. It is therefore preferred not to use the `USE` command, but always specify the database / schema where a table is located.

**Debugging parameterized SQL**

Debugging parameterized SQL can be a bit complicated; Only the rendered SQL can be tested against a database server, but changes to the code should be made in the parameterized (pre-rendered) SQL.

A Shiny app is included in the SqlRender package for interactively editing source SQL and generating rendered and translated SQL. The app can be started using:

```
launchSqlRenderDeveloper()
```

Which will open the default browser with the app shown in Figure 10.1. The app is also publicly available on the web[1].

In the app you can enter OHDSI SQL, select the target dialect as well as provide values for the parameters that appear in your SQL, and the translation will automatically appear at the bottom.

## 10.2 DatabaseConnector

DatabaseConnector is an R package for connecting to various database platforms using Java's JDBC drivers. The DatabaseConnector package is available on CRAN (the Comprehensive R Archive Network), and can therefore be installed using:

```
install.packages("DatabaseConnector")
```

[1] http://data.ohdsi.org/SqlDeveloper/

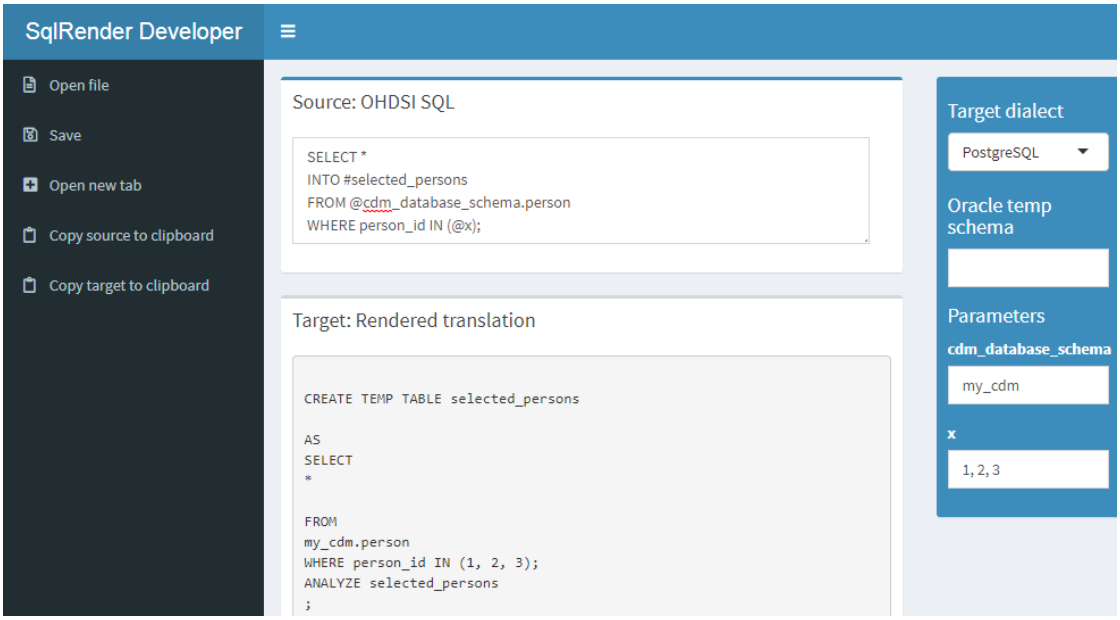



Figure 10.1: The SqlDeveloper Shiny app.

DatabaseConnector supports a wide array of technical platforms including traditional database systems (PostgreSQL, Microsoft SQL Server, SQLite, and Oracle), parallel data warehouses (Microsoft APS, IBM Netezza, and Amazon RedShift), as well as Big Data platforms (Hadoop through Impala, and Google BigQuery). The package already contains most drivers, but because of licensing reasons the drivers for BigQuery, Netezza and Impala are not included but must be obtained by the user. Type `?jdbcDrivers` for instructions on how to download these drivers. Once downloaded, you can use the `pathToDriver` argument of the `connect`, `dbConnect`, and `createConnectionDetails` functions.

### 10.2.1 Creating a connection

To connect to a database a number of details need to be specified, such as the database platform, the location of the server, the user name, and password. We can call the `connect` function and specify these details directly:

```
conn <- connect(dbms = "postgresql",
                server = "localhost/postgres",
                user = "joe",
                password = "secret",
                schema = "cdm")
```

```
## Connecting using PostgreSQL driver
```

See `?connect` for information on which details are required for each platform. Don't forget to close

any connection afterwards:

```
disconnect(conn)
```

Note that, instead of providing the server name, it is also possible to provide the JDBC connection string if this is more convenient:

```
connString <- "jdbc:postgresql://localhost:5432/postgres"
conn <- connect(dbms = "postgresql",
                connectionString = connString,
                user = "joe",
                password = "secret",
                schema = "cdm")
```

```
## Connecting using PostgreSQL driver
```

Sometimes we may want to first specify the connection details, and defer connecting until later. This may be convenient for example when the connection is established inside a function, and the details need to be passed as an argument. We can use the `createConnectionDetails` function for this purpose:

```
details <- createConnectionDetails(dbms = "postgresql",
                                   server = "localhost/postgres",
                                   user = "joe",
                                   password = "secret",
                                   schema = "cdm")
conn <- connect(details)
```

```
## Connecting using PostgreSQL driver
```

### 10.2.2 Querying

The main functions for querying database are the `querySql` and `executeSql` functions. The difference between these functions is that `querySql` expects data to be returned by the database, and can handle only one SQL statement at a time. In contrast, `executeSql` does not expect data to be returned, and accepts multiple SQL statements in a single SQL string.

Some examples:

```
querySql(conn, "SELECT TOP 3 * FROM person")
```

```
##   PERSON_ID GENDER_CONCEPT_ID YEAR_OF_BIRTH
## 1         1              8507          1975
## 2         2              8507          1976
## 3         3              8507          1977
```

```
executeSql(conn, "TRUNCATE TABLE foo; DROP TABLE foo;")
```

Both function provide extensive error reporting: When an error is thrown by the server, the error message and the offending piece of SQL are written to a text file to allow better debugging. The `executeSql` function also by default shows a progress bar, indicating the percentage of SQL statements that has been executed. If those attributes are not desired, the package also offers the `lowLevelQuerySql` and `lowLevelExecuteSql` functions.

### 10.2.3 Querying using ffdf objects

Sometimes the data to be fetched from the database is too large to fit into memory. As mentioned in Section 9.4.2, in such a case we can use the `ff` package to store R data objects on file, and use them as if they are available in memory. `DatabaseConnector` can download data directly into ffdf objects:

```
x <- querySql.ffdf(conn, "SELECT * FROM person")
```

Where x is now an ffdf object.

### 10.2.4 Querying different platforms using the same SQL

The following convenience functions are available that first call the `render` and `translate` functions in the SqlRender package: `renderTranslateExecuteSql`, `renderTranslateQuerySql`, `renderTranslateQuerySql.ffdf`. For example:

```
x <- renderTranslateQuerySql(conn,
                             sql = "SELECT TOP 10 * FROM @schema.person",
                             schema = "cdm_synpuf")
```

Note that the SQL Server-specific 'TOP 10' syntax will be translated to for example 'LIMIT 10' on PostgreSQL, and that the SQL parameter `@schema` will be instantiated with the provided value 'cdm_synpuf'.

### 10.2.5 Inserting tables

Although it is also possible to insert data in the database by sending SQL statements using the `executeSql` function, it is often more convenient and faster (due to some optimization) to use the `insertTable` function:

```
data(mtcars)
insertTable(conn, "mtcars", mtcars, createTable = TRUE)
```

In this example, we're uploading the mtcars data frame to a table called 'mtcars' on the server, which will be automatically created.

## 10.3 Querying the CDM

In the following examples we use OHDSI SQL to query a database that adheres to the CDM. These queries use `@cdm` to denote the database schema where the data in CDM can be found.

We can start by just querying how many people are in the database:

```
SELECT COUNT(*) AS person_count FROM @cdm.person;
```

| PERSON_COUNT |
|---|
| 26299001 |

Or perhaps we're interested in the average length of an observation period:

```
SELECT AVG(DATEDIFF(DAY,
                    observation_period_start_date,
                    observation_period_end_date) / 365.25) AS num_years
FROM @cdm.observation_period;
```

| NUM_YEARS |
|---|
| 1.980803 |

We can join tables to produce additional statistics. A join combines fields from multiple tables, typically by requiring specific fields in the tables to have the same value. For example, here we join the PERSON table to the OBSERVATION_PERIOD table on the person_id fields in both tables. In other words, the result of the join is a new table-like set that has all the fields of the two tables, but in all rows the person_id fields from the two tables must have the same value. We can now for example compute the maximum age at observation end by using the observation_period_end_date field from the OBSERVATION_PERIOD table together with the year_of_birth field of the PERSON table:

```
SELECT MAX(YEAR(observation_period_end_date) -
           year_of_birth) AS max_age
FROM @cdm.person
INNER JOIN @cdm.observation_period
  ON person.person_id = observation_period.person_id;
```

| MAX_AGE |
|---|
| 90 |

A much more complicated query is needed to determine the distribution of age at the start of observation. In this query, we first join the PERSON to the OBSERVATION_PERIOD table to compute age at start of observetion. We also compute the ordering for this joined set based on age, and store it as order_nr. Because we want to use the result of this join multiple times, we define it as a common table expression (CTE) (defined using `WITH ... AS`) that we call "ages", meaning We can refer to ages as if it is an existing table. We count the number of rows in ages to produce "n", and then for each quantile find the minimum age where the order_nr is smaller than the fraction times n. For example, median we use the minimum age where $order\_nr < .50 * n$. The minimum and maximum age are computed separately:

```
WITH ages
AS (
    SELECT age,
        ROW_NUMBER() OVER (
            ORDER BY age
            ) order_nr
    FROM (
        SELECT YEAR(observation_period_start_date) - year_of_birth AS age
        FROM @cdm.person
        INNER JOIN @cdm.observation_period
            ON person.person_id = observation_period.person_id
        ) age_computed
    )
SELECT MIN(age) AS min_age,
    MIN(CASE
            WHEN order_nr < .25 * n
                THEN 9999
            ELSE age
            END) AS q25_age,
    MIN(CASE
            WHEN order_nr < .50 * n
                THEN 9999
            ELSE age
            END) AS median_age,
    MIN(CASE
            WHEN order_nr < .75 * n
                THEN 9999
            ELSE age
            END) AS q75_age,
    MAX(age) AS max_age
FROM ages
CROSS JOIN (
    SELECT COUNT(*) AS n
```

```
    FROM ages
    ) population_size;
```

| MIN_AGE | Q25_AGE | MEDIAN_AGE | Q75_AGE | MAX_AGE |
|---|---|---|---|---|
| 0 | 6 | 17 | 34 | 90 |

More complex computations can also be performed in R instead of using SQL. For example, we can get the same answer using this R code:

```
sql <- "SELECT YEAR(observation_period_start_date) -
               year_of_birth AS age
FROM @cdm.person
INNER JOIN @cdm.observation_period
  ON person.person_id = observation_period.person_id;"
age <- renderTranslateQuerySql(conn, sql, cdm = "cdm")
quantile(age[, 1], c(0, 0.25, 0.5, 0.75, 1))
```

```
##   0%  25%  50%  75% 100%
##    0    6   17   34   90
```

Here we compute age on the server, download all ages, and then compute the age distribution. However, this requires millions of rows of data to be downloaded from the database server, and is therefore not very efficient. You will need to decide on a case-by-case basis whether a computation is best performed in SQL or in R.

Queries can use the source values in the CDM. For example, we can retrieve the top 10 most frequent condition source codes using:

```
SELECT TOP 10 condition_source_value,
  COUNT(*) AS code_count
FROM @cdm.condition_occurrence
GROUP BY condition_source_value
ORDER BY -COUNT(*);
```

| CONDITION_SOURCE_VALUE | CODE_COUNT |
|---|---|
| 4019 | 49094668 |
| 25000 | 36149139 |
| 78099 | 28908399 |
| 319 | 25798284 |
| 31401 | 22547122 |
| 317 | 22453999 |
| 311 | 19626574 |
| 496 | 19570098 |

| CONDITION_SOURCE_VALUE | CODE_COUNT |
|---|---|
| I10 | 19453451 |
| 3180 | 18973883 |

Here we grouped records in the CONDITION_OCCURRENCE table by values of the condition_source_value field, and counted the number of records in each group. We retrieve the condition_source_value and the count, and reverse-order it by the count.

## 10.4 Using the vocabulary when querying

Many operations require the vocabulary to be useful. The Vocabulary tables are part of the CDM, and are therefore available using SQL queries. Querying the Vocabulary is already described at length in Chapter 6. Here we show how queries against the Vocabulary can be combined with queries against the CDM. Many fields in the CDM contain concept IDs which can be resolved using the CONCEPT table. For example, we may wish to count the number of persons in the database stratified by gender, and it would be convenient to resolve the GENDER_CONCEPT_ID field to a concept name:

```
SELECT COUNT(*) AS subject_count,
  concept_name
FROM @cdm.person
INNER JOIN @cdm.concept
  ON person.gender_concept_id = concept.concept_id
GROUP BY concept_name;
```

| SUBJECT_COUNT | CONCEPT_NAME |
|---|---|
| 14927548 | FEMALE |
| 11371453 | MALE |

A very powerful feature of the Vocabulary is its hierarchy. A very common query looks for a specific concept *and all of its descendants*. For example, image we wish to count the number of prescriptions containing the ingredient ibuprofen:

```
SELECT COUNT(*) AS prescription_count
FROM @cdm.drug_exposure
INNER JOIN @cdm.concept_ancestor
  ON drug_concept_id = descendant_concept_id
INNER JOIN @cdm.concept ingredient
  ON ancestor_concept_id = ingredient.concept_id
WHERE LOWER(ingredient.concept_name) = 'ibuprofen'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S';
```

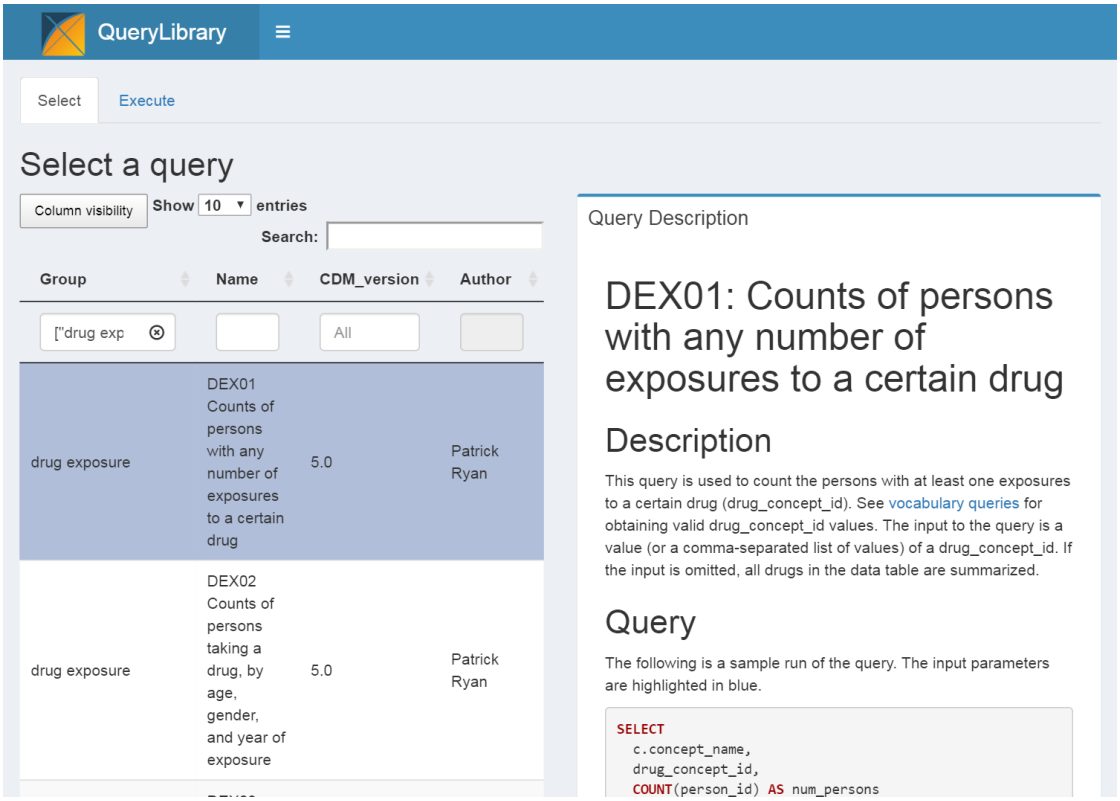



Figure 10.2: QueryLibrary: a library of SQL queries against the CDM.

| PRESCRIPTION_COUNT |
|---|
| 26871214 |

## 10.5 QueryLibrary

TODO: update this section when QueryLibrary is finalized.

QueryLibrary is a library of commonly-used SQL queries for the CDM. It is available as an online application [2] shown in Figure 10.2, and as an R package[3].

The purpose of the library is to help new users learn how to query the CDM. The queries in the library have been reviewed and approved by the OHDSI community. The query library is primarily intended for training purposes, but is also a valuable resource for experienced users.

[2] http:/data.ohdsi.org/QueryLibrary
[3] https://github.com/OHDSI/QueryLibrary

The QueryLibrary makes use of SqlRender to output the queries in the SQL dialect of choice. Users can also specify the CDM database schema, vocabulary database schema (if separate), and the Oracle temp schema (if needed), so the queries will be automatically rendered with these settings.

## 10.6 Designing a simple study

### 10.6.1 Problem definition

Angioedema is a well-known side-effect of ACE inhibitors (ACEi). Slater et al. (1988) estimate the incidence rate of angioedema in the first week of ACEi treatment to be one case per 3,000 patients per week. Here we seek to replicate this finding, and stratify by age and gender, thus answering the question

> What is the rate of angioedema in the first week following ACEi treatment initiation, stratified by age and gender?

### 10.6.2 Exposure

We'll define exposure as first exposure to a drug containing an ingredient in the ACEi class. By first we mean no earlier exposure to any ingredient in the class. We require 365 days of continuous observation time prior to the first exposure.

### 10.6.3 Outcome

We define angioedema as any occurrence of an angioedema diagnose code during an inpatient or emergency room (ER) visit.

### 10.6.4 Time-at-risk

We will compute the incidence rate in the first week following treatment initiation, irrespective of whether patients were exposed for the full week.

## 10.7 Implementing the study using SQL and R

Although we are not bound to any of the OHDSI tool conventions, it is helpful to follow the same principles. In this case, we will use SQL to populate a cohort table, similarly to how the OHDSI tools work. The COHORT table is defined in the CDM, and has a predefined set of fields that we will also use. We first must create the COHORT table in a database schema where we have write access, which likely is not the same as the database schema that holds the data in CDM format.

```
library(DatabaseConnector)
conn <- connect(dbms = "postgresql",
                server = "localhost/postgres",
                user = "joe",
                password = "secret")
cdmDbSchema <- "cdm"
cohortDbSchema <- "scratch"
cohortTable <- "my_cohorts"

sql <- "
CREATE TABLE @cohort_db_schema.@cohort_table (
  cohort_definition_id INT,
  cohort_start_date DATE,
  cohort_end_date DATE,
  subject_id BIGINT
);
"
renderTranslateExecuteSql(conn, sql,
                          cohort_db_schema = cohortDbSchema,
                          cohort_table = cohortTable)
```

Here we have parameterized the database schema and table names, so we can easily adapt them to different environments. The result is an empty table on the database server.

### 10.7.1 Exposure cohort

Next we create our exposure cohort, and insert it into our COHORT table:

```
sql <- "
INSERT INTO @cohort_db_schema.@cohort_table (
  cohort_definition_id,
  cohort_start_date,
  cohort_end_date,
  subject_id
)
SELECT 1 AS cohort_definition_id,
  cohort_start_date,
  cohort_end_date,
  subject_id
FROM (
  SELECT MIN(drug_exposure_start_date) AS cohort_start_date,
    MIN(drug_exposure_end_date) AS cohort_end_date,
    person_id AS subject_id
  FROM @cdm_db_schema.drug_exposure
  INNER JOIN @cdm_db_schema.concept_ancestor
    ON drug_concept_id = descendant_concept_id
  WHERE ancestor_concept_id IN (1335471, 1340128, 1341927,
```

```
    1363749, 1308216, 1310756, 1373225, 1331235, 1334456,
    1342439) -- ACE inhibitors
  GROUP BY person_id
) first_exposure
INNER JOIN @cdm_db_schema.observation_period
  ON subject_id = person_id
    AND observation_period_start_date < cohort_start_date
    AND observation_period_end_date > cohort_start_date
WHERE DATEDIFF(DAY,
               observation_period_start_date,
               cohort_start_date) >= 365;
"

renderTranslateExecuteSql(conn, sql,
                          cohort_db_schema = cohortDbSchema,
                          cohort_table = cohortTable,
                          cdm_db_schema = cdmDbSchema)
```

Here we use the DRUG_EXPOSURE table, and join it the CONCEPT_ANCESTOR table, thus allowing us to search for the ACEi ingredients and all their descendants, i.e. all drugs containing an ACEi. We take the first drug exposure per person, and then join to the OBSERVATION_PERIOD table, and because a person can have several observation periods we must make sure we only join to the period containing the drug exposure. We then require at least 365 days between the observation_period_start_date and the cohort_start_date.

### 10.7.2 Outcome cohort

Finally, we must create our outcome cohort:

```
sql <- "
INSERT INTO @cohort_db_schema.@cohort_table (
 cohort_definition_id,
 cohort_start_date,
 cohort_end_date,
subject_id
)
SELECT 2 AS cohort_definition_id,
  cohort_start_date,
  cohort_end_date,
  subject_id
FROM (
  SELECT DISTINCT person_id AS subject_id,
    condition_start_date AS cohort_start_date,
    condition_end_date AS cohort_end_date
  FROM @cdm_db_schema.condition_occurrence
  INNER JOIN @cdm_db_schema.concept_ancestor
```

```
    ON condition_concept_id = descendant_concept_id
  WHERE ancestor_concept_id = 432791 -- Angioedema
) distinct_occurrence
INNER JOIN @cdm_db_schema.visit_occurrence
  ON subject_id = person_id
  AND visit_start_date <= cohort_start_date
  AND visit_end_date >= cohort_start_date
WHERE visit_concept_id IN (262, 9203,
    9201) -- Inpatient or ER;
"

renderTranslateExecuteSql(conn, sql,
                          cohort_db_schema = cohortDbSchema,
                          cohort_table = cohortTable,
                          cdm_db_schema = cdmDbSchema)
```

Here we join the CONDITION_OCCURRENCE table to the CONCEPT_ANCESTOR table to find all occurrences of angioedema or any of its descendants. We use DISTINCT to make sure we only select one record per day, as we believe multiple angioedema diagnoses on the same day are more likely to be the same occurrence rather than multiple angioedema events. We join these occurrences to the VISIT_OCCURRENCE table to ensure the diagnose was made in and inpatient or ER setting.

### 10.7.3 Incidence rate calculation

Now that our cohorts are in place, we can compute the incidence rate, stratified by age and gender:

```
sql <- "
WITH tar AS (
  SELECT concept_name AS gender,
    FLOOR((YEAR(cohort_start_date) -
          year_of_birth) / 10) AS age,
    subject_id,
    cohort_start_date,
    CASE WHEN DATEADD(DAY, 7, cohort_start_date) >
      observation_period_end_date
    THEN observation_period_end_date
    ELSE DATEADD(DAY, 7, cohort_start_date)
    END AS cohort_end_date
  FROM @cohort_db_schema.@cohort_table
  INNER JOIN @cdm_db_schema.observation_period
    ON subject_id = observation_period.person_id
      AND observation_period_start_date < cohort_start_date
      AND observation_period_end_date > cohort_start_date
  INNER JOIN @cdm_db_schema.person
    ON subject_id = person.person_id
  INNER JOIN @cdm_db_schema.concept
```

```
    ON gender_concept_id = concept_id
  WHERE cohort_definition_id = 1 -- Exposure
)
SELECT days.gender,
    days.age,
    days,
    CASE WHEN events IS NULL THEN 0 ELSE events END AS events
FROM (
  SELECT gender,
    age,
    SUM(DATEDIFF(DAY, cohort_start_date,
      cohort_end_date)) AS days
  FROM tar
  GROUP BY gender,
    age
) days
LEFT JOIN (
  SELECT gender,
      age,
      COUNT(*) AS events
  FROM tar
  INNER JOIN @cohort_db_schema.@cohort_table angioedema
    ON tar.subject_id = angioedema.subject_id
      AND tar.cohort_start_date <= angioedema.cohort_start_date
      AND tar.cohort_end_date >= angioedema.cohort_start_date
  WHERE cohort_definition_id = 2 -- Outcome
  GROUP BY gender,
    age
) events
ON days.gender = events.gender
  AND days.age = events.age;
"

results <- renderTranslateQuerySql(conn, sql,
                                   cohort_db_schema = cohortDbSchema,
                                   cohort_table = cohortTable,
                                   cdm_db_schema = cdmDbSchema,
                                   snakeCaseToCamelCase = TRUE)
```

We first create "tar", a CTE that contains all exposures with the appropriate time-at-risk. Note that we truncate the time-at-risk at the observation_period_end_date. We also compute the age in 10-year bins, and identify the gender. The advantage of using a CTE is that we can use the same set of intermediate results several times in a query. In this case we use it to count the total amount of time-at-risk, as well as the number of angioedema events that occur during the time-at-risk.

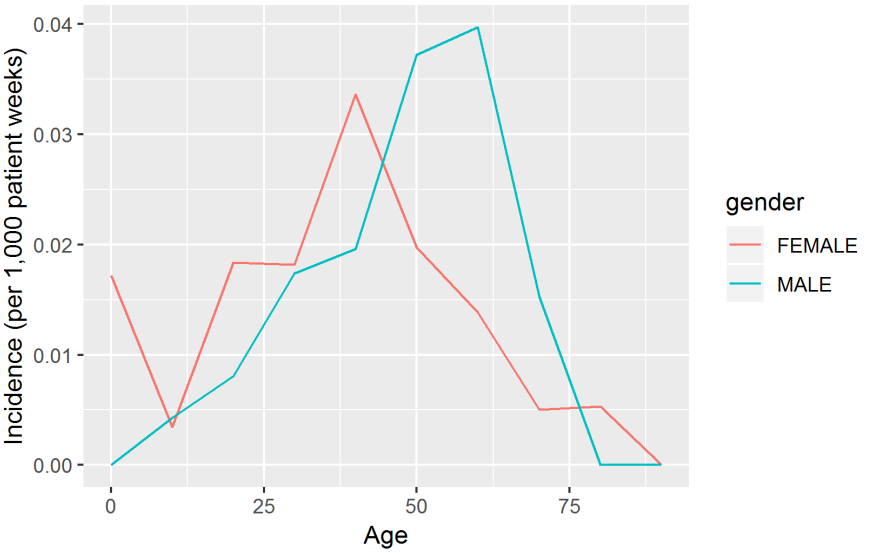
We use `snakeCaseToCamelCase = TRUE` because in SQL we tend to use snake_case for field names (because SQL in case-insensitive), whereas in R we tend to use camelCase (because R is case-sensitive). The `results` data frame column names will now be in camelCase.

With the help of the ggplot2 package we can easily plot our results:

```
# Compute incidence rate (IR) :
results$ir <- 1000 * results$events / results$days / 7

# Fix age scale:
results$age <- results$age * 10

library(ggplot2)
ggplot(results, aes(x = age, y = ir, group = gender, color = gender)) +
  geom_line() +
  xlab("Age") +
  ylab("Incidence (per 1,000 patient weeks)")
```




### 10.7.4 Clean up

Don't forget to clean up the table we created, and to close the connection:

```
sql <- "
TRUNCATE TABLE @cohort_db_schema.@cohort_table;
DROP TABLE @cohort_db_schema.@cohort_table;
"
renderTranslateExecuteSql(conn, sql,
                          cohort_db_schema = cohortDbSchema,
                          cohort_table = cohortTable)

disconnect(conn)
```

### 10.7.5 Compatibility

Because we use OHDSI SQL together with DatabaseConnector and SqlRender throughout, the code we reviewed here will run on any database platform supported by OHDSI.

Note that for demonstration purposes we chose to create our cohorts using hand-crafted SQL. It would probably have been more convenient to construct cohort definition in ATLAS, and use the SQL generated by ATLAS to instantiate the cohorts. ATLAS also produced OHDSI SQL, and can therefore easily be used together with SqlRender and DatabaseConnector.

## 10.8 Summary

- **SQL** (Structured Query Language) is a standard language for querying databases, including those that conform to the Common Data Model (CDM).
- Different database platforms have different SQL dialects, and require different tools to query them.
- The **SqlRender** and **DatabaseConnector** R packages provide a unified way to query data in the CDM, allowing the same analysis code to be run in different environments without modificiation.
- By using R and SQL together we can implement custom analyses that are not supported by the OHDSI tools.
- The **QueryLibrary** provides a collection of re-usable SQL queries for the CDM.

## 10.9 Exercises

#### Prerequisites

For these exercises we assume R, R-Studio and Java have been installed as described in Section 9.4.5. Also required are the SqlRender, DatabaseConnector, and Eunomia packages, which can be installed using:

```
install.packages(c("SqlRender", "DatabaseConnector", "devtools"))
devtools::install_github("ohdsi/Eunomia")
```

The Eunomia package provides a simulated dataset in the CDM that will run inside your local R session. The connection details can be obtained using:

```
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
```

The CDM database schema is “main”.

**Exercise 10.1.** Using SQL and R, compute how many people are in the database.

**Exercise 10.2.** Using SQL and R, compute how many people have at least one prescription of celecoxib.

**Exercise 10.3.** Using SQL and R, compute how many diagnoses of gastrointestinal haemorrhage occur during exposure to celecoxib. (Hint: the concept ID for gastrointestinal haemorrhage is 192671.)

Suggested answers can be found in Appendix D.1.

# Chapter 11

# Building the building blocks: cohorts

Contributors: Kristin Kostka, Patrick Ryan, Jon Duke, Juan Banda & Joel Swerdel

Cohorts are used throughout OHDSI analytical tools and network studies as the primary building blocks for running high quality, systematic research. Cohort definitions vary from study to study depending on the research question of interest. Each cohort defines a specific way to represent a person with a condition or exposure using data in an observational health database. Thus, cohorts are an important component in documenting the methods of an observational research study.

The chapter serves to explain what is meant by creating and sharing cohort definitions, the methods for developing cohorts, and examples of how to build your own cohorts using ATLAS (see Chapter 9) and SQL queries against the Common Data Model (CDM).

## 11.1 Theory

**OHDSI Cohort Definition:** A cohort is defined as a set of persons who satisfy one or more inclusion criteria for a duration of time.

In many peer-reviewed scientific manuscripts, a cohort definition is suggested to be analogous to a codeset of specific clinical codes (e.g. ICD-9/ICD-10, NDC, HCPCS, etc). While codesets are an important piece to assembling a cohort, a cohort definition is not simply a codesets. A cohort definition requires logic for how to use the codeset in a criteria. A well documented cohort specifies how a patient enters a cohort, a patient exits a cohort and any additional inclusion criteria that impacts how to observe a patient's time-at-risk.

The term *cohort* is often interchanged with the term phenotype. The term *phenotype* is applied to patient characteristics inferred from electronic health record (EHR) data (Hripcsak citation). The goal is to draw conclusions about a target concept based on raw EHR data, claims data, or other clinically relevant data. Thus, a *cohort* is a set of persons who satisfy one or more

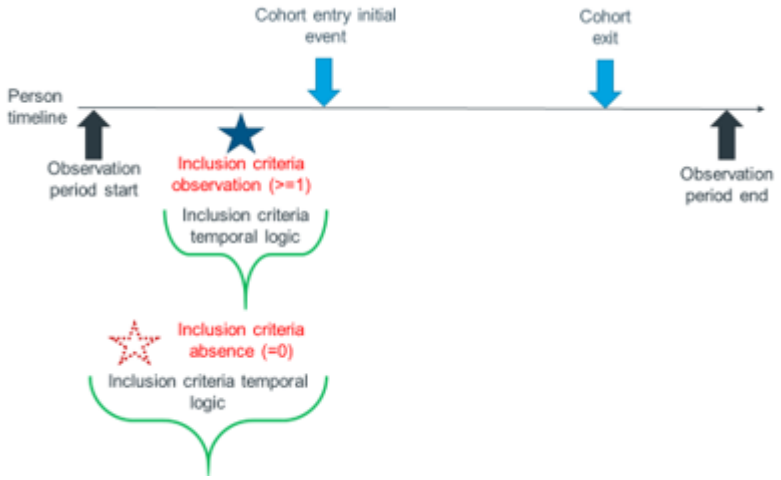



Figure 11.1: Cohort Creation

> inclusion criteria (a phenotype) for a duration of time. A cohort in itself is not a phenotype but a phenotype can be used to create a cohort.

There are two main approaches to constructing a cohort: **1) rules-based design** or **2) probabilistic design**. A rules-based cohort design relies heavily on the domain expertise of the individual designing the cohort to use their knowledge of the therapeutic area of interest to build rules to qualify potential cohort membership. Conversely, a probabilistic design mines already available data to identify and qualify potential cohort membership through machine-suggested patterns. The next sections will discuss these approaches in further detail.

### 11.1.1 Rules-based cohort design

A rules-based OHDSI cohort definition begins by an expert-consensus stating one or more inclusion criteria (e.g. "people with angioedema") in a specific duration of time (e.g. "who developed this condition within the last 6 months").

When creating a cohort definition, you need to ask yourself the following questions:

- *What initial event(s) define cohort entry?*
- *What inclusion criteria are applied to the initial events?*
- *What defines a person's cohort exit?*

To visualize the importance of these criteria, think of how this information comes together in a person's timeline. The OBSERVATION_PERIOD table creates the window for which we see the person in the data.

*Cohort entry criteria:* The cohort entry event can be one or many clinical attributes which dictate an individual patient's eligibility to be included in a cohort. Events are recorded time-stamped observa-

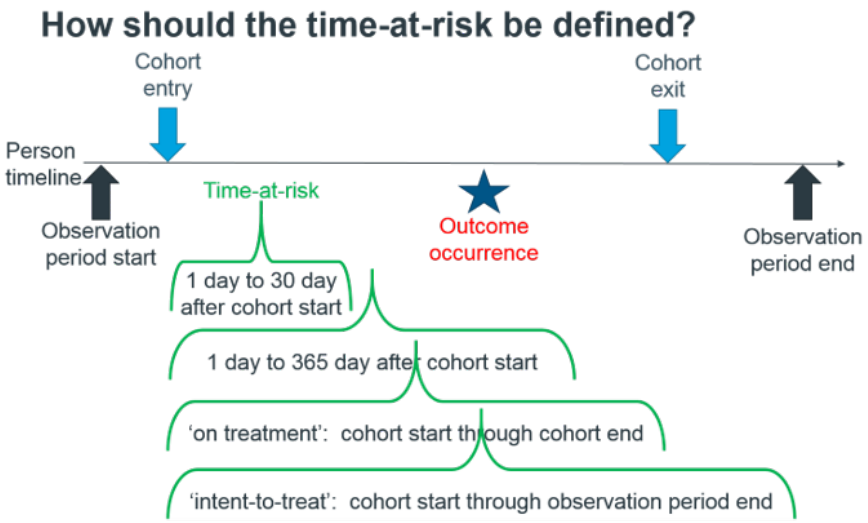



Figure 11.2: Time-at-Risk Construction

tions for the persons, such as drug exposures, conditions, procedures, measurements and visits. The event index date is set to be equal to the event start date. Initial events defined by a domain, concept set, and any domain-specific attributes required.

*Inclusion criteria:* The qualifying cohort will be defined as all persons who have an initial event and satisfy all qualifying inclusion criteria. Each inclusion criterion is defined by domain(s), concept set(s), domain-specific attributes, and the temporal logic relative to initial events. Each qualifying inclusion criterion can be evaluated to determine the impact of the criteria on the attrition of persons from the initial cohort.

*Cohort exit criteria:* The cohort exit event signifies when a person no longer qualifies for cohort membership. Cohort exit can be defined in multiple ways such as the end of the observation period, a fixed time interval relative to the initial entry event, the last event in a sequence of related observations (e.g. persistent drug exposure) or through other censoring of observation period. Cohort exit strategy will impact whether a person can belong to the cohort multiple times during different time intervals.

*Time-at-risk:* In order to interpret risk of a specific outcome, which will be defined as a separate cohort definition, it is necessary to know the length of time that applies. A time-at-risk criteria states the period of time in which the cohort must be in the data following the cohort entry criteria. The time-at-risk will vary based on whether you're observing an acute/short term trend or a chronic/long term trend.

In traditional study design, we would categorize time-at-risk for 'on treatment' as the entirety of the time between when a person meets cohort entry through the cohort exit criteria. An 'intent-to-treat' design would be the entirety of the time from the cohort start through the observation period ending (e.g. when the person leaves the data because they've switched physicians, insurance carriers, etc).

The use of these criteria may present a number of unique nuances to an OHDSI cohort including:

- One person may belong to multiple cohorts
- One person may belong to the same cohort at multiple different time periods
- One person may not belong to the same cohort multiple times during the same period of time
- One cohort may have zero or more members

Throughout the Book of OHDSI, we will detail how to address these consequences in your overall study design. In each respective methodology, we will discuss how you can configure a methods package to address how one person shows up in multiple cohorts being studied.

### 11.1.2 Probabilistic cohort design using APHRODITE

Rules-based cohort design are a popular method for assembling cohort definitions. However, assembling necessary expert consensus to create a study cohort can be prohibitively time consuming. Probabilistic cohort design is an alternative, machine-driven method to expedite the selection of cohort attributes. In this method, supervised learning allows a phenotyping algorithm to learn from a set of labeled examples (cases) of what attributes contribute to cohort membership. This algorithm can then be used to better ascertain the defining characteristics of a phenotype and what trade offs occur in overall study accuracy when choosing to modify phenotype criteria.

To apply this approach on OMOP data, OHDSI community researchers created Automated PHenotype Routine for Observational Definition, Identification, Training and Evaluation (APHRODITE), an R-package cohort building framework that combines the ability of learning from imperfectly labeled data and the Anchor learning framework for improving selected features in the phenotype models, for use with the OHDSI/OMOP CDM (reference: https://www.ncbi.nlm.nih.gov/pubmed/28815104). APHRODITE is an open-source package (https://github.com/OHDSI/Aphrodite) available for use which provides the OHDSI data network to the ability to start building electronic phenotype models that leverage machine learning techniques and go beyond traditional rule based approaches to cohort building.

## 11.2 Phenotype Evaluation

The systematic reuse of cohort definitions and the subsequent evaluation of phenotypes to characterize components of disease remains an ongoing piece of work within the OHDSI Community. A literature review of over 33 studies found significant heterogeneity in phenotype algorithms used, validation methods, and results (Swerdel reference). In general, the validation of a rules-based cohort definition or probabilistic algorithm can be thought of as a test of the proposed cohort compared to some form of "gold standard" reference (e.g. manual chart review of cases).

For a complete validation of an algorithm, we need to calculate:

- **Sensitivity** = True Positive (TP) / (True Positive + False Negative)
- **Specificity** = True Negative (TN) / (True Negative + False Positive)
- **Positive Predictive Value** = TP / (True Positive + False Positive)

| | | Truth | |
|---|---|---|---|
| | | Positive | Negative |
| Test | Positive | True Positive (TP) | False Positive (FP) |
| | Negative | False Negative (FN) | True Negative (TN) |

Figure 11.3: Algorithm Evaluation

This framework continues to be utilized across cohort definition research to evaluate the utility of reuse of cohorts across different electronic health data.

## 11.3 OHDSI Gold Standard Phenotype Library

To assist the community in evaluation of existing cohort definitions and algorithms, the OHDSI Gold Standard Phenotype Library (GSPL) Workgroup was formed. The purpose of the GSPL workgroup is to provide additional leadership to the development of community-backed cohort libraries from rules-based and probabilistic methods. The GPSL enable members of the OHDSI community to find, evaluate, and utilize community-validated cohort definitions for research and other activities. These "gold standard" definitions will reside in a library, the entries of which are held to specific standards of design and evaluation. For additional information related to the GSPL, consult the OHDSI workgroup page (https://www.ohdsi.org/web/wiki/doku.php?id=projects:workgroups:gold-library-wg).

## 11.4 Practice

Building a cohort starts with asking a question: "I want to find patients who initiate ACE inhibitors monotherapy as first-line treatments for hypertension."

Before you can define a cohort, you will need to construct OMOP concept sets. OMOP concept sets represent the sets of clinical codes that are strung together with other logical expressions to create your cohort. A detailed discussion of OMOP concept sets can be found 6. Cohort inclusion criteria are created using specific attributes of data in the OMOP CDM (e.g. condition occurrence, drug era, drug exposure, observation period, visit, etc). OHDSI domains are analogous to building blocks to contribute cohort attributes:

Prior to building a cohort, refer to the Common Data Model (Chapter 5) to understand what data elements are available for defining a cohort. When you are building a cohort, you should consider which of these is more important to you, finding all the eligible patients? *vs.* Getting only the ones you are confident about?

Your strategy to construct your cohort will depend on your definition stringency. The right cohort design will depend on the question you're trying to answer. You may opt to build a cohort definition that: uses everything you can get, uses the lowest common denominator so you can share or is a


Figure 11.4: Building Blocks of Cohorts

compromise of the two. It is ultimately at the researcher's discretion what, threshold of stringency is necessary to adequately study the cohort of interest.

### 11.4.1 Using ATLAS

Missing: need to add high quality screenshots.

### 11.4.2 Using SQL

Missing: need to build tables for equivalent code.

## 11.5 Exercises

To be created.

# Chapter 12

# Characterization

ATLAS' incidence rate calculator + cohort characterization tool

FeatureExtraction package: https://github.com/OHDSI/FeatureExtraction

Case study: characteristics + IRs of some cohorts

Example .. http://www.pnas.org/content/113/27/7329

# Chapter 13

# Population-level estimation

*Chapter leads: Martijn Schuemie, David Madigan, Marc Suchard & Patrick Ryan*

Observational healthcare data, such as administrative claims and electronic health records, offer opportunities to generate real-world evidence about the effect of treatments that can meaningfully improve the lives of patients. In this chapter we focus on population-level effect estimation, that is, the estimation of average causal effects of exposures (e.g. medical interventions such as drug exposures or procedures) on specific health outcomes of interest. In what follows, we consider two different estimation tasks:

- **Direct effect estimation**: estimating the effect of an exposure on the risk of an outcome, as compared to no exposure.
- **Comparative effect estimation**: estimating the effect of an exposure (the target exposure) on the risk of an outcome, as compared to another exposure (the comparator exposure). \index{comparative effect estimation

In both cases, the patient-level causal effect contrasts a factual outcome, i.e., what happened to the exposed patient, with a counterfactual outcome, i.e., what would have happened had the exposure not occurred (direct) or had a different exposure occurred (comparative). Since any one patient reveals only the factual outcome (the fundamental problem of causal inference), the various effect estimation designs employ different analytic devices to shed light on the counterfactual outcomes.

Use-cases for population-level effect estimation include treatment selection, safety surveillance, and comparative effectiveness. Methods can test specific hypotheses one-at-a-time (e.g. 'signal evaluation') or explore multiple-hypotheses-at-once (e.g. 'signal detection'). In all cases, the objective remains the same: to produce a high-quality estimate of the causal effect.

In this chapter we first describe various population-level estimation study designs, all of which are implemented as R packages in the OHDSI Methods Library. We then detail the design of an example estimation study, followed by step-by-step guides of how to implement the design using ATLAS and R. Finally, we review the various outputs generated by the study, including study diagnostics and effect size estimates.

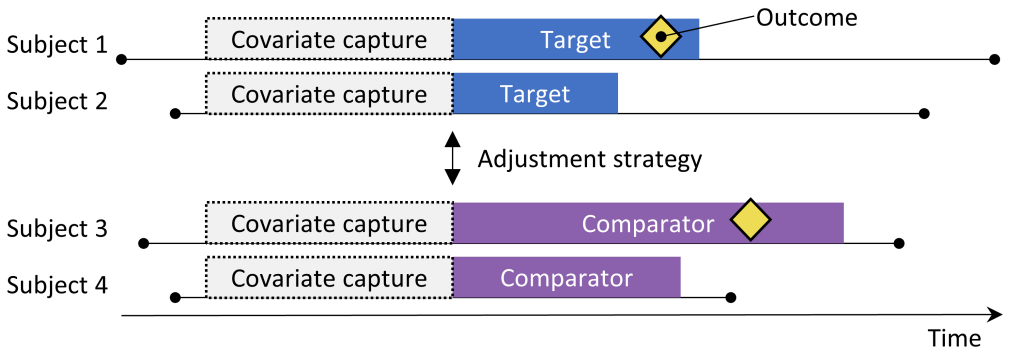



Figure 13.1: The new-user cohort design. Subjects observed to initiate the target treatment are compared to those initiating the comparator treatment. To adjust for differences between the two treatment groups several adjustment strategies can be used, such as stratification, matching, or weighting by the propensity score, or by adding baseline characateristcs to the outcome model. The chararacteristics included in the propensity model or outcome model are captured prior to treatment initiation.

## 13.1 The cohort method design

The cohort method attempts to emulate a randomized clinical trial (Hernan and Robins, 2016). Subjects that are observed to initiate one treatment (the target) are compared to subjects initiating another treatment (the comparator) and are followed for a specific amount of time following treatment initiation, for example the time they stay on the treatment. We can specify the questions we wish to answer in a cohort study by making the five choices highlighted in Table 13.1.

Table 13.1: Main design choices in a comparative cohort design.

| Choice | Description |
|---|---|
| Target cohort | A cohort representing the target treatment |
| Comparator cohort | A cohort representing the comparator treatment |
| Outcome cohort | A cohort representing the outcome of interest |
| Time-at-risk | At what time (often relative to the target and comparator cohort start and end dates) do we consider the risk of the outcome? |
| Model | The model used to estimate the effect while adjusting for differences between the target and comparator |

The choice of model specifies, among others, the type of model. For example, we could use a logistic regression, which evaluates whether or not the outcome has occurred, and produces an odds ratio. A logistic regression assumes the time-at-risk is of the same length for both target and comparator, or is irrelevant. Alternatively, we could choose a Poisson regression which estimates the incidence rate ratio, assuming a constant incidence rate. Often a Cox regression is used which considers time to first outcome to estimate the hazard ratio, assuming proportional hazards between target and comparator.

The new-user cohort method inherently is a method for comparative effect estimation, comparing one treatment to another. It is difficult to use this method to compare a treatment against no treatment, since it is hard to define a group of unexposed people that is comparable with the exposed group. If one wants to use this design for direct effect estimation, the preferred way

maybe the target patients that experienced stroke might well have done so even if they had received the comparator. In this context, age is a "confounder".

### 13.1.1 Propensity scores

In a randomized trial, a (virtual) coin toss assigns patients to their respective groups. Thus, by design, the probability that a patient receives the target treatment as against the comparator treatment does not relate in any way to patient characteristics such as age. The coin has no knowledge of the patient, and, what's more, we know with certainty the exact probability that a patient receives the target exposure. As a consequence, and with increasing confidence as the number of patients in the trial increases, the two groups of patients essentially *cannot* differ systematically with respect to *any* patient characteristic. This guaranteed balance holds true for characteristics that the trial measured (such as age) as well as characteristics that the trial failed to measure.

For a given patient, the *propensity score* (PS) is the probability that that patient received the target treatment as against the comparator. (Rosenbaum and Rubin, 1983) In a balanced two-arm randomized trial, the propensity score is 0.5 for every patient. In a propensity score-adjusted observational study, we estimate the probability of a patient receiving the target treatment based on what we can observe in the data on and before the time of treatment initiation (irrespective of the treatment they actually received). This a straightforward predictive modeling application; we fit a model (e.g. a logistic regression) that predicts whether a subject receives the target treatment, and use this model to generate predicted probabilities (the PS) for each subject. Unlike in a standard randomized trial, different patients will have different probabilities of receiving the target treatment. The PS can be used in several ways, for example by matching target subjects to comparator subjects with similar PS, by stratifying the study population based on the PS, or by weighting subjects using Inverse Probability of Treatment Weighting (IPTW) derived from the PS. When matching we can select just one comparator subject for each target subject, or we can allow more than one comparator subject per target subject, a technique know as variable-ratio matching. (Rassen et al., 2012)

For example, suppose we use one-on-one PS matching, and that Jan has a priori probability of 0.4 of receiving the target treatment and in fact receives the target treatment. If we can find a patient (named Jun) that also had an a priori probability of 0.4 of receiving the target treatment but in fact received the comparator, the comparison of Jan and Jun's outcomes is like a mini-randomized trial, at least with respect to measured confounders. This comparison will yield an estimate of the Jan-Jun causal contrast that is as good as the one randomization would have produced. Estimation then proceeds as follows: for every patient that received the target, find one or more matched patients that received the comparator but had the same a priori probability of receiving the target. Compare the outcome for the target patient with the outcomes for the comparator patients within each of these matched groups.

Propensity scoring controls for measured confounders. In fact, if treatment assignment is "strongly ignorable" given measured characteristics, propensity scoring will yield an unbiased estimate of the causal effect. "Strongly ignorable" essentially means that there are no unmeasured confounders, and that the measured confounders are adjusted for appropriately. Unfortunately this is not a testable assumption. See Chapter 19 for further discussion of this issue.

### 13.1.2 Variable selection

In the past, PS were computed based on manually selected characteristics, and although the OHDSI tools can support such practices, we prefer using many generic characteristics (i.e. characteristics that are not selected based on the specific exposures and outcomes in the study). (Tian et al., 2018) These characteristics include demographics, as well as all diagnoses, drug exposures, measurement, and medical procedures observed prior to and on the day of treatment initiation. A model typically involves 10,000 to 100,000 unique characteristics, which we fit using large-scale regularized regression (Suchard et al., 2013) implemented in the Cyclops package. In essence, we let the data appropriately weigh the characteristics.

We typically include the day of treatment initiation in the covariate capture window because many relevant data points such as the diagnosis leading to the treatment are recorded on that date. This does require us to explicitly exclude the target and comparator treatment from the set of covariates, because these are the things we are trying to predict.

Some have argued that a data-driven approach to covariate selection that does not depend on clinical expertise to specify the "right" causal structure runs the risk of erroneously including so-called instrumental variables and colliders, thus increasing variance and potentially introducing bias. (Hernan et al., 2002) However, these concerns are unlikely to have a large impact in real-world scenarios. (Schneeweiss, 2018) Furthermore, in medicine the true causal structure is rarely known, and when different researchers are asked to identify the 'right' covariates to include for a specific research question, each researcher invariably comes up with a different list, thus making the process irreproducible. Most importantly, our diagnostics such as inspection of the propensity model, evaluating balance on all covariates, and including negative controls would identify most problems related to colliders and instrumental variables.

### 13.1.3 Caliper

Since propensity scores fall on a continuum from 0 to 1, exact matching is rarely possible. Instead, the matching process finds patients that match the propensity score of a target patient(s) within some tolerance known as a "caliper." Following Austin (2011), we use a default caliper of 0.2 standard deviations on the logit scale.

### 13.1.4 Overlap: preference scores

The propensity method requires that matching patients exist! As such, a key diagnostic shows the distribution of the propensity scores in the two groups. To facilitate interpretation, the OHDSI tools plot a transformation of the propensity score called the "preference score". (Walker et al., 2013) The preference score adjusts for the "market share" of the two treatments. For example, if 10% of patients receive the target treatment (and 90% receive the comparator treatment), then patients with a preference score of 0.5 have a 10% probability of receiving the target treatment. Mathematically, the preference score is

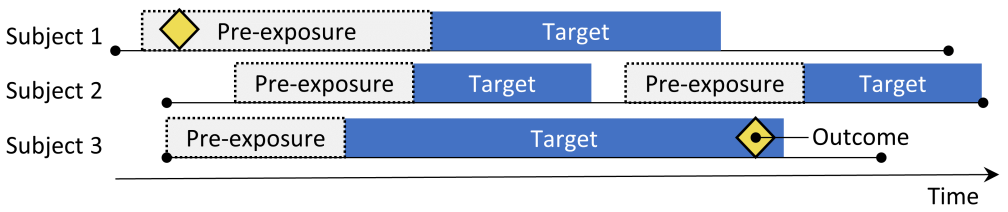



Figure 13.2: The self-controlled cohort design. The rate of outcomes during exposure to the target is compared to the rate of outcomes in the time pre-exposure.

$$\ln\left(\frac{F}{1-F}\right) = \ln\left(\frac{S}{1-S}\right) - \ln\left(\frac{P}{1-P}\right)$$

Where $F$ is the preference score, $S$ is the propensity score, and $P$ is the proportion of patients receiving the target treatment.

Walker et al. (2013) discuss the concept of "empirical equipoise". They accept exposure pairs as emerging from empirical equipoise if at least half of the exposures are to patients with a preference score of between 0.3 and 0.7.

### 13.1.5 Balance

Good practice always checks that the PS adjustment succeeds in creating balanced groups of patients. Figure 13.18 shows the standard OHDSI output for checking balance. For each patient characteristic, this plots the standardized difference between means between the two exposure groups before and after PS adjustment. Some guidelines recommend an after-adjustment standardized difference upper bound of 0.1. (Rubin, 2001)

## 13.2 The self-controlled cohort design

The self-controlled cohort (SCC) design (Ryan et al., 2013) compares the rate of outcomes during exposure to the rate of outcomes in the time just prior to the exposure. The four choices shown in Table 13.2 define a self-controlled cohort question.

Table 13.2: Main design choices in a self-controlled cohort design.

| Choice | Description |
|---|---|
| Target cohort | A cohort representing the treatment |
| Outcome cohort | A cohort representing the outcome of interest |
| Time-at-risk | At what time (often relative to the target cohort start and end dates) do we consider the risk of the outcome? |
| Control time | The time period used as the control time |

Because the same subject that make up the exposed group are also used as the control group, no adjustment for between-person differences need to be made. However, the method is vulnerable

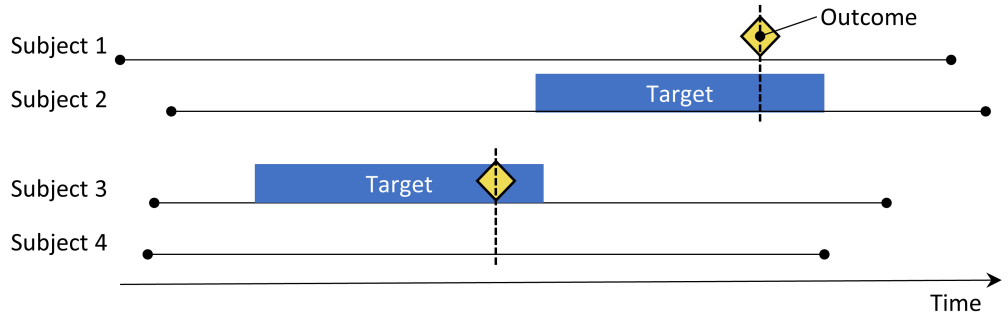



Figure 13.3: The case-control design. Subjects with the outcome ('cases') are compared to subjects without the outcome ('controls') in terms of their exposure status. Often, cases and controls are matched on various characteristics such as age and sex.

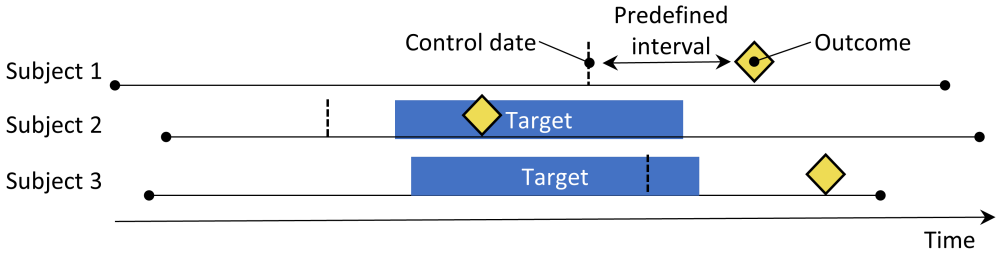



Figure 13.4: The case-crossover design. The time around the outcome is compared to a control date set at a predefined interval prior to the outcome date.

## 13.3 The case-control design

Case-control studies (Vandenbroucke and Pearce, 2012) consider the question "are persons with a specific disease outcome exposed more frequently to a specific agent than those without the disease?" Thus, the central idea is to compare "cases", i.e., subjects that experience the outcome of interest with "controls", i.e., subjects that did not experience the outcome of interest. The choices in Table 13.3 define a case-control question.

Table 13.3: Main design choices in a case-control design.

| Choice | Description |
|---|---|
| Outcome cohort | A cohort representing the cases (the outcome of interest) |
| Control cohort | A cohort representing the controls. Typically the control cohort is automatically derived from the outcome cohort using some selection logic |
| Target cohort | A cohort representing the treatment |
| [Nesting cohort] | Optionally, a cohort defining the subpopulation from which cases and controls are drawn |
| Time-at-risk | At what time (often relative to the index date) do we consider exposure status? |

Often, one selects controls to match cases based on characteristics such as age and sex to make them

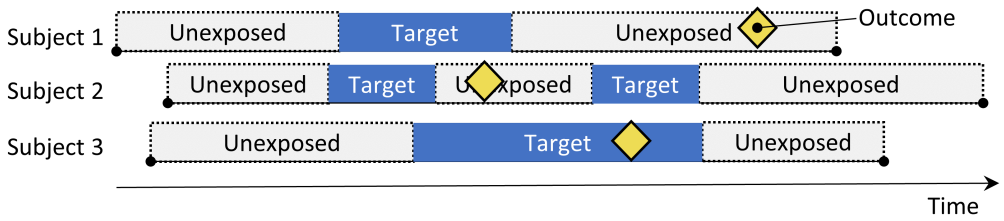



Figure 13.5: The Self-Controlled Case Series design. The rate of outcomes during exposure is compared to the rate of outcomes when not exposed.

determine whether there is something special about the day the outcome occurred. Table 13.4 shows the choices that define a case-crossover question:

Table 13.4: Main design choices in a case-crossover design.

| Choice | Description |
|---|---|
| Outcome cohort | A cohort representing the cases (the outcome of interest) |
| Target cohort | A cohort representing the treatment |
| Time-at-risk | At what time (often relative to the index date) do we consider exposure status? |
| Control time | The time period used as the control time |

Since cases serve as their own control, it is a self-controlled design, and should therefore be robust to confounding due to between-person differences. One concern is that, because the outcome date is always later than the control date, the method will be positively biased if the overall frequency of exposure increases over time (or negatively biased if there is a decrease). To address this, the case-time-control design (Suissa, 1995) was developed, which adds controls, matched for example on age and sex, to the case-crossover design to adjust for exposure trends.

## 13.5 The self-controlled case series design

The Self-Controlled Case Series (SCCS) design (Farrington, 1995; Whitaker et al., 2006) compares the rate of outcomes during exposure to the rate of outcomes during all unexposed time, both before, between, and after exposures. It is a Poisson regression that is conditioned on the person. Thus, it seeks to answer the question: "Given that a patient has the outcome, is the outcome more likely during exposed time compared to non-exposed time?". The choices in Table 13.5 define an SCCS question.

Table 13.5: Main design choices in a self-controlled case series design.

| Choice | Description |
|---|---|
| Target cohort | A cohort representing the treatment |
| Outcome cohort | A cohort representing the outcome of interest |
| Time-at-risk | At what time (often relative to the target cohort start and end dates) do we consider the risk of the outcome? |
| Model | The model to estimate the effect, including any adjustments for time-varying confounders |

Like other self-controlled designs, the SCCS is robust to confounding due to between-person differences, but vulnerable to confounding due to time-varying effects. Several adjustments are possible to attempt to account for these, for example by including age and season. A special variant of the SCCS includes not just the exposure of interest, but all other exposures to drugs recorded in the database, (Simpson et al., 2013) potentially adding thousands of additional variables to the model. L1-regularization using cross-validation to select the regularization hyperparameter is applied to the coefficients of all exposures except the exposure of interest.

One important assumption underlying the SCCS is that the observation period end is independent of the date of the outcome. Because for some outcomes, especially ones that can be fatal such as stroke, this assumption can be violated. An extension to the SCCS has been developed that corrects for any such dependency. (Farrington et al., 2011)

## 13.6 Designing a hypertension study

### 13.6.1 Problem definition

ACE inhibitors (ACEi) are widely used in patients with hypertension or ischemic heart disease, especially those with other comorbidities such as congestive heart failure, diabetes mellitus, or chronic kidney disease. (Zaman et al., 2002) Angioedema, a serious and sometimes life-threatening adverse event that usually manifests as swelling of the lips, tongue, mouth, larynx, pharynx, or periorbital region, has been linked to the use of these medications. (Sabroe and Black, 1997) However, limited information is available about the absolute and relative risks for angioedema associated with the use of these medications. Existing evidence is primarily based on investigations of specific cohorts (e.g., predominantly male veterans or Medicaid beneficiaries), whose findings may not be generalizable to other populations, or based on investigations with few events, which provide unstable risk estimates (Powers et al., 2012). Several observational studies compare ACEi to beta-blockers for the risk of angioedema, (Magid et al., 2010; Toh et al., 2012) but beta-blockers are no longer recommend as first-line treatment of hypertension. (Whelton et al., 2018) A viable alternative treatment could be thiazides or thiazide-like diuretics (THZ), which could be just as effective in managing hypertension

and its associated risks such as acute myocardial infarction (AMI), but without increasing the risk of angioedema.

The following will demonstrate how to apply our population-level estimation framework to observational healthcare data to address the following comparative estimation questions:

> What is the risk of angioedema in new users of ACE inhibitors compared to new users of thiazide and thiazide-like diuretics?
>
> What is the risk of acute myocaridal infarction in new users of ACE inhibitors compared to new users of thiazide and thiazide-like diuretics?

Since these are comparative effect estimation questions we will apply the cohort method as described in Section 13.1.

### 13.6.2 Target and comparator

We consider patients new-users if their first observed treatment for hypertension was monotherapy with any active ingredient in either the ACEi or THZ class. We define mono therapy as not starting on any other anti-hypertensive drug in the seven days following treatment initiation. We require patients to have at least one year of prior continuous observation in the database before first exposure and a recorded hypertension diagnosis at or in the year preceding treatment initiation.

### 13.6.3 Outcome

We define angioedema as any occurrence of an angioedema condition concept during an inpatient or emergency room (ER) visit, and require there to be no angioedema diagnosis recorded in the seven days prior. We define AMI as any occurrence of an AMI condition concept during an inpatient or ER visit, and require there to be no AMI diagnosis record in the 180 days prior.

### 13.6.4 Time-at-risk

We define time-at-risk to start on the day after treatment initiation, and stop when exposure stops, allowing for a 30-day gap between subsequent drug exposures.

### 13.6.5 Model

We fit a PS model using the default set of covariates, including demographics, conditions, drugs, procedures, measurements, observations, and several co-morbidity scores. We exclude ACEi and THZ from the covariates. We perform variable-ratio matching and condition the Cox regression on the matched sets.

### 13.6.6 Study summary

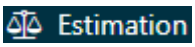
Table 13.6: Main design choices four our comparative cohort study.

| Choice | Value |
|---|---|
| Target cohort | New users of ACE inhibitors as first-line monotherapy for hypertension. |
| Comparator cohort | New users of thiazides or thiazide-like diuretics as first-line monotherapy for hypertension. |
| Outcome cohort | Angioedema or acute myocardial infarction. |
| Time-at-risk | Starting the day after treatment initiation, stopping when exposure stops. |
| Model | Cox proportional hazards model using variable-ratio matching. |

### 13.6.7 Control questions

To evaluate whether our study design produces estimates in line with the truth, we additionally include a set of control questions where the true effect size is known. Control questions can be divided in negative controls, having a hazard ratio of 1, and positive controls, having a known hazard ratio greater than 1. For several reasons we use real negative controls, and synthesize positive controls based on these negative controls. How to define and use control questions is discussed in detail in Chapter 19.

## 13.7 Implementing the study using ATLAS

Here we demonstrate how this study can be implemented using the Estimation function in ATLAS. Click on 

in the left bar of ATLAS, and create a new estimation study. Make sure to give the study an easy-to-recognize name. The study design can be saved at any time by clicking the  button.

In the Estimation design function, there are three sections: Comparisons, Analysis Settings, and Evaluation Settings. We can specify multiple comparisons and multiple analysis settings, and ATLAS will execute all combinations of these as separate analyses. Here we discuss each section:

### 13.7.1 Comparative cohort settings

A study can have one or more comparisons. Click on "Add Comparison", which will open a new dialog. Click on  to the select the target and comparator cohorts. By clicking on "Add Outcome" we can add our two outcome cohorts. We assume the cohorts have already been created in ATLAS as described in Chapter 11. The Appendix provides the full definitions of the target (Appendix B.2), comparator (Appendix B.5), and outcome (Appendix B.4 and Appendix B.3) cohorts. When done, the dialog should look like Figure 13.6.

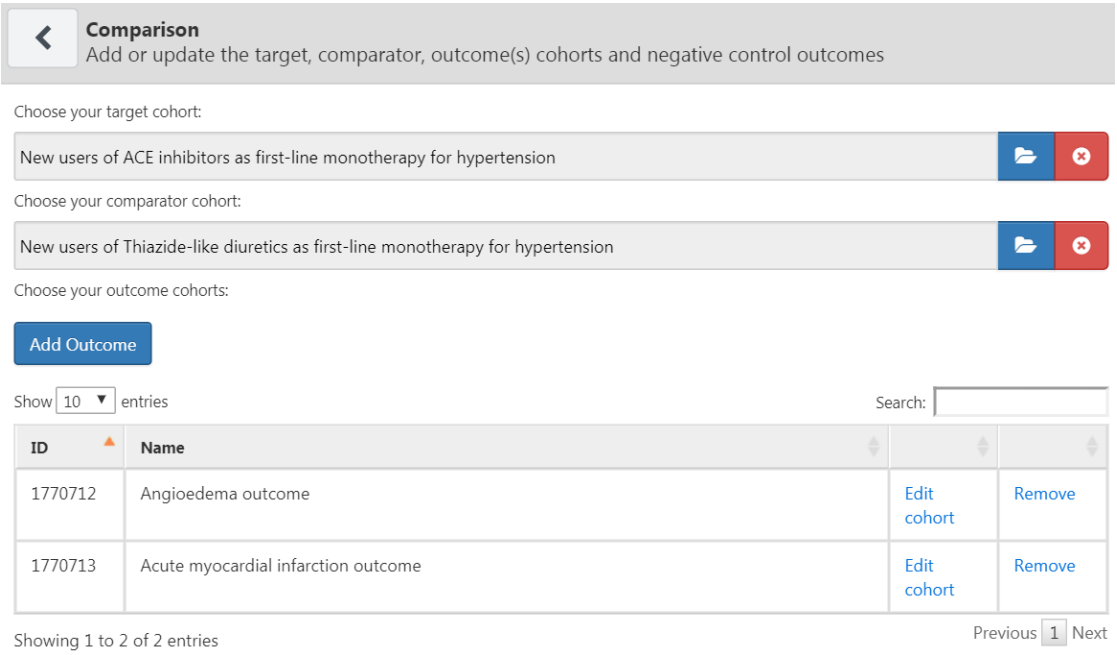



Figure 13.6: The comparison dialog

Note that we can select multiple outcomes for a target-comparator pair. Each outcome will be treated independently, and will result in a separate analysis.

**Negative control outcomes**

Negative controls outcomes are outcomes that are not believed to be caused by either the target or the comparator, and where therefore the true hazard ratio equals 1. Ideally, we would have proper cohort definitions for each outcome cohort. However, typically, we only have a concept set, with one concept per negative control outcome, and some standard logic to turn these into outcome cohorts. Here we assume the concept set has already been created as described in Chapter 19 and can simply be selected. The negative control concept set should contain a concept per negative control, and not include descendants. Figure 13.7 shows the negative control concept set used for this study.

**Concepts to include**

TODO: Update these sections when ATLAS interface has been updated.

When selecting concept to include, we can specify which covariates we would like to generate, for example to use in a propensity model. When specifying covariates here, all other covariates (aside from those you specified) are left out. We usually want to include all baseline covariates, letting the regularized regression build a model that balances all covariates. The only reason we might want to specify particular covariates is to replicate an existing study that manually picked covariates. These inclusions can be specified in this comparison section or in the analysis section, because sometimes they pertain to a specific comparison (e.g. know confounders in a comparison), or sometimes they pertain to an analysis (e.g. when evaluating a particular covariate selection strategy).

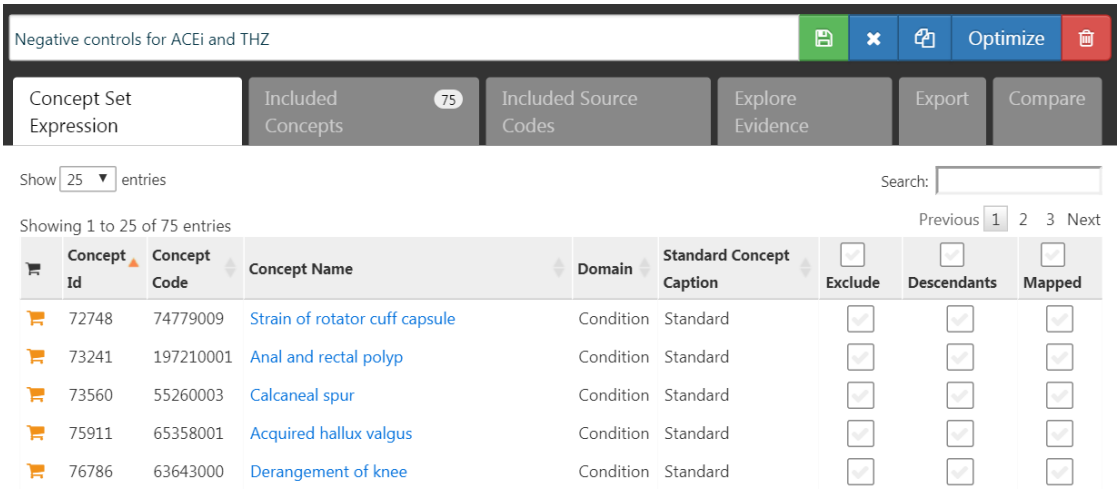



Figure 13.7: Negative Control concept set.

**Concepts to exclude**

Rather than specifying which concepts to include, we can instead specify concepts to *exclude*. When we submit a concept set in this field, we use every covariate except for those that we submitted. When using the default set of covariates, which includes all drugs and procedures occurring on the day of treatment initiation, we must exclude the target and comparator treatment, and any concepts that are directly related to these. For example, if the target exposure is an injectable, we should not only exclude the drug, but also the injection procedure from the propensity model. In this example, the covariates we want to exclude are ACEi and THZ. Figure 13.8 shows we select a concept set that includes all these concepts.

After selecting the negative controls and covariates to exclude, the lower half of the comparisons dialog should look like Figure 13.9.

### 13.7.2 Effect estimation analysis settings

After closing the comparisons dialog we can click on "Add Analysis Settings". In the box labeled "Analysis Name", we can give the analysis a unique name that is easy to remember and locate in the future. For example, we could set the name to "Propensity score matching".

**Study population**

There are a wide range of options to specify the study population; the set of subjects that will enter the analysis. Many of these overlap with options available when designing the target and comparator cohorts in the cohort definition tool. One reason for using the options in Estimation instead of in the cohort definition is re-usability: We can define the target, comparator, and outcome cohorts completely independently, and add dependencies between these at a later point in time. For example, if we wish to remove people who had the outcome before treatment initiation, we could do so in the definitions of the target and comparator cohort, but then we would need to create separate cohorts

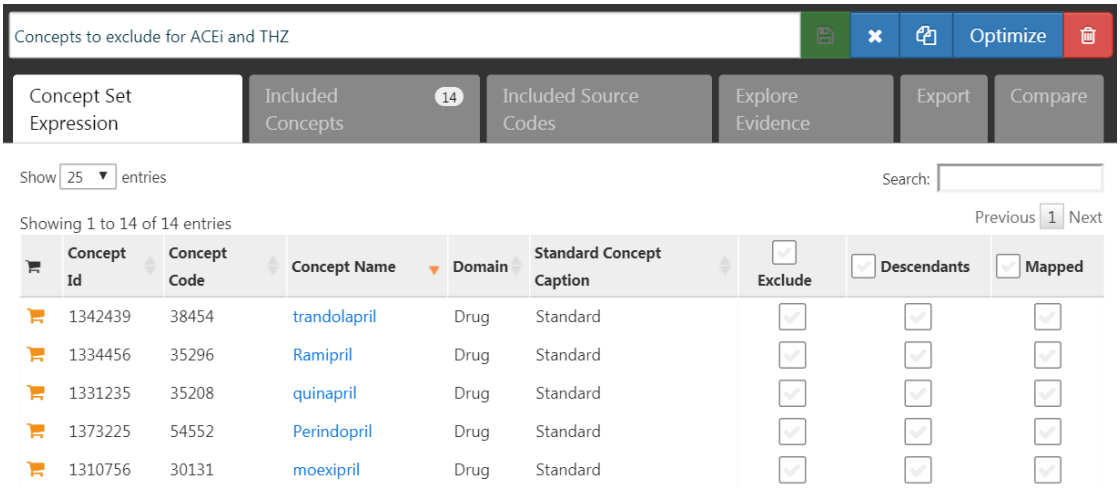



Figure 13.8: The concept set defining the concepts to exclude.

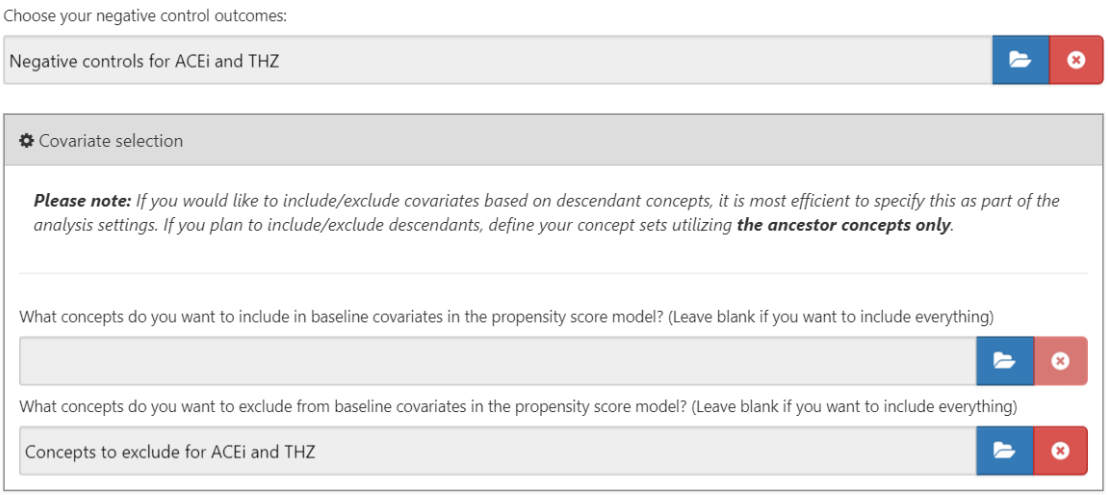



Figure 13.9: The comparison window showing concept sets for negative controls and concepts to exclude.

for every outcome! Instead, we can choose to have people with prior outcomes be removed in the analysis settings, and now we can reuse our target and comparator cohorts for our two outcomes of interest (as well as our negative control outcomes).

The **study start and end dates** can be used to limit the analyses to a specific period. The study end date also truncates risk windows, meaning no outcomes beyond the study end date will be considered. One reason for selecting a study start date might be that one of the drugs being studied is new and did not exist in an earlier time. Automatically adjusting for this can be done by answering "yes" to the question "**Restrict the analysis to the period when both exposures are observed?**". Another reason to adjust study start and end dates might be that medical practice changed over time (e.g., due to a drug warning) and we are only interested in the time where medicine was practiced a specific way.

The option "**Should only the first exposure per subject be included?**" can be used to restrict to the first exposure per patient. Often this is already done in the cohort definition, as is the case in this example. Similarly, the option "**The minimum required continuous observation time prior to index date for a person to be included in the cohort**" is often already set in the cohort definition, and can therefore be left at 0 here. Having observed time (as defined in the OBSERVATION_PERIOD table) before the index date ensures that there is sufficient information about the patient to calculate a propensity score, and is also often used to ensure the patient is truly a new user, and therefore was not exposed before.

"**Remove subjects that are in both the target and comparator cohort?**" defines, together with the option "**If a subject is in multiple cohorts, should time-at-risk be censored when the new time-at-risk starts to prevent overlap?**" what happens when a subject is in both target and comparator cohort. The first setting has three choices:

- "**Keep All**" indicating to keep the subjects in both cohorts. With this option it might be possible to double-count subjects and outcomes.
- "**Keep First**" indicating to keep the subject in the first cohort that occurred.
- "**Remove All**" indicating to remove the subject from both cohorts.

If the options "keep all" or "keep first" are selected, we may wish to censor the time when a person is in both cohorts. This is illustrated in Figure 13.10. By default, the time-at-risk is defined relative to the cohort start and end date. In this example, the time-at-risk starts one day after cohort entry, and stops at cohort end. Without censoring the time-at-risk for the two cohorts might overlap. This is especially problematic if we choose to keep all, because any outcome that occurs during this overlap (as shown) will be counted twice. If we choose to censor, the first cohort's time-at-risk ends when the second cohort's time-at-risk starts.

We can choose to **remove subjects that have the outcome prior to the risk window start**, because often a second outcome occurrence is the continuation of the first one. For instance, when someone develops heart failure, a second occurrence is likely, which means the heart failure probably never fully resolved in between. On the other hand, some outcomes are episodic, and it would be expected for patients to have more than one independent occurrence, like an upper respiratory infection. If we choose to remove people that had the outcome before, we can select **how many days we should look back when identifying prior outcomes**.

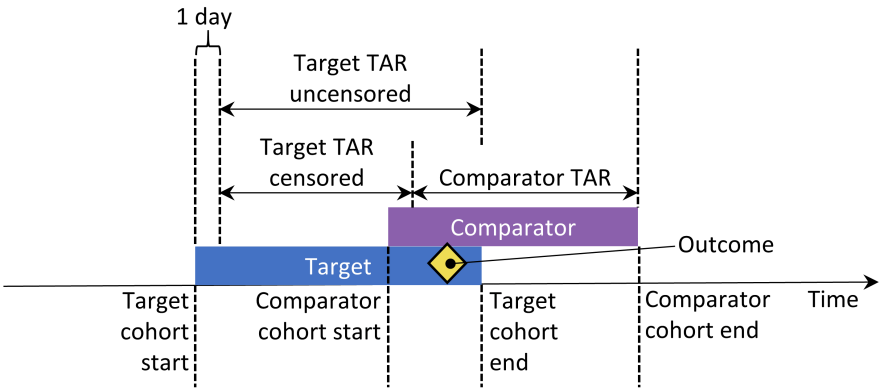



Figure 13.10: Time-at-risk (TAR) for subjects who are in both cohorts, assuming time-at-risk starts the day after treatment initiation, and stops at exposure end.

Our choices for our example study are shown in Figure 13.11. Because our target and comparator cohort definitions already restrict to the first exposure and require observation time prior to treatment initiation, we do not apply these criteria here.

**Covariate settings**

Here we specify the covariates to construct. These covariates are typically used in the propensity model, but can also be included in the outcome model (the Cox proportional hazards model in this case). If we **click to view details** of our covariate settings, we can select which sets of covariates to construct. However, the recommendation is to use the default set, which constructs covariates for demographics, all conditions, drugs, procedures, measurements, etc.

We can modify the set of covariates by specifying concepts to **include** and/or **exclude**. These settings are the same as the ones found in Section 13.7.1 on comparison settings. The reason why they can be found in two places is because sometimes these settings are related to a specific comparison, as is the case here because we wish to exclude the drugs we are comparing, and sometimes the settings are related to a specific analysis. When executing an analysis for a specific comparison using specific analysis settings, the OHDSI tools will take the union of these sets.

The choice to **add descendants to include or exclude** affects this union of the two settings. So in this example we specified only the ingredients to exclude when defining the comparisons. Here we set "Should descendant concepts be added to the list of excluded concepts?" to "Yes" to also add all descendants.

Figure 13.12 shows our choices for this study. Note that we have selected to add descendants to the concept to exclude, which we defined in the comparison settings in Figure 13.9.

**Time at risk**

Time-at-risk is defined relative to the start and end dates of our target and comparator cohorts. In our example, we had set the cohort start date to start on treatment initiation, and cohort end date when exposure stops (for at least 30 days). We set the start of time-at-risk to one day after cohort start, so one day after treatment initiation. A reason to set the time-at-risk start to be later than the cohort

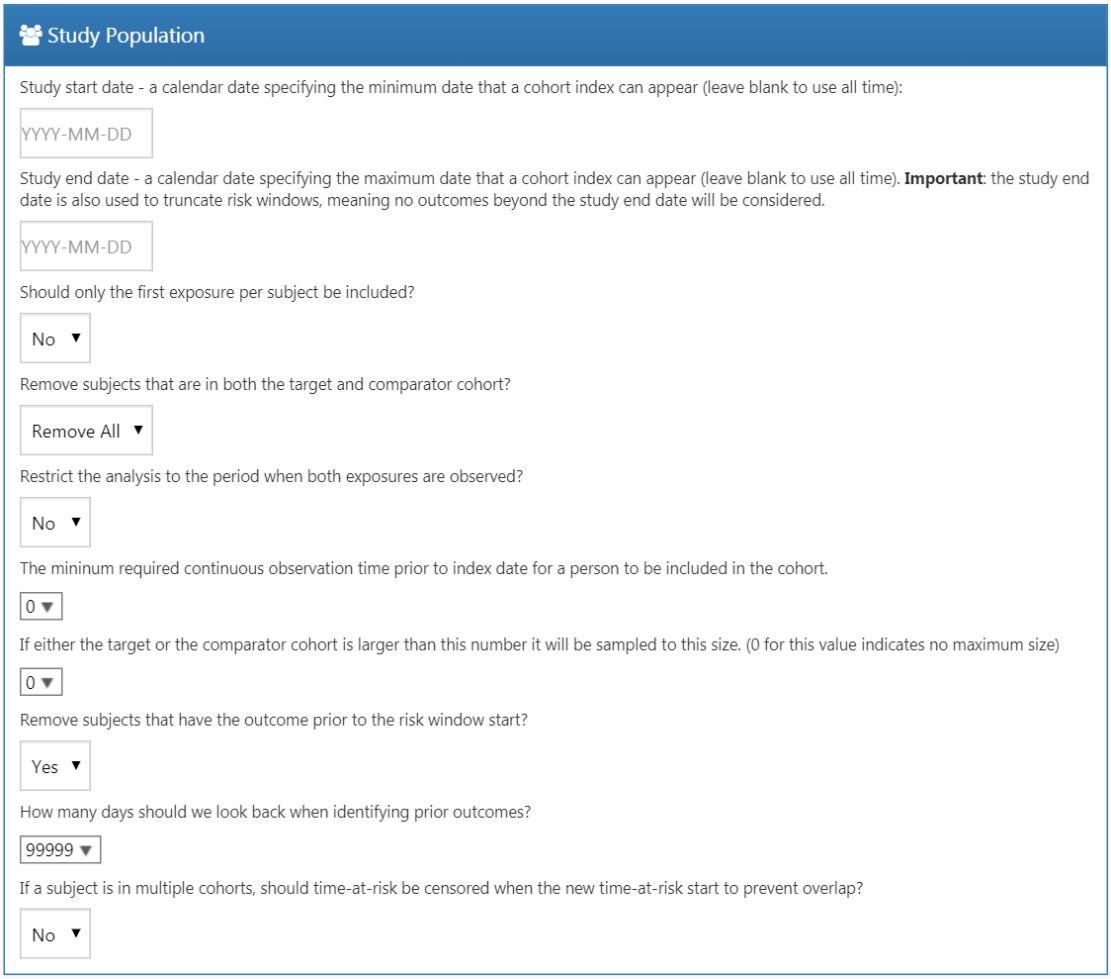



Figure 13.11: Study population settings..

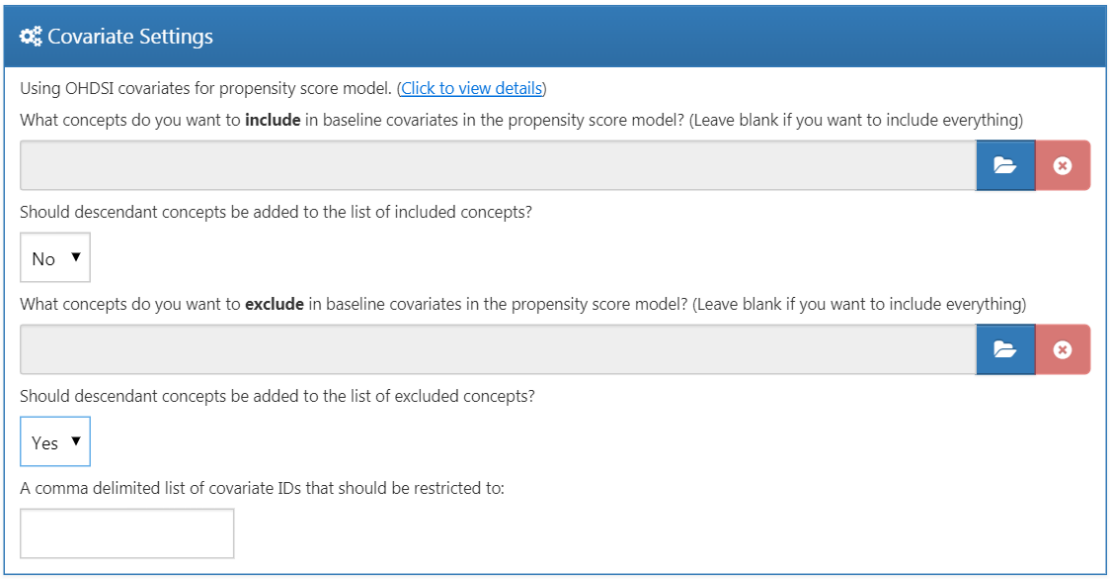



Figure 13.12: Covariate settings.

start is because we may want to exclude outcome events that occur on the day of treatment initiation if we do not believe it biologically plausible they can be caused by the drug.

We set the end of the time-at-risk to the cohort end, so when exposure stops. We could choose to set the end date later if for example we believe events closely following treatment end may still be attributable to the exposure. In the extreme we could set the time-at-risk end to a large number of days (e.g. 99999) after the cohort end date, meaning we will effectively follow up subjects until observation end. Such a design is sometimes referred to as an *intent-to-treat* design.

A patient with zero days at risk adds no information, so the **minimum days at risk** is normally set at one day. If there is a known latency for the side effect, then this may be increased to get a more informative proportion. It can also be used to create a cohort more similar to that of a randomized trial it is being compared to (e.g., all the patients in the randomized trial were observed for at least N days).

A golden rule in designing a cohort study is to never use information that falls after the cohort start date to define the study population, as this may introduce bias. For example, if we require everyone to have at least a year of time-at-risk, we will likely have limited our analyses to those who tolerate the treatment well. This setting should therefore be used with extreme care.

**Propensity score adjustment**

We can opt to **trim** the study population, removing people with extreme PS values. We can choose to remove the top and bottom percentage, or we can remove subjects whose preference score falls outside the range we specify. Trimming the cohorts is generally not recommended because it requires discarding observations, which reduces statistical power. It may be desirable to trim in some cases,

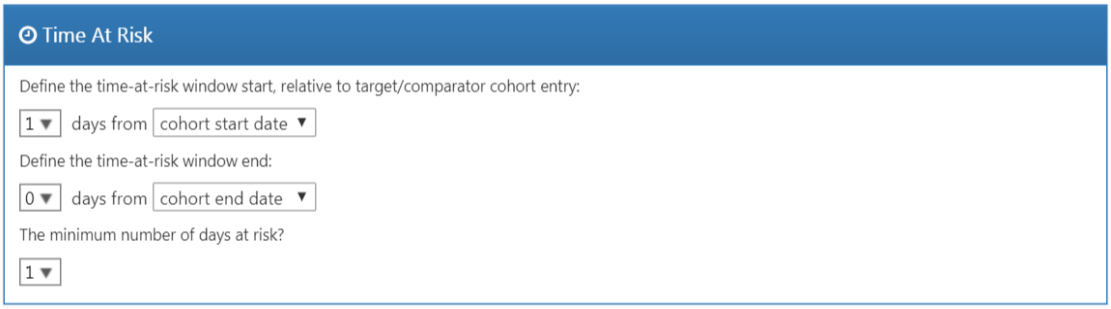



Figure 13.13: Time-at-risk settings.

for example when using IPTW.

In addition to, or instead of trimming, we can choose to **stratify** or **match** on the propensity score. When stratifying we need to specify the **number of strata** and whether to select the strata based on the target, comparator, or entire study population. When matching we need to specify the **maximum number of people from the comparator group to match to each person in the target group**. Typical values are 1 for one-on-one matching, or a large number (e.g. 100) for variable-ratio matching. We also need to specify the **caliper**: the maximum allowed difference between propensity scores to allow a match. The caliper can be defined on difference **caliper scales**:

- **The propensity score scale**: the PS itself
- **The standardized scale**: in standard deviations of the PS distributions
- **The standardized logit scale**: in standard deviations of the PS distributions after the logit transformation to make the PS more normally distributed.

In case of doubt, we suggest using the default values, or consult the work on this topic by Austin (2011).

Fitting large-scale propensity models can be computationally expensive, so we may want to restrict the data used to fit the model to just a sample of the data. By default the maximum size of the target and comparator cohort is set to 250,000. In most studies this limit will not be reached. It is also unlikely that more data will lead to a better model. Note that although a sample of the data may be used to fit the model, the model will be used to compute PS for the entire population.

**Test each covariate for correlation with the target assignment?** If any covariate has an unusually high correlation (either positive or negative), this will throw an error. This avoids lengthy calculation of a propensity model only to discover complete separation. Finding very high univariate correlation allows you to review the covariate to determine why it has high correlation and whether it should be dropped.

Figure 13.14 shows our choices for this study. Note that we select variable-ratio matching by setting the maximum number of people to match to 100.

**Outcome model settings**

First, we need to **specify the statistical model we will use to estimate the relative risk of the out-**

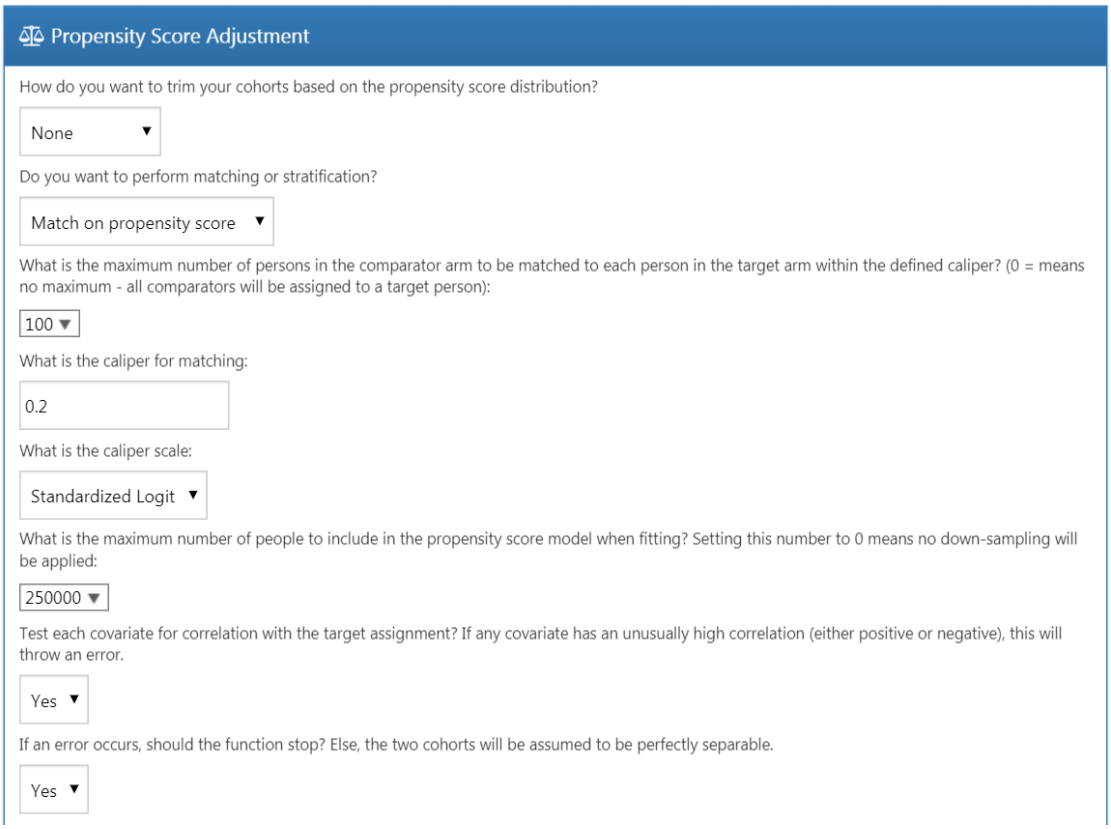



Figure 13.14: Propenstity score adjustment settings.

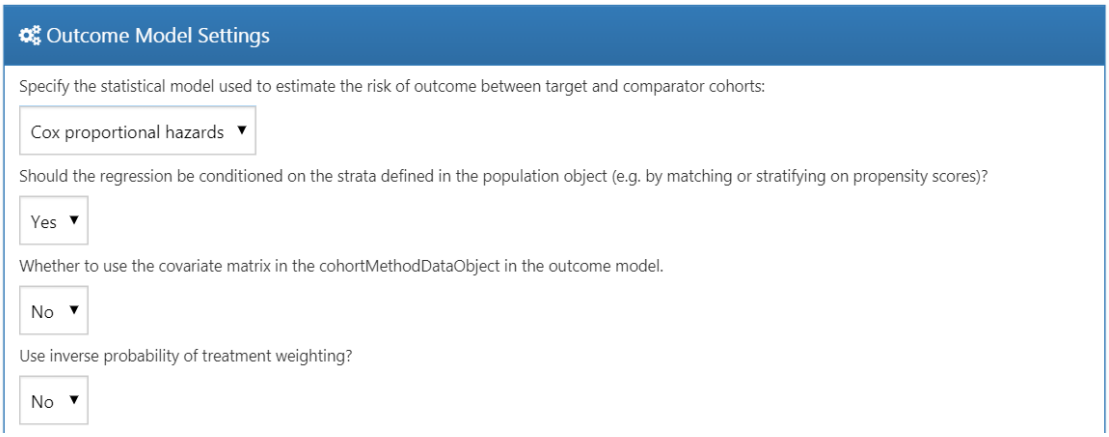



Figure 13.15: Outcome model settings.

**come between target and comparator cohorts**. We can choose between Cox, Poisson, and logistic regression, as discussed briefly in Section 13.1. For our example we choose a Cox proportional hazards model, which considers time to first event with possible censoring. Next, we need to specify **whether the regression should be conditioned on the strata**. One way to understand conditioning is to imagine a separate estimate is produced in each stratum, and then combined across strata. For one-to-one matching this is likely unnecessary and would just lose power. For stratification or variable-ratio matching it is required.

We can also choose to **add all covariates to the outcome model** to adjust the analysis. This can be done in addition or instead of using a propensity model. However, whereas there usually is ample data to fit a propensity model, with many people in both treatment groups, there is typically very little data to fit the outcome model, with only few people having the outcome. We therefore recommend keeping the outcome model as simple as possible and not include additional covariates.

Instead of stratifying or matching on the propensity score we can also choose to **use inverse probability of treatment weighting** (IPTW). If weighting is used it is often recommended to use some form of trimming to avoid extreme weights and therefore unstable estimates.

Figure 13.14 shows our choices for this study. Because we use variable-ratio matching, we must condition the regression on the strata (i.e. the matched sets).

### 13.7.3 Evaluation settings

As described in Chapter 19, negative and positive controls should be included in our study to evaluate the operating characteristics, and perform empirical calibration.

**Negative control outcome cohort definition**

In Section 13.7.1 we selected a concept set representing the negative control outcomes. However, we need logic to convert concepts to cohorts to be used as outcomes in our analysis. ATLAS provides standard logic with three choices The first choice is whether to **use all occurrences** or just the **first**

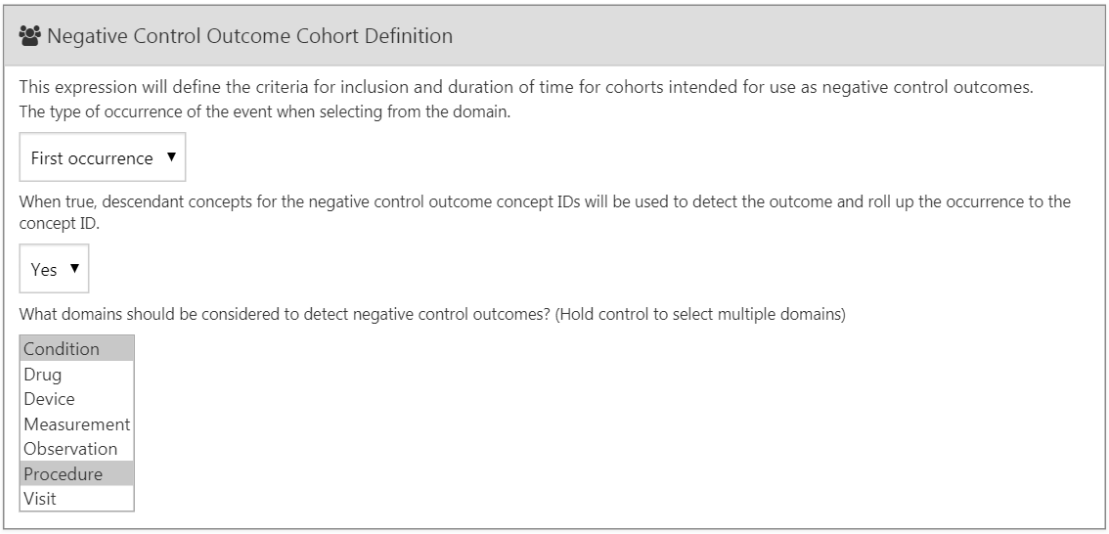



Figure 13.16: Negative control outcome cohort definition settings.

**occurrence** of the concept. The second choice determines **whether occurrences of descendant concepts should be considered**. For example, occurrences of the descendant "ingrown nail of foot" can also be counted as an occurrence of the ancestor "ingrown nail". The third choice specifies which domains should be considered when looking for the concepts.

**Positive control synthesis**

In addition to negative controls we can also include positive controls, which are exposure-outcome pairs where a causal effect is believed to exist with known effect size. For various reasons real positive controls are problematic, so instead we rely on synthetic positive controls, derived from negative controls as described in Chapter 19. Positive control synthesis is an advanced topic that we will skip for now.

TODO: Add positive control synthesis settings when ATLAS interface is updated.

### 13.7.4 Running the study package

Now that we have fully defined our study, we can export it as an executable R package. This package contains everything that is needed to execute the study at a site that has data in CDM. This includes the cohort definitions that can be used to instantiate the target, comparator and outcome cohorts, the negative control concept set and logic to create the negative control outcome cohorts, as well as the R code to execute the analysis. Before generating the package make sure to save your study, then click on the **Utilities** tab. Here we can review the set of analyses that will be performed. As mentioned before, every combination of a comparison and an analysis setting will results in a separate analysis. In our example we have specified two analyses: ACEi versus THZ for AMI, and ACEi versus THZ for angioedema, both using propensity score matching.

We must provide a name for our package, after which we can click on "Download" to download the

zip file. The zip file contains an R package, with the usual required folder structure for R packages. (Wickham, 2015) To use this package we recommend using R Studio. If you are running R Studio locally, unzip the file, and double click the .Rproj file to open it in R Studio. If you are running R Studio on an R studio server, click 

to upload and unzip the file, then click on the .Rproj file to open the project.

Once you have opened the project in R Studio, you can open the README file, and follow the instructions. Make sure to change all file paths to existing paths on your system.

A common error message that may appear when running the study is "High correlation between covariate(s) and treatment detected". This indicates that when fitting the propensity model, some covariates were observed to be highly correlated with the exposure. Please review the covariates mentioned in the error message, and exclude them from the set of covariates if appropriate (see Section 13.1.2).

## 13.8 Implementing the study using R

Instead of using ATLAS to write the R code that executes the study, we can also write the R code ourselves. One reason we might want to do this is because R offers far greater flexibility than is exposed in ATLAS. If we for example wish to use custom covariates, or a linear outcome model, we will need to write some custom R code, and combine it with the functionality provided by the OHDSI R packages.

For our example study we will rely on the CohortMethod package to execute our study. CohortMethod extracts the necessary data from a database in the CDM and can use a large set of covariates for the propensity model. In the following example we first only consider angioedema as outcome. In Section 13.8.6 we then describe how this can be extended to include AMI and the negative control outcomes.

### 13.8.1 Cohort instantiation

We first need to instantiate the target and outcome cohorts. Instantiating cohorts is described in Chapter 11. The Appendix provides the full definitions of the target (Appendix B.2), comparator (Appendix B.5), and outcome (Appendix B.4 ) cohorts. We will assume the ACEi, THZ, and angioedema cohorts have been instantiated in a table called `scratch.my_cohorts` with cohort definition IDs 1,2, and 3 respectively.

### 13.8.2 Data extraction

We first need to tell R how to connect to the server. CohortMethod uses the DatabaseConnector package, which provides a function called `createConnectionDetails`. Type

`?createConnectionDetails` for the specific settings required for the various database management systems (DBMS). For example, one might connect to a PostgreSQL database using this code:

```
library(CohortMethod)
connDetails <- createConnectionDetails(dbms = "postgresql",
                                       server = "localhost/ohdsi",
                                       user = "joe",
                                       password = "supersecret")

cdmDbSchema <- "my_cdm_data"
cohortDbSchema <- "scratch"
cohortTable <- "my_cohorts"
cdmVersion <- "5"
```

The last four lines define the `cdmDbSchema`, `cohortDbSchema`, and `cohortTable` variables, as well as the CDM version. We will use these later to tell R where the data in CDM format live, where the cohorts of interest have been created, and what version CDM is used. Note that for Microsoft SQL Server, database schemas need to specify both the database and the schema, so for example `cdmDbSchema <- "my_cdm_data.dbo"`.

Now we can tell CohortMethod to extract the cohorts, construct covariates, and extract all necessary data for our analysis:

```
# target and comparator ingredient concepts:
aceI <- c(1335471,1340128,1341927,1363749,1308216,1310756,1373225,
          1331235,1334456,1342439)
thz <- c(1395058,974166,978555,907013)

# Define which types of covariates must be constructed:
cs <- createDefaultCovariateSettings(excludedCovariateConceptIds = c(aceI,
                                                                     thz),
                                     addDescendantsToExclude = TRUE)

#Load data:
cmData <- getDbCohortMethodData(connectionDetails = connectionDetails,
                                cdmDatabaseSchema = cdmDatabaseSchema,
                                oracleTempSchema = NULL,
                                targetId = 1,
                                comparatorId = 2,
                                outcomeIds = 3,
                                studyStartDate = "",
                                studyEndDate = "",
                                exposureDatabaseSchema = cohortDbSchema,
                                exposureTable = cohortTable,
                                outcomeDatabaseSchema = cohortDbSchema,
                                outcomeTable = cohortTable,
                                cdmVersion = cdmVersion,
```

```
                                    firstExposureOnly = FALSE,
                                    removeDuplicateSubjects = FALSE,
                                    restrictToCommonPeriod = FALSE,
                                    washoutPeriod = 0,
                                    covariateSettings = cs)
cmData
```

```
## CohortMethodData object
##
## Treatment concept ID: 1
## Comparator concept ID: 2
## Outcome concept ID(s): 3
```

There are many parameters, but they are all documented in the CohortMethod manual. The `createDefaultCovariateSettings` function is described in the FeatureExtraction package. In short, we are pointing the function to the table containing our cohorts and specify which cohort definition IDs in that table identify the target, comparator and outcome. We instruct that the default set of covariates should be constructed, including covariates for all conditions, drug exposures, and procedures that were found on or before the index date. As mentioned in Section 13.1 we must exclude the target and comparator treatments from the set of covariates, and here we achieve this by listing all ingredients in the two classes, and tell FeatureExtraction to also exclude all descendants, thus excluding all drugs that contain these ingredients.

All data about the cohorts, outcomes, and covariates are extracted from the server and stored in the `cohortMethodData` object. This object uses the package `ff` to store information in a way that ensures R does not run out of memory, even when the data are large, as mentioned in Section 9.4.2.

We can use the generic `summary()` function to view some more information of the data we extracted:

```
summary(cmData)
```

```
## CohortMethodData object summary
##
## Treatment concept ID: 1
## Comparator concept ID: 2
## Outcome concept ID(s): 3
##
## Treated persons: 67166
## Comparator persons: 35333
##
## Outcome counts:
##          Event count Person count
## 3                980          891
##
## Covariates:
```

```
## Number of covariates: 58349
## Number of non-zero covariate values: 24484665
```

Creating the `cohortMethodData` file can take considerable computing time, and it is probably a good idea to save it for future sessions. Because `cohortMethodData` uses `ff`, we cannot use R's regular save function. Instead, we'll have to use the `saveCohortMethodData()` function:

```
saveCohortMethodData(cmData, "AceiVsThzForAngioedema")
```

We can use the `loadCohortMethodData()` function to load the data in a future session.

**Defining new users**

Typically, a new user is defined as first time use of a drug (either target or comparator), and typically a washout period (a minimum number of days prior first use) is used to increase the probability that it is truly first use. When using the CohortMethod package, you can enforce the necessary requirements for new use in three ways:

1. When defining the cohorts.
2. When loading the cohorts using the `getDbCohortMethodData` function, you can use the `firstExposureOnly`, `removeDuplicateSubjects`, `restrictToCommonPeriod`, and `washoutPeriod` arguments.
3. When defining the study population using the `createStudyPopulation` function (see below) using the `firstExposureOnly`, `removeDuplicateSubjects`, `restrictToCommonPeriod`, and `washoutPeriod` arguments.

The advantage of option 1 is that the input cohorts are already fully defined outside of the CohortMethod package, and external cohort characterization tools can be used on the same cohorts used in this analysis. The advantage of options 2 and 3 is that they save you the trouble of limiting to first use yourself, for example allowing you to directly use the DRUG_ERA table in the CDM. Option 2 is more efficient than 3, since only data for first use will be fetched, while option 3 is less efficient but allows you to compare the original cohorts to the study population.

### 13.8.3 Defining the study population

Typically, the exposure cohorts and outcome cohorts will be defined independently of each other. When we want to produce an effect size estimate, we need to further restrict these cohorts and put them together, for example by removing exposed subjects that had the outcome prior to exposure, and only keeping outcomes that fall within a defined risk window. For this we can use the `createStudyPopulation` function:

```
studyPop <- createStudyPopulation(cohortMethodData = cmData,
                                  outcomeId = 3,
                                  firstExposureOnly = FALSE,
                                  restrictToCommonPeriod = FALSE,
                                  washoutPeriod = 0,
```

```
                                    removeDuplicateSubjects = "remove all",
                                    removeSubjectsWithPriorOutcome = TRUE,
                                    minDaysAtRisk = 1,
                                    riskWindowStart = 1,
                                    addExposureDaysToStart = FALSE,
                                    riskWindowEnd = 0,
                                    addExposureDaysToEnd = TRUE)
```

Note that we've set `firstExposureOnly` and `removeDuplicateSubjects` to FALSE, and `washoutPeriod` to 0 because we already applied those criteria in the cohort definitions. We specify the outcome ID we will use, and that people with outcomes prior to the risk window start date will be removed. The risk window is defined as starting on the day after the cohort start date (`riskWindowStart = 1` and `addExposureDaysToStart = FALSE`), and the risk windows ends when the cohort exposure ends (`riskWindowEnd = 0` and `addExposureDaysToEnd = TRUE`), which was defined as the end of exposure in the cohort definition. Note that the risk windows are automatically truncated at the end of observation or the study end date. We also remove subjects who have no time at risk. To see how many people are left in the study population we can always use the `getAttritionTable` function:

```
getAttritionTable(studyPop)
```

```
##                    description targetPersons comparatorPersons          ...
## 1             Original cohorts         67212              35379          ...
## 2 Removed subs in both cohorts         67166              35333          ...
## 3             No prior outcome         67061              35238          ...
## 4 Have at least 1 days at risk         66780              35086          ...
```

### 13.8.4 Propensity scores

We can fit a propensity model using the covariates constructed by the `getDbcohortMethodData()` function, and compute a PS for each person:

```
ps <- createPs(cohortMethodData = cmData, population = studyPop)
```

The `createPs` function uses the Cyclops package to fit a large-scale regularized logistic regression. To fit the propensity model, Cyclops needs to know the hyperparameter value which specifies the variance of the prior. By default Cyclops will use cross-validation to estimate the optimal hyperparameter. However, be aware that this can take a really long time. You can use the `prior` and `control` parameters of the `createPs` function to specify Cyclops' behavior, including using multiple CPUs to speed-up the cross-validation.

Here we use the PS to perform variable-ratio matching:

```
matchedPop <- matchOnPs(population = ps, caliper = 0.2,
                        caliperScale = "standardized logit", maxRatio = 100)
```

Alternatively, we could have used the PS in the `trimByPs`, `trimByPsToEquipoise`, or `stratifyByPs` functions.

### 13.8.5 Outcome models

The outcome model is a model describing which variables are associated with the outcome. Under strict assumptions, the coefficient for the treatment variable can be interpreted as the causal effect. In this case we fit a Cox proportional hazards model, conditioned (stratified) on the matched sets:

```
outcomeModel <- fitOutcomeModel(population = matchedPop,
                                modelType = "cox",
                                stratified = TRUE)
outcomeModel
```

```
## Model type: cox
## Stratified: TRUE
## Use covariates: FALSE
## Use inverse probability of treatment weighting: FALSE
## Status: OK
##
##           Estimate lower .95 upper .95   logRr seLogRr
## treatment   4.3203    2.4531    8.0771 1.4633   0.304
```

### 13.8.6 Running multiple analyses

Often we want to perform more than one analyses, for example for multiple outcomes including negative controls. The CohortMethod offers functions for performing such studies efficiently. This is described in detail in the package vignette on running multiple analyses. Briefly, assuming the outcome of interest and negative control cohorts have already been created, we can specify all target-comparator-outcome combinations we wish to analyse:

```
# Outcomes of interest:
ois <- c(3, 4) # Angioedema, AMI

# Negative controls:
ncs <- c(434165,436409,199192,4088290,4092879,44783954,75911,137951,77965,
         376707,4103640,73241,133655,73560,434327,4213540,140842,81378,432303,
         4201390,46269889,134438,78619,201606,76786,4115402,45757370,433111
         433527,4170770,4092896,259995,40481632,4166231,433577,4231770,440329,
         4012570,4012934,441788,4201717,374375,4344500,139099,444132,196168,
```

```
          432593,434203,438329,195873,4083487,4103703,4209423,377572,40480893,
          136368,140648,438130,4091513,4202045,373478,46286594,439790,81634,
          380706,141932,36713918,443172,81151,72748,378427,437264,194083,
          140641,440193,4115367)

tcos <- createTargetComparatorOutcomes(targetId = 1,
                                       comparatorId = 2,
                                       outcomeIds = c(ois, ncs))

tcosList <- list(tcos)
```

Next, we specify what arguments should be used when calling the various functions described previously in our example with one outcome:

```
aceI <- c(1335471,1340128,1341927,1363749,1308216,1310756,1373225,
          1331235,1334456,1342439)
thz <- c(1395058,974166,978555,907013)

cs <- createDefaultCovariateSettings(excludedCovariateConceptIds = c(aceI,
                                                                     thz),
                                     addDescendantsToExclude = TRUE)

cmdArgs <- createGetDbCohortMethodDataArgs(
  studyStartDate = "",
  studyEndDate = "",
  firstExposureOnly = FALSE,
  removeDuplicateSubjects = FALSE,
  restrictToCommonPeriod = FALSE,
  washoutPeriod = 0,
  covariateSettings = cs)

spArgs <- createCreateStudyPopulationArgs(
  firstExposureOnly = FALSE,
  restrictToCommonPeriod = FALSE,
  washoutPeriod = 0,
  removeDuplicateSubjects = "remove all",
  removeSubjectsWithPriorOutcome = TRUE,
  minDaysAtRisk = 1,
  riskWindowStart = 1,
  addExposureDaysToStart = FALSE,
  riskWindowEnd = 0,
  addExposureDaysToEnd = TRUE)

psArgs <- createCreatePsArgs()

matchArgs <- createMatchOnPsArgs(
  caliper = 0.2,
  caliperScale = "standardized logit",
```

```
  maxRatio = 100)

fomArgs <- createFitOutcomeModelArgs(
  modelType = "cox",
  stratified = TRUE)
```

We then combine these into a single analysis settings object, which we provide a unique analysis ID and some description. We can combine one or more analysis settings objects into a list:

```
cmAnalysis <- createCmAnalysis(
  analysisId = 1,
  description = "Propensity score matching",
  getDbCohortMethodDataArgs = cmdArgs,
  createStudyPopArgs = spArgs,
  createPs = TRUE,
  createPsArgs = psArgs,
  matchOnPs = TRUE,
  matchOnPsArgs = matchArgs
  fitOutcomeModel = TRUE,
  fitOutcomeModelArgs = fomArgs)

cmAnalysisList <- list(cmAnalysis)
```

We can now run the study including all comparisons and analysis settings:

```
result <- runCmAnalyses(connectionDetails = connectionDetails,
                        cdmDatabaseSchema = cdmDatabaseSchema,
                        exposureDatabaseSchema = cohortDbSchema,
                        exposureTable = cohortTable,
                        outcomeDatabaseSchema = cohortDbSchema,
                        outcomeTable = cohortTable,
                        cdmVersion = cdmVersion,
                        outputFolder = outputFolder,
                        cmAnalysisList = cmAnalysisList,
                        targetComparatorOutcomesList = tcosList)
```

The `result` object contains references to all the artifacts that were created. For example, we can retrieve the outcome model for AMI:

```
omFile <- result$outcomeModelFile[result$targetId == 1 &
                                  result$comparatorId == 2 &
                                  result$outcomeId == 4 &
                                  result$analysisId == 1]
outcomeModel <- readRDS(file.path(outputFolder, omFile))
outcomeModel
```

```
## Model type: cox
## Stratified: TRUE
## Use covariates: FALSE
## Use inverse probability of treatment weighting: FALSE
## Status: OK
##
##           Estimate lower .95 upper .95   logRr seLogRr
## treatment   1.1338    0.5921    2.1765 0.1256   0.332
```

We can also retrieve the effect size estimates for all outcomes with one command:

```
summ <- summarizeAnalyses(result, outputFolder = outputFolder)
head(summ)
```

```
##     analysisId targetId comparatorId outcomeId        rr    ci95lb  ...
## 1            1        1            2     72748 0.9734698 0.5691589  ...
## 2            1        1            2     73241 0.7067981 0.4009951  ...
## 3            1        1            2     73560 1.0623951 0.7187302  ...
## 4            1        1            2     75911 0.9952184 0.6190344  ...
## 5            1        1            2     76786 1.0861746 0.6730408  ...
## 6            1        1            2     77965 1.1439772 0.5173222  ...
```

## 13.9 Study outputs

Our estimates are only valid if several assumptions have been met. We use a wide set of diagnostics to evaluate whether this is the case. These are available in the results produced by the R package generated by ATLAS, or can be generated on the fly using specific R functions.

### 13.9.1 Propensity scores and model

We first need to evaluate whether the target and comparator cohort are to some extend comparable. For this we can compute the Area Under the Receiver Operator Curve (AUC) statistic for the propensity model. An AUC of 1 indicates the treatment assignment was completely predictable based on baseline covariates, and that the two groups are therefore incomparable. We can use the `computePsAuc` function to compute the AUC, which in our example is 0.79. Using the `plotPs` function, we can also generate the preference score distribution as shown in Figure 13.17. Here we see that for many people the treatment they received was predictable, but there is also a large amount of overlap, indicating that adjustment can be used to select comparable groups.

In general it is a good idea to also inspect the propensity model itself, and especially so if the model is very predictive. That way we may discover which variables are most predictive. Table 13.7 shows the top predictors in our propensity model. Note that if a variable is too predictive, the CohortMethod

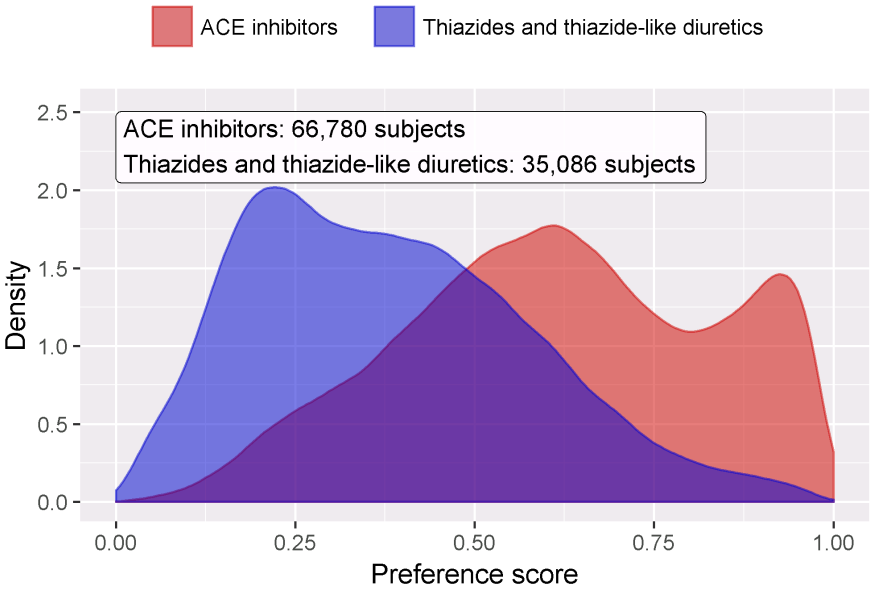



Figure 13.17: Preference score distribution.

package will throw an informative error rather than attempt to fit a model that is already known to be perfectly predictive.

Table 13.7: Top 10 predictors in the propensity model for ACEi and THZ. Positive values mean subjects with the covariate are more likely to receive the target treatment.

| Beta | Covariate |
|---|---|
| -1.42 | condition_era group during day -30 through 0 days relative to index: Edema |
| -1.11 | drug_era group during day 0 through 0 days relative to index: Potassium Chloride |
| 0.68 | age group: 05-09 |
| 0.64 | measurement during day -365 through 0 days relative to index: Renin |
| 0.63 | condition_era group during day -30 through 0 days relative to index: Urticaria |
| 0.57 | condition_era group during day -30 through 0 days relative to index: Proteinuria |
| 0.55 | drug_era group during day -365 through 0 days relative to index: INSULINS AND ANALOGUES |
| -0.54 | race = Black or African American |
| 0.52 | (Intercept) |
| 0.50 | gender = MALE |

If a variable is found to be highly predictive, there are two possible conclusions: Either we find that the variable is clearly part of the exposure itself and should be removed before fitting the model, or else we must conclude that the two populations are truly incomparable, and the analysis must be stopped.

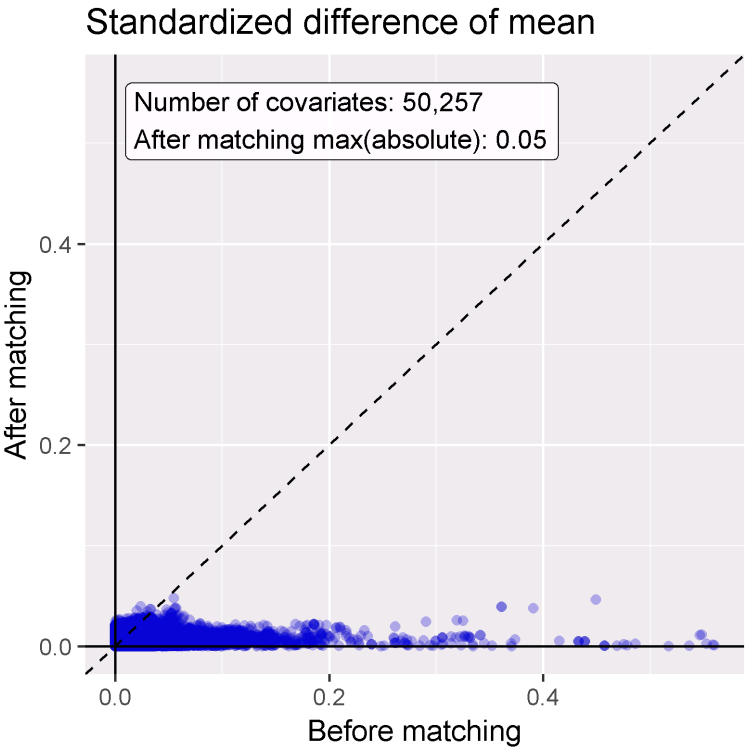



Figure 13.18: Covariate balance, showing the absolute standardized difference of mean before and after propensity score matching. Each blue dot represents a covariate.

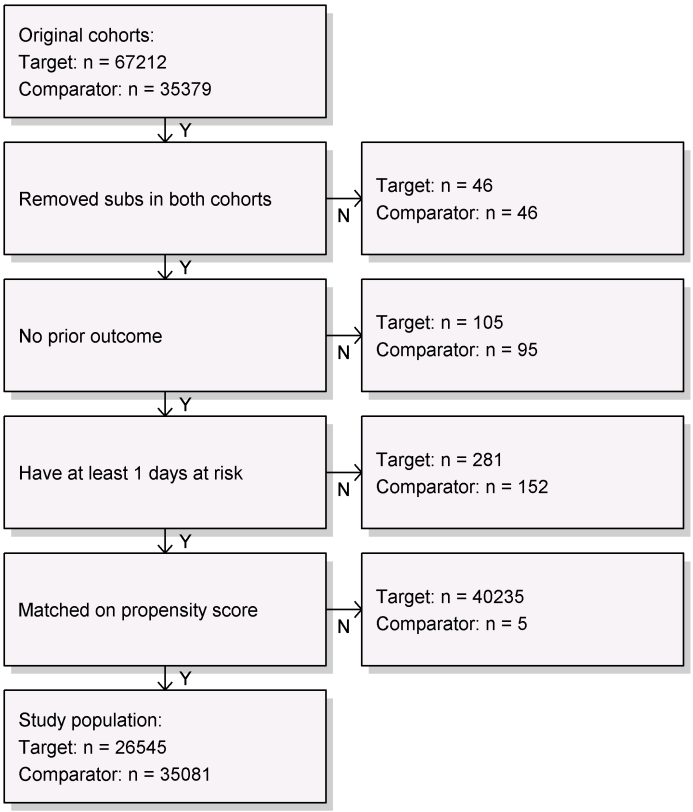



Figure 13.19: Attrition diagram. The counts shown at the top are those that meet our target and comparator cohort definitions. The counts at the bottom are those that enter our outcome model, in this case a Cox regression.

attrition of subjects in our study using the `drawAttritionDiagram` function as shown in Figure 13.19.

Since the sample size is fixed in retrospective studies (the data has already been collected), and the true effect size is unknown, it is therefore less meaningful to compute the power given an expected effect size. Instead, the CohortMethod package provides the `computeMdrr` function to compute the minimum detectable relative risk (MDRR). In our example study the MDRR is 1.69.

To gain a better understanding of the amount of follow-up available we can also inspect the distribution of follow-up time. We defined follow-up time as time at risk, so not censored by the occurrence of the outcome. The `getFollowUpDistribution` can provide a simple overview as shown in Figure 13.20, which suggests the follow-up time for both cohorts is comparable.

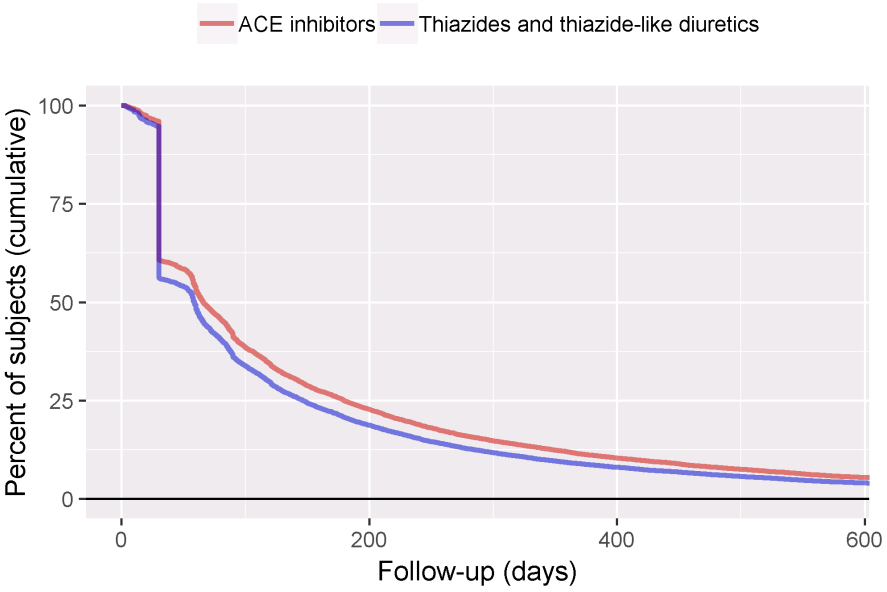



Figure 13.20: Distribution of follow-up time for the target and comparator cohorts.

### 13.9.4 Kaplan Meier

One last check is to review the Kaplan Meier plot, showing the survival over time in both cohorts. Using the `plotKaplanMeier` function we can create 13.21, which we can check for example if our assumption of proportionality of hazards holds. The Kaplan-Meier plot automatically adjusts for stratification or weighting by PS. In this case, because variable-ratio matching is used, the survival curve for the comparator groups is adjusted to mimick what the curve had looked like for the target group had they been exposued to the comparator instead.

### 13.9.5 Effect size estimate

We observe a hazard ratio of 4.32 (95% confidence interval: 2.45 - 8.08) for angioedema, which tells us that ACEi appear to increase the risk of angioedema compared to THZ. Similarly, we observe a hazard ratio of 1.13 (95% confidence interval: 0.59 - 2.18) for AMI, suggesting little or no effect for AMI. Our diagnostics, as reviewed earlier, give no reason for doubt. However, ultimately the quality of this evidence, and whether we choose to trust it, depends on many factors that are not covererd by the study diagnostics as described in Chapter 15.

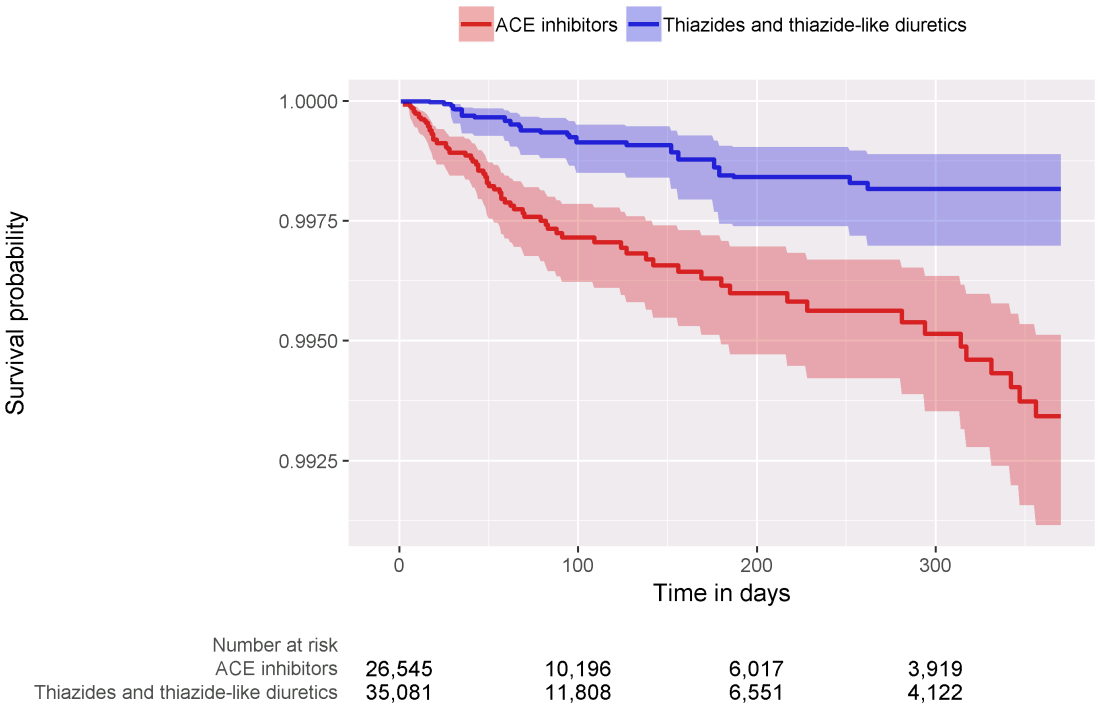



Figure 13.21: Kaplan Meier plot.

## 13.10 Summary

- Population-level estimation aims to infer causal effects from observational data.
- The **counterfactual**, what would have happened if the subject had received an alternative exposure or no exposure, cannot be observed.
- Different designs aim to construct the counterfactual in different ways.
- The various designs as implemented in the OHDSI Methods Library provide diagnostics to evaluate whether the assumptions for creating an appropriate counterfactual have been met.

## 13.11 Excercises

Note: The excercises still have to be defined. The idea is to require readers to define a study that estimates the effect of celecoxib on GI bleed, compared to diclofenac. For this they must use the Eunomia package, which is still under development.

# Chapter 14

# Patient Level Prediction

*Chapter leads: Peter Rijnbeek & Jenna Reps*

Clinical decision making is a complicated task in which the clinician has to infer a diagnosis or treatment pathway based on the available medical history of the patient and the current clinical guidelines. Clinical prediction models have been developed to support this decision-making process and are used in clinical practice in a wide spectrum of specialties. These models predict a diagnostic or prognostic outcome based on a combination of patient characteristics, e.g. demographic information, disease history, treatment history.

The number of publications describing clinical prediction models has increased strongly over the last 10 years. Most currently-used models are estimated using small datasets and consider only a small set of patient characteristics. This low sample size, and thus low statistical power, forces the data analyst to make strong modelling assumptions. The selection of the limited set of patient characteristics is strongly guided by the expert knowledge at hand. This contrasts sharply with the reality of modern medicine wherein patients generate a rich digital trail, which is well beyond the power of any medical practitioner to fully assimilate. Presently, health care is generating a huge amount of patient-specific information stored in Electronic Health Records (EHRs). This includes structured data in the form of diagnose, medication, laboratory test results, and unstructured data contained in clinical narratives. It is unknown how much predictive accuracy can be gained by leveraging the large amount of data originating from the complete EHR of a patient.

Advances in machine learning for large dataset analysis have led to increased interest in applying patient-level prediction on this type of data. However, many published efforts in patient-level prediction do not follow the model development guidelines, fail to perform extensive external validation, or provide insufficient model details which limits the ability of independent researchers to reproduce the models and perform external validation. This makes it hard to fairly evaluate the predictive performance of the models and reduces the likelihood of the model being used appropriately in clinical practice. To improve standards, several papers have been written detailing guidelines for best practices in developing and reporting prediction models. For example, the Transparent Reporting of a multivariable prediction model for Individual Prognosis Or Diagnosis (TRIPOD) statement [1]

[1] https://www.equator-network.org/reporting-guidelines/tripod-statement/

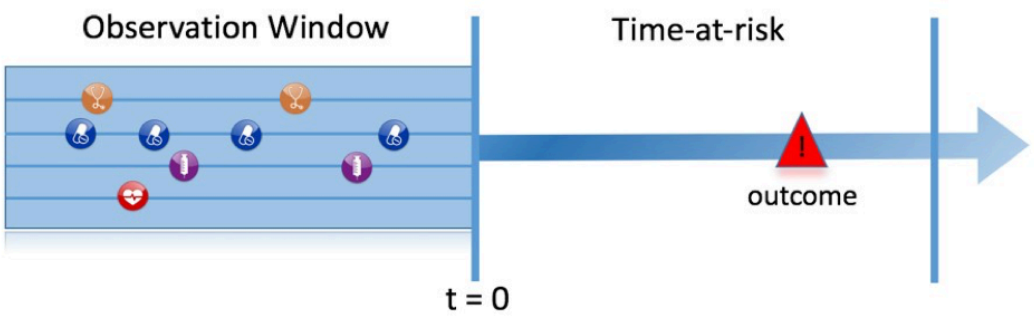



Figure 14.1: The prediction problem.

provides clear recommendations for reporting prediction model development and validation and addresses some of the concerns related to transparency.

Massive-scale, patient-specific predictive modeling has become reality due to OHDSI, where the common data model (CDM) allows for uniform and transparent analysis at an unprecedented scale. The databases available in the CDM contain rich data to build highly predictive large-scale models and also provide immediate opportunity to serve large communities of patients who are in most need of improved quality of care. Such models can inform truly personalized medical care leading hopefully to sharply improved patient outcomes.

In this chapter we describe OHDSI's standardized framework for patient-level prediction, (Reps et al., 2018) and discuss the PatientLevelPrediction R package that implements established best practices. We start with providing the necessary theory behind the development and evaluation of patient-level prediction and provide a high-level overview of the implemented machine learning algorithms. We then discuss an example prediction problem and provide step-by-step guidance on its definition and implementation using ATLAS or custom R code. Finally, we review two Shiny apps that allow viewing the study outputs. One app explores a single prediction model, while the other explores many models at once.

## 14.1 The prediction problem

Figure 14.1, illustrates the prediction problem we address. Among a population at risk, we aim to predict which patients at a defined moment in time (t = 0) will experience some outcome during a time-at-risk. Prediction is done using only information about the patients in an observation window prior to that moment in time.

As shown in Table 14.1, to define a prediction problem we have to define t=0 by a target cohort, the outcome we like to predict by an outcome cohort, and the time-at-risk. We define the standard prediction question as:

> Among *[target cohort definition]*, who will go on to have *[outcome cohort definition]* within *[time-at-risk period]*?

Furthermore, we have to make design choices for the model we like to develop, and determine the observational datasets to perform internal and external validation.

Table 14.1: Main design choices in a prediction design.

| Choice | Description |
|---|---|
| Target cohort | A cohort for whom we wish to predict |
| Outcome cohort | A cohort representing the outcome we wish to predict |
| Time-at-risk | For what time relative to t=0 do we want to make the prediction? |
| Model | What algorithms using which parameters do we want use, and what predictor variables do we want to include? |

This conceptual framework works for all type of prediction problems, for example:

- Disease onset and progression
  - **Structure**: Among patients who are newly diagnosed with *[a disease]*, who will go on to have *[another disease or complication]* within *[time horizon from diagnosis]*?
  - **Example**: Among newly diagnosed atrial fibrillation patients, who will go on to have ischemic stroke in the next three years?
- Treatment choice
  - **Structure**: Among patients with *[indicated disease]* who are treated with either *[treatment 1]* or *[treatment 2]*, which patients were treated with *[treatment 1]*?
  - **Example**: Among patients with atrial fibrillation who took either warfarin or rivaroxaban, which patients gets warfarin? (e.g. for a propensity model)
- Treatment response
  - **Structure**: Among new users of *[a treatment]*, who will experience *[some effect]* in *[time window]* ?
  - **Example**: Which patients with diabetes who start on metformin stay on metformin for three years?
- Treatment safety
  - **Structure**: Among new users of *[a treatment]*, who will experience *[adverse event]* in *[time window]*?
  - **Example**: Among new users of warfarin, who will have a gastrointestinal bleed in one year?
- Treatment adherence
  - **Structure**: Among new users of *[a treatment]*, who will achieve *[adherence metric]* at *[time window]*?
  - **Example**: Which patients with diabetes who start on metformin achieve >=80% proportion of days covered at one year?

## 14.2 Data extraction

The observational data we use in OHDSI consist of time-stamped records of interactions of the patient with the healthcare system, as well as anonymized patient details such as gender and year of birth, stored in the CDM (Chapter 5). To use this information in a prediction problem, we must convert this data into two datasets:

1. A set of **covariates** (also referred to as "features" or "independent variables"). These describe the characteristics of the patients. Covariates can include age, gender, presence of specific condition and exposure codes in a patient's record, etc.
2. A set describing the **outcome status** (also referred to as the "labels" or "classes"). Did a patient experience the outcome of interest in the time-at-risk?

When creating a predictive model we use a process know as supervised learning - a form of machine learning - to infer the relationship between the covariates and the outcome status. Once the model is created, we can apply it to patients where we know their characteristics (their covariates), but do not know their outcome status, to create a prediction.

Converting the data in the CDM to these two datasets requires selecting a set of people, and for these people selecting a specific date. We will refer to this date as the index date. Data prior to (and on) the index date is used to construct the covariates. Covariates are typically constructed using the FeatureExtraction package, described in more detail in Chapter 12. Data after (or on) the index date is used to construct the outcome status. The group of people and their index date are defined by the **target cohort**. The outcome status is determined by the **time-at-risk**, which is usually defined relative to the target cohort start and end date, and the **outcome cohort**; If the outcome occurs within the time-at-risk, the outcome status is "positive".

### 14.2.1 Data extraction example

Table 14.2 shows an example COHORT table with two cohorts. The cohort with cohort definition ID 1 is the target cohort (e.g. "people recently diagnosed with atrial fibrillation"). Cohort definition ID 2 implies the outcome cohort (e.g. "stroke").

Table 14.2: Example COHORT table. For simplicity the cohort_end_date has been omitted.

| cohort_definition_id | subject_id | cohort_start_date |
|---|---|---|
| 1 | 1 | 2000-06-01 |
| 1 | 2 | 2001-06-01 |
| 2 | 2 | 2001-07-01 |

Table 14.3 provides an example CONDITION_OCCURRENCE table. Concept ID 320128 refers to "Essential hypertension".

Table 14.3: Example CONDITION_OCCURRENCE table. For simplicity only three columns are shown.

| person_id | condition_concept_id | condition_start_date |
|---|---|---|
| 1 | 320128 | 2000-10-01 |
| 2 | 320128 | 2001-05-01 |

Based on this example data, and assuming the time at risk is the year following the index date (the target cohort start date), we can construct the covariates and the outcome status. A covariate indicating "Essential hypertension in the year prior" will have the value 0 (not present) for person ID 1 (the condition occurred *after* the index date), and the value 1 (present) for person ID 2. Similarly, the outcome status will be 0 for person ID 1 (this person had no entry in the outcome cohort), and 1 for person ID 2 (the outcome occurred within a year following the index date).

### 14.2.2 Missingness

Observational healthcare data rarely reflects whether data is missing. In the prior example, we simply observed the person with ID 1 had no essential hypertension occurrence prior to the index date. This could be because the condition was not present at that time, or because it was not recorded. There is no way to distinguish between the two. For machine learning this does not matter as much as one might think. If a covariate is predictive of the outcome status it will end up in the model, else not. However, we should be aware that our interpretation of covariates should be nuanced: it is not the actual condition that is the predictor, but rather the recording of the condition in the data.

## 14.3 Fitting the model

When fitting a model (using supervised learning) we are trying to establish the relationship between the covariates and the observed outcome status, so that if we do not yet know the outcome status, we can predict it. If we consider the situation where we have two covariates (for example systolic and diastolic blood pressure), then we can represent each patient as a plot in two dimensional space as shown in Figure 14.2. The shape of a data points corresponds to the patient's outcome status (e.g. stroke). The idea of supervised learning is to generalize what we see and fill in where there are no current data points. A supervised learning model will try to partition the space via a decision boundary that aims to minimize the cases where the outcome status does not match the models prediction. Different supervised learning techniques lead to different decision boundaries and there are often hyper-parameters that can impact the complexity of the decision boundary.

In Figure 14.2 we can see three different decision boundaries. The boundaries are used to infer the outcome status of any new data point. If a new data point falls into the shaded area then the model will predict "has outcome", otherwise it will predict "no outcome". Ideally a decision boundary should perfectly partition the two classes. However, generalizability is an issue, as complex models can "overfit" the data; boundaries may be fit too closely and may not work for new data. For example,

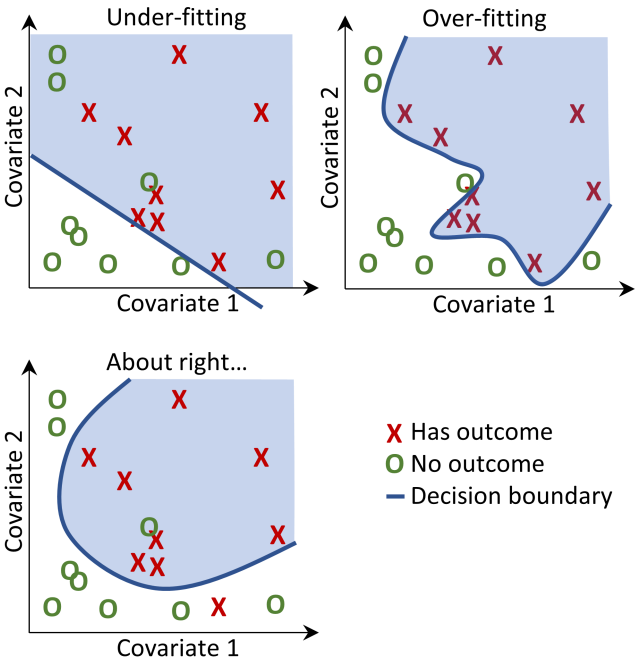



Figure 14.2: Decision boundary.

if the data contains noise, with mislabelled or incorrectly positioned data points, we would not want to fit our model to that noise. We may pref to define a decision boundary that does not perfectly discriminate those with known outcome status, to get a model that better predicts for now, previously unseen patients. We want a model that appears to partition the labelled data well but is also as simple as possible. Techniques such as regularization aim to maximize model performance on the data with known outcome status while minimizing complexity. Complexity can also be controlled by picking classifier hyper-parameters such that a simpler decision boundary is used.

Another way to think about supervised learning is finding a function that maps from a patient's covariates to their label: $p(outcome\ status = 1|covariates) = f(covariates)$. Each supervised learning algorithm has a different way to learn the mapping function and the no free lunch theorem states that no one algorithm is always going to outperform the others. The performance of each type of supervised learning algorithm depends on how the labelled data points are distributed in space. Therefore we recommend trying multiple supervised learning techniques with various hyper-parameter settings when developing patient-level prediction models.

The following algorithms are available in the PatientLevelPrediction package:

### 14.3.1 Regularized logistic regression

LASSO (least absolute shrinkage and selection operator) logistic regression belongs to the family of generalized linear models, where a linear combination of the variables is learned and finally a logistic function maps the linear combination to a value between 0 and 1. The LASSO regularization adds a

cost based on model complexity to the objective function when training the model. This cost is the sum of the absolute values of the linear combination of the coefficients. The model automatically performs feature selection by minimizing this cost. We use the Cyclops (Cyclic coordinate descent for logistic, Poisson and survival analysis) package to perform large-scale regularized logistic regression.

Table 14.4: Hyper-parameters for the regularized logistic regression.

| Parameter | Description | Typical values |
|---|---|---|
| Starting variance | The starting variance of the prior distribution. | 0.1 |

Note that the variance is optimized by maximizing the out-of-sample likelihood in a cross-validation, so the starting variance has little impact on the performance of the resulting model. However, picking a starting variance that is too far from the optimal value may lead to long fitting time.

### 14.3.2 Gradient boosting machines

Gradient boosting machines is a boosting ensemble technique and in our framework it combines multiple decision trees. Boosting works by iteratively adding decision trees but adds more weight to the data-points that are misclassified by prior decision trees in the cost function when training the next tree. We use Extreme Gradient Boosting, which is an efficient implementation of the gradient boosting framework implemented in the xgboost R package available from CRAN.

Table 14.5: Hyper-parameters for gradient boosting machines.

| Parameter | Description | Typical values |
|---|---|---|
| mtry | Number of features in each tree | ? |
| ntree | Number of trees | ? |
| maxDepth | Max levels in a tree | ? |
| minRows | Min data points in a node | ? |
| balance | Should class sizes be balanced? | ? |

### 14.3.3 Random forest

Random forest is a bagging ensemble technique that combines multiple decision trees. The idea behind bagging is to reduce the likelihood of overfitting, by using weak classifiers, but combining multiple diverse weak classifiers into a strong classifier. Random forest accomplishes this by training multiple decision trees but only using a subset of the variables in each tree and the subset of variables differ between trees. Our packages uses the sklearn implementation of Random Forest in Python.

Table 14.6: Hyper-parameters for random forests.

| Parameter | Description | Typical values |
|---|---|---|
| mtry | Number of features in each tree | ? |
| ntree | Number of trees | ? |
| maxDepth | Max levels in a tree | ? |
| minRows | Min data points in a node | ? |
| balance | Should class sizes be balanced? | ? |

### 14.3.4 K-nearest neighbors

K-nearest neighbors (KNN) is an algorithm that uses some distance metric to find the K closest labelled data-points to a new unlabelled data-point. The prediction of the new data-points is then the most prevalent class of the K-nearest labelled data-points. There is a sharing limitation of KNN, as the model requires labelled data to perform the prediction on new data, and it is often not possible to share this data across data sites. We included the BigKnn package developed in OHDSI which is a large scale KNN classifier.

Table 14.7: Hyper-parameters for K-nearest neighbors.

| Parameter | Description | Typical values |
|---|---|---|
| k | Number of neighbors | ? |
| weighted | Weight by inverse frequency? | ? |

### 14.3.5 Naive Bayes

The Naive Bayes algorithm applies the Bayes theorem with the naive assumption of conditional independence between every pair of features given the value of the class variable. Based on the likelihood the data belongs to a class and the prior distribution of the class, a posterior distribution is obtained. Naive Bayes has no hyper-parameters.

### 14.3.6 AdaBoost

AdaBoost is a boosting ensemble technique. Boosting works by iteratively adding classifiers but adds more weight to the data-points that are misclassified by prior classifiers in the cost function when training the next classifier. We use the sklearn AdaboostClassifier implementation in Python.

Table 14.8: Hyper-parameters for AdaBoost.

| Parameter | Description | Typical values |
|---|---|---|
| nEstimators | The maximum number of estimators at which boosting is terminated | ? |
| learningRate | Learning rate shrinks the contribution of each classifier by learning_rate. There is a trade-off between learningRate and nEstimators | ? |

### 14.3.7 Decision Tree

A decision tree is a classifier that partitions the variable space using individual tests selected using a greedy approach. It aims to find partitions that have the highest information gain to separate the classes. The decision tree can easily overfit by enabling a large number of partitions (tree depth) and often needs some regularization (e.g., pruning or specifying hyper-parameters that limit the complexity of the model). We use the sklearn DecisionTreeClassifier implementation in Python.

Table 14.9: Hyper-parameters for decision trees.

| Parameter | Description | Typical values |
|---|---|---|
| maxDepth | The maximum depth of the tree | ? |
| minSamplesSplit | ? | ? |
| minSamplesLeaf | ? | ? |
| minImpuritySplit | ? | ? |
| classWeight | “Balance”” or “None” | ? |

### 14.3.8 Multilayer Perceptron

Neural networks containing multiple layers of nodes that weight their inputs using a non-linear function. The first layer is the input layer, the last layer is the output layer, and in between are the hidden layers. Neural networks are generally trained using back-propagation, meaning the training input is propagated forward through the network to produce an output, the error between the output and the outcome status is computed, and this error is propagated backwards through the network to update the linear function weights.

| Parameter | Description | Typical values |
|---|---|---|

Table 14.10: Hyper-parameters for Multilayer Perceptrons.

| Parameter | Description | Typical values |
|---|---|---|
| size | The number of hidden nodes | ? |
| alpha | The l2 regularization) | ? |

### 14.3.9 Deep Learning

Deep learning such as deep nets, convolutional neural networks or recurrent neural networks are similar to Multilayer Perceptrons but have multiple hidden layers that aim to learn latent representations useful for prediction. In a separate vignette in the PatientLevelPrediction package we describe these models and hyper-parameters in more detail.

### 14.3.10 Other algorithms

Other algorithms can be added to the patient-level prediction framework. This is out-of-scope for this chapter. Details can be found in the "Adding Custom Patient-Level Prediction Algorithms" vignette in the PatientLevelPrediction package.

## 14.4 Evaluating prediction models

### 14.4.1 Evaluation Types

We can evaluate a prediction model by measuring the agreement between the model's prediction and observed outcome status, which means we need data where the outcome status is known.

For evaluation we must use a different dataset than was used to develop the model, or else we run the risk of favoring models that are over-fitted (see Section 14.3) and may not perform well for new patients.

We distinguish between

- **Internal validation**: Using different sets of data extracted from the same database to develop and evaluate the model.
- **External validation**: Developing the model in one database, and evaluating in another database.

There are two ways to perform internal validation:

- A **holdout set** approach splits the labelled data into two independent sets: a train set and a test set (the hold out set). The train set is used to learn the model and the test set is used to evaluate it. We can simply divide our patients randomly into a train and test set, or we may choose to:
  - Split the data based on time (temporal validation), for example training on data before a specific date, and evaluating on data after that date. This may inform us on whether our model generalizes to different time periods.
  - Split the data based on geographic location (spatial validation).
- **Cross validation** is useful when the data are limited. The data is split into $n$ equally-sized sets, where $n$ needs to be prespecified (e.g. $n = 10$). For each of these sets a model is trained on all data except the data in that set, and used to generate predictions for the held-out set. In this way, all data is used once to evaluate the model-building algorithm. In the patient-level prediction framework we use cross validation to pick the optimal hyper-parameters.

Note that some may consider bootstrapping to be another approach to internal validation, which is often used specifically to express the uncertainty around a model's prediction, typically as confidence intervals. We currently do not use bootstrapping in the patient-level prediction framework. TODO: maybe elaborate on this.

External validation is when a model trained in one database is validated on a data from another database. This is important as different databases may represent different patient populations, but also perhaps different healthcare systems and different data-capture processes. External validation can therefore inform us on how well a model will perform outside of the settings it was developed in.

### 14.4.2 Performance Metrics

#### Threshold measures

A prediction model assigns a value between 0 and 1 for each patient corresponding to the risk of the patient having the outcome during the time at risk. A value of 0 means 0% risk, a value of 0.5 means 50% risk and a value of 1 means 100% risk. Common metrics such as accuracy, sensitivity, specificity, positive predictive value can be calculated by first specifying a threshold that is used to classify patients as having the outcome or not during the time at risk. For example, given Table 14.11, if we set the threshold as 0.5, the patients 1, 3, 7 and 10 have a predicted risk greater than or equal to the threshold of 0.5 so they would be predicted to have the outcome. All other patients had a predicted risk less than 0.5, so would be predicted to not have the outcome.

Table 14.11: Example of using a threshold on the predicted probability.

| Patient ID | Predicted risk | Predicted class at 0.5 threshold | Has outcome during time-at-risk | Type |
|---|---|---|---|---|
| 1 | 0.8 | 1 | 1 | TP |
| 2 | 0.1 | 0 | 0 | TN |
| 3 | 0.7 | 1 | 0 | FP |

| Patient ID | Predicted risk | Predicted class at 0.5 threshold | Has outcome during time-at-risk | Type |
|---|---|---|---|---|
| 4 | 0 | 0 | 0 | TN |
| 5 | 0.05 | 0 | 0 | TN |
| 6 | 0.1 | 0 | 0 | TN |
| 7 | 0.9 | 1 | 1 | TP |
| 8 | 0.2 | 0 | 1 | FN |
| 9 | 0.3 | 0 | 0 | TN |
| 10 | 0.5 | 1 | 0 | FP |

If a patient is predicted to have the outcome and has the outcome (during the time-at-risk) then this is called as a true positive (TP). If a patient is predicted to have the outcome but does not have the outcomethen this is called a false positive (FP). If a patient is predicted to not have the outcome and does not have the outcome then this is called a true negative (TN). Finally, if a patient is predicted to not have the outcome but does have the outcome then this is called a false negative (FN).

The following threshold based metrics are:

- accuracy: $(TP + TN)/(TP + TN + FP + FN)$
- sensitivity: $TP/(TP + FN)$
- specificity: $TN/(TN + FP)$
- positive predictive value: $TP/(TP + FP)$

Note that these values can either decrease or increase if the threshold is lowered. Lowering the threshold of a classifier may increase the denominator, by increasing the number of results returned. If the threshold was previously set too high, the new results may all be true positives, which will increase positive predictive value. If the previous threshold was about right or too low, further lowering the threshold will introduce false positives, decreasing positive predictive value. For sensitivity the denominator does not depend on the classifier threshold ($TP + FN$ is a constant). This means that lowering the classifier threshold may increase sensitivity, by increasing the number of true positive results. It is also possible that lowering the threshold may leave sensitivity unchanged, while the positive predictive value fluctuates.

**Discrimination**

Discrimination is the ability to assign a higher risk to patients who will experience the outcome during the time at risk. The Receiver Operating Characteristics (ROC) is determined by plotting 1 – specificity on the x-axis and sensitivity on the y-axis at all possible thresholds. An example ROC plot is presented later in this chapter in Figure 14.17. The area under the receiver operating characteristic curve (AUC) gives an overall measure of discrimination where a value of 0.5 corresponds to randomly assigning the risk and a value of 1 means perfect discrimination. In reality, most prediction models obtain AUCs between 0.6-0.8.

For rare outcomes even a model with a high AUC may not be practical, because for every positive above a given threshold there could also be many negatives (i.e. the positive predictive value will be low). Depending on the severity of the outcome and cost (health risk and/or monetary) of some

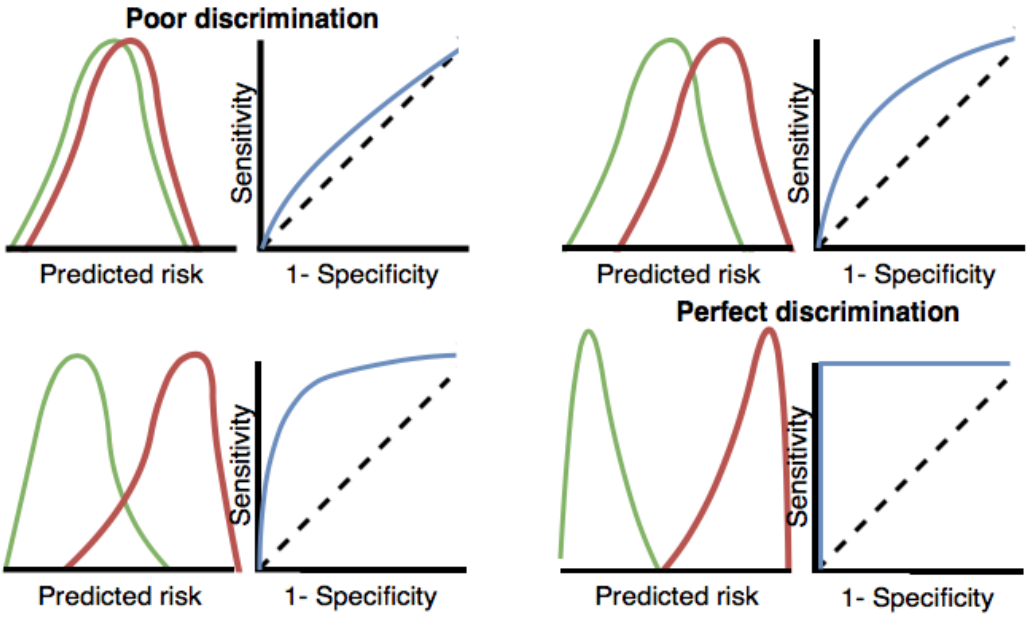



Figure 14.3: How the ROC plots are linked to discrimination. If the two classes have similar distributions of prediced risk, the ROC will be close to the diagonal, with AUC close to 0.5.

intervention, a high false positive rate may be impractical. When the outcome is rare another measure known as the area under the precision-recall curve (AUPRC) is therefore recommended. The AUPRC is the area under the line generated by plotting the sensitivity on the x-axis (also known as the recall) and the positive predictive value (also known as the precision) on the y-axis.

The AUC provides a way to determine how different the predicted risk distributions are between the patients who experience the outcome during the time at risk and those who do not. If the AUC is high, then the distributions will be mostly disjointed, whereas when there is a lot of overlap, the AUC will be closer to 0.5, see Figure 14.3.

**Calibration**

Calibration is the ability of the model to assign the correct risk. For example, if the model assigned one hundred patients a risk of 10% then ten of the patients should experience the outcome during the time at risk. If the model assigned 100 patients a risk of 80% then eighty of the patients should experience the outcome during the time at risk. The calibration is generally calculated by partitioning the patients into deciles based on the predicted risk and in each group calculating the mean predicted risk and the fraction of the patients who experienced the outcome during the time at risk. We then plot these ten points (predicted risk on the y-axis and observed risk on the x-axis) and see whether they fall on the x = y line, indicating the model is well calibrated. An example calibration plot is presented later in this chapter in Figure 14.18. We also fit a linear model using the points to calculate the intercept (which should be close to zero) and the gradient (which should be close to one). If the gradient is greater than one then the model is assigning a higher risk than the true risk and if the

gradient is less than one the model is assigning a lower risk than the true risk.

## 14.5 Designing a patient-level prediction Study

In this section we will demonstrate how to design a prediction study. The first step is to clearly define the prediction problem. Interestingly, in many published papers the prediction problem is poorly defined, for example it is unclear how the index date (start of the target cohort) is defined. A poorly defined prediction problem does not allow for external validation by others let alone implementation in clinical practice. In the patient-level prediction framework we enforce proper specification of the prediction problem by requiring the key choices defined in Table 14.1 to be explicitly defined. Here we will walk through this process using a "treatment safety" type prediction problem as an example.

### 14.5.1 Problem definition

Angioedema is a well known side-effect of ACE inhibitors, and the incidence of angioedema reported in the labeling for ACE inhibitors is in the range of 0.1% to 0.7% (Byrd et al., 2006). Monitoring patients for this adverse effect is important, because although angioedema is rare, it may be life-threatening, leading to respiratory arrest and death (Norman et al., 2013). Further, if angioedema is not initially recognized, it may lead to extensive and expensive workups before it is identified as a cause (Norman et al., 2013; Thompson and Frable, 1993). Other than the higher risk among African-American patients, there are no known predisposing factors for the development of ACE inhibitor related angioedema (Byrd et al., 2006). Most reactions occur within the first week or month of initial therapy and often within hours of the initial dose (Cicardi et al., 2004). However, some cases may occur years after therapy has begun (O'Mara and O'Mara, 1996). No diagnostic test is available that specifically identifies those at risk. If we could identify those at risk, doctors could act, for example by discontinuing the ACE inhibitor in favor of another hypertension drug.

We will apply the patient-level prediction framework to observational healthcare data to address the following patient-level prediction question:

> Among patients who have just started on an ACE inhibitor for the first time, who will experience angioedema in the following year?

### 14.5.2 Study population definition

The final study population in which we will develop our model is often a subset of the target cohort, because we may for example apply criteria that are dependent on the outcome, or we want to perform sensitivity analyses with sub-populations of the target cohort. For this we have to answer the following questions:

- *What is the minimum amount of observation time we require before the start of the target cohort?* This choice could depend on the available patient time in the training data, but also on the time we expect to be available in the data sources we want to apply the model on in the

future. The longer the minimum observation time, the more baseline history time is available for each person to use for feature extraction, but the fewer patients will qualify for analysis. Moreover, there could be clinical reasons to choose a short or longer look-back period. For our example, we will use a 365-day prior history as look-back period (washout period).

- *Can patients enter the target cohort multiple times?* In the target cohort definition, a person may qualify for the cohort multiple times during different spans of time, for example if they had different episodes of a disease or separate periods of exposure to a medical product. The cohort definition does not necessarily apply a restriction to only let the patients enter once, but in the context of a particular patient-level prediction problem we may want to restrict the cohort to the first qualifying episode. In our example, a person can only enter the target cohort once since our criteria was based on first use of an ACE inhibitor.

- *Do we allow persons to enter the cohort if they experienced the outcome before?* Do we allow persons to enter the target cohort if they experienced the outcome before qualifying for the target cohort? Depending on the particular patient-level prediction problem, there may be a desire to predict incident first occurrence of an outcome, in which case patients who have previously experienced the outcome are not at risk for having a first occurrence and therefore should be excluded from the target cohort. In other circumstances, there may be a desire to predict prevalent episodes, whereby patients with prior outcomes can be included in the analysis and the prior outcome itself can be a predictor of future outcomes. For our prediction example, we will choose not to include those with prior angioedema.

- *How do we define the period in which we will predict our outcome relative to the target cohort start?* We have to make two decisions to answer this question. First, does the time-at-risk window start at the date of the start of the target cohort or later? Arguments to make it start later could be that we want to avoid outcomes that were entered late in the record that actually occurred before the start of the target cohort or we want to leave a gap where interventions to prevent the outcome could theoretically be implemented. Second, we need to define the time-at-risk by setting the risk window end, as some specification of days offset relative to the target cohort start or end dates. For our problem we will predict in a time-at-risk window starting 1 day after the start of the target cohort up to 365 days later.

- *Do we require a minimum amount of time-at-risk?* We have to decide if we want to include patients that did not experience the outcome but did leave the database earlier than the end of our time-at-risk period. These patients may experience the outcome when we no longer observe them. For our prediction problem we decide to answer this question with "yes", requiring a minimum time-at-risk for that reason. Furthermore, we have to decide if this constraint also applies to persons who experienced the outcome or we will include all persons with the outcome irrespective of their total time at risk. For example, if the outcome is death, then persons with the outcome are likely censored before the full time-at-risk period is complete.

### 14.5.3 Model development settings

To develop the prediction model we have to decide which algorithm(s) we like to train. We see the selection of the best algorithm for a certain prediction problem as an empirical question, i.e. we

prefer to let the data speak for itself and try different approaches to find the best one. There is no algorithm that will work best for all problems (no free lunch). In our framework we have therefore implemented many algorithms as described in Section 14.3, and allow others to be added. In this example, to keep things simple, we select just one algorithm: Gradient Boosting Machines.

Furthermore, we have to decide on the covariates that we will use to train our model. In our example, we like to add gender, age, all conditions, drugs and drug groups, and visit counts. We will look for these clinical events in the year before and any time prior the index date.

### 14.5.4 Model evaluation

Finally, we have to define how we will evaluate our model. For simplicity, we here choose internal validation. We have to decide how we divide our dataset in a training and test dataset and how we assign patients to these two sets. Here we will use a typical 75% - 25% split. Note that for very large datasets we could use more data for training.

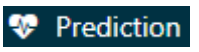
### 14.5.5 Study summary

We have now completely defined our study as shown in Table 14.12.

Table 14.12: Main design choices for our study.

| Choice | Value |
|---|---|
| Target cohort | Patients who have just started on an ACE inhibitor for the first time. Patients are excluded if they have less than 365 days of prior observation time or have prior angioedema. |
| Outcome cohort | Angioedema. |
| Time-at-risk | 1 day till 365 days from cohort start. We will require at least 364 days at risk. |
| Model | Gradient Boosting Machine with hyper-parameters ntree: 5000, max depth: 4 or 7 or 10 and learning rate: 0.001 or 0.01 or 0.1 or 0.9. Covariates will include gender, age, conditions, drugs, drug groups, and visit count. Data split: 75% train - 25% test, randomly assigned by person. |

## 14.6 Implementing the study in ATLAS

The interface for designing a prediction study can be opened by clicking on the 

button in the left hand side ATLAS menu. Create a new prediction study. Make sure to give the study an easy-to-recognize name. The study design can be saved at any time by clicking the  button.

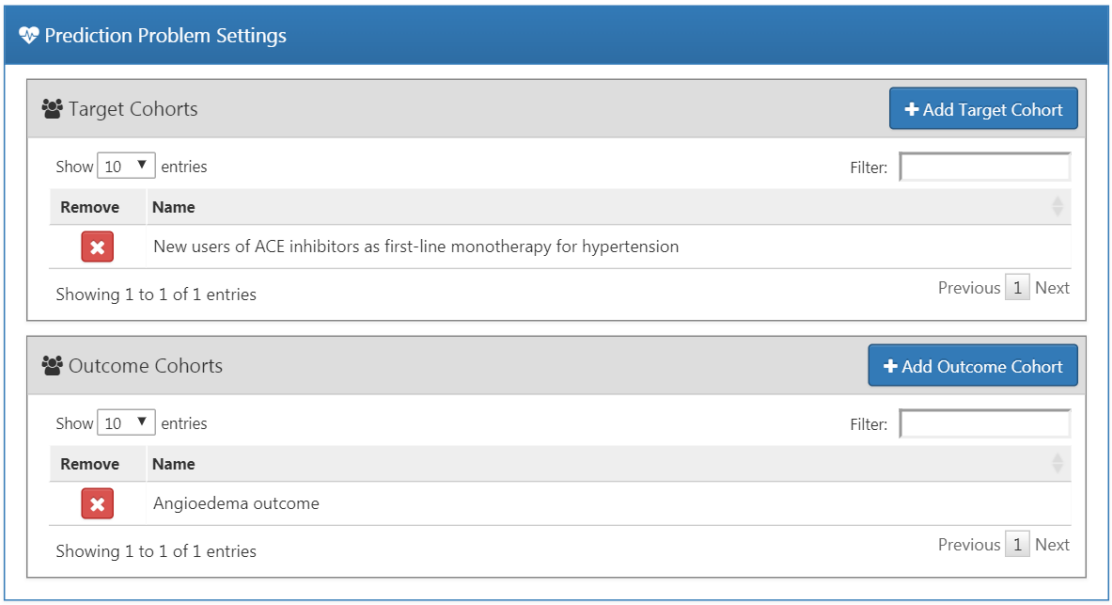



Figure 14.4: Prediction problem settings.

In the Prediction design function, there are four sections: Prediction Problem Settings, Analysis Settings, Execution Settings, and Training Settings. Here we discuss each section:

### 14.6.1 Prediction Problem Settings

Here we select the target population cohorts and outcome cohorts for the analysis. A prediction model will be developed for all combinations of the target population cohorts and the outcome cohorts. For example, if we specify two target populations and two outcomes, we have specified four prediction problems.

To select a target population cohort we need to have previously defined it ATLAS. Instantiating cohorts is described in Chapter 11. The Appendix provides the full definitions of the target (Appendix B.1) and outcome (Appendix B.4) cohorts used in this example. To add a target population to the cohort, click on the "Add Target Cohort" button. Adding outcome cohorts similarly works by clicking the "Add Outcome Cohort" button. When done, the dialog should look like Figure 14.4.

### 14.6.2 Analysis Settings

The analysis settings enables selection of the supervised learning algorithms, the covariates and population settings.

**Model Settings**

We can pick one or more supervised learning algorithms for model development. To add a supervised learning algorithms click on the "Add Model Settings" button. A dropdown containing all the models

currently supported in the ATLAS interface will appear. We can select the supervised learning model we want to include in the study by clicking on the name in the dropdown menu. This will then show a view for that specific model, allowing the selection of the hyper-parameter values. If multiple values are provided, a grid search is performed across all possible combinations of values to select the optimal combination using cross-validation.

For our example we select gradient boosting machines, and set the hyper-parameters as specified in Figure 14.5.

**Covariate Settings**

We have defined a set of standard covariates that can be extracted from the observational data in the CDM format. In the covariate settings view, it is possible to select which of the standard covariates to include. We can define different types of covariate settings, and each model will be created separately with each specified covariate setting.

To add a covariate setting into the study, click on the "Add Covariate Settings". This will open the covariate setting view.

The first part of the covariate settings view is the exclude/include option. Covariates are generally constructed for any concept. However, we may want to include or exclude specific concepts, for example if a concept is linked to the target cohort definition. To only include certain concepts, create a concept set in ATLAS and then under the "**What concepts do you want to include in baseline covariates in the patient-level prediction model? (Leave blank if you want to include everything)**" select the concept set by clicking on  . We can automatically add all descendant concepts to the concepts in the concept set by answering "yes" to the question "**Should descendant concepts be added to the list of included concepts?**" The same process can be repeated for the question "**What concepts do you want to exclude in baseline covariates in the patient-level prediction model? (Leave blank if you want to include everything)**", allowing covariates corresponding to the selected concepts to be removed. The final option "**A comma delimited list of covariate IDs that should be restricted to**" enables us to add a set of covariate IDs (rather than concept IDs) comma separated that will only be included in the model. This option is for advanced users only. Once done, the inclusion and exclusion settings should look like Figure 14.6.

The next section enables the selection of non-time bound variables.

- Gender: a binary variable indicating male or female gender
- Age: a continuous variable corresponding to age in years
- Age group: binary variables for every 5 years of age (0-4, 5-9, 10-14, …, 95+)
- Race: a binary variable for each race, 1 means the patient has that race recorded, 0 otherwise
- Ethnicity: a binary variable for each ethnicity, 1 means the patient has that ethnicity recorded, 0 otherwise
- Index year: a binary variable for each cohort start date year, 1 means that was the patients cohort start date year, 0 otherwise. **It often does not make sense to include index year, since we would like to apply our model to the future**.
- Index month - a binary variable for each cohort start date month, 1 means that was the patients cohort start date month, 0 otherwise

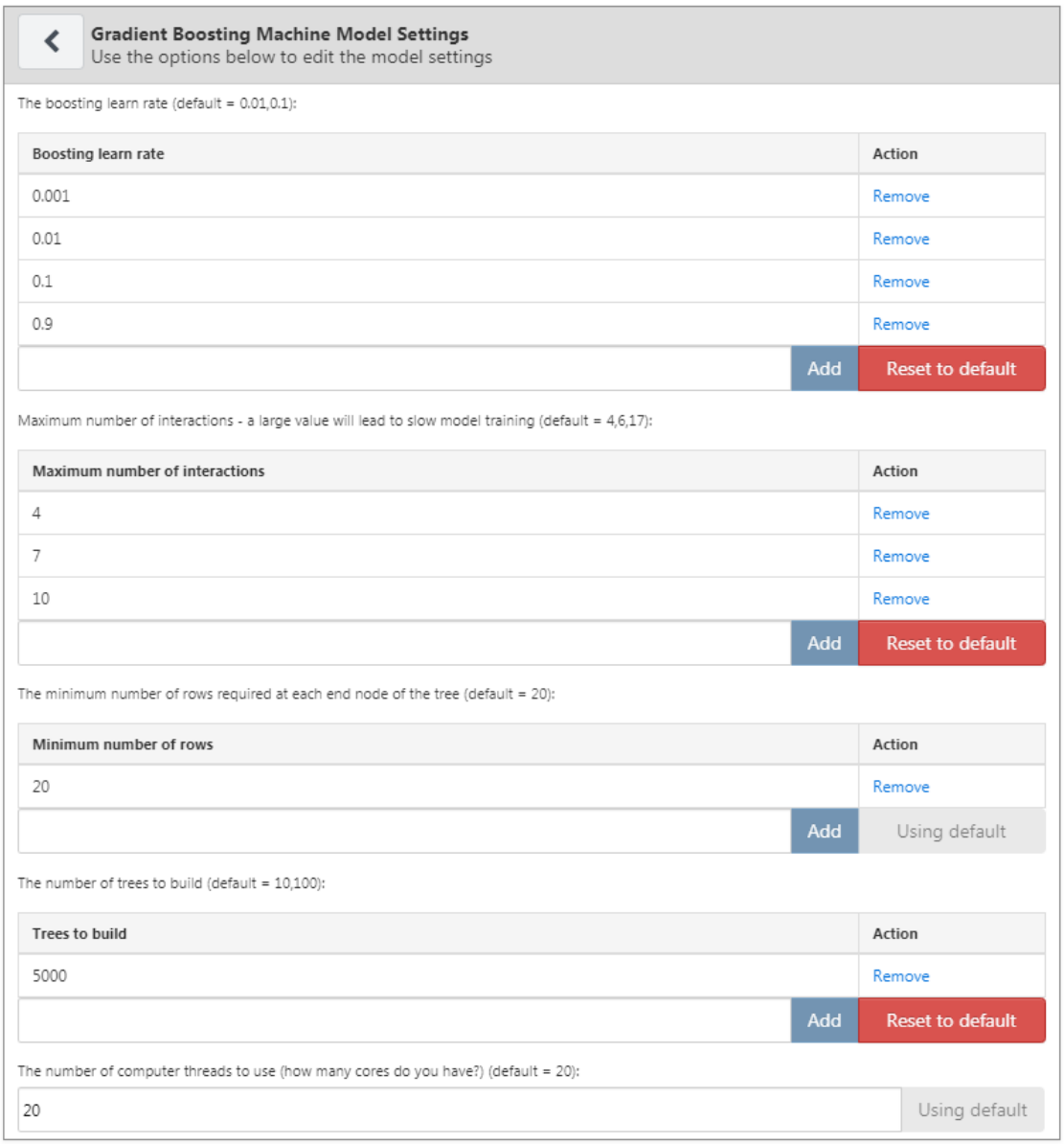



Figure 14.5: Gradient boosting machine settings

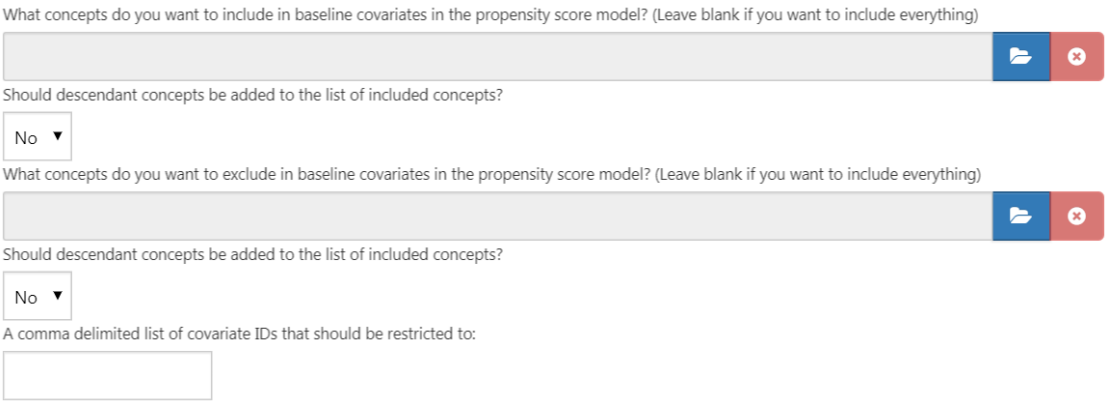



Figure 14.6: Covariate inclusion and exclusion settings.


Figure 14.7: Select covariates.

- Prior observation time: [Not recommended for prediction] a continuous variable corresponding to how long in days the patient was in the database prior to the cohort start date
- Post observation time: [Not recommended for prediction] a continuous variable corresponding to how long in days the patient was in the database post cohort start date
- Time in cohort: a continuous variable corresponding to how long in days the patient was in the cohort (cohort end date minus cohort start date)
- Index year and month: [Not recommended for prediction] a binary variable for each cohort start date year and month combination, 1 means that was the patients cohort start date year and month, 0 otherwise

Once done, this section should look like Figure 14.7.

The standard covariates enable three flexible time intervals for the covariates:

- end days: when to end the time intervals relative to the cohort start date [default is 0]
- long term [default -365 days to end days prior to cohort start date]
- medium term [default -180 days to end days prior to cohort start date]
- short term [default -30 days to end days prior to cohort start date]

Once done, this section should look like Figure 14.8.

The next option is the covariates extracted from the era tables:

- Condition: Construct covariates for each condition concept ID and time interval selected and

Time bound covariates

Set the time windows for the time bound covariates in days relative to the cohort index

| | Any Time Prior | Long Term | Medium Term | Short Term | End Days |
|---|---|---|---|---|---|
| Time Windows | All Time | -365 | -180 | -30 | 0 |

Figure 14.8: Time bound covariates.

Set the time bound era covariates

| Domain | Any Time Prior | Long Term (-365 days) | Medium Term (-180 days) | Short Term (-30 days) | Overlapping | Era Start | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Long Term (-365 days) | Medium Term (-180 days) | Short Term (-30 days) |
| Condition | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Condition Group | ☐ | ☑ | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ |
| Drug | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Drug Group | ☐ | ☑ | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ |

Figure 14.9: Time bound era covariates.

if a patient has the concept ID with an era (i.e., the condition starts or ends during the time interval or starts before and ends after the time interval) during the specified time interval prior to the cohort start date in the condition era table, the covariate value is 1, otherwise 0.

- Condition group: Construct covariates for each condition concept ID and time interval selected and if a patient has the concept ID **or any descendant concept ID** with an era during the specified time interval prior to the cohort start date in the condition era table, the covariate value is 1, otherwise 0.
- Drug: Construct covariates for each drug concept ID and time interval selected and if a patient has the concept ID with an era during the specified time interval prior to the cohort start date in the drug era table, the covariate value is 1, otherwise 0.
- Drug group: Construct covariates for each drug concept ID and time interval selected and if a patient has the concept ID **or any descendant concept ID** with an era during the specified time interval prior to the cohort start date in the drug era table, the covariate value is 1, otherwise 0.

Overlapping time interval setting means that the drug or condition era should start prior to the cohort start date and end after the cohort start date, so it overlaps with the cohort start date. The **era start** option restricts to finding condition or drug eras that start during the time interval selected.

Once done, this section should look like Figure 14.9.

The next option selects covariates corresponding to concept IDs in each domain for the various time intervals:

- Condition: Construct covariates for each condition concept ID and time interval selected and if a patient has the concept ID recorded during the specified time interval prior to the cohort start date in the condition occurrence table, the covariate value is 1, otherwise 0.

- Condition Primary Inpatient: TODO
- Drug: Construct covariates for each drug concept ID and time interval selected and if a patient has the concept ID recorded during the specified time interval prior to the cohort start date in the drug exposure table, the covariate value is 1, otherwise 0.
- Procedure: Construct covariates for each procedure concept ID and time interval selected and if a patient has the concept ID recorded during the specified time interval prior to the cohort start date in the procedure occurrence table, the covariate value is 1, otherwise 0.
- Measurement: Construct covariates for each measurement concept ID and time interval selected and if a patient has the concept ID recorded during the specified time interval prior to the cohort start date in the measurement table, the covariate value is 1, otherwise 0.
- Measurement Value: Construct covariates for each measurement concept ID with a value and time interval selected and if a patient has the concept ID recorded during the specified time interval prior to the cohort start date in the measurement table, the covariate value is the measurement value, otherwise 0.
- Measurement range group: TODO
- Observation: Construct covariates for each observation concept ID and time interval selected and if a patient has the concept ID recorded during the specified time interval prior to the cohort start date in the observation table, the covariate value is 1, otherwise 0.
- Device: Construct covariates for each device concept ID and time interval selected and if a patient has the concept ID recorded during the specified time interval prior to the cohort start date in the device table, the covariate value is 1, otherwise 0.
- Visit Count: Construct covariates for each visit and time interval selected and count the number of visits recorded during the time interval as the covariate value
- Visit Concept Count: Construct covariates for each visit, domain and time interval selected and count the number of records per domain recorded during the visit type and time interval as the covariate value

The distinct count option counts the number of records per domain and time interval [TODO].

Once done, this section should look like Figure 14.10.

The final option is whether to include commonly used risk scores as covariates. Once done, the risk score settings should look like Figure 14.11.

**Population Settings**

The population settings is where addition inclusion criteria can be applied to the target population and is also where the time-at-risk is defined. To add a population setting into the study, click on the "Add Population Settings" button. This will open up the population setting view.

The first set of options enable the user to specify the time-at-risk period. This is the time interval where we look to see whether the outcome of interest occurs. If a patient has the outcome during the time-at-risk period then we will classify them as "Has outcome", otherwise they are classified as "No outcome". "**Define the time-at-risk window start, relative to target cohort entry:**" defines the start of the time-at-risk, relative to the target cohort start or end date. Similarly, "**Define the time-at-risk window end:**" defines the end of the time-at-risk.

"**Minimum lookback period applied to target cohort**" specifies the minimum baseline period; the

Set the time bound covariates

| Domain | Any Time Prior | Long Term (-365 days) | Medium Term (-180 days) | Short Term (-30 days) | Distinct Count | | |
|---|---|---|---|---|---|---|---|
| | | | | | Long Term (-365 days) | Medium Term (-180 days) | Short Term (-30 days) |
| Condition | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Condition - Primary Inpatient | ☐ | ☐ | ☐ | ☐ | | | |
| Drug | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Procedure | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Measurement | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Measurement - Value | ☐ | ☐ | ☐ | ☐ | | | |
| Measurement - Range Group | ☐ | ☑ | ☐ | ☐ | | | |
| Observation | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Device | ☐ | ☑ | ☐ | ☐ | | | |
| Visit - Count | | ☑ | ☐ | ☐ | | | |
| Visit - Concept Count | | ☐ | ☐ | ☐ | | | |

Figure 14.10: Time bound covariates.

Set the index score covariates

| Index Score Type | |
|---|---|
| $CHADS_2$ | ☐ |
| $CHA_2DS_2VASc$ | ☑ |
| DCSI | ☑ |
| Charlson | ☑ |

Figure 14.11: Risk score covariate settings.

minimum number of days prior to the cohort start date that a patient is continuously observed. The default is 365 days. Expanding the minimum look-back will give a more complete picture of a patient (as they must have been observed for longer) but will filter patients who do not have the minimum number of days prior observation.

If "**Should subjects without time at risk be removed?**" is set to yes, then a value for "**Minimum time at risk:**" is also required. This allows removing people who are lost to follow-up (i.e. that have left the database during the time-at-risk period). For example, if the time-at-risk period was 1 day from cohort start until 365 days from cohort start, then the full time-at-risk interval is 364 days (365-1). If we only want to include patients who are observed the whole interval, then we set the minimum time at risk to be 364. If we are happy as long as people are in the time-at-risk for the first 100 days, then we select minimum time at risk to be 100. In this case as the time-at-risk start as 1 day from the cohort start, a patient will be include if they remain in the database for at least 101 days from the cohort start date. If we set "Should subjects without time at risk be removed?" to 'No', then this will keep all patients, even those who drop out from the database during the time-at-risk.

The option "**Include people with outcomes who are not observed for the whole at risk period?**" is related to the previous option. If set to "yes", then people who experience the outcome during the time-at-risk are always kept, even if they are not observed for the specified minimum amount of time.

The option "**Should only the first exposure per subject be included?**" is only useful if our target cohort contains patients multiple times with different cohort start dates. In this situation, picking "yes" will result in only keeping the earliest target cohort date per patient in the analysis. Otherwise a patient can be in the dataset multiple times.

Setting "**Remove patients who have observed the outcome prior to cohort entry?**" to "yes" will remove patients who have the outcome prior to the time-at-risk start date, so the model is for patients who have never experience the outcome before. If "no" is selected, then patients could have had the outcome prior. Often, having the outcome prior is very predictive of having the outcome during the time-at-risk.

Once done, the population settings dialog should look like Figure 14.12.

Now that we are finished with the Analysis Settings, the entire dialog should look like Figure 14.13.

### 14.6.3 Execution settings

There are three options:

- "**Perform sampling**": here we choose whether to perform sampling (default = "no"). If set to "yes", another option will appear: "**How many patients to use for a subset?**", where the sample size can be specified. Sampling can be an efficient means to determine if a model for a large population (e.g. 10 million patients) will be predictive, by creating and testing the model with a sample of patients. For example, if the AUC is close to 0.5 in the sample, we might abandon the model.

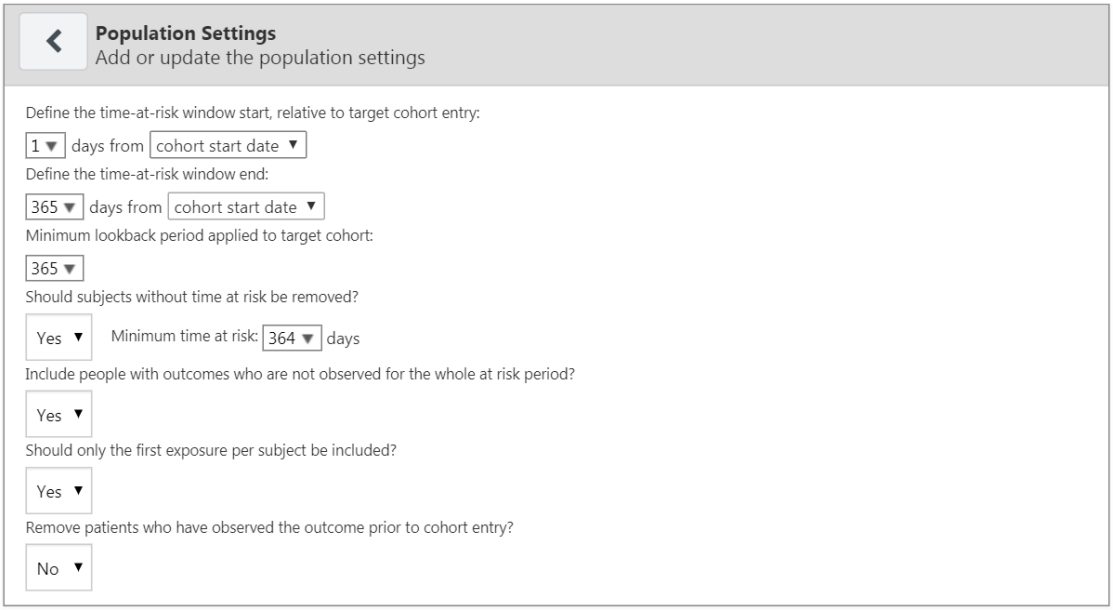



Figure 14.12: Population settings.

- "**Minimum covariate occurrence: If a covariate occurs in a fraction of the target population less than this value, it will be removed:**": here we choose then minimum covariate occurrence (default = 0.001). A minimum threshold value for covariate occurrence is necessary to remove rare events that are not representative of the overall population.
- "**Normalize covariate**": here we choose whether to normalize covariates (default = "yes"). Normalization of the covariates is usually necessary for successful implementation of a LASSO model.

For our example we make the choices shown in Figure 14.14.

### 14.6.4 Training settings

There are four options:

- "**Specify how to split the test/train set:**" Select whether to differentiate the train/test data by person (stratified by outcome) or by time (older data to train the model, later data to evaluate the model).
- "**Percentage of the data to be used as the test set (0-100%)**": Select the percentage of data to be used as test data (default = 25%).
- "**The number of folds used in the cross validation**": Select the number of folds for cross-validation used to select the optimal hyper-parameter (default = 3).
- "**The seed used to split the test/train set when using a person type testSplit (optional):**": Select the random seed used to split the train/test set when using a person type test split.

For our example we make the choices shown in Figure 14.15.

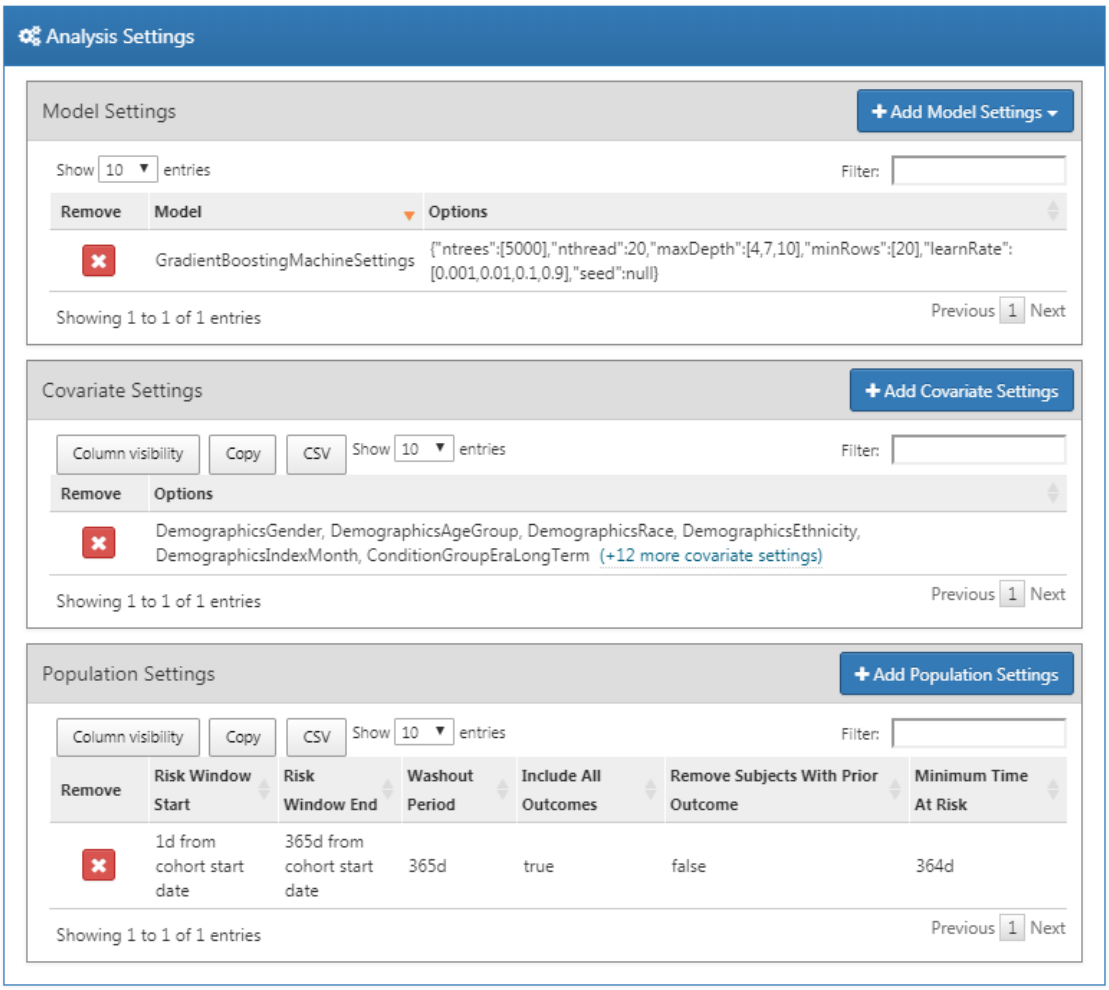



Figure 14.13: Analysis settings.

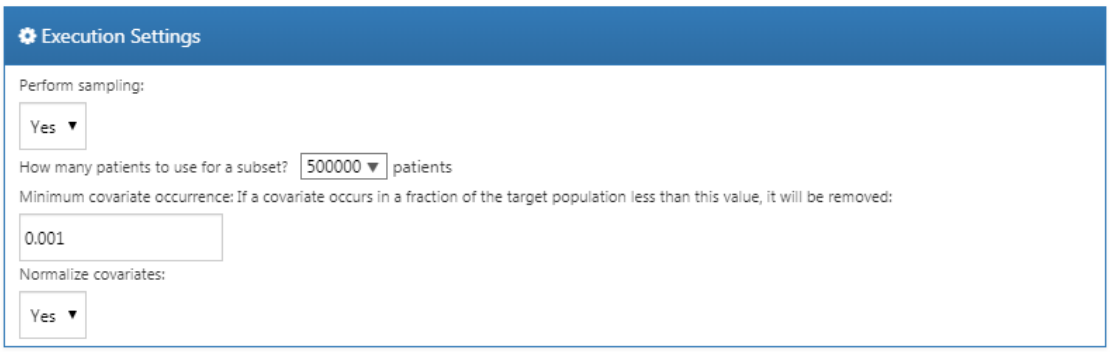



Figure 14.14: Execution settings.

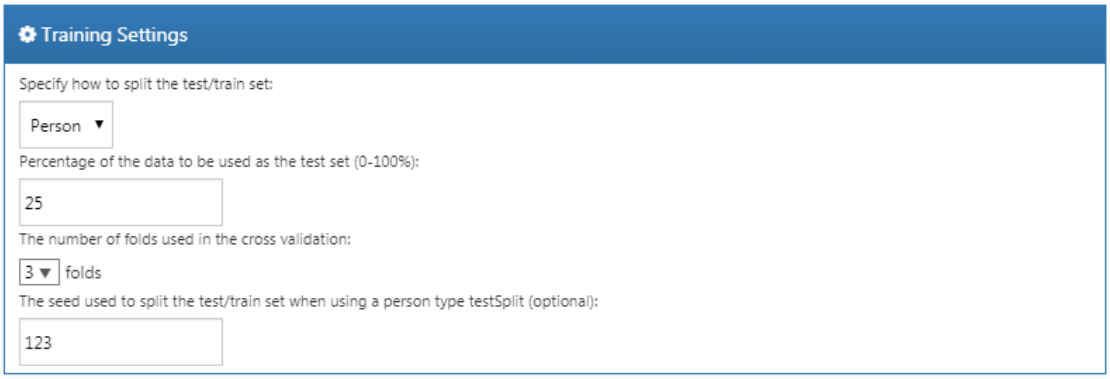



Figure 14.15: Training settings.

### 14.6.5 Importing and exporting a study

To export a study, click on the "Export" tab under "Utilities". ATLAS will produce JSON that can be directly copied and pasted into a file that contains all of the data, such as the study name, cohort definitions, models selected, covariates, settings, needed to run the study.

To import a study, click on the "Import" tab under "Utilities". Paste the contents of a patient-level prediction study JSON into this window, then click on the Import button below the other tab buttons. Note that this will overwrite all previous settings for that study, so this is typically done using a new, empty study design.

### 14.6.6 Downloading the study package

Click on the "Review & Download" tab under "Utilities". In the "Download Study Package" section, enter a descriptive name for the R package, noting that any illegal characters in R will automatically be removed from the file name by ATLAS. Click on Download to download the R package to a local folder.

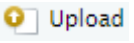
### 14.6.7 Running the study

To run the R package requires having R, RStudio, and Java installed as described in Section 9.4.5. Also required is the PatientLevelPrediction package, which can be installed in R using:

```
install.packages("drat")
drat::addRepo("OHDSI")
install.packages("PatientLevelPrediction")
```

Some of the machine learning algorithms require additional software to be installed. For a full description of how to install the PatientLevelPrediction package, see the "Patient-Level Prediction

Installation Guide" vignette.

To use the study R package we recommend using R Studio. If you are running R Studio locally, unzip the file generated by ATLAS, and double click the .Rproj file to open it in R Studio. If you are running R Studio on an R studio server, click 

to upload and unzip the file, then click on the .Rproj file to open the project.

TODO: full instructions for running the package should be in the package README, not in the book.

After running the R package analysis we can view the results in an interactive shiny app by running:

```
PatientLevelPrediction::viewMultiplePlp(outputFolder)
```

## 14.7 Implementing the study in R

An alternative to implementing our study design using ATLAS is to write the study code outselves in R. We can make use of the functions provided in the PatientLevelPrediction package. The package enables data extraction, model building, and model evaluation using data from databases that are translated into the OMOP CDM.

### 14.7.1 Cohort instantiation

We first need to instantiate the target and outcome cohorts. Instantiating cohorts is described in Chapter 11. The Appendix provides the full definitions of the target (Appendix B.1) and outcome (Appendix B.4) cohorts. In this example we will assume the ACE inhibitors cohort has ID 1, and the angioedema cohort has ID 2.

### 14.7.2 Data extraction

We first need to tell R how to connect to the server. `PatientLevelPrediction` uses the `DatabaseConnector` package, which provides a function called `createConnectionDetails`. Type `?createConnectionDetails` for the specific settings required for the various database management systems (DBMS). For example, one might connect to a PostgreSQL database using this code:

```
library(PatientLevelPrediction)
connDetails <- createConnectionDetails(dbms = "postgresql",
                                       server = "localhost/ohdsi",
                                       user = "joe",
                                       password = "supersecret")

cdmDbSchema <- "my_cdm_data"
```

```
cohortsDbSchema <- "scratch"
cohortsDbTable <- "my_cohorts"
cdmVersion <- "5"
```

The last four lines define the `cdmDbSchema`, `cohortsDbSchema`, and `cohortsDbTable` variables, as well as the CDM version. We will use these later to tell R where the data in CDM format live, where the cohorts of interest have been created, and what version CDM is used. Note that for Microsoft SQL Server, database schemas need to specify both the database and the schema, so for example `cdmDbSchema <- "my_cdm_data.dbo"`.

First it makes sense to verify that the cohort creation has succeeded, by counting the number of cohort entries:

```
sql <- paste("SELECT cohort_definition_id, COUNT(*) AS count",
"FROM @cohortsDbSchema.cohortsDbTable",
"GROUP BY cohort_definition_id")
conn <- connect(connDetails)
renderTranslateQuerySql(connection = conn,
                        sql = sql,
                        cohortsDbSchema = cohortsDbSchema,
                        cohortsDbTable = cohortsDbTable)
```

```
##   cohort_definition_id  count
## 1                    1 527616
## 2                    2   3201
```

Now we can tell PatientLevelPrediction to extract all necessary data for our analysis. Covariates are extracted using the `FeatureExtraction` package. For more detailed information on the FeatureExtraction package see its vignettes. For our example study we decided to use these settings:

```
covSettings <- createCovariateSettings(useDemographicsGender = TRUE,
                                       useDemographicsAge = TRUE,
                                       useConditionGroupEraLongTerm = TRUE,
                                       useConditionGroupEraAnyTimePrior = TRUE,
                                       useDrugGroupEraLongTerm = TRUE,
                                       useDrugGroupEraAnyTimePrior = TRUE,
                                       useVisitConceptCountLongTerm = TRUE,
                                       longTermStartDays = -365,
                                       endDays = -1)
```

The final step for extracting the data is to run the `getPlpData` function and input the connection details, the database schema where the cohorts are stored, the cohort definition IDs for the cohort and outcome, and the washout period which is the minimum number of days prior to cohort index date that the person must have been observed to be included into the data, and finally input the previously constructed covariate settings.

```
plpData <- getPlpData(connectionDetails = connDetails,
                      cdmDatabaseSchema = cdmDbSchema,
                      cohortDatabaseSchema = cohortsDbSchema,
                      cohortTable = cohortsDbSchema,
                      cohortId = 1,
                      covariateSettings = covariateSettings,
                      outcomeDatabaseSchema = cohortsDbSchema,
                      outcomeTable = cohortsDbSchema,
                      outcomeIds = 2,
                      sampleSize = 10000
)
```

There are many additional parameters for the `getPlpData` function which are all documented in the PatientLevelPrediction manual. The resulting `plpData` object uses the package `ff` to store information in a way that ensures R does not run out of memory, even when the data are large.

Creating the `plpData` object can take considerable computing time, and it is probably a good idea to save it for future sessions. Because `plpData` uses `ff`, we cannot use R's regular save function. Instead, we'll have to use the `savePlpData` function:

```
savePlpData(plpData, "angio_in_ace_data")
```

We can use the `loadPlpData()` function to load the data in a future session.

### 14.7.3 Additional inclusion criteria

The final study population is obtained by applying additional constraints on the two earlier defined cohorts, e.g., a minimum time at risk can be enforced (`requireTimeAtRisk, minTimeAtRisk`) and we can specify if this also applies to patients with the outcome (`includeAllOutcomes`). Here we also specify the start and end of the risk window relative to target cohort start. For example, if we like the risk window to start 30 days after the at-risk cohort start and end a year later we can set `riskWindowStart = 30` and `riskWindowEnd = 365`. In some cases the risk window needs to start at the cohort end date. This can be achieved by setting `addExposureToStart = TRUE` which adds the cohort (exposure) time to the start date.

In the example below all the settings we defined for our study are imposed:

```
population <- createStudyPopulation(plpData = plpData,
                                     outcomeId = 2,
                                     washoutPeriod = 364,
                                     firstExposureOnly = FALSE,
                                     removeSubjectsWithPriorOutcome = TRUE,
                                     priorOutcomeLookback = 9999,
                                     riskWindowStart = 1,
                                     riskWindowEnd = 365,
```

```
                                    addExposureDaysToStart = FALSE,
                                    addExposureDaysToEnd = FALSE,
                                    minTimeAtRisk = 364,
                                    requireTimeAtRisk = TRUE,
                                    includeAllOutcomes = TRUE,
                                    verbosity = "DEBUG"
)
```

### 14.7.4 Model Development

In the set function of an algorithm the user can specify a list of eligible values for each hyper-parameter. All possible combinations of the hyper-parameters are included in a so-called grid search using cross-validation on the training set. If a user does not specify any value then the default value is used instead.

For example, if we use the following settings for the gradient boosting machine: `ntrees = c(100,200), maxDepth = 4` the grid search will apply the gradient boosting machine algorithm with `ntrees = 100` and `maxDepth = 4` plus the default settings for other hyper-parameters and `ntrees = 200` and `maxDepth = 4` plus the default settings for other hyper-parameters. The hyper-parameters that lead to the best cross-validation performance will then be chosen for the final model. For our problem we choose to build a gradient boosting machine with several hyper-parameter values:

```
gbmModel <- setGradientBoostingMachine(ntrees = 5000,
                                       maxDepth = c(4,7,10),
                                       learnRate = c(0.001,0.01,0.1,0.9))
```

The `runPlP` function uses the population, `plpData`, and model settings to train and evaluate the model. We can use the `testSplit` (person/time) and `testFraction` parameters to split the data in a 75%-25% split and run the patient-level prediction pipeline:

```
gbmResults <- runPlp(population = population,
                     plpData = plpData,
                     modelSettings = gbmModel,
                     testSplit = 'person',
                     testFraction = 0.25,
                     nfold = 2,
                     splitSeed = 1234)
```

Under the hood the package will now use the R xgboost package to fit a a gradient boosting machine model using 75% of the data and will evaluate the model on the remaining 25%. A results data structure is returned containing information about the model, its performance etc.

In the `runPlp` function there are several parameters to save the `plpData`, `plpResults`, `plpPlots`, `evaluation`, etc. objects which are all set to `TRUE` by default.

We can save the model using:

```
savePlpModel(gbmResults$model, dirPath = "model")
```

We can load the model using:

```
plpModel <- loadPlpModel("model")
```

You can also save the full results structure using:

```
savePlpResult(gbmResults, location = "gbmResults")
```

To load the full results structure use:

```
gbmResults <- loadPlpResult("gbmResults")
```

### 14.7.5 Internal Validation

Once we execute the study, the `runPlp` function returns the trained model and the evaluation of the model on the train/test sets. You can interactively view the results by running: `viewPlp(runPlp = gbmResults)`. This will open a Shiny App in which we can view all performance measures created by the framework, including interactive plots, as shown in Figure 14.16.

To generate and save all the evaluation plots to a folder run the following code:

```
plotPlp(gbmResults, "plots")
```

The plots are described in more detail in Section 14.4.2.

#### 14.7.5.1 External validation

We recommend to always perform external validation, i.e. apply the final model on as much new datasets as feasible and evaluate its performance. Here we assume the data extraction has already been performed on a second database and stored in the `newData` folder. We load the model we previously fitted from the `model` folder:

```
# load the trained model
plpModel <- loadPlpModel("model")

#load the new plpData and create the population
plpData <- loadPlpData("newData")
```

```
population <- createStudyPopulation(plpData = plpData,
                                     outcomeId = 2,
                                     washoutPeriod = 364,
                                     firstExposureOnly = FALSE,
                                     removeSubjectsWithPriorOutcome = TRUE,
                                     priorOutcomeLookback = 9999,
                                     riskWindowStart = 1,
                                     riskWindowEnd = 365,
                                     addExposureDaysToStart = FALSE,
                                     addExposureDaysToEnd = FALSE,
                                     minTimeAtRisk = 364,
                                     requireTimeAtRisk = TRUE,
                                     includeAllOutcomes = TRUE
)

# apply the trained model on the new data
validationResults <- applyModel(population, plpData, plpModel)
```

To make things easier we also provide the `externalValidatePlp` function for performing external validation that also extracts the required data. Assuming we ran `result <- runPlp(...)` then we can extract the data required for the model and evaluated it on new data. Assuming the validation cohorts are in the table `mainschema.dob.cohort` with IDs 1 and 2 and the CDM data is in the schema `cdmschema.dob`:

```
valResult <- externalValidatePlp(plpResult = result,
                                 connectionDetails = connectionDetails,
                                 validationSchemaTarget = 'mainschema.dob',
                                 validationSchemaOutcome = 'mainschema.dob',
                                 validationSchemaCdm = 'cdmschema.dbo',
                                 databaseNames = 'new database',
                                 validationTableTarget = 'cohort',
                                 validationTableOutcome = 'cohort',
                                 validationIdTarget = 1,
                                 validationIdOutcome = 2
          )
```

If we have multiple databases to validate the model on then we can run:

```
valResults <- externalValidatePlp(plpResult = result,
                                  connectionDetails = connectionDetails,
                                  validationSchemaTarget = list('mainschema.dob',
                                                                'difschema.dob',
                                                                'anotherschema.dob'),
                                  validationSchemaOutcome = list('mainschema.dob',
                                                                 'difschema.dob',
                                                                 'anotherschema.dob'),
```

```
                                    validationSchemaCdm = list('cdms1chema.dbo',
                                                               'cdm2schema.dbo',
                                                               'cdm3schema.dbo'),
                                    databaseNames = list('new database 1',
                                                         'new database 2',
                                                         'new database 3'),
                                    validationTableTarget = list('cohort1',
                                                                 'cohort2',
                                                                 'cohort3'),
                                    validationTableOutcome = list('cohort1',
                                                                  'cohort2',
                                                                  'cohort3'),
                                    validationIdTarget = list(1,3,5),
                                    validationIdOutcome = list(2,4,6)
          )
```

## 14.8 Single model viewer app

Exploring the performance of a prediction model is easiest with the `viewPlp` function. This requires a results object as the input. If developing models in R we can use the result of `runPLp` as the input. If using the ATLAS-generated study package, then we need to load one of the models (in this example we will load Analysis_1):

```
plpResult <- loadPlpResult(file.path(outputFolder,
                                     'Analysis_1',
                                     'plpResult'))
```

Here "Analysis_1" corresponds to the analysis we specified earlier.

We can then launch the shiny app by running:

```
viewPlp(plpResult)
```

The Shiny app opens with a summary of the performance metrics on the test and train sets, see Figure 14.16. The results show that the AUC on the train set was 0.78 and this dropped to 0.74 on the test set. The test set AUC is the more accurate measure. Overall, the model appears to be able to discriminate those who will develop the outcome in new users of ACE inhibitors but it slightly overfit as the performance on the train set is higher than the test set. The ROC plot is presented in Figure 14.17.

The calibration plot in Figure 14.18 shows that generally the observed risk matches the predicted risk as the dots are around the diagonal line. The demographic calibration plot in Figure 14.19 however shows that the model is not well calibrated for the younger patients, as the blue line (the predicted risk) differs from the red line (the observed risk) for those aged below 40. This may indicate we need

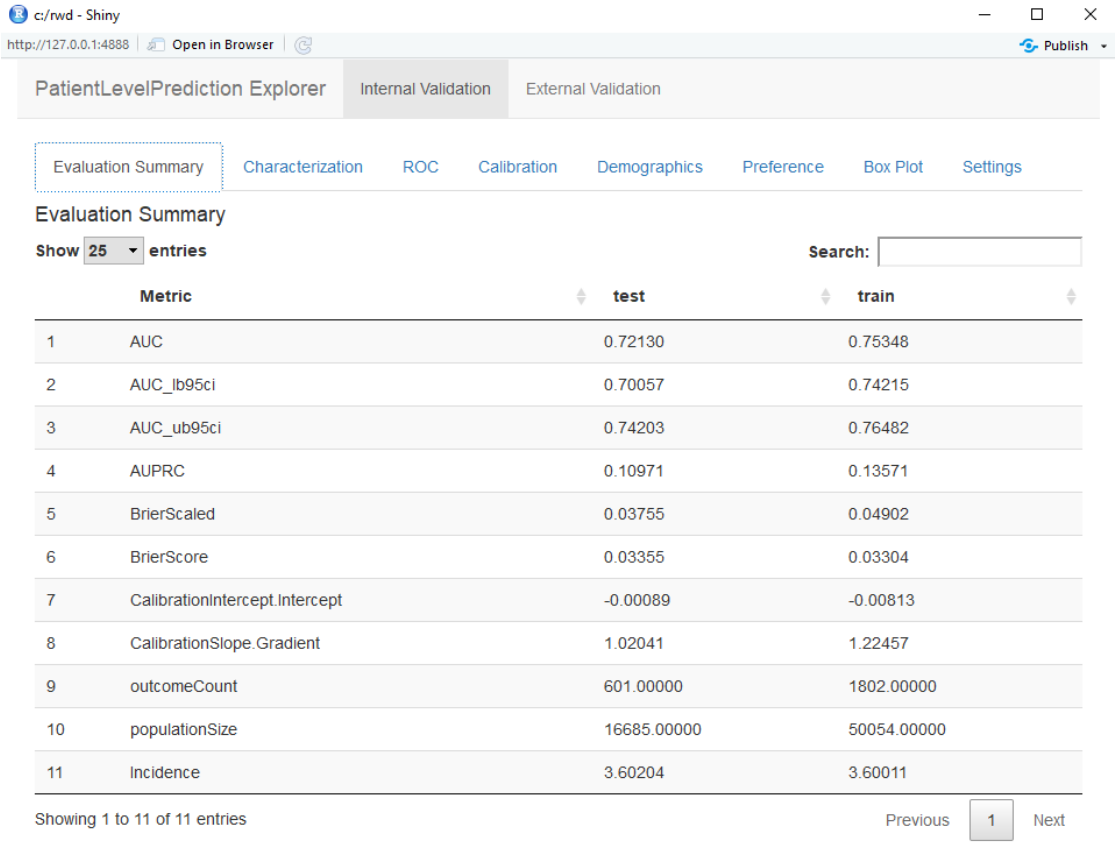



Figure 14.16: Summary evaluation statistics in the Shiny app.

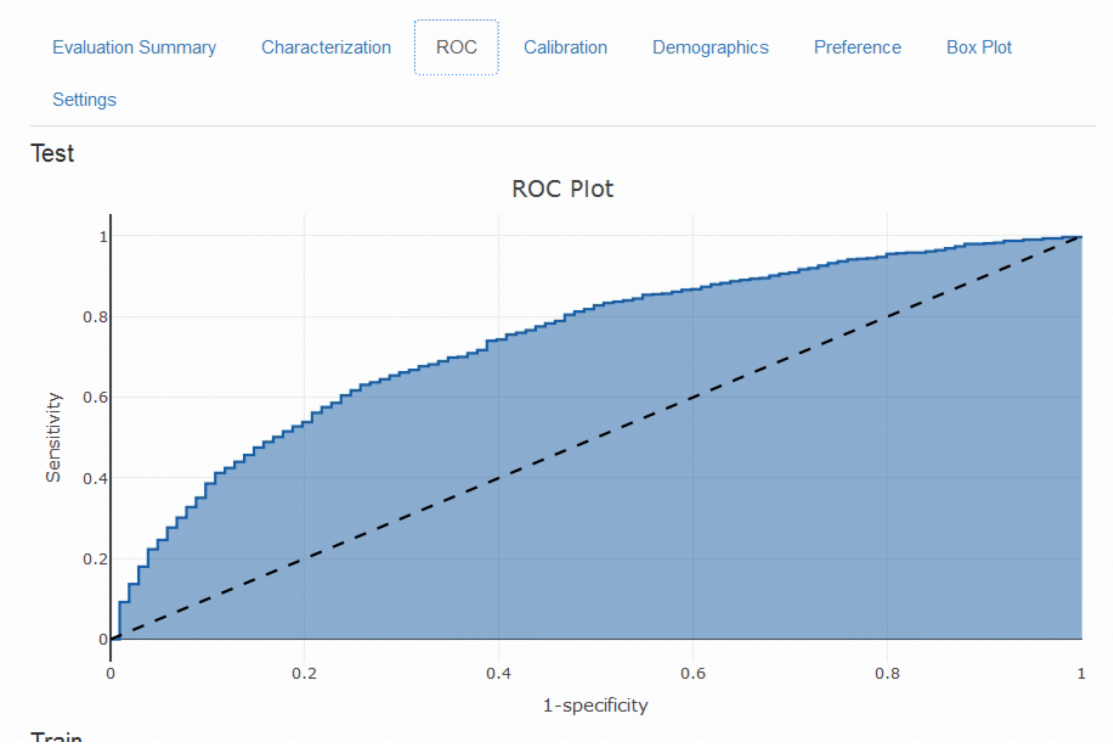



Figure 14.17: The ROC plot.

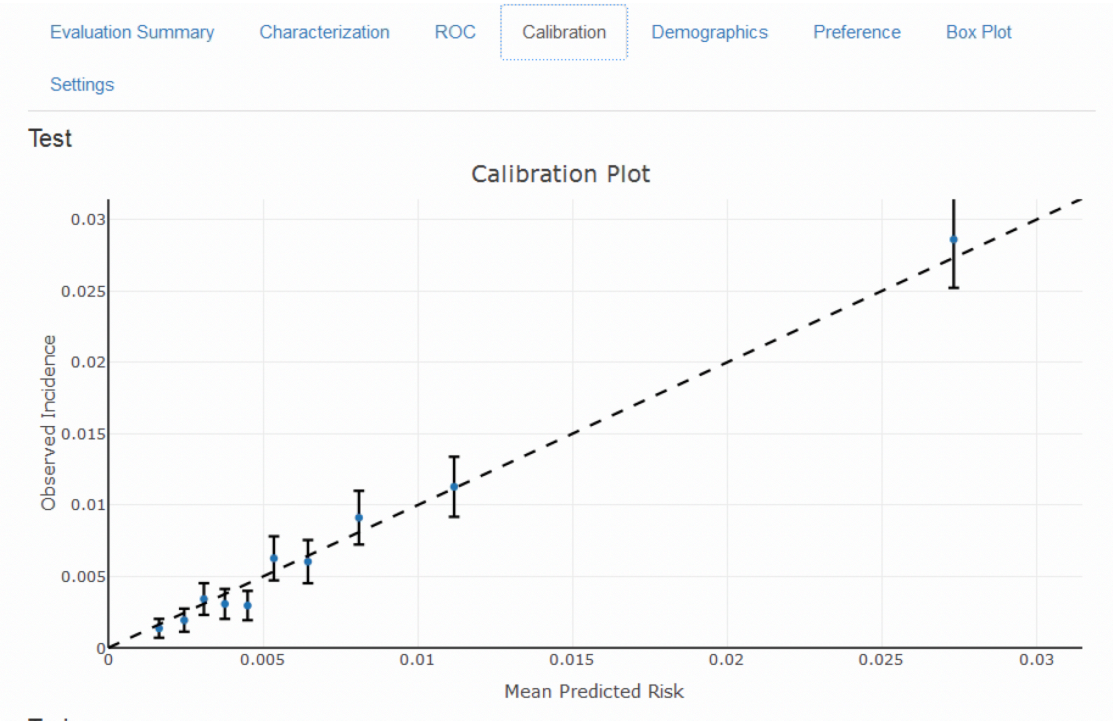



Figure 14.18: The calibration of the model

to remove the under 40s from the target population (as the observed risk for the younger patients is nearly zero).

Finally, the attrition plot shows the loss of patients from the labelled data based on inclusion/exclusion criteria, see Figure 14.20. The plot shows that we lost a large portion of the target population due to them not being observed for the whole time at risk (1 year follow up). Interestingly, not as many patients with the outcome lacked the complete time at risk.

## 14.9 Multiple model viewer app

The study package as generated by ATLAS allows generating and evaluating many different prediction models, for different prediction problems. Therefore, specifically for the output generated by the study package an additional Shiny app has been developed for viewing multiple models. To start this app, run `viewMultiplePlp(outputFolder)` where `outputFolder` is the path containing the analysis results as specified when running the `execute` command (and should for example contain a sub-folder named "Analysis_1").

### 14.9.1 Viewing the model summary and settings

The interactive shiny app will start at the summary page as shown in Figure 14.21.

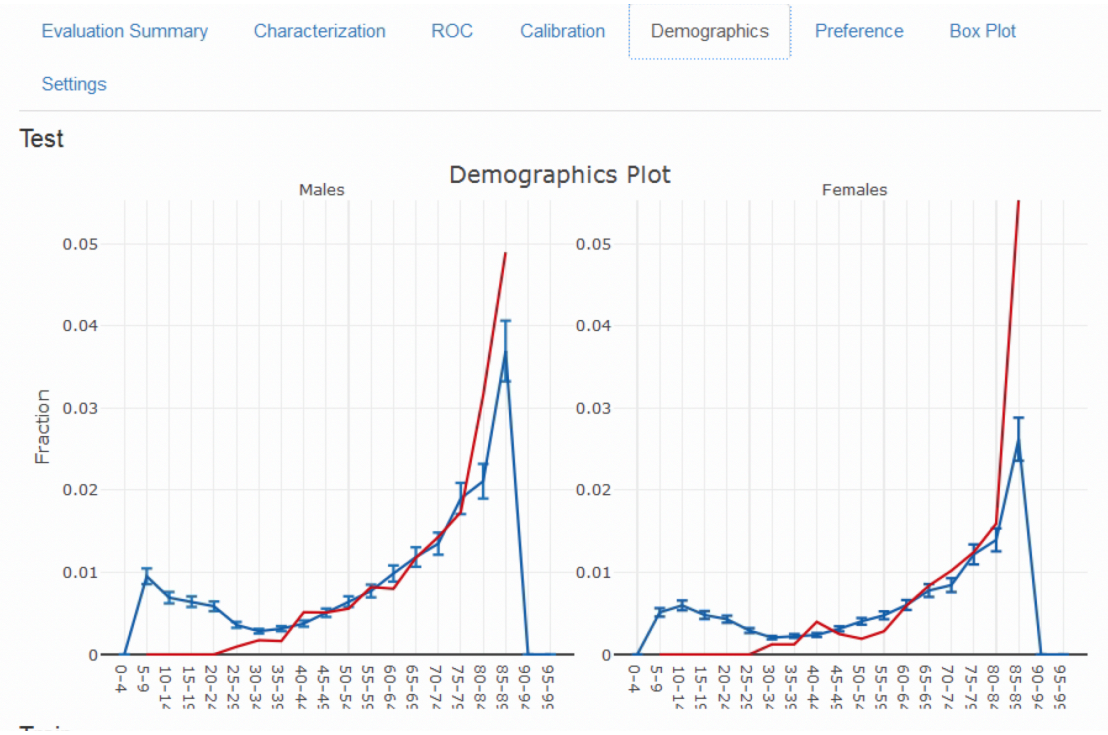



Figure 14.19: The demographic calibration of the model

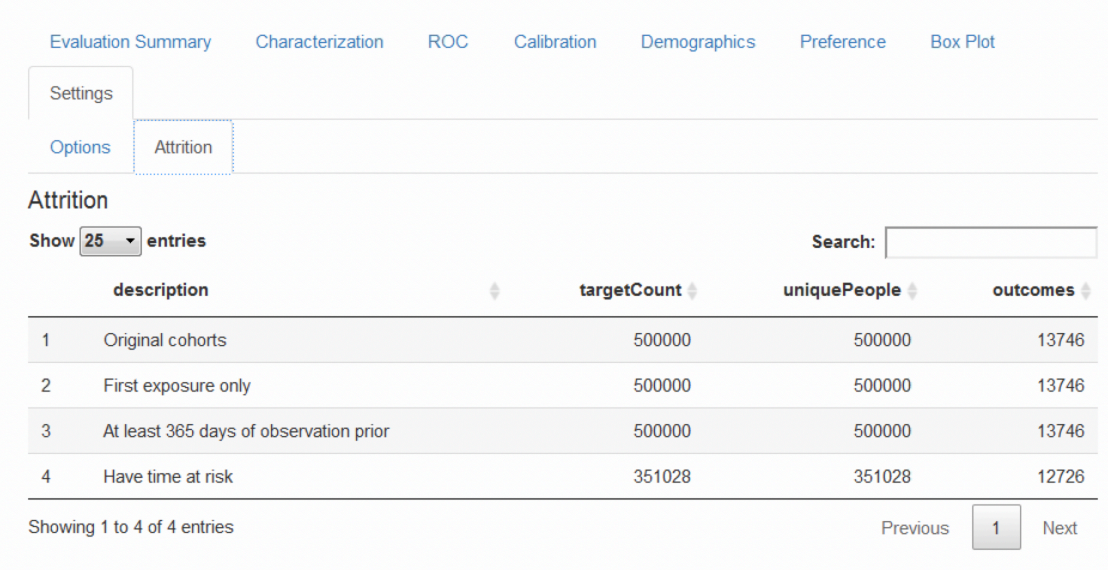



Figure 14.20: The attrition plot for the prediction problem

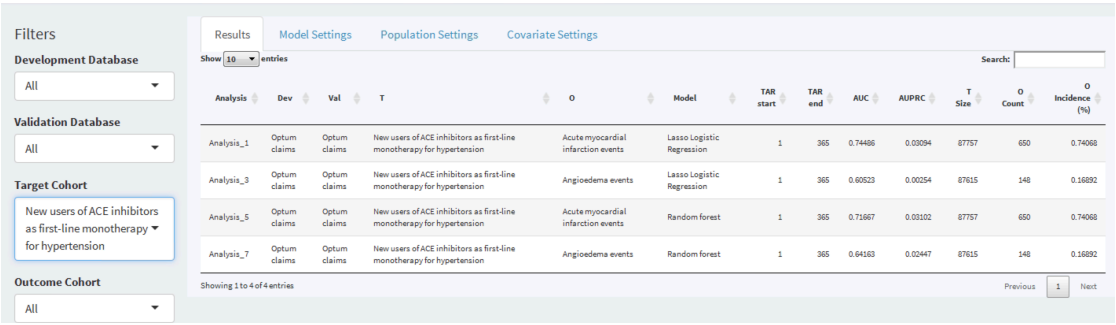



Figure 14.21: The shiny summary page containing key hold out set performance metrics for each model trained

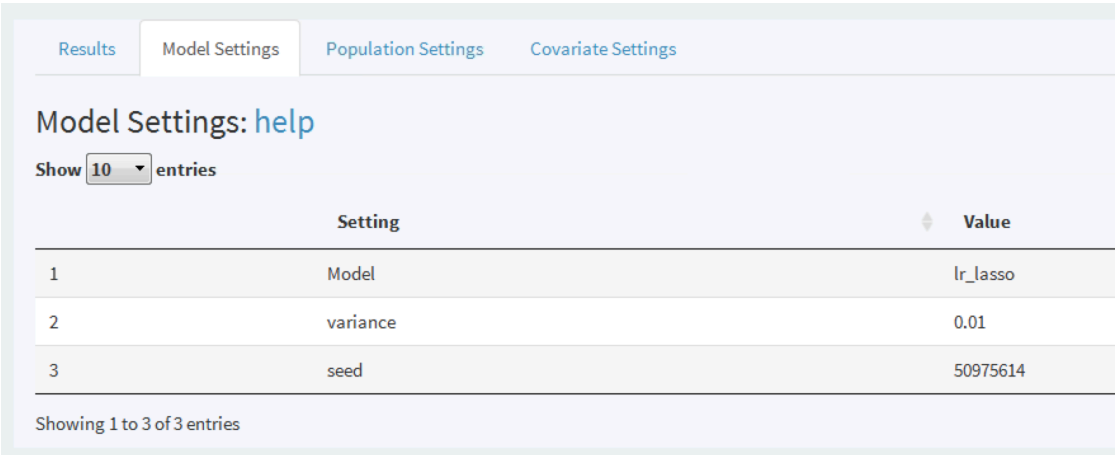



Figure 14.22: To view the model settings used when developing the model.

This summary page table contains:

- basic information about the model (e.g., database information, classifier type, time at risk settings, target population and outcome names)
- hold out target population count and incidence of outcome
- discrimination metrics: AUC, AUPRC

To the left of the table is the filter option, where we can specify the development/validation databases to focus on, the type of model, the time at risk settings of interest and/or the cohorts of interest. For example, to pick the models corresponding to the target population "New users of ACE inhibitors as first line monotherapy for hypertension", select this in the *Target Cohort* option.

To explore a model click on the corresponding row, a selected row will be highlighted. With a row selected, we can now explore the model settings used when developing the model by clicking on the *Model Settings* tab:

Similarly, we can explore the population and covariate settings used to generate the model in the other tabs.

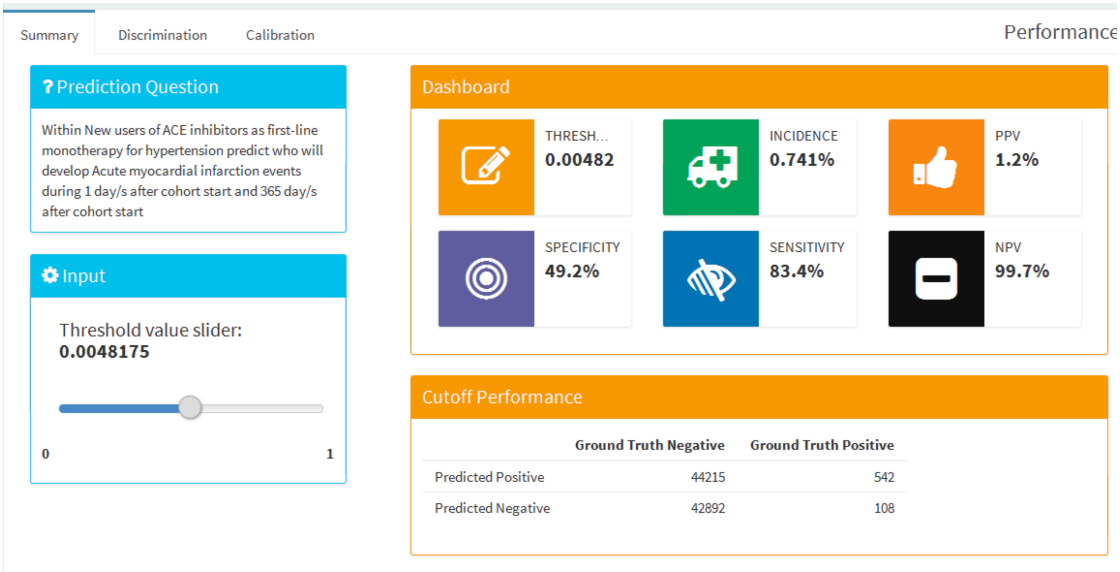



Figure 14.23: The summary performance measures at a set threshold.

### 14.9.2 Viewing model performance

Once a model row had been selected we can also view the model performance. Click on Performance to open the threshold performance summary shown in Figure 14.23.

This summary view shows the selected prediction question in the standard format, a threshold selector and a dashboard containing key threshold based metrics such as positive predictive value (PPV), negative predictive value (NPV), sensitivity and specificity (see Section 14.4.2). In Figure 14.23 we see that at a threshold of 0.00482 the sensitivity is 83.4% (83.4% of patients with the outcome in the following year have a risk greater than or equal to 0.00482) and the PPV is 1.2% (1.2% of patients with a risk greater than or equal to 0.00482 have the outcome in the following year). As the incidence of the outcome within the year is 0.741%, identifying patients with a risk greater than or equal to 0.00482 would find a subgroup of patients that have nearly double the risk of the population average risk. We can adjust the threshold using the slider to view the performance at other values.

To look at the overall discrimination of the model click on the "Discrimination" tab to view the ROC plot, precision-recall plot, and distribution plots. The line on the plots corresponds to the selected threshold point. Figure 14.24 show the ROC and precision-recall plots. The ROC plot shows the model was able to discriminate between those who will have the outcome within the year and those who will not. However, the performance looks less impressive when we see the precision-recall plot, as the low incidence of the outcome means there is a high false positive rate.

Figure 14.25 shows the prediction and preference score distributions.

Finally, we can also inspect the calibration of the model by clicking on the "Calibration" tab. This displays the calibration plot and the demographic calibration shown in Figure 14.26.

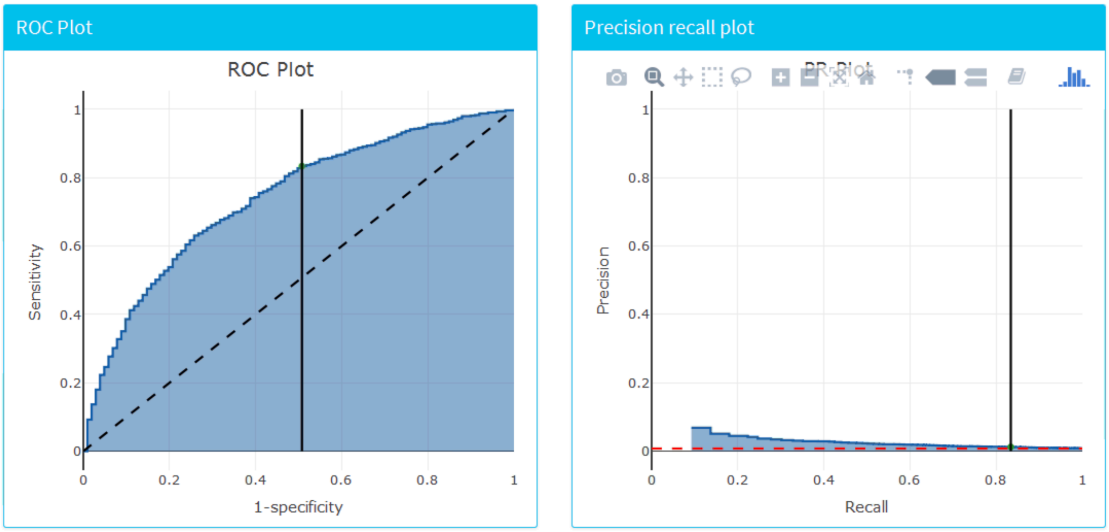



Figure 14.24: The ROC and precision-recall plots used to access the overal discrimination ability of the model.

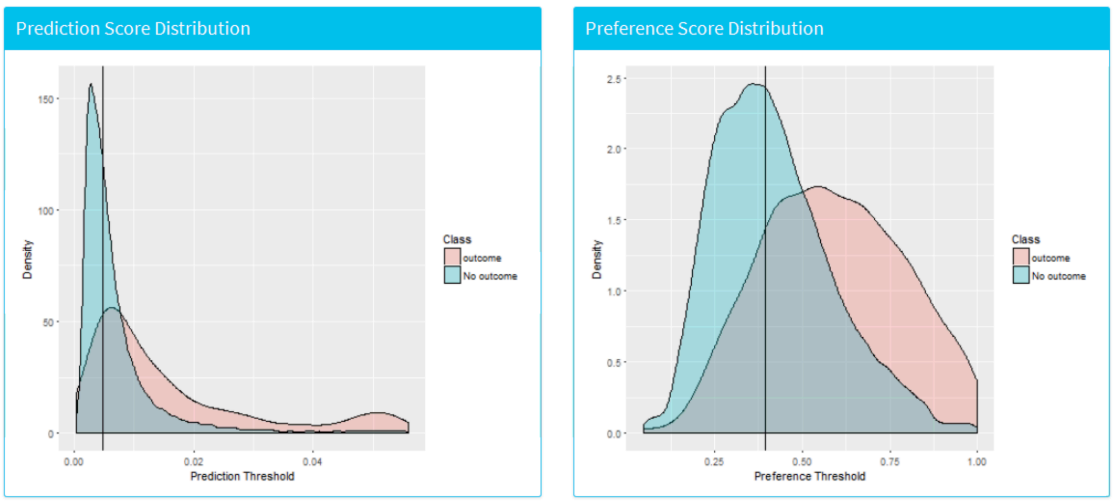



Figure 14.25: The predicted risk distribtion for those with and without the outcome. The more these overlap the worse the discrimination

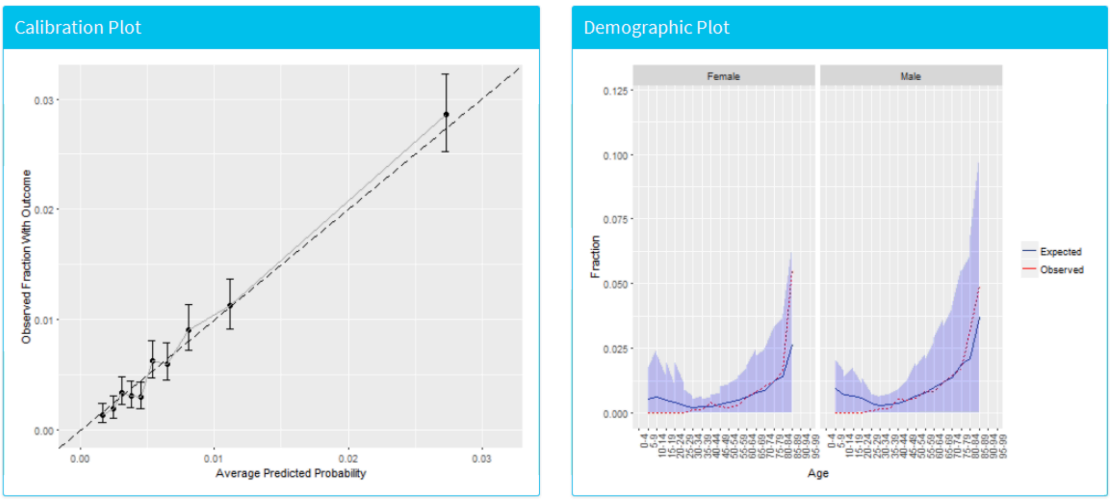



Figure 14.26: The risk stratified calibration and demographic calibration

We see that the average predicted risk appears to match the observed fraction who experienced the outcome within a year, so the model is well calibrated. Interestingly, the demographic calibration shows that the blue line is higher than the red line for young patients, so we are predicting a higher risk for young age groups. Conversely, for the patients above 80 the model is predicting a lower risk than the observed risk. This may prompt us to develop separate models for the younger or older patients.

### 14.9.3 Viewing the model

To inspect the final model, select the Model option from the left hand menu. This will open a view containing plots for each variable in the model, shown in Figure 14.27, and a table summarizing all the candidate covariates, shown in Figure 14.28. The variable plots are separated into binary variables and continuous variables. The x-axis is the prevalence/mean in patients without the outcome and the y-axis is the prevalence/mean in patients with the outcome. Therefore, any variable's dot falling above the diagonal is more common in patients with the outcome and any variable's dot falling below the diagonal is less common in patients with the outcome.

The table in Figure 14.28 displays the name, value (coefficient if using a general linear model, or variable importance otherwise) for all the candidate covariates, outcome mean (the mean value for those who have the outcome) and non-outcome mean (the mean value for those who do not have the outcome).

Predictive models are not causal models, and predictors should not be mistaken for causes. There is no guarantee that modifying any of the variables in Figure 14.28 will have an effect on the risk of the outcome.

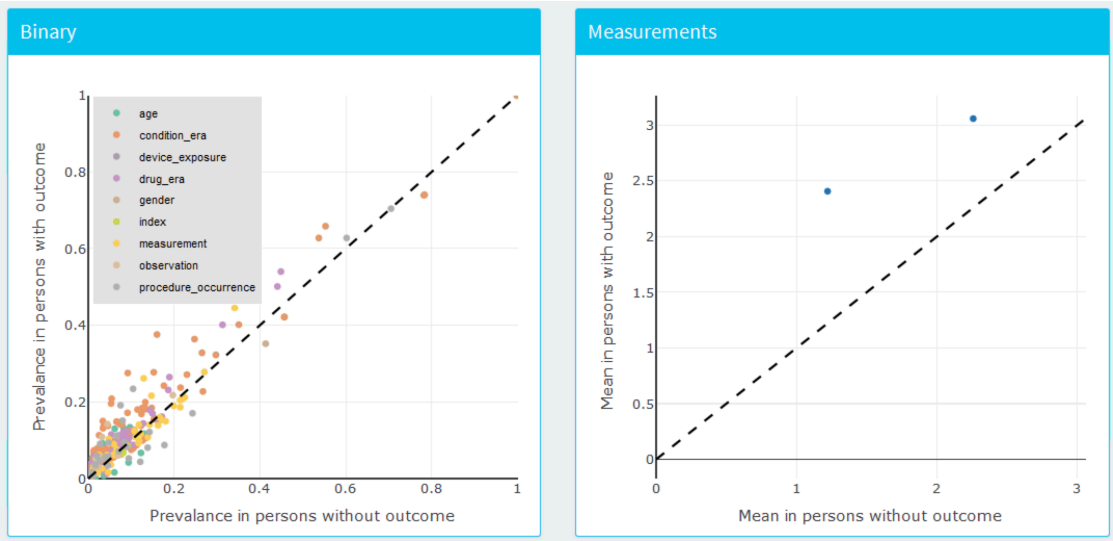



Figure 14.27: Model summary plots. Each dot corresponds to a variable included in the model.

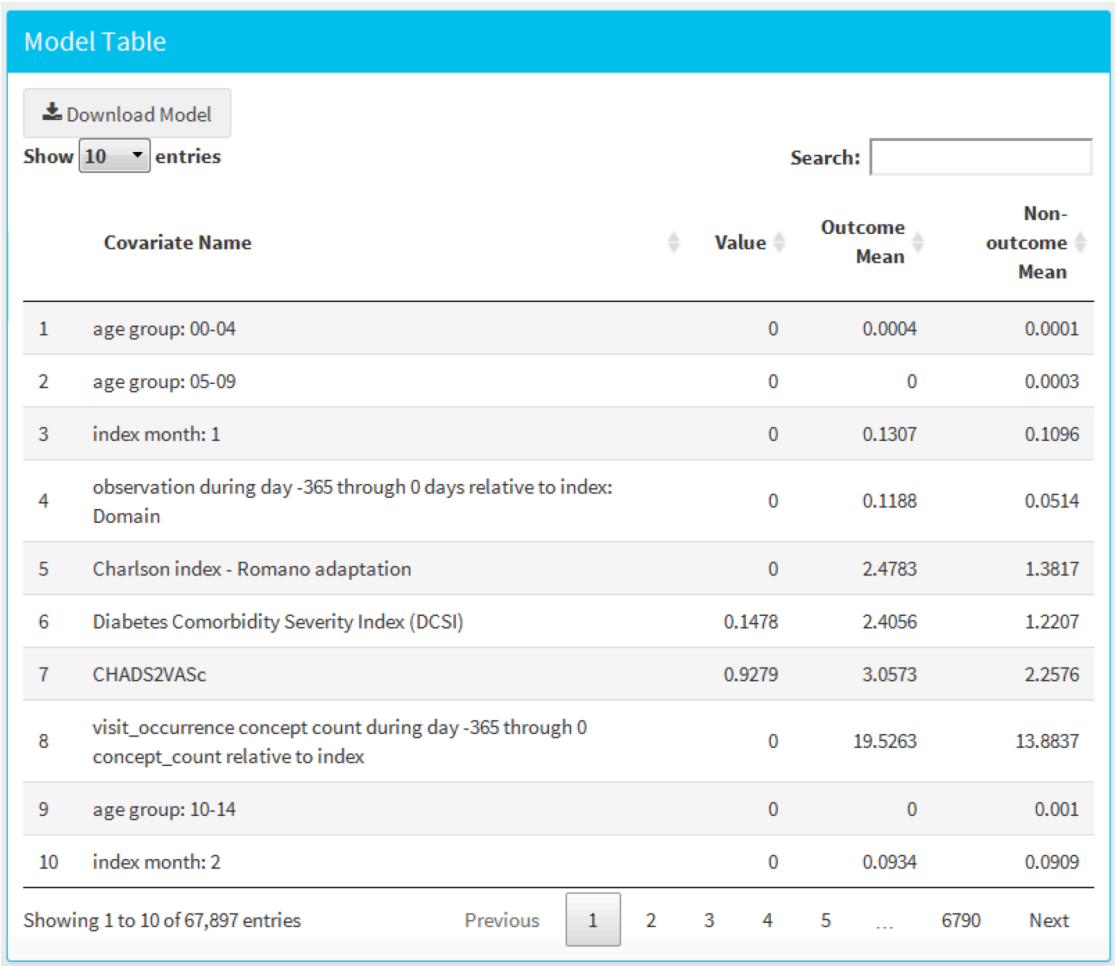
## 14.10 Additional Patient-level Prediction Features

### 14.10.1 Journal paper generation

We have added functionality to automatically generate a word document we can use as start of a journal paper. It contains many of the generated study details and results. If we have performed external validation these results will can be added as well. Optionally, we can add a "Table 1" that contains data on many covariates for the target population. We can create the draft journal paper by running this function:

```
createPlpJournalDocument(plpResult = <your plp results>,
            plpValidation = <your validation results>,
            plpData = <your plp data>,
            targetName = "<target population>",
            outcomeName = "<outcome>",
            table1 = F,
            connectionDetails = NULL,
            includeTrain = FALSE,
            includeTest = TRUE,
            includePredictionPicture = TRUE,
            includeAttritionPlot = TRUE,
            outputLocation = "<your location>")
```

For more details see the help page of the function.



Model Table

Download Model

Show 10 entries Search:

| | Covariate Name | Value | Outcome Mean | Non-outcome Mean |
|---|---|---|---|---|
| 1 | age group: 00-04 | 0 | 0.0004 | 0.0001 |
| 2 | age group: 05-09 | 0 | 0 | 0.0003 |
| 3 | index month: 1 | 0 | 0.1307 | 0.1096 |
| 4 | observation during day -365 through 0 days relative to index: Domain | 0 | 0.1188 | 0.0514 |
| 5 | Charlson index - Romano adaptation | 0 | 2.4783 | 1.3817 |
| 6 | Diabetes Comorbidity Severity Index (DCSI) | 0.1478 | 2.4056 | 1.2207 |
| 7 | CHADS2VASc | 0.9279 | 3.0573 | 2.2576 |
| 8 | visit_occurrence concept count during day -365 through 0 concept_count relative to index | 0 | 19.5263 | 13.8837 |
| 9 | age group: 10-14 | 0 | 0 | 0.001 |
| 10 | index month: 2 | 0 | 0.0934 | 0.0909 |

Showing 1 to 10 of 67,897 entries Previous 1 2 3 4 5 … 6790 Next

Figure 14.28: Model details table.

## 14.11 Summary

- ToDo

## 14.12 Excercises

ToDo

# Part IV

# Evidence Quality

# Chapter 15

# Evidence Quality

*Chapter lead: Jon Duke*

## 15.1 Understanding Evidence Quality

How do we know if the results of a study are reliable? Can they be trusted for use in clinical settings? What about in regulatory decision-making? Can they serve as a foundation for future research? Each time a new study is published or disseminated, readers must consider these questions, regardless of whether the work was a randomized controlled trial, an observational study, or other type of analysis.

One of the concerns that is often raised around observational studies and the use of "real world data" is the topic of data quality (Botsis et al., 2010; Hersh et al., 2013; Sherman et al., 2016). Commonly noted is that data used in observational research were not originally gathered for research purposes and thus may suffer from incomplete or inaccurate data capture as well inherent biases. These concerns have given rise to a growing body of research around how to measure, characterize, and ideally improve data quality (Kahn et al., 2012; Liaw et al., 2013; Weiskopf and Weng, 2013a). The OHDSI community is a strong advocate of such research and community members have led and participated in many studies looking at data quality in the OMOP CDM and the OHDSI network (Huser et al., 2016; Kahn et al., 2015; Callahan et al., 2017; Yoon et al., 2016).

Given the findings of the past decade in this area, it has become apparent that data quality is not perfect and never will be. This notion is nicely reflected in this quote from Dr Clem McDonald, a pioneer in the field of medical informatics:

> Loss of fidelity begins with the movement of data from the doctor's brain to the medical record.

Thus, as a community we must ask the question– *given imperfect data, how can we achieve the most reliable evidence?* The OHDSI community is seeking to address this question through a holistic focus on "evidence quality". Evidence quality considers not only the quality of observational data

but also the validity of the methods, software, and clinical definitions used in our observational analyses.

In the following chapters, we will explore four components of evidence quality:

| Component of Evidence Quality | What it Measures |
|---|---|
| Data Quality | Are the data completely captured with plausible values in a manner that is conformant to agreed structure and conventions? |
| Clinical Validity | To what extent does the analysis conducted match the clinical intention? |
| Software Validity | Can we trust that the process transforming and analyzing the data does what it is supposed to do? |
| Method Validity | Is the methodology appropriate for the question, given the strengths and weaknesses of the data? |

## 15.2 Communicating Evidence Quality

An important aspect of evidence quality is the ability to express the uncertainty that comes from the data being imperfect. Thus, our efforts around evidence quality include not only concepts but also specific tools and community processes. The overarching goal of OHDSI's work around evidence quality is to produce confidence in health care decision-makers that the evidence generated by OHDSI– while undoubtedly imperfect in many ways– has been consistently measured for its weaknesses and strengths and that this information has been communicated in a rigorous and open manner.

# Chapter 16

# Data Quality

## 16.1 Introduction

Kahn et al. define data quality as consisting of three components: (1) conformance (do data values adhere to do specified standard and formats?; subtypes: value, relational and computational conformance); (2) completeness (are data values present?); and (3) plausibility (are data values believable?; subtypes uniqueness, atemporal; temporal) (Kahn et al., 2016)

Kahn additionaly defines two contexts: verification and validation. Verification focuses on model and data constraints and does not rely on external reference. Validation focuses on data expectations that are derived from comparison to a relative gold standard and uses external knowledge.

Table below shows examples of the above defined data quality (DQ) constructs.

| Term | Subtype | Validation example |
|---|---|---|
| Conformance | Value | Providers are only assigned valid medical specialties. |
| | Relational | Prescribing provider identifier is present in drug dispensation data. |
| | Computational | Computed eGFR value conforms to the expected value for a test case patient scenario. |
| Completeness | n/a (no subtypes defined) | A drug product withdrawn from the market at a specific absolute historic date shows expected drop in dispensation. |
| Plausibility | Uniqueness | A zip code for a location does not refer to vastly conflicting geographical areas. |
| | Atemporal | Use of a medication (by age group) for a specific disease agrees with the age pattern for that disease. |
| | Temporal | Temporal pattern of an outbreak of a disease (e.g., Zika) agrees with external source pattern. |

Kahn introduces the term *data quality check* (sometimes refered to as data quality rule) that tests

whether data conform to a given requirement (e.g., implausible age of 141 of a patient (due to incorrect birth year or missing death event)). In support of checks, he also defines *data quality measure* (sometimes refered to as pre-computed analysis) as data analysis that supports evaluation of a check. For example, distribution of days of supply by drug concept.

Two types of DQ checks can be distinguished(Weiskopf and Weng, 2013b)

- general checks
- study-specific checks

From the point of researcher analyzing the data, the desired situation is that data is free from erros that could have been prevented. *ETL data errors* are errors introduced during extract-tranform-load proces. A special type of ETL data error is *mapping error* that results from incorrect mapping of the data from the source terminology (e.g., Korean national drug terminology) into the target data model's standard terminology (e.g., RxNorm and RxNorm Extension). A *source data error* is an error that is already present in the source data due to various cuases (e.g., human typo during data entry).(Huser et al., 2016)

Data quality can also be seen as a component in a larger effort refered to as *evidence quality* or *evidence validation*. Data quality would fall in this framework under *data validation*.

## 16.2 Achilles Heel tool

Since 2014, a component of the OHDSI Achilles tool called Heel was used to check data quality.(Huser et al., 2018)

### 16.2.1 Precomputed Analyses

In support of data characterization, Achilles tool pre-computes number of data analyses. Each pre-computed analysis has an analysis ID and a short description of the analysis. For example, "715: Distribution of days_supply by drug_concept_id" or "506: Distribution of age at death by gender". List of all pre-computed analyses (for Achilles version 1.6.3) as available at https://github.com/OHDSI/Achilles/blob/v1.6.3/inst/csv/achilles/achilles_analysis_details.csv

Achilles has more than 170 pre-computed analysis that support not only data quality checks but also general data characterization (outside data quality context) such as data density visualizations. The pre-computations are largely guided by the CDM relational database schema and analyze most terminology-based data columns, such as condition_concept_id or place_of_service_concept_id. Pre-computations results are stored in table ACHILLES_RESULTS and ACHILLES_RESULTS_DIST.

### 16.2.2 Example DQ check

In complete data about general population, a range of services is provided by a range of providers (with many specialties). A data completness rule with rule_id of 38 evaluates data completness in the PROVIDER table. Checking optional fields in CDM (such as provider specialty) lead to a notification severity output. Analysis Rule 38 triggers a notification if count of distinct specialties <2. It relies on a derived measure `Provider:SpeciatlyCnt`. The rule SQL-formulated logic can be found here: https://github.com/OHDSI/Achilles/blob/v1.6.3/inst/sql/sql_server/heels/serial/rule_38.sql

### 16.2.3 Overview of existing DQ Heel checks

Achilles developers maintain a list of all DQ checks in an overview file. For version 1.6.3, this overview is available here https://github.com/OHDSI/Achilles/blob/v1.6.3/inst/csv/heel/heel_rules_all.csv Each DQ check has a rule_id.

Checks are classified into CDM conformance checks and DQ checks.

Depending on the severity of the problem, the Heel output can be error, warning or notification.

## 16.3 Study-specific checks

The chapter has so far focused on general DQ checks. Such checks are executed regardless of the single research question context. The assumption is that a researcher would formulate additional DQ checks that are required for a specific research question.

We use case studies to demostrate study-specific checks.

### 16.3.1 Outcomes

For an international analysis, part of OHDSI study diagnostics (for a give dataset) may involve checking whether coding practices (that are country specific) affect a cohort definition. A stringent cohort definition may lead to zero cohort size in one (or multiple dataesets).

### 16.3.2 Laboratory data

A diabetes study may utilize HbA1c measurement. A 2018 OHDSI study (https://www.ncbi.nlm.nih.gov/pubmed/30646124) defined a cohort ‘HbA1c8Moderate’ (see https://github.com/rohit43/DiabetesTxPath/blob/master/inst/settings/CohortsToCreate.csv)

## 16.4 ETL unit testing

Extract Transform Load (ETL) process that transforms data from source (in EHR system or claims sys) to target (OMOP CDM) can contain errors. Unit testing of ETL code allows for preventing coding errors in ETL to cause data errors.

### 16.4.1 Unit testing framwork in Rabbit-in-a-Hat

OHDSI tool Rabbit-in-a-Hat includes an ETL unit testing framwork. This framework defines an a set of function for each table in the source schema and a set of functions for each table in target OMOP CDM schema. Detailed description is available at https://www.ohdsi.org/web/wiki/doku.php?id=documentation:software:whiterabbit:test_framework.

# Chapter 17

# Clinical Validity

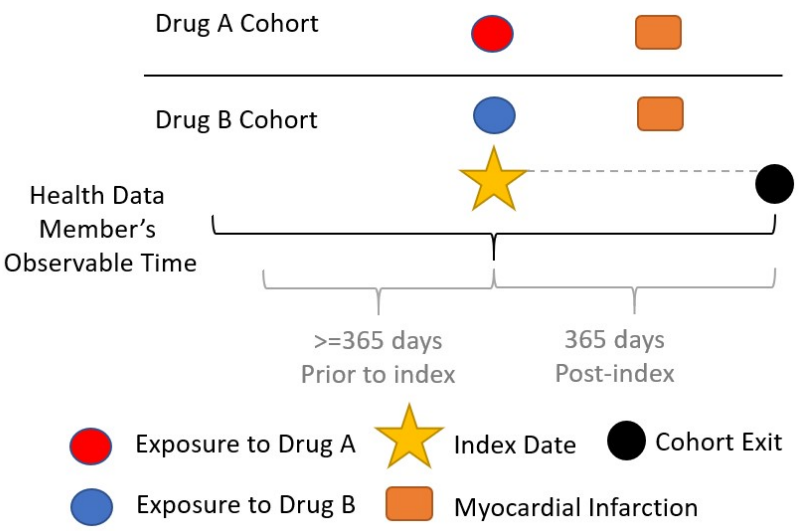
*Chapter leads: Joel N. Swerdel, Seng Chan You*

> The likelihood of transforming matter into energey is something akin to shooting birds in the dark in a country where there are only a few birds. *Einstein, 1935*

The goal of the Research Network is to lower the barrier to performing large-scale collaborative research using observational data to generate high-quality evidence through peer review across study design, execution, and data analysis (Hripcsak et al., 2015). Patient care and policy decisions demand high-quality evidence. Some have hypothesized that the volume, velocity, and veracity of observational data from electronic health records, administrative claims, and investment in data networks can be positioned to meet this demand. Analyses in large data sets are not necessarily correct simply because they are larger. Deficient studies may lead to misuse of resources and result in poor health care oucomes for patients (Morton et al). This chapter will focus on the question: 'To what extent does the analysis conducted match the clinical intention?'

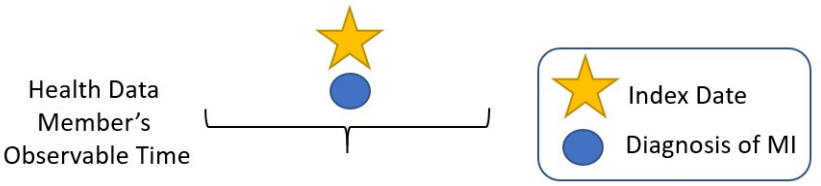
Studies using observational data usually begin by developing cohorts of subjects to compare for determining some effect estimate. During the development of these cohorts certain assumptions are made concerning the validity of the created cohorts, the most important of which is that the subjects in the cohorts have the characteristic that is the basis of the study. For example, if the study involves subjects with myocardial infarction (MI) each of the subjects in the cohort must have had an MI. Ideally, we would definitive evidence of the diagnosis of MI. However, with observational data, we use subject records derived from limited data collected for a specific purpose. In some cases, the data may be derived from data sets specifically collected from subjects with the health condition of interest such as disease registries. While this is high quality data, it is usually a subset of all the subject's data. For example, the data for subjects with MI may include the interpretation of the electrocardiogram (ECG) and not the actual data from the ECG. Subjects are included in these registries based on a clinical review of each subject's health record. In other cases, the data is derived from administrative datasets from insurance claims. These data are usually much more limited than data from health registries but have the advantage of usually including a larger number of subjects from a broader population which may be more generalizable to the overall population. In the case of administrative data, determination of the health claim for a subject is based on administrative codes

for the health condition. In the US, for example, these codes are from the International Classification of Diseases (ICD). Regardless of the origin of the data, the validity of the health conditions based on this data need to be validated. An example of a typical epidemiological study is shown in the diagram below:

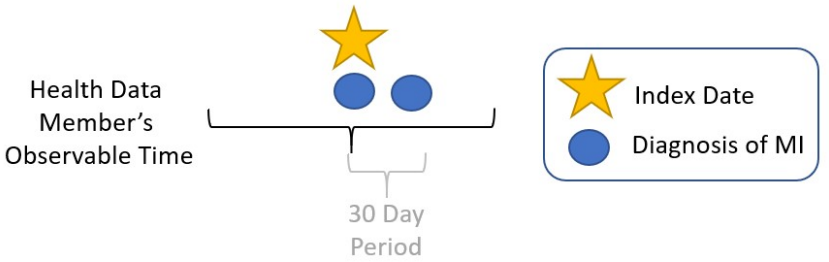



In this example, we are comparing the rates of the occurrence of MI in one year in cohorts of subjects who initiated either Drug A or Drug B. In this example, it is critical for the validity of the study to have valid measures of the rate of MI occurrence. For studies where administrative data is used, the determination of MI is typically from the use of a phenotype algorithm (PA). A PA is a heuristic-based set of rules used to determine the health condition with good precision. These algorithms are often derived from prior research some of which may have been validated. Examples of typical PAs for MI are illustrated below:

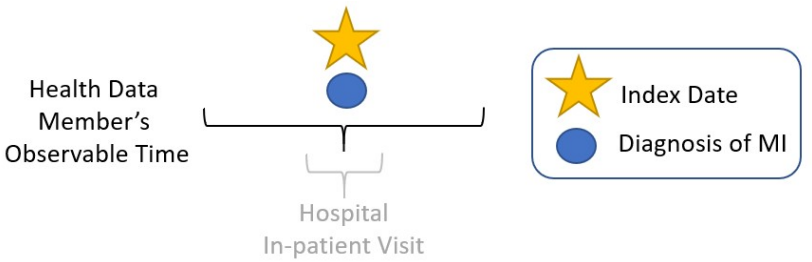
1) One or more occurrences of MI in the subject's record




2) One occurrence of MI in the subject's record followed by a second occurrence within 30 days



3) One or more occurrences of MI in the subject's record from a hospital in-patient setting




Once the PA for the study has been determined, the validity of the definition needs to be determined. To determine the validity of the different algorithms, we need to examine several performance characteristics of the PA including:

1) Sensitivity of the PA – what proportion of the subjects with the health condition in the whole cohort were determined to have the health outcome based on the PA?
2) Specificity of the PA - what proportion of the subjects without the health condition in the whole cohort were determined to not have the health outcome based on the PA?
3) Positive predictive value of the PA - what proportion of the subjects determined by the PA to have the health condition actually had the health condition?
4) Negative predictive value of the PA - what proportion of the subjects determined by the PA to not have the health condition actually did not have the health condition?

The methods used to determine these performance characteristics are described in the remainder of this section. Validation Methods for Phenotype Algorithms The traditional method that has been used to validate PAs has been through a thorough examination of subject records by one or more people with sufficient clinical knowledge to accurately determine the health condition of interest. The method generally follows these steps:

1) Select a random subset of subjects from the overall cohort.
2) Obtain permission from these subjects to receive health records.
3) Obtain the health records from those subjects who have granted permission to do so from their physicians.
4) Select one or more persons with sufficient clinical expertise to review subject records.
5) Determine the guidelines for adjudicating whether a subject is positive or negative for the health condition.
6) Use the results from the clinical adjudication to calculate the performance characteristics of the PA used in the study.

Each step in the above process has the potential to bias the results of the study. For example, obtaining permission from subjects may be difficult and may introduce selection bias if those subjects who provide permission differ from those who do not. In addition, obtaining the patient records and conducting a clinical review of those records is a time consuming and costly process. In order to

complete this process, many studies only examine the records of those subjects the PA identified as cases for the health conditions. Under those conditions, the only performance characteristic that can be calculated is positive predictive value. In the OHDSI community, we were in the process of developing a different approach. We are attempting to use diagnostic predictive models as an alternative method for cohort validation. The general idea is to simulate the ascertainment of the health outcome similar to the way clinicians would in a traditional phenotype algorithm validation but at scale. In this process we develop a diagnostic predictive model for a health outcome and then use that model to determine the probability of a health outcome in a large set of subjects, the "evaluation" cohort, within the data set. We then use that evaluation cohort to test our phenotype algorithms. Using this method, we are able to determine the full set of performance characteristics (i.e., sensitivity, specificity, and positive and negative predictive value) at scale. The tool is being developed as an open-source R package called **PheValuator**.

The process is as follows:

1) Develop a diagnostic predictive model for a phenotype: Diagnostic predictive models are used to estimate the probability that a specific outcome or disease is present in an individual.(Moons et al., 2015) The output of the model is a set of weighted predictors for diagnosing a phenotype.

2) Determine the probability of a phenotype for each individual in a large group of subjects: The set of predictors from the model can be used to estimate the probability of the presence of a phenotype in an individual. We use these predictions as a 'probabilistic gold standard'.

3) Evaluate the performance characteristics of the PAs: We compare the predicted probability to the binary classification of a PA (the test conditions for the confusion matrix). Using the test conditions and the estimates for the true conditions, we can fully populate the confusion matrix and determine the full set of performance characteristics, i.e., sensitivity, specific, and predictive values. There are limitations using this approach. First, we can only use data that is in dataset; we are limited to diagnoses, procedures, observations, measurements, and drug exposures. Moreover, measurements are usually incomplete in administrative datasets as these do not include the actual values for the measures and, as such, we are limited to the existence or absence of the measurements as part of the model. Clinical notes are also not usually present in our data sets. Patient complaints or symptoms may not be recorded in administrative data sets. These may include things like lethargy and acute pain. In diagnostic predictive modeling we create a model that discriminates between those with the disease and those without the disease. For the PheValuator process, we use an extremely specific cohort definition, the "xSpec" cohort, to determine the cases for the models. The xSpec cohort uses a definition to find those with the disease with a very high probability of having the disease of interest. The xSpec cohort is may be defined as those subjects who have multiple condition code codes in their record for the health outcome of interest. For example, for atrial fibrillation, we may have subjects who have 10 or more codes for atrial fibrillation in their record. For MI, an acute outcome, we may use 5 occurrences of MI and include the requirement of having at least two occurrences from a hospital inpatient setting. For our non-cases, we exclude anybody from the data set who have any of the condition occurrences for the health outcome of interest. There are limitations to this method. It is possible that these xSpec cohort subjects may have may be more severe than other cases of the disease. It may also be that these subjects had longer

observation time after initial diagnosis than the average patient. We use logistic regression to create these models, using the LASSO method.(Tibshirani, 1996) This method produces a parsimonious model. In so doing it removes many of the collinear covariates which may be present in many subjects. In the current version of our software we also do not include the temporality of events in the patient's record.

**Practice of Phenotype Algorithm Validation**

An example of the process to conduct a PA validation using chart review is provided by Cutrona and colleagues who validated their PA for MI for the US Food and Drug Administration's (FDA) Sentinel program.(Cutrona et al., 2013) The steps this group used to conduct the validation were as follows:

1) Develop a PA for MI: They used the PA "those with an ICD-9-CM code for AMI (410.x0, 410.x1) in the principal or primary position on facility claims for hospitalizations."

2) Randomly select enough cases to be requested from the primary health care providers in order to achieve a sufficient successful return of records to achieve appropriate statistical precision for their performance measure, in this case PPV.

3) Request approximately an equal number of patient records from each of their four data providers.

4) Receive records from data providers redacted to ensure patient confidentiality.

5) Abstract the data from the records using the clinical expertise of two nurses. Abstracted information included EKG images, cardiac biomarkers, information on ischemic symptoms and results of cardiac diagnostic tests.

6) Test inter-rater reliability of the two data abstractors.

7) Develop an adjudication protocol based on standardized criteria from the joint European Society of Cardiology and American College of Cardiology Global Task Force

8) Review abstracted records using the clinical expertise of two cardiologists. The cardiologists were provided with all abstracted information listed above and an abstracted case summary. Detailed discharge information was not provided to the reviewers.

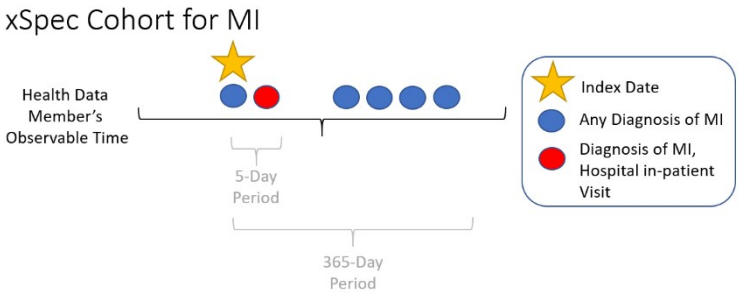
9) Classify each subject as: (1) definite MI; (2) probable MI; (3) no MI; or (4) unable to determine. Distinguishing between definite and probable was not defined for the clinicians but was to based on their own clinical judgement. Subjects were considered cases if the clinicians determined the subject to be definite or probable for MI.

10) Disagreement between the two clinicians, i.e., one clinician determining the subject to be definite or probable and the other either not an MI or unable to determine, was settled through a joint discussion between the clinicians until agreement could be reached.

It appears evident that the above process is both very thorough as well as very time consuming and costly. In this example neither sensitivity nor specificity of the PA was determined likely owing to the cost of including records from a random selection of patients not determined to have an MI based on the PA. Overall the process examined the records of 143 of the 153 records requested. One advantage

this group had was that patient permission was not required as the FDA Sentinel Initiative activities did not require institutional review board (IRB) approval. At the end of this process the researchers determined the PPV for this algorithm to be 86.0% (95% confidence interval; 79.2%, 91.2%). The above study was included in a review of validation efforts for MI PAs by Rubbo et al.(Rubbo et al., 2015) In this review, the authors examined 33 studies involving validation of phenotype algorithms for acute myocardial infarction. They found that there was significant heterogeneity in the phenotype algorithms used in the studies as well as in the validation methods and the results reported. The authors concluded that for acute myocardial infarction there is no gold standard phenotype algorithm available. They noted that the process was both costly and time consuming. Due to that limitation most studies had small sample sizes in their validation leading to wide variations in the estimates for the performance characteristics. They also noted that in the 33 studies, while all the studies reported positive predictive value, only 11 studies reported sensitivity and only five studies reported specificity. The question then needs to be asked is this really validation of the phenotype algorithm? As discussed previously an alternative approach currently being developed in the OHDSI community is through the use of diagnostic predictive modeling tool using a tool called PheValuator. The following are the steps for testing PAs for MI using PheValuator:

1) Develop an extremely specific, xSpec, cohort to determine those with MI with a high probability. For MI, we used an occurrence of MI with one or more occurrences of MI recorded from a hospital in-patient visit within 5 days, and 4 or more occurrences of MI in the patient record within 365 days. The following illustrates this PA for MI:




2) Develop the diagnostic predictive model for MI

   - Create a sample of subjects labeled as cases (from the xSpec cohort for MI) or non-cases in approximate proportion to the prevalence of MI in the population.
   - Use the OHDSI Patient Level Prediction package to develop a LASSO logistic regression model using all available data in the subject record.

3) Randomly select 2M subjects from the database, the evaluation cohort, and apply the model to this cohort to determine the probability of MI in each subject.

4) Test possible algorithms for MI, e.g., 1 occurrence of MI in the patient record, using the evaluation cohort.

5) Compare performance characteristics for the various algorithms to determine the PA for MI for use within a study.

Using this process, Table 1 displays the performance characteristics for four PAs for MI across five

datasets. For a PA similar to the one evaluated by Cutrona and colleagues, ">=1 X HOI, In-Patient", we found a mean PPV of 67% (range: 59%-74%).

Table 1: Performance Characteristics of Four Phenotype Algorithms using Diagnostic Condition Codes to Determine Myocardial Infarction on Multiple Datasets using PheValuator. The continuous 3-color heat map for the data in the table was defined as Red (value = 0), Yellow (value = 0.5), and Green (value = 1).

| | | Acute Myocardial Infarction | | | |
|---|---|---|---|---|---|
| Phenotype Algorithm | Database | Sens | PPV | Spec | NPV |
| >=1 X HOI | CCAE | 0.761 | 0.598 | 0.997 | 0.999 |
| | Optum1862 | 0.723 | 0.530 | 0.995 | 0.998 |
| | OptumGE66 | 0.643 | 0.534 | 0.973 | 0.982 |
| | MDCD | 0.676 | 0.468 | 0.990 | 0.996 |
| | MDCR | 0.665 | 0.553 | 0.977 | 0.985 |
| >= 2 X HOI | CCAE | 0.585 | 0.769 | 0.999 | 0.998 |
| | Optum1862 | 0.495 | 0.693 | 0.998 | 0.996 |
| | OptumGE66 | 0.382 | 0.644 | 0.990 | 0.971 |
| | MDCD | 0.454 | 0.628 | 0.996 | 0.993 |
| | MDCR | 0.418 | 0.674 | 0.991 | 0.975 |
| >=1 X HOI, In-Patient | CCAE | 0.674 | 0.737 | 0.999 | 0.998 |
| | Optum1862 | 0.623 | 0.693 | 0.998 | 0.997 |
| | OptumGE66 | 0.521 | 0.655 | 0.987 | 0.977 |
| | MDCD | 0.573 | 0.593 | 0.995 | 0.994 |
| | MDCR | 0.544 | 0.649 | 0.987 | 0.980 |
| 1 X HOI, In-Patient, 1st Position | CCAE | 0.633 | 0.788 | 0.999 | 0.998 |
| | Optum1862 | 0.581 | 0.754 | 0.999 | 0.997 |
| | OptumGE66 | 0.445 | 0.711 | 0.991 | 0.974 |
| | MDCD | 0.499 | 0.666 | 0.997 | 0.993 |
| | MDCR | 0.445 | 0.711 | 0.991 | 0.974 |

Sens – Sensitivity ; PPV – Positive Predictive Value ; Spec – Specificity; NPV – Negative Predictive Value; Dx Code – Diagnosis code for the phenotype; CCAE - IBM® MarketScan® Commercial Claims and Encounters Database, ages 18-62 years; MDCR - IBM® MarketScan® Medicare Supplemental and Coordination of Benefits Database, ages 66 years and greater; MDCD - IBM® MarketScan® Multi-State Medicaid, ages 18-62 years; Optum1862 - Optum© De-Identified Clinformatics® Data Mart Database – Date of Death, ages 18-62 years; OptumGE66 - ages 66 years and greater

# Chapter 18

# Software Validity

*Chapter lead: Martijn Schuemie*

The central question of software validity is

> Does the software do what it is expected to do?

Software validity is an essential component of evidence quality: only if our analysis software does what it is expected to do can we produce reliable evidence. As described in Section 18.1.1, it is essential to view every study as a software development exercise, creating an automated script that executes the entire analysis, from data in the Common Data Model (CDM) to the results such as estimates, figures as tables. It is this script, and any software used in this script, that must be validated. As described in Section 9.1, we can write te entire analysis as custom code, or we can use the functionality available in the OHDSI Methods Library. The advantage of using the Methods Library is that great care has already been taken to ensure its validity, so establishing the validity of the entire analysis becomes less burdensome.

In this chapter we first describe best practices for writing valid analysis code. After this we discuss how the Methods library is validated through its software development process and testing.

## 18.1 Study code validity

### 18.1.1 Automation as a requirement for reproducibility

Traditionally, observational studies are often viewed as a journey rather than a process: a database expert may extract a data set from the database and hands this over to the data analyst, who may open it in a spreadsheet editor or other interactive tool, and starts working on the analysis. In the end, a result is produced, but little is preserved of how it came about. The destination of the journey was reached, but it is not possible to retrace the exact steps taken to get there. This practice is entirely unacceptable, both because it is not reproducible, but also because it lacks transparency; we do not

know exactly what was done to produce the result, so we also cannot verify that no mistakes were made.

Every analysis generating evidence must therefore be fully automated. By automated we mean the analysis should be implemented as a single script, and we should be able to redo the entire analysis from database in CDM format to results, including tables and figures, with a single command. The analysis can be of arbitrary complexity, perhaps producing just a single count, or generating empirically calibrated estimates for millions of research questions, but the same principle applies. The script can invoke other scripts, which in turn can invoke even lower-level analysis processes.

The analysis script can be implemented in any computer language, although in OHDSI the preferred language is R. Thanks to the DatabaseConnector R package, we can connect directly to the data in CDM format, and many advanced analytic are available through the other R packages in the OHDSI Methods Library.

### 18.1.2 Programming best practices

Observational analyses can become very complex, with many steps needed to produce the final results. This complexity can make it harder to maintain the analysis code, and increase the likelihood of making errors as well as making it harder to notice errors. Luckily, computer programmers have over many years developed best practices for writing code that can deal with complexity, and is easy to read, reuse, adapt, and verify. (Martin, 2008) A full discussion of these best practices could fill many books. Here, we highlight these four import principles:

- **Abstraction**: Rather than write a single large script that does everything, leading to so-call "spaghetti code" where dependencies between lines of code can go from anywhere to anywhere (e.g. a value set on line 10 is used in line 1,000), we can organize our code in units called "functions". A function should have a clear goal, for example "take random sample", and once created we can then use this function in our larger script without having to think of the minutiae of what the function does; We can abstract the function to a simple-to-understand concept.
- **Encapsulation**: For abstraction to work, we should make sure that dependencies of a function are minimized and clearly defined. Our example sampling function should have a few arguments (e.g. a dataset and a sample size), and one output (e.g. the sample). Nothing else should influence what the function does. So-called "global variables", variables that are set outside a function, are not arguments of a function, but are nevertheless used in the function, should be avoided.
- **Clear naming**: Variables and functions should have clear names, making code read almost like natural language. For example, instead of `x <- spl(y, 100)`, we can write code that reads `sampledPatients <- takeSample(patients, sampleSize = 100)`. Try to resist the urge to abbreviate. Modern languages have no limits on the length of variable and function names.
- **Reuse**: One advantage of writing clear, well encapsulated functions is that they can often be reused. This not only saves time, it also means there will be less code, so less complexity and fewer opportunities for errors.

### 18.1.3 Code validation

Several approaches exist to verify the validity of software code, but two are especially relevant for code implementing an observational study:

- **Code review**: One person writes the code, and another person reviews the code.
- **Double coding**: Two persons both independently write the analysis code, and afterwards the results of the two scripts are compared.

Code review has the advantage that it is usually less work, but the disadvantage is that the reviewer might miss some errors. Double coding on the other hand is usually very labor intensive, but it is less likely, although not impossible, that errors are missed. Another disadvantages of double coding is that two separate implementations *almost always* produce different results, due to the many minor arbitrary choices that need to made (e.g. should "until exposure end" be interpreted as including the exposure end date, or not?). As a consequence, the two supposedly independent programmers often need to work together to align their analyses, thus breaking their independence.

Other software validation practices such as unit testing are less relevant here because a study is typically a one-time activity with highly complex relationship between input (the data in CDM) and outputs (the study results), making these practices less usable. Note that these practices are applied in the Methods Library.

### 18.1.4 Using the Methods Library

The OHDSI Methods Library provides a large set of functions, allowing most observational studies to be implemented using only a few lines of code. Using the Methods Library therefore shifts most of the burden of establishing the validity of one's study code to the Library. Validity of the Methods Library is ensured by its software development process, and by extensive testing.

## 18.2 Methods Library software development process

The OHDSI Methods Library is developed by the OHDSI community. Proposed changes to the Library are discussed in two venues: The GitHub issue trackers (for example the CohortMethod issue tracker[1]) and the OHDSI Forums[2]. Both are open to the public. Any member of the community can contribute software code to the Library, however, final approval of any changes incorporated in the released versions of the software is performed by the OHDSI Population-Level Estimation Workgroup leadership (Drs. Marc Suchard and Martijn Schuemie) and OHDSI Patient-Level Prediction Workgroup leadership (Drs. Peter Rijnbeek and Jenna Reps) only.

Users can install the Methods Library in R directly from the master branches in the GitHub repositories, or through a system known as "drat" that is always up-to-date with the master branches. A

[1] https://github.com/OHDSI/CohortMethod/issues

[2] http://forums.ohdsi.org/

number of the Methods Library packages are available through R's Comprehensive R Archive Network (CRAN), and this number is expected to increase over time.

Reasonable software development and testing methodologies are employed by OHDSI to maximize the accuracy, reliability and consistency of the Methods Library performance. Importantly, as the Methods Library is released under the terms of the Apache License V2, all source code underlying the Methods Library, whether it be in R, C++, SQL, or Java is available for peer review by all members of the OHDSI community, and the public in general. Thus, all the functionality embodied within Methods Library is subject to continuous critique and improvement relative to its accuracy, reliability and consistency.

### 18.2.1 Source Code Management

All of the Methods Library's source code is managed in the source code version control system "git" publicly accessible via GitHub. The OHDSI Methods Library repositories are access-controlled. Anyone in the world can view the source code, and any member of the OHDSI community can submit changes through so-called pull requests. Only the OHDSI Population-Level Estimation Workgroup and Patient-Level Prediction Workgroup leadership can approve such request, make changes to the master branches, and release new versions. Continuous logs of code changes are maintained within the GitHub repositories and reflect all aspects of changes in code and documentation. These commit logs are available for public review.

New versions are released by the OHDSI Population-Level Estimation Workgroup and Patient-Level Prediction Workgroup leadership as needed. A new release starts by pushing changes to a master branch with a package version number (as defined in the DESCRIPTION file inside the package) that is greater than the version number of the previous release. This automatically triggers checking and testing of the package. If all tests are passed, the new version is automatically tagged in the version control system and the package is automatically uploaded to the OHDSI drat repository. New versions are numbered using three-component version number:

- New **micro versions** (e.g. from 4.3.2 to 4.3.3) indicate bug fixes only. No new functionality, and forward and backward compatibility are guaranteed
- New **minor versions** (e.g. from 4.3.3 to 4.4.0) indicate added functionality. Only backward compatibility is guaranteed
- New **major versions** (e.g. from 4.4.0 to 5.0.0) indicate major revisions. No guarantees are made in terms of compatibility

### 18.2.2 Documentation

All packages in the Methods Library are documented through R's internal documentation framework. Each package has a package manual that describes every function available in the package. To promote alignment between the function documentation and the function implementation, the roxygen2 software is used to combine a function's documentation and source code in a single file. The package manual is available on demand through R's command line interface, as a PDF in the package reposi-

tories, and as a web page. In addition, many packages also have vignettes that highlight specific use cases of a package. All documentation can be viewed though the Methods Library website[3].

All Method Library source code is available to end users. Feedback from the community is facilitated using GitHub's issue tracking system and the OHDSI Forums.

### 18.2.3 Availability of Current and Historical Archive Versions

Current and historical versions of the Methods Library packages are available in two locations: First, the GitHub version control system contains the full development history of each package, and the state of a package at each point in time can be reconstructed and retrieved. Most importantly, each released version is tagged in GitHub. Second, the released R source packages are stored in the OHDSI GitHub drat repository.

### 18.2.4 Maintenance, Support and Retirement

Each current version of the Methods Library is actively supported by OHDSI with respect to bug reporting, fixes and patches. Issues can be reported through GitHub's issue tracking system, and through the OHDSI forums. Each package has a package manual, and zero, one or several vignettes. Online video tutorials are available, and in-person tutorials are provided from time to time.

### 18.2.5 Qualified Personnel

Members of OHDSI community represent multiple statistical disciplines and are based at academic, not-for-profit and industry-affiliated institutions on multiple continents.

All leaders of the OHDSI Population-Level Estimation Workgroup and OHDSI Patient-Level Prediction Workgroup hold PhDs from accredited academic institutions and have published extensively in peer reviewed journals.

### 18.2.6 Physical and Logical Security

The OHDSI Methods Library is hosted on the GitHub[4] system. GitHub's security measures are described at https://github.com/security. Usernames and passwords are required by all members of the OHDSI community contribute modifications to the Methods Library, and only the Population-Level Estimation Workgroup and Patient-Level Prediction Workgroup leadership can makes changes to the master branches. User accounts are limited in access based upon standard security policies and functional requirements.

[3] https://ohdsi.github.io/MethodsLibrary/
[4] https://github.com/

### 18.2.7 Disaster Recovery

The OHDSI Methods Library is hosted on the GitHub system. GitHub's disaster recovery facilities are described at https://github.com/security.

## 18.3 Methods Library testing

We distinguish between two types of tests performed on the Methods Library: Tests for individual functions in the packages (so-called "unit tests"), and tests for more complex functionality using simulations.

### 18.3.1 Unit test

A large set of automated validation tests is maintained and upgraded by OHDSI to enable the testing of source code against known data and known results. Each test begins with specifying some simple input data, then executes a function in one of the packages on this input, and evaluates whether the output is exactly what would be expected. For simple functions, the expected result is often obvious (for example when performing propensity score matching on example data containing only a few subjects), for more complicated functions the expected result may be generated using combinations of other functions available in R (for example, Cyclops, our large-scale regression engine, is tested among others by comparing results on simple problems with other regression routines in R). We aim for these tests in total to cover 100% of the lines of executable source code.

These tests are automatically performed when changes are made to a package (specifically, when changes are pushed to the package repository). Any errors noted during testing automatically trigger emails to the leadership of the Workgroups, and must be resolved prior to release of a new version of a package. The source code and expected results for these tests are available for review and use in other applications as may be appropriate. These tests are also available to end users and/or system administrators and can be run as part of their installation process to provide further documentation and objective evidence as to the accuracy, reliability and consistency of their installation of the Methods Library.

### 18.3.2 Simulation

For more complex functionality it is not always obvious what the expected output should be given the input. In these cases simulations are sometimes used, generating input given a specific statistical model, and establishing whether the functionality produces results in line with this known model. For example, in the SelfControlledCaseSeries package simulations are used to verify that the method is able to detect and appropriately model temporal trends in simulated data.

## 18.4 Summary

- An observational study should be implemented as an automated script that executes the entire analysis, from data in the CDM to the results, to ensure reproducability and transparancy.
- Custom study code should adhere to best programming practices, including abstraction, encapsulation, clear naming, and code reuse.
- Custom study code can be validated using code review or double coding.
- The Methods Library provided validated functionality that can be used in observational studies.
- The Methods Library is valdated by using a software development process aimed at creating valid software, and by testing.

# Chapter 19

# Method Validity

*Chapter lead: Martijn Schuemie*

When considering method validity we aim to answer the question

> Is this method valid for answering this question?

Where "method" includes not only the study design, but also the data and the implementation of the design. Method validity is therefore somewhat of a catch-all; It is often not possible to observe good method validity without good data quality, clinical validity, and software validity. Those aspects of evidence quality should have already been addressed separately before we consider method validity.

The core activity when establishing method validity is evaluating whether important assumptions in the analysis have been met. For example, we assume that propensity-score matching makes two populations comparable, but we need to evaluate whether this is the case. Where possible, empirical tests should be performed to verify these assumptions. We can for example generate diagnostics to show that our two populations are indeed comparable on a wide range of characteristics after matching. In OHDSI we have developed many standardized diagnostics that should be generated and evaluated whenever an analysis is performed.

In this chapter we will focus on the validity of methods use in population-level estimation. We will first briefly highlight some study design-specific diagnostics, and will then discuss diagnostics that are applicable to most if not all population-level estimation studies. Following this is a step-by-step description of how to execute these diagnostics using the OHDSI tools. We close this chapter with an advanced topic, reviewing the OHDSI Methods Benchmark and its application to the OHDSI Methods Library.

## 19.1 Design-specific diagnostics

For each study design there are diagnostics specific to such a design. Many of these diagnostics are implemented and readily available in the R packages of the OHDSI Methods Library. For example, Section 13.9 lists a wide range of diagnostics generated by the CohortMethod package, including:

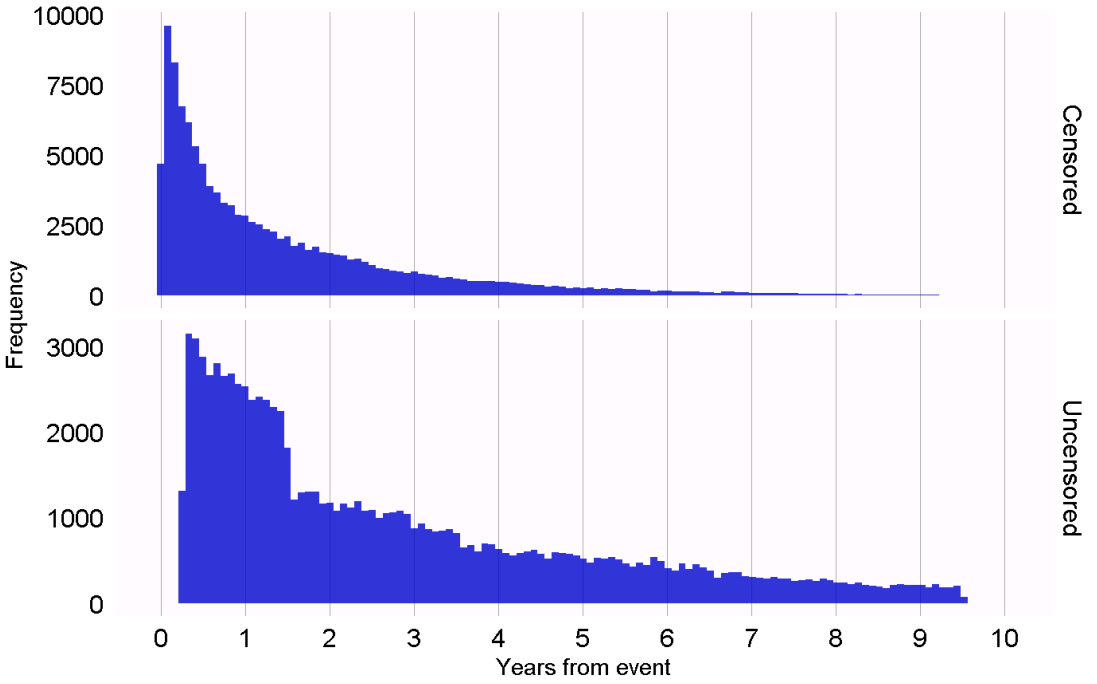



Figure 19.1: Time to observation end for those that are censored, and those that uncensored.

- **Propensity score distribution** to asses initial comparability of cohorts.
- **Propensity model** to identify potential variables that should be excluded from the model.
- **Covariate balance** to evaluate whether propensity score adjustment has made the cohorts comparable (as measured through baseline covariates).
- **Attrition** to observe how many subjects were excluded in the various analysis steps, which may inform on the generalizability of the results to the initial cohorts of interest.
- **Power** to assess whether enough data is available to answer the question.
- **Kaplan Meier curve** to asses typical time to onset, and whether the proportionality assumption underlying Cox models is met.

Other study designs require different diagnostics to test the different assumptions in those designs. For example, for the self-controlled case series (SCCS) design we may check the necessary assumption that the end of observation is independent of the outcome. This assumption is often violated in the case of serious, potentially lethal, events such as myocardial infarction. We can evaluate whether the assumption holds by generating the plot shown in Figure 19.1, which shows histograms of the time to observation period end for those that are censored, and those that uncensored. In our data we consider those whose observation period ends at the end date of data capture (the date when observation stopped for the entire data base, for example the date of extraction, or the study end date) to be uncensored, and all others to be censored. In Figure 19.1 we see only minor differences between the two distributions, suggesting our assumptions holds.

## 19.2 Diagnostics for all estimation

Next to the design-specific diagnostics, there are also several diagnostics that are applicable across all causal effect estimation methods. Many of these rely on the use of control hypotheses, research questions where the answer is already known. Using control hypotheses we can then evaluate whether our design produces results in line with the truth. Controls can be divided into negative controls and positive controls.

### 19.2.1 Negative controls

Negative controls are exposure-outcome pairs where one believes no causal effect exists, and including negative controls or "falsification endpoints" (Prasad and Jena, 2013) has been recommended as a means to detect confounding, (Lipsitch et al., 2010) selection bias and measurement error. (Arnold et al., 2016) For example, in one study (Zaadstra et al., 2008) investigating the relationship between childhood diseases and later multiple sclerosis (MS), the authors include three negative controls that are not believed to cause MS: a broken arm, concussion, and tonsillectomy. Two of these three controls produce statistically significant associations with MS, suggesting that the study may be biased.

We should select negative controls that are comparable to our hypothesis of interest, which means we typically select exposure-outcome pairs that either have the same exposure as the hypothesis of interest (so-called "outcome controls") or the same outcome ("exposure controls"). Our negative controls should further meet these criteria:

- The exposure **should not cause** the outcome. One way to think of causation is to think of the counterfactual: could the outcome be caused (or prevented) if a patient was not exposed, compared to if the patient had been exposed? Sometimes this is clear, for example ACEi are known to cause angioedema. Other times this is far less obvious. For example, a drug that may cause hypertension can therefore indirectly cause cardiovascular diseases that are a consequence of the hypertension.
- The exposure should also **not prevent or treat** the outcome. This is just another causal relationship that should be absent if we are to believe the true effect size (e.g. the hazard ratio) is 1.
- The negative control should **exist in the data**, ideally with sufficient numbers. We try to achieve this by prioritizing candidate negative controls based on prevalence.
- Negative controls should ideally be **independent**. For example, we should avoid having negative controls that are either ancestors of each other (e.g. "ingrown nail" and "ingrown nail of foot") or siblings (e.g. "fracture of left femur" and "fracture of right femur").
- Negative controls should ideally have **some potential for bias**. For example, the last digit of someone's social security number is basically a random number, and is unlikely to show confounding. It should therefore not be used as a negative control.

Some argue that negative controls should also have the same confounding structure as the exposure-outcome pair of interest. (Lipsitch et al., 2010) However, we believe this confounding structure is unknowable; The relationships between variables found in reality is often far more complex than people imagine. Also, even if the confounder structure were known, it is unlikely that a negative

control exists having that exact same confounding structure, but lacking the direct causal effect. For this reason in OHDSI we rely on a large number of negative controls, assuming that such a set represents many different types of bias, including the ones present in the hypothesis of interest.

The absence of a causal relationship between an exposure and an outcome is rarely documented. Instead, we often make the assumption that a lack of evidence of a relationship implies the lack of a relationship. This assumption is more likely to hold if the exposure and outcome have both been studies extensively, so a relationship could have been detected. For example, the lack of evidence for a completely novel drug likely implies a lack of knowledge, not the lack of a relationship. With this Principe in mind we have developed a semi-automated procedure for selecting negative controls (Voss et al., 2016). In brief, information from literature, product labels, and spontaneous reporting is automatically extracted and synthesized to produce a candidate list of negative controls. This list must then undergo manual review, not only to verify that the automated extraction was accurate, but also to impose additional criteria such as biological plausibility.

### 19.2.2 Positive controls

To understand the behavior of a method when the true relative risk is smaller or greater than one requires the use of positive controls, where the null is believed to not be true. Unfortunately, real positive controls for observational research tend to be problematic for three reasons. First, in most research contexts, for example when comparing the effect of two treatments, there is a paucity of positive controls relevant for that specific context. Second, even if positive controls are available, the magnitude of the effect size may not be known with great accuracy, and often depends on the population in which one measures it. Third, when treatments are widely known to cause a particular outcome, this shapes the behavior of physicians prescribing the treatment, for example by taking actions to mitigate the risk of unwanted outcomes, thereby rendering the positive controls useless as a means for evaluation. (Noren et al., 2014)

In OHDSI we therefore use synthetic positive controls, (Schuemie et al., 2018a) created by modifying a negative control through injection of additional, simulated occurrences of the outcome during the time at risk of the exposure. For example, assume that, during exposure to ACEi, n occurrences of our negative control outcome “ingrowing nail” were observed. If we now add an additional n simulated occurrences during exposure, we have doubled the risk. Since this was a negative control, the relative risk compared to the counterfactual was one, but after injection, it becomes two.

One issue that stands important is the preservation of confounding. The negative controls may show strong confounding, but if we inject additional outcomes randomly, these new outcomes will not be confounded, and we may therefore be optimistic in our evaluation of our capacity to deal with confounding for positive controls. To preserve confounding, we want the new outcomes to show similar associations with baseline subject-specific covariates as the original outcomes. To achieve this, for each outcome we train a model to predict the survival rate with respect to the outcome during exposure using covariates captured prior to exposure. These covariates include demographics, as well as all recorded diagnoses, drug exposures, measurements, and medical procedures. An L1-regularized Poisson regression (Suchard et al., 2013) using 10-fold cross-validation to select the regularization hyperparameter fits the prediction model. We then use the predicted rates to sample

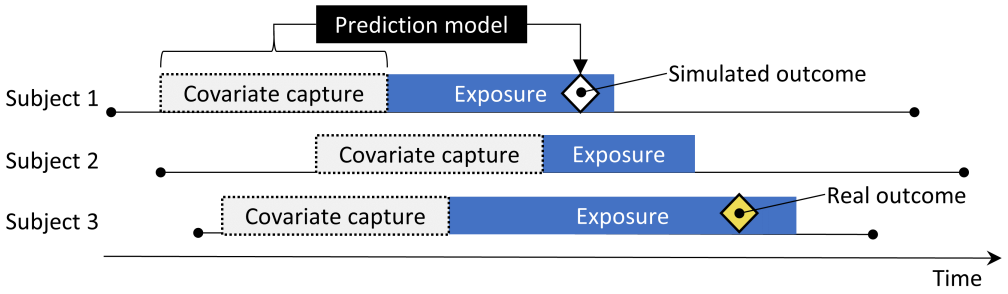



Figure 19.2: Synthesizing positive controls from negative controls.

simulated outcomes during exposure to increase the true effect size to the desired magnitude. The resulting positive control thus contains both real and simulated outcomes.

Figure 19.2 depicts this process. Note that although this procedure simulates several important sources of bias, it does not capture all. For example, some effects of measurement error are not present. The synthetic positive controls imply constant positive predictive value and sensitivity, which may not be true in reality.

Although we refer to a single true "effect size" for each control, different methods estimate different statistics of the treatment effect. For negative controls, where we believe no causal effect exists, all such statistics, including the relative risk, hazard ratio, odds ratio, incidence rate ratio, both conditional and marginal, as well as the average treatment effect in the treated (ATT) and the overall average treatment effect (ATE) will be identical to 1. Our process for creating positive controls synthesizes outcomes with a constant incidence rate ratio over time and between patients, using a model conditioned on the patient where this ratio is held constant, up to the point where the marginal effect is achieved. The true effect size is thus guaranteed to hold as the marginal incidence rate ratio in the treated. Under the assumption that our outcome model used during synthesis is correct, this also holds for the conditional effect size and the ATE. Since all outcomes are rare, odds ratios are all but identical to the relative risk.

### 19.2.3 Empirical evaluation

Based on the estimates of a particular method for the negative and positive controls, we can then understand the operating characteristic by computing a range of metrics, for example:

- **Area Under the receiver operator Curve (AUC)**: the ability to discriminate between positive and negative controls.
- **Coverage**: how often the true effect size is within the 95% confidence interval.
- **Mean precision**: precision is computed as $1/(standard\ error)^2$, higher precision means narrower confidence intervals. We use the geometric mean to account for the skewed distribution of the precision.
- **Mean squared error (MSE)**: Mean squared error between the log of the effect size point-estimate and the log of the true effect size.

- **Type 1 error**: For negative controls, how often was the null rejected (at $\alpha = 0.05$). This is equivalent to the false positive rate and $1 - specificity$.
- **Type 2 error**: For positive controls, how often was the null not rejected (at $\alpha = 0.05$). This is equivalent to the false negative rate and $1 - sensitivity$.
- **Non-estimable**: For how many of the controls was the method unable to produce an estimate? There can be various reasons why an estimate cannot be produced, for example because there were no subjects left after propensity score matching, or because no subjects remained having the outcome.

Depending on our use case, we can evaluate whether these operating characteristics are suitable for our goal. For example, if we wish to perform signal detection, we may care about type 1 and type 2 error, or if we are willing to modify our $\alpha$ threshold, we may inspect the AUC instead.

### 19.2.4 P-value calibration

Often the type 1 error (at $\alpha = 0.05$) is larger than 5%. In other words, we are often more likely than 5% to reject the null hypothesis when in fact the null hypothesis is true. The reason is that the p-value only reflects random error, the error due to having a limited sample size. It does not reflect systematic error, for example the error due to confounding. OHDSI has developed a process for calibrating p-values to restore the type 1 error to nominal. (Schuemie et al., 2014) We derive an empirical null distribution from the actual effect estimates for the negative controls. These negative control estimates give us an indication of what can be expected when the null hypothesis is true, and we use them to estimate an empirical null distribution.

Formally, we fit a Gaussian probability distribution to the estimates, taking into account the sampling error of each estimate. Let $\hat{\theta}_i$ denote the estimated log effect estimate (relative risk, odds or incidence rate ratio) from the $i$th negative control drug–outcome pair, and let $\hat{\tau}_i$ denote the corresponding estimated standard error, $i = 1, \ldots, n$. Let $\theta_i$ denote the true log effect size (assumed 0 for negative controls), and let $\beta_i$ denote the true (but unknown) bias associated with pair $i$ , that is, the difference between the log of the true effect size and the log of the estimate that the study would have returned for control $i$ had it been infinitely large. As in the standard p-value computation, we assume that $\hat{\theta}_i$ is normally distributed with mean $\theta_i + \beta_i$ and standard deviation $\hat{\tau}_i^2$. Note that in traditional p-value calculation, $\beta_i$ is always assumed to be equal to zero, but that we assume the $\beta_i$'s, arise from a normal distribution with mean $\mu$ and variance $\sigma^2$. This represents the null (bias) distribution. We estimate $\mu$ and $\sigma^2$ via maximum likelihood. In summary, we assume the following:

$$\beta_i \sim N(\mu, \sigma^2) \text{ and} \hat{\theta}_i \sim N(\theta_i + \beta_i, \tau_i^2)$$

where $N(a, b)$ denotes a Gaussian distribution with mean $a$ and variance $b$, and estimate $\mu$ and $\sigma^2$ by maximizing the following likelihood:

$$L(\mu, \sigma|\theta, \tau) \propto \prod_{i=1}^{n} \int p(\hat{\theta}_i|\beta_i, \theta_i, \hat{\tau}_i) p(\beta_i|\mu, \sigma) \mathrm{d}\beta_i$$

yielding maximum likelihood estimates $\hat{\mu}$ and $\hat{\sigma}$. We compute a calibrated p-value that uses the empirical null distribution. Let $\hat{\theta}_{n+1}$ denote the log of the effect estimate from a new drug–outcome pair, and let $\hat{\tau}_{n+1}$ denote the corresponding estimated standard error. From the aforementioned assumptions and assuming $\beta_{n+1}$ arises from the same null distribution, we have the following:

$$\hat{\theta}_{n+1} \sim N(\hat{\mu}, \hat{\sigma} + \hat{\tau}_{n+1})$$

When $\hat{\theta}_{n+1}$ is smaller than $\hat{\mu}$, the one-sided calibrated p-value for the new pair is then

$$\phi\left(\frac{\theta_{n+1} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \hat{\tau}_{n+1}^2}}\right)$$

where $\phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution. When $\hat{\theta}_{n+1}$ is bigger than $\hat{\mu}$, the one-sided calibrated p-value is then

$$1 - \phi\left(\frac{\theta_{n+1} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \hat{\tau}_{n+1}^2}}\right)$$

### 19.2.5 Confidence interval calibration

Similarly, we typically observe that the coverage of the 95% confidence interval is less than 95%: the true effect size is inside the 95% confidence interval less than 95% of the time. For confidence interval calibration (Schuemie et al., 2018a) we extend the framework for p-value calibration by also making use of our positive controls. Typically, but not necessarily, the calibrated confidence interval is wider than the nominal confidence interval, reflecting the problems unaccounted for in the standard procedure (such as unmeasured confounding, selection bias and measurement error) but accounted for in the calibration.

Formally, we assume that $beta_i$, the bias associated with pair $i$, again comes from a Gaussian distribution, but this time using a mean and standard deviation that are linearly related to $theta_i$, the true effect size:

$$\beta_i \sim N(\mu(\theta_i), \sigma^2(\theta_i))$$

where

$$\mu(\theta_i) = a + b \times \theta_i \text{ and} \sigma(\theta_i)^2 = c + d \times \mid \theta_i \mid$$

We estimate $a$, $b$, $c$ and $d$ by maximizing the marginalized likelihood in which we integrate out the unobserved $\beta_i$:

$$l(a,b,c,d|\theta,\hat{\theta},\hat{\tau}) \propto \prod_{i=1}^{n} \int p(\hat{\theta}_i|\beta_i,\theta_i,\hat{\tau}_i)p(\beta_i|a,b,c,d,\theta_i)\mathrm{d}\beta_i,$$

yielding maximum likelihood estimates $(\hat{a},\hat{b},\hat{c},\hat{d})$.

We compute a calibrated CI that uses the systematic error model. Let $\hat{\theta}_{n+1}$ again denote the log of the effect estimate for a new outcome of interest, and let $\hat{\tau}_{n+1}$ denote the corresponding estimated standard error. From the assumptions above, and assuming $\beta_{n+1}$ arises from the same systematic error model, we have:

$$\hat{\theta}_{n+1} \sim N(\theta_{n+1} + \hat{a} + \hat{b} \times \theta_{n+1}, \hat{c} + \hat{d}\times \mid \theta_{n+1} \mid) + \hat{\tau}_{n+1}^2).$$

We find the lower bound of the calibrated 95% CI by solving this equation for $\theta_{n+1}$:

$$\Phi\left(\frac{\theta_{n+1} + \hat{a} + \hat{b} \times \theta_{n+1} - \hat{\theta}_{n+1}}{\sqrt{(\hat{c} + \hat{d}\times \mid \theta_{n+1} \mid) + \hat{\tau}_{n+1}^2}}\right) = 0.025,$$

where $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution. We find the upper bound similarly for probability 0.975. We define the calibrated point estimate by using probability 0.5.

Both p-value calibration and confidence interval calibration are implemented in the EmpiricalCalibration package.

### 19.2.6 Replication across sites

Another form of method validation comes from executing the study across several different databases that represent different populations, different health care systems, and/or different data capture processes. Prior research has shown that executing the same study design across different databases can produce vastly different effect size estimates, (Madigan et al., 2013) suggesting that either the effect differs greatly for different populations, or that the design does not adequately address the different biases found in the different databases. In fact, we observe that accounting for residual bias in a database through empirical calibration of confidence intervals can greatly reduce between-study heterogeneity. (Schuemie et al., 2018a)

One way to express between-database heterogeneity is the $I^2$ score, describing the percentage of total variation across studies that is due to heterogeneity rather than chance. (Higgins et al., 2003) A naive categorization of values for $I^2$ would not be appropriate for all circumstances, although one could tentatively assign adjectives of low, moderate, and high to $I^2$ values of 25%, 50%, and 75%. In a study estimating the effects for many depression treatments using a new-user cohort design with large-scale propensity score adjustment, Schuemie et al. (2018b) observed only 58% of the estimates to have an $I^2$ below 25%. After empirical calibration this increased to 83%.

Observing between-database heterogeneity casts doubt on the validity of the estimates. Unfortunately, the inverse is not true. Not observing heterogeneity does not guarantee an unbiased estimate. It is not unlikely that all databases share a similar bias, and that all estimates are therefore consistently wrong.

### 19.2.7 Sensitivity analyses

When designing a study there are often design choices that are uncertain. For example, should propensity score matching of stratification be used? If stratification is used, how many strata? What is the appropriate time-at-risk? When faced with such uncertainty, one solution is to evaluate various options, and observe the sensitivity of the results to the design choice. If the estimate remains the same under various options, we can say the study is robust to the uncertainty.

This definition of sensitivity analysis should not be confused with the definitions used by others such as Rosenbaum (2005), who define sensitivity analysis to "appraise how the conclusions of a study might be altered by hidden biases of various magnitudes".

## 19.3 Method validation in practice

Here we build on the example in Chapter 13, where we investigate the effect of ACE inhibitors (ACEi) on the risk of angioedema and acute myocardial infarction (AMI), compared to thiazides and thiazide-like diuretics (THZ). In that chapter we already explore many of the diagnostics specific to the design we used: the cohort method. Here, we apply additional diagnostics that could also have been applied had other designs been used. If the study is implemented using ATLAS as described in Section 13.7 these diagnostics are available in the Shiny app that is included in the study R package generated by ATLAS. If the study is implemented using R instead, as described in Section 13.8, then R functions available in the various packages should be used, as described in the next sections.

### 19.3.1 Selecting negative controls

We must select negative controls, exposure-outcome pairs where no causal effect is believed to exist. For comparative effect estimation such as our example study, we select negative control outcomes that are believed to be neither caused by the target nor the comparator exposure. We want enough negative controls to make sure we have a diverse mix of biases represented in the controls, and also to allow empirical calibration. As a rule-of-thumb we typically aim to have 50-100 such negative controls. We could come up with these controls completely manually, but fortunately ATLAS provides features to aid the selection of negative controls using data from literature, product labels, and spontaneous reports.

To generate a candidate list of negative controls, we first must create a concept set containing all exposures of interest. In this case we select all ingredients in the ACEi and THZ classes, as shown

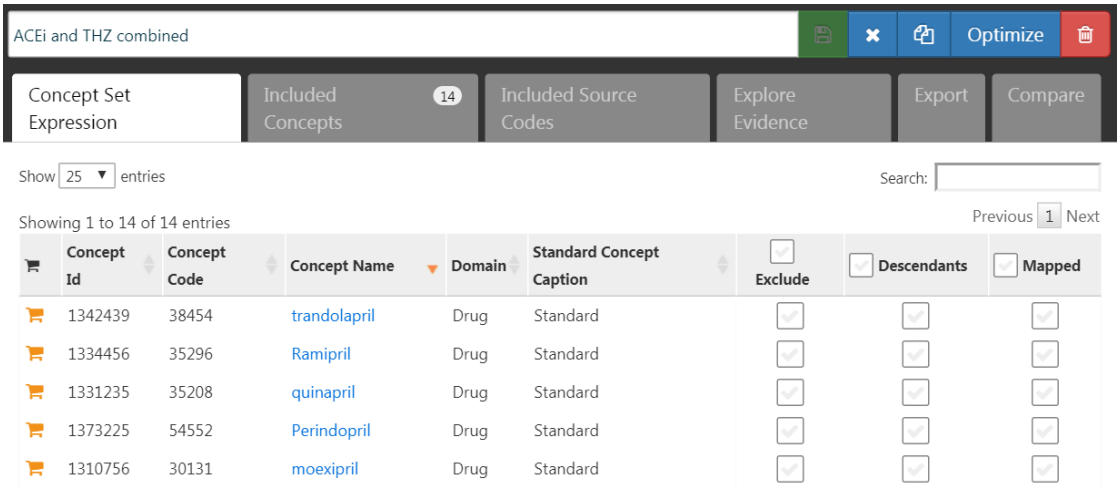



Figure 19.3: A concept set containing the concepts defining the target and comparator exposures.

in Figure 19.3.

Next, we go to the "Explore Evidence" tab, and click on the Generate button. Generating the evidence overview will take a few minutes, after which you can click on the View Evidence button. This will open the list of outcomes as shown in Figure 19.4.

This list shows condition concepts, along with an overview of the evidence linking the condition to any of the exposures we defined. For example, we see the number of publications that link the exposures to the outcomes found in PubMed using various strategies, the number of product labels of our exposures of interest that list the condition as a possible adverse effect, and the number of spontaneous reports. By default the list is sorted to show candidate negative controls first. It is then sorted by the "Sort Order", which represents the prevalence of the condition in a collection of observational databases. The higher the Sort Order, the higher the prevalence. Although the prevalence in these databases might not correspond with the prevalence in the database we wish to run the study, it is likely a good approximation.

The next step is to manually review the candidate list, typically starting at the top, so with the most prevalent condition, and working our way down until we are satisfied we have enough. One typical way to do this is to export the list to a CSV (comma separated values) file, and have clinicians review these, considering the criteria mentioned in Section 19.2.1.

For our example study we select the 76 negative controls listed in Appendix C.1.

### 19.3.2 Including controls

Once we have defined our set of negative controls we must include them in our study. First we must define some logic for turning our negative control condition concepts into outcome cohorts. Section 13.7.3 discusses how ATLAS allows creating such cohorts based on a few choices the user must

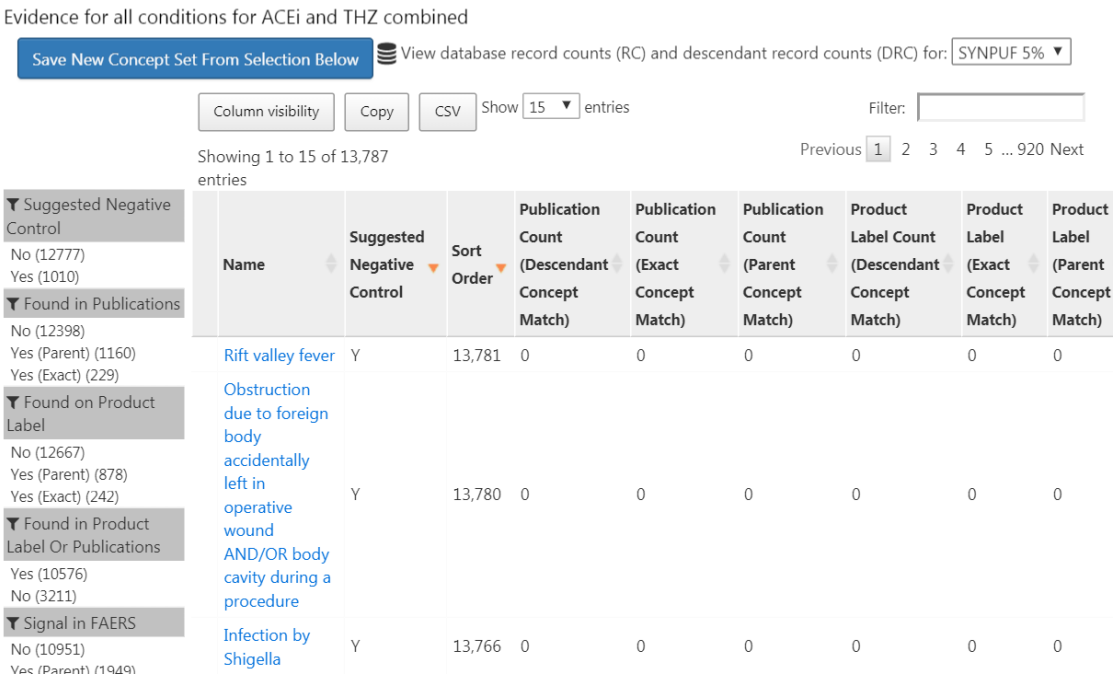



Figure 19.4: Candidate control outcomes with an overview of the evidence found in literature, product labels, and spontaneous reports.

make. Often we simply choose to create a cohort based on any occurrence of a negative control concept or any of its descendants. If the study is implemented in R then SQL (Structured Query Language) can be used to construct the negative control cohorts. Chapter 10 describes how cohorts can be created using SQL and R. We leave it as an exercise for the reader to write the appropriate SQL and R.

The OHDSI tools also provide functionality for automatically generating and including positive controls derived from the negative controls. This functionality can be found in the Evaluation Settings section in ATLAS described in Section 13.7.3, and is implemented in the `injectSignals` function in the MethodEvaluation package. Here we generate three positive controls for each negative control, with true effect sizes of 1.5, 2, and 4, using a survival model:

```
library(MethodEvaluation)
# Create a data frame with all negative control exposure-
# outcome pairs, using only the target exposure (ACEi = 1).
eoPairs <- data.frame(exposureId = 1,
                      outcomeId = ncs)

pcs <- injectSignals(connectionDetails = connectionDetails,
                     cdmDatabaseSchema = cdmDbSchema,
                     exposureDatabaseSchema = cohortDbSchema,
                     exposureTable = cohortTable,
```

```
                    outcomeDatabaseSchema = cohortDbSchema,
                    outcomeTable = cohortTable,
                    outputDatabaseSchema = cohortDbSchema,
                    outputTable = cohortTable,
                    createOutputTable = FALSE,
                    modelType = "survival",
                    firstExposureOnly = TRUE,
                    firstOutcomeOnly = TRUE,
                    removePeopleWithPriorOutcomes = TRUE,
                    washoutPeriod = 365,
                    riskWindowStart = 1,
                    riskWindowEnd = 0,
                    addExposureDaysToEnd = TRUE,
                    exposureOutcomePairs = eoPairs,
                    effectSizes = c(1.5, 2, 4),
                    cdmVersion = cdmVersion,
                    workFolder = file.path(outputFolder,
                                           "pcSynthesis"))
```

Note that we must mimic the time-at-risk settings used in our estimation study design. The `injectSignals` function will extract information about the exposures and negative controls outcomes, fit outcome models per exposure-outcome pair, and synthesize outcomes. The positive control outcome cohorts will be added to the cohort table specified by `cohortDbSchema` and `cohortTable`. The resulting `pcs` data frame contains the information on the synthesized positive controls.

Next we must execute the same study used to estimate the effect of interest to also estimate effects for the negative and positive controls. Setting the set of negative controls in the comparisons dialog in ATLAS instructs ATLAS to compute estimates for these controls. Similarly, specifying that positive controls be generated in the Evaluation Settings includes these in our analysis. In R, the negative and positive controls should be treated as any other outcome. All estimation packages in the OHDSI Methods Library readily allow estimation of many effects in an efficient manner.

### 19.3.3 Empirical performance

Figure 19.5 shows the estimated effect sizes for the negative and positive controls included in our example study, stratified by true effect size. This plot is included in the Shiny app that comes with the study R package generated by ATLAS, and can be generated using the `plotControls` function in the MethodEvaluation package. Note that the number of controls is often lower than what was defined because there was not enough data to either produce an estimate, or to synthesize a positive control.

Based on these estimates we can compute the metrics shown in Table 19.1 using the `computeMetrics` function in the MethodEvaluation package.

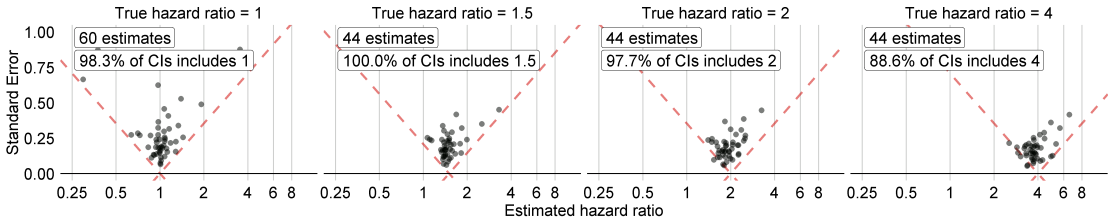



Figure 19.5: Estimates for the negative (true hazard ratio = 1) and positive controls (true hazard ratio > 1). Each dot represents a control. Estimates below the dashed line have a confidence interval that doesn't include the true effect size.

Table 19.1: Method performance metrics derived from the negative and positive control estimates.

| Metric | Value |
|---|---|
| AUC | 0.96 |
| Coverage | 0.97 |
| Mean Precision | 19.33 |
| MSE | 2.08 |
| Type 1 error | 0.00 |
| Type 2 error | 0.18 |
| Non-estimable | 0.08 |

We see that coverage and type 1 error are very close to their nominal values of 95% and 5%, respectively, and that the AUC is very high. This is certainly not always the case.

Note that although in Figure 19.5 not all confidence intervals include one when the true hazard ratio is one, the type 1 error in Table 19.1 is 0%. This is an exceptional situation, caused by the fact that confidence intervals in the Cyclops package are estimated using likelihood profiling, which is more accurate than traditional methods but can result in asymmetric confidence intervals. The p-value instead is computed assuming symmetrical confidence intervals, and this is what was used to compute the type 1 error.

### 19.3.4 P-value calibration

We can use the estimates for our negative controls to calibrate our p-values. This is done automatically in the Shiny app, and can be done manually in R. Assuming we have created the summary object `summ` as described in Section 13.8.6, we can plot the empirical calibration effect plot:

```
# Estimates for negative controls (ncs) and outcomes of interest (ois):
ncEstimates <- summ[summ$outcomeId %in% ncs, ]
oiEstimates <- summ[summ$outcomeId %in% ois, ]

library(EmpiricalCalibration)
plotCalibrationEffect(logRrNegatives = ncEstimates$logRr,
                      seLogRrNegatives = ncEstimates$seLogRr,
                      logRrPositives = oiEstimates$logRr,
```

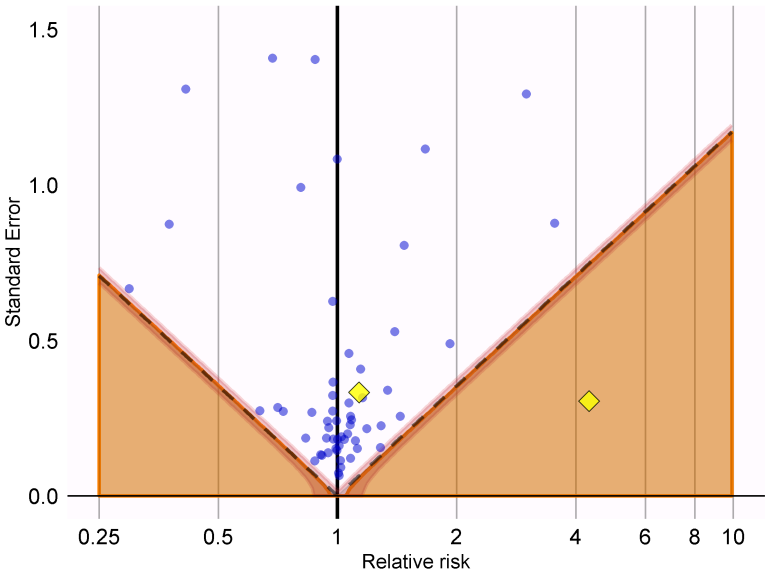



Figure 19.6: P-value calibration: estimates below the dashed line have a conventional p < 0.05. Estimates in the orange area have calibrated p < 0.05. The pink area denotes the 95% credible interval around the edge of the orange area. Blue dots indicate negative controls. Yellow diamonds indicate outcomes of interest.

stands out from the negative control, and falls well within the area where both uncalibrated and calibrated p-values are smaller than 0.05.

We can compute the calibrated p-values:

```
null <- fitNull(logRr = ncEstimates$logRr,
                seLogRr = ncEstimates$seLogRr)
calibrateP(null,
           logRr= oiEstimates$logRr,
           seLogRr = oiEstimates$seLogRr)
```

```
## [1] 1.604351e-06 7.159506e-01
```

And contrast these with the uncalibrated p-values:

```
oiEstimates$p
```

```
## [1] [1] 1.483652e-06 7.052822e-01
```

As expected, because little to no bias was observed, the uncalibrated and calibrated p-values are very similar.

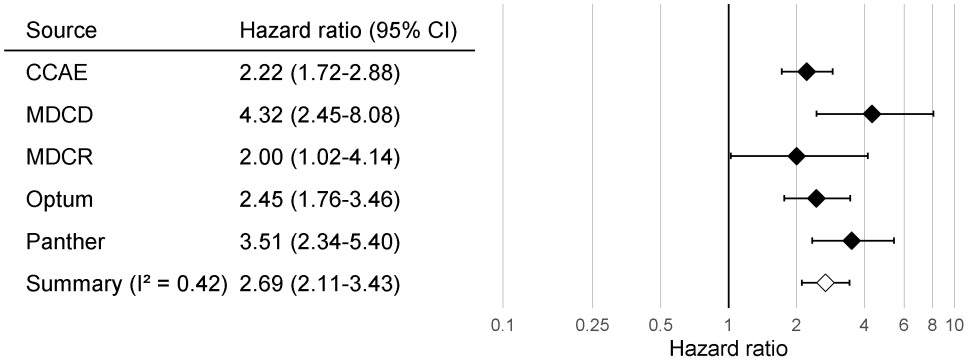



Figure 19.7: Effect size estimates and 95% confidence intervals (CI) from five different databases and a meta-analytic estimatewhen comparing ACE inhibitors to thiazides and thiazide-like diuretics for the risk of angioedema.

### 19.3.5 Confidence interval calibration

Similarly, we can use the estimates for our negative and positive controls to calibrate the confidence intervals. The Shiny app automatically reports the calibrate confidence intervals. In R we can calibrate intervals using the `fitSystematicErrorModel` and `calibrateConfidenceInterval` functions in the EmpiricalCalibration package, as described in detail in the "Empirical calibration of confidence intervals" vignette.

Before calibration, the estimated hazard ratios (95% confidence interval) are 4.32 (2.45 - 8.08) and 1.13 (0.59 - 2.18), for angioedema and AMI respectively. The calibrated hazard ratios are 4.75 (2.52 - 9.04) and 1.15 (0.58 - 2.30).

### 19.3.6 Between-database heterogeneity

Just as we executed our analysis on one database, in this case the IBM MarketScan Medicaid (MDCD) database, we can also run the same analysis code on other databases that adhere to the Common Data Model (CDM). Figure 19.7 shows the forest plot and meta-analytic estimates (assuming random effects) (DerSimonian and Laird, 1986) across a total of five databases for the outcome of angioedema. This figure was generated using the `plotMetaAnalysisForest` function in the EvidenceSynthesis package.

Although all confidence intervals are above one, suggesting agreement on the fact that there is an effect, the $I^2$ suggests between-database heterogeneity. However, if we compute the $I^2$ using the calibrated confidence intervals as shown in Figure 19.8, we see that this heterogeneity can be explained by the bias measured in each database through the negative and positive controls. The empirical calibration appears to properly taken this bias into account.

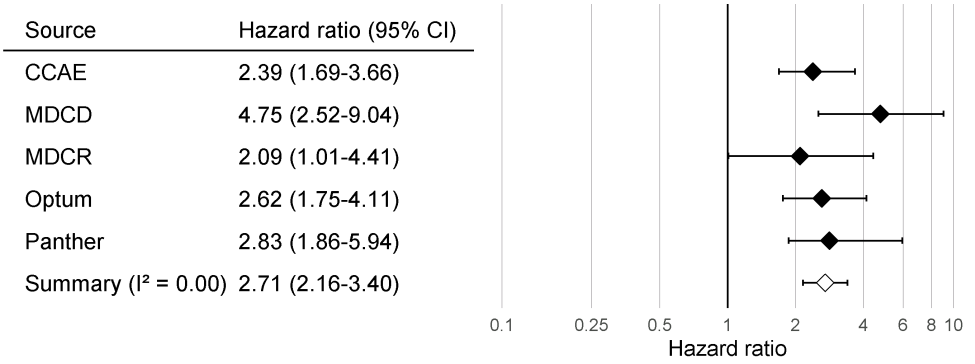



Figure 19.8: Calibrated Effect size estimates and 95% confidence intervals (CI) from five different databases and a meta-analytic estimate for the hazard ratio of angioedema when comparing ACE inhibitors to thiazides and thiazide-like diuretics.

### 19.3.7 Sensitivity analyses

One of the design choices in our analysis was to use variable-ratio matching on the propensity score. However, we could have also used stratification on the propensity score. Because we we are uncertain about this choice, we may decide to use both. Table 19.2 shows the effect size estimates for AMI and angioedema, both calibrated and uncalibrated, when using variable-ratio matching and stratification (with 10 equally-sized strata).

Table 19.2: Uncalibrated and calibrated hazard ratios (95% confidence interval) for the two analysis variants.

| Outcome | Adjustment | Uncalibrated | Calibrated |
|---|---|---|---|
| Angioedema | Matching | 4.32 (2.45 - 8.08) | 4.75 (2.52 - 9.04) |
| Angioedema | Stratification | 4.57 (3.00 - 7.19) | 4.52 (2.85 - 7.19) |
| Acute myocardial infarction | Matching | 1.13 (0.59 - 2.18) | 1.15 (0.58 - 2.30) |
| Acute myocardial infarction | Stratification | 1.43 (1.02 - 2.06) | 1.45 (1.03 - 2.06) |

We see that the estimates from the matched and stratified analysis are in strong agreement, with the confidence intervals for stratification falling completely inside of the confidence intervals for matching. This suggests that our uncertainty around this design choice does not impact the validity of our estimates. Stratification does appear to give us more power (narrower confidence intervals), which is not surprising since matching results in loss of data, whereas stratification does not. The price for this could be an increase in bias, due to within-strata residual confounding, although we see no evidence of increased bias reflected in the calibrated confidence intervals.

Study diagnostics allow us to evaluate design choices even before fully executing a study. It is recommended not to finalize the protocol before generating and reviewing all study diagnostics. To avoid p-hacking (adjusting the design to achieve a desired result), this should be done while blinded to the effect size estimate of interest.

## 19.4 OHDSI Methods Benchmark

carefully selected negative controls that can be stratified into eight categories, with the controls in each category either sharing the same exposure or the same outcome. From these 200 negative controls, 600 synthetic positive controls are derived as described in Section 19.2.2. To evaluate a method, it must be used to produce effect size estimates for all controls, after which the metrics described in Section 19.2.3 can be computed. The benchmark is publicly available, and can be deployed as described in the Running the OHDSI Methods Benchmark vignette in the MethodEvaluation package.

We have run all the methods in the OHDSI Methods Library through this benchmark, with various analysis choices per method. For example, the cohort method was evaluated using propensity score matching, stratification, and weighting. This experiment was executed on four large observational healthcare databases. The results, viewable in an online Shiny app[1], show that although several methods show high AUC (the ability to distinguish positive controls from negative controls), most methods in most settings demonstrate high type 1 error and low coverage of the 95% confidence interval, as shown in Figure 19.9.

This emphasizes the need for empirical evaluation and calibration: if no empirical evaluation is performed, which is true for almost all published observational studies, we must assume a prior informed by the results in Figure 19.9, and conclude that it is likely that the true effect size is not contained in the 95% confidence interval!

Our evaluation of the designs in the Methods Library also shows that empirical calibration restores type 1 error and coverage to their nominal values, although often at the cost of increasing type 2 error and decreasing precision.

## 19.5 Summary

- A method's validity depends on whether the assumptions underlying the method are met.
- Where possible, these assumptions should be empirically tested using study diagnostics.
- Control hypotheses, questions where the answer is known, should be used to evaluate whether a specific study design produces answers in line with the truth.
- Often, p-values and confidence intervals do not demonstrate nominal characteristics as measured using control hypotheses.
- These characteristics can often be restored to nominal using empirical calibration.
- Study diagnostics can be used to guide analytic design choices and adapt the protocol, as long as the researcher remains blinded to the effect of interest to avoid p-hacking.

[1] http://data.ohdsi.org/MethodEvalViewer/


Figure 19.9: Coverage of the 95% confidence interval for the methods in the Methods Library. Each dot represents the performance of a specific set of analysis choices. The dashed line indicates nominal performance (95% coverage). SCCS = Self-Controlled Case Series, GI = Gastrointestinal, IBD = inflammatory bowel disease.

## 19.6 Exercises

Todo

# Part V

# OHDSI Studies

# Chapter 20

# Study steps

Writing the protocol, OHDSI style: http://www.ohdsi.org/web/wiki/lib/exe/fetch.php?media=projects:workgroups:wg_study_protocols_eastern_hemisphere.pptx

Study reproducibility (Martijn has some slides that might help: http://www.ohdsi.org/web/wiki/lib/exe/fetch.php?media=projects:workgroups:wg_study_reproducability.pptx )

# Chapter 21

# OHDSI Network Research

Contributors: Greg Klebanov, Kristin Kostka & Vojtech Huser

The mission of OHDSI is to generate high-quality evidence through observational research. A primary way this is accomplished is through collaborative research studies. In prior chapters we discussed how the OHDSI community has authored standards and tools to facilitate high-quality, reproducible research, including Standardized Vocabularies , the Common Data Model (CDM) , analytical methods packages, ATLAS and the study steps (Chapter20) to run a retrospective database study. OHDSI Network Studies represent the culmination of a transparent, consistent and reproducible way to conduct research across a large number of geographically dispersed data. In this chapter we will discuss what constitutes an OHDSI network study, how to run a network study and discuss enabling technologies such as the ARACHNE Research Network.

## 21.1 What is the OHDSI Research Network?

The OHDSI Research Network is an international collaboration of researchers seeking to advance observational data research in healthcare. Today, the network consists of over 1.2 billion patient records (~650 million de-duplicated patient records) in the OMOP CDM. This includes more than 200 researchers and 82 observational health databases across 17 countries with regional central coordinating centers housed at Columbia University (USA), Erasmus Medical Center (Europe) and Ajou University (South Korea). The OHDSI community continues to grow rapidly across Europe (in collaboration with the IMI EHDEN project), Central America (e.g. Argentina, Brazil, Colombia), and Asia (e.g. China, Japan, Singapore).

OHDSI is open network, inviting healthcare institutions across the globe with active patient data to join the network and convert data to the OMOP CDM. As OMOP data conversions are complete, collaborators are invited to report site information in the Data Network census maintained by the OHDSI Program Manager (beaton@ohdsi.org). Each OHDSI network site participates voluntarily. There are no hard obligations. Each site opts-in to each respective network study. In each study, data remains at the site behind a firewall. No data pooling across network sites. Only aggregate results

are shared.

**Benefits of Joining the OHDSI Network** Unlock the power of institutional data: Transforming institutional EHR data in the OMOP Common Data Model enables clinical research on populations of your patients, something EHR systems don't support. Access to free tools: OHDSI publishes free, open source tools for data characterization and analytics (e.g. browsing the clinical concepts, defining and characterizing cohorts, running Population-Level Estimation and Patient-Level Prediction studies). Participate in a premier research community: Author and publish network research, gain access to eminent leaders in global real-world evidence community. Buildout Quality Benchmarks: Network can validate quality improvement benchmarks against other institutions (e.g. On average how long does it take to get an appendectomy discharged?)

## 21.2 What is an OHDSI Network Study?

In the study steps chapter (Chapter20), we discussed the steps to execute a retrospective database study using the OMOP CDM. A study may be conducted on a single OMOP CDM or on multiple OMOP CDMs. It can be conducted within a single institution's OMOP CDM data or across many institutions. There is no requirement that an OHDSI research study package be shared across the entire OMOP network. In fact, there may be legitimate instances when a study protocol is written for specific clinical practice that cannot be generalized to the entirety of the network. The principal investigator of each OHDSI research study will determine which, if any, sites they would like to include in an analysis.

**When is a study considered a network study?** An OHDSI research project becomes a network study when it is published and shared for execution across the OHDSI community.

Elements of a Network Study: - Must have a protocol (a description of the analysis to be performed) - Must have a study code package designed for the OMOP CDM - Must be executed across two or more network sites (not just 2 or more databases at a single site) - Encouraged to publish all documentation on GitHub - At the end of the analysis, the results are made available in Github or other public repository (e.g. a Shiny Application)

## 21.3 Executing an OHDSI Network Study

Conducting an OHDSI Network Study requires a substantial amount of preparation to ensure success.

"You'll never walk alone in your OHDSI journey." - Peter Rijnbeek

**New to Network Studies?** The OHDSI Study Nurture Committee is a resource for you as you navigate your journey. This committee helps train and guide researchers to complete OHDSI Studies including how to effectively use OHDSI tools, providing guidance to the OHDSI study design for increase reproducibility and reliability and assisting with helping study investigators recruit data partners to run study packages.

Running an OHDSI Network Study has three distinct stages:

- Study Feasibility and Design
- Study Execution
- Results Dissemination and Publication

### 21.3.1 Study Feasibility and Design

The study feasibility stage *(pre-study stage)* is focused on supporting a definition of a study and a creation of the study protocol, *e.g. undertaking activities to make sure the study is feasible to be executed as described in the formal protocol.*

The feasibility stage does not have a well-defined process but rather is driven by various supporting activities, including identification and enrollment of relevant databases that contain the targeted patient population with required drug exposure, procedure information, condition or demographics information through data characterization, validating and agreeing on target analytical methods and algorithms. These activities may involve sharing JSON files of cohort definitions from ATLAS and provisional test of study R packages. A study lead may have enough data to do this inside their own organization or may opt for support from other OHDSI network sites.

The outcome of the feasibility stage is generation of a final protocol as well as a list of target collaborators. The formal protocol will detail the study team, including the designated study lead (often the corresponding author for publication purposes), and information on the timeline for the study. The protocol is a critical component for additional network sites to review, approve and execute the study package on their OMOP CDM data. A protocol must include information on study population, the methods being used, how the results will be stored and analyzed as well as how the study results will be disseminated after completion (e.g. a publication, a poster, etc).

### 21.3.2 Study Execution

After completing feasibility, a study advances to the execution phase. The key activities in executing a network study include the following:

- The study lead formally initiates a new OHDSI network study with the OHDSI Coordinating Center. *In tandem, this may include undertaking other organization-specific processes to approve an OHDSI study.*
- The study lead publishes the study protocol to the OHDSI GitHub.

- The study lead announces the study on the OHDSI Community Call and OHDSI Forum, inviting participating centers and collaborators.
- Study participating organizations assemble teams within each site, assign study roles (e.g. data analyst(s) executing the study package, site leadership reviewing the study design and manuscript).
- The data scientist/statisticians for the study lead will use a study protocol to design study analyses and generate study code.
- The data scientist/statisticians will conduct a feasibility test of study code within their own environment. The package will be shared to 1-2 network sites for additional validation.
- The data scientist/statistician will publish the validated study code in the OHDSI GitHub for execution at participating sites.
- Site data scientists/statisticians access the OHDSI study package and generate results in the standardized format following OHDSI guidelines. Each participating site will follow internal institutional processes regarding data sharing rules. **Sites should not share results unless approval is obtained from IRB or other institutional approval processes.**
- Data scientist/statisticians and Study Lead collect and review the analysis execution results.
- Iterate steps 5-7, if reasonable adjustments required.
- Collaboratively finalize study results. Study lead disseminates study results (e.g. a Shiny Application).
- Study lead formally closes the study out with OHDSI Coordinating Center.

While OHDSI processes can be executed rapidly, it is advised to allow for a few weeks to months for all participating sites to execute the study and receive appropriate approvals to publish results. A study leads should set study milestones and anticipated closure date in advance to assist with managing the overall study timeline.

### 21.3.3 Results Dissemination and Publication

During this stage, the study lead will collaborate with other sites on various administrative tasks, such as manuscript development and optimizing data visualizations.

Not sure where to publish your OHDSI network study? Consult JANE (Journal/Author Name Estimator), a tool which takes your abstract and scans publications for relevance and fit (http://jane.biosemantics.org/).

Researchers are also invited to present OHDSI Network Studies on weekly OHDSI community calls and at OHDSI Symposiums across the globe.

## 21.4 Types of Network Studies

The network studies can be of different types - ranging from simple characterization questions to more advanced predictive studies. A large number of studies conducted today are focused on epidemiology and drug efficacy and safety and thus carry different type of characterization analyses

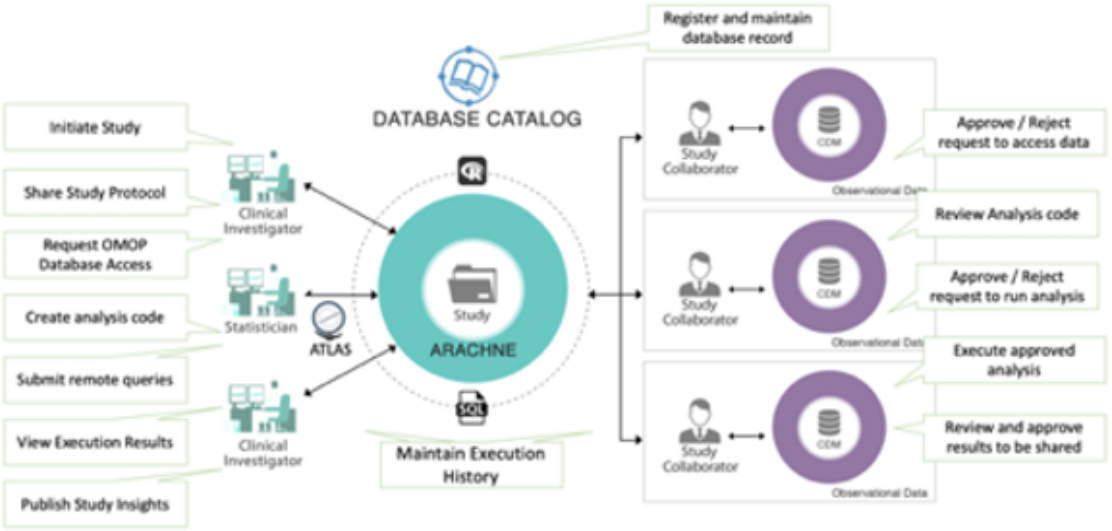



Figure 21.1: The ARACHNE Network Study Process.

such as patient population characterization, incidence rates of certain outcomes/conditions, comparative drug effectiveness comparison, prevalence of disease and similar. However, more and more studies cary predictive nature including the probability of an outcome for a certain type of a patient (personalized medicine).

## 21.5 Forward Looking: Using Network Study Automation

The current network study process is manual and rudimentary - with study team members using various mechanisms (including Wiki, GitHub and email) to collaborate on study design, share code and results. This process is not consistent and scalable and to solve that issue, the OHDSI community is actively working to systemize study processes. The ARACHNE Research Network platform is a community-driven solution to streamline and automate the process of network studies.

The ARACHNE Platform includes multiple core components:

- The **ARACHNE Data Catalog** where different network participants register and maintain information about data sets available for network research
- The **Study Workflow Manager** that allows study teams to orchestrate and end to end network study process. The ARACHNE Research Network platform It is taking full advantage of OHDSI standards and establishes a consistent, transparent, secure and compliant observational research process, across multiple organizations. ARACHNE standardizes the communication protocol to access the data and exchange analysis results, while enabling authentication and authorization for restricted content. It brings participating organizations - data providers, investigators, sponsors and data scientists - into a single collaborative study team and facilitates an end-to-end observational study. The tool enables the creation of a complete, standards-

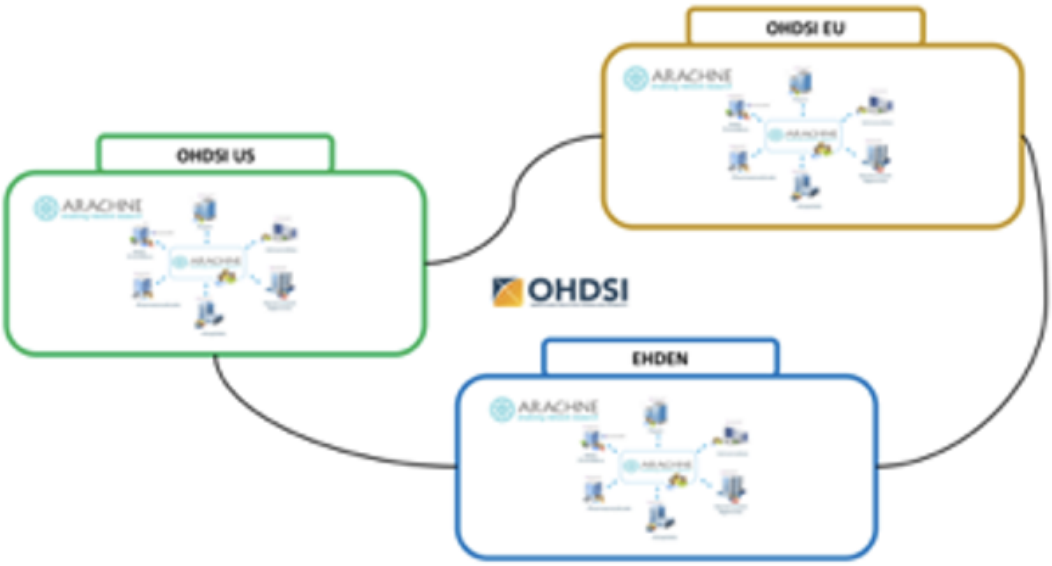



Figure 21.2: The ARACHNE Network of Networks.

based R, Python and SQL execution environment including approval workflows controlled by the data custodian.

ARACHNE is built to provide a seamless integration with other OHDSI tools, including ACHILLES reports and an ability to import ATLAS design artefacts, create self-contained packages and automatically execute those across multiple sites. The future vision is to eventually enable multiple networks to be linked together for the purpose of conducting research not only between organizations within a single network, but also between organizations across multiple networks.

## 21.6 Best Practices for Network Research

There are multiple best practices that study teams should be following while executing the network study:

- Make sure that study questions can be supported by data available. Perform study feasibility to identify the best databases.
- Write code in generic way and parametrize all functions and variables e.g. do not hard database connection, local hard drive path, assume a certain operating system.
- Ensure the target databases have required OMOP CDM version and OMOP Standardized Vocabularies.
- Ensure the target database ETL has followed THEMIS business rules and conventions and correct data was placed into correct CDM tables and fields.
- Do not tweak the study code to get desired results

## 21.7 Example: LEGEND - Hypertension

To be added.

# Appendix A

# Glossary

Cohort A cohort is a list of person_ids with start and end date. It is stored in a study specific cohort table or a CDM specified cohort table can also be used. Cohort can be represented as .json file. It is used for import and export but not during an analysis. OHDSI tools use SQL so Atlas also generates a .sql file that creates the cohort during analysis.

Parametized SQL code An SQL code that allows for use of parameters. Parameters are prefixed with @. Such code has to be “rendered”. Synonym: OHDSI SQL code.

# Appendix B

# Cohort definitions

This Appendix contains cohort definitions used throughout the book.

## B.1 ACE inhibitors

**Initial Event Cohort**

People having any of the following:

- a drug exposure of *ACE inhibitors* (Table B.1) for the first time in the person's history

with continuous observation of at least 365 days prior and 0 days after event index date, and limit initial events to: all events per person.

Limit qualifying cohort to: all events per person.

**End Date Strategy**

Custom Drug Era Exit Criteria This strategy creates a drug era from the codes found in the specified concept set. If the index event is found within an era, the cohort end date will use the era's end date. Otherwise, it will use the observation period end date that contains the index event.

Use the era end date of *ACE inhibitors* (Table B.1)

- allowing 30 days between exposures
- adding 0 days after exposure end

**Cohort Collapse Strategy**

Collapse cohort by era with a gap size of 30 days.

**Concept Set Definitions**

Table B.1: ACE inhibitors

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 1308216 | Lisinopril | NO | YES | NO |
| 1310756 | moexipril | NO | YES | NO |
| 1331235 | quinapril | NO | YES | NO |
| 1334456 | Ramipril | NO | YES | NO |
| 1335471 | benazepril | NO | YES | NO |
| 1340128 | Captopril | NO | YES | NO |
| 1341927 | Enalapril | NO | YES | NO |
| 1342439 | trandolapril | NO | YES | NO |
| 1363749 | Fosinopril | NO | YES | NO |
| 1373225 | Perindopril | NO | YES | NO |

## B.2 New users of ACE inhibitors as first-line monotherapy for hypertension

**Initial Event Cohort**

People having any of the following:

- a drug exposure of *ACE inhibitors* (Table B.2) for the first time in the person's history

with continuous observation of at least 365 days prior and 0 days after event index date, and limit initial events to: earliest event per person.

**Inclusion Rules**

Inclusion Criteria #1: has hypertension diagnosis in 1 yr prior to treatment

Having all of the following criteria:

- at least 1 occurrences of a condition occurrence of *Hypertensive disorder* (Table B.3) where event starts between 365 days Before and 0 days After index start date

Inclusion Criteria #2: Has no prior antihypertensive drug exposures in medical history

Having all of the following criteria:

- exactly 0 occurrences of a drug exposure of *Hypertension drugs* (Table B.4) where event starts between all days Before and 1 days Before index start date

Inclusion Criteria #3: Is only taking ACE as monotherapy, with no concomitant combination treatments

Having all of the following criteria:

- exactly 1 distinct occurrences of a drug era of *Hypertension drugs* (Table B.4) where event starts between 0 days Before and 7 days After index start date

Limit qualifying cohort to: earliest event per person.

**End Date Strategy**

Custom Drug Era Exit Criteria. This strategy creates a drug era from the codes found in the specified concept set. If the index event is found within an era, the cohort end date will use the era's end date. Otherwise, it will use the observation period end date that contains the index event.

Use the era end date of *ACE inhibitors* (Table B.2)

- allowing 30 days between exposures
- adding 0 days after exposure end

**Cohort Collapse Strategy**

Collapse cohort by era with a gap size of 0 days.

**Concept Set Definitions**

Table B.2: ACE inhibitors

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 1308216 | Lisinopril | NO | YES | NO |
| 1310756 | moexipril | NO | YES | NO |
| 1331235 | quinapril | NO | YES | NO |
| 1334456 | Ramipril | NO | YES | NO |
| 1335471 | benazepril | NO | YES | NO |
| 1340128 | Captopril | NO | YES | NO |
| 1341927 | Enalapril | NO | YES | NO |
| 1342439 | trandolapril | NO | YES | NO |
| 1363749 | Fosinopril | NO | YES | NO |
| 1373225 | Perindopril | NO | YES | NO |

Table B.3: Hypertensive disorder

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 316866 | Hypertensive disorder | NO | YES | NO |

Table B.4: Hypertension drugs

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 904542 | Triamterene | NO | YES | NO |
| 907013 | Metolazone | NO | YES | NO |

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 932745 | Bumetanide | NO | YES | NO |
| 942350 | torsemide | NO | YES | NO |
| 956874 | Furosemide | NO | YES | NO |
| 970250 | Spironolactone | NO | YES | NO |
| 974166 | Hydrochlorothiazide | NO | YES | NO |
| 978555 | Indapamide | NO | YES | NO |
| 991382 | Amiloride | NO | YES | NO |
| 1305447 | Methyldopa | NO | YES | NO |
| 1307046 | Metoprolol | NO | YES | NO |
| 1307863 | Verapamil | NO | YES | NO |
| 1308216 | Lisinopril | NO | YES | NO |
| 1308842 | valsartan | NO | YES | NO |
| 1309068 | Minoxidil | NO | YES | NO |
| 1309799 | eplerenone | NO | YES | NO |
| 1310756 | moexipril | NO | YES | NO |
| 1313200 | Nadolol | NO | YES | NO |
| 1314002 | Atenolol | NO | YES | NO |
| 1314577 | nebivolol | NO | YES | NO |
| 1317640 | telmisartan | NO | YES | NO |
| 1317967 | aliskiren | NO | YES | NO |
| 1318137 | Nicardipine | NO | YES | NO |
| 1318853 | Nifedipine | NO | YES | NO |
| 1319880 | Nisoldipine | NO | YES | NO |
| 1319998 | Acebutolol | NO | YES | NO |
| 1322081 | Betaxolol | NO | YES | NO |
| 1326012 | Isradipine | NO | YES | NO |
| 1327978 | Penbutolol | NO | YES | NO |
| 1328165 | Diltiazem | NO | YES | NO |
| 1331235 | quinapril | NO | YES | NO |
| 1332418 | Amlodipine | NO | YES | NO |
| 1334456 | Ramipril | NO | YES | NO |
| 1335471 | benazepril | NO | YES | NO |
| 1338005 | Bisoprolol | NO | YES | NO |
| 1340128 | Captopril | NO | YES | NO |
| 1341238 | Terazosin | NO | YES | NO |
| 1341927 | Enalapril | NO | YES | NO |
| 1342439 | trandolapril | NO | YES | NO |
| 1344965 | Guanfacine | NO | YES | NO |
| 1345858 | Pindolol | NO | YES | NO |
| 1346686 | eprosartan | NO | YES | NO |
| 1346823 | carvedilol | NO | YES | NO |
| 1347384 | irbesartan | NO | YES | NO |
| 1350489 | Prazosin | NO | YES | NO |

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 1351557 | candesartan | NO | YES | NO |
| 1353766 | Propranolol | NO | YES | NO |
| 1353776 | Felodipine | NO | YES | NO |
| 1363053 | Doxazosin | NO | YES | NO |
| 1363749 | Fosinopril | NO | YES | NO |
| 1367500 | Losartan | NO | YES | NO |
| 1373225 | Perindopril | NO | YES | NO |
| 1373928 | Hydralazine | NO | YES | NO |
| 1386957 | Labetalol | NO | YES | NO |
| 1395058 | Chlorthalidone | NO | YES | NO |
| 1398937 | Clonidine | NO | YES | NO |
| 40226742 | olmesartan | NO | YES | NO |
| 40235485 | azilsartan | NO | YES | NO |

## B.3 Acute myocardial infarction (AMI)

**Initial Event Cohort**

People having any of the following:

- a condition occurrence of *Acute myocardial Infarction* (Table B.5)

with continuous observation of at least 0 days prior and 0 days after event index date, and limit initial events to: all events per person.

For people matching the Primary Events, include: Having any of the following criteria:

- at least 1 occurrences of a visit occurrence of *Inpatient or ER visit* (Table B.6) where event starts between all days Before and 0 days After index start date and event ends between 0 days Before and all days After index start date

Limit cohort of initial events to: all events per person.

Limit qualifying cohort to: all events per person.

**End Date Strategy**

Date Offset Exit Criteria. This cohort defintion end date will be the index event's start date plus 7 days

**Cohort Collapse Strategy**

Collapse cohort by era with a gap size of 180 days.

**Concept Set Definitions**

Table B.5: Inpatient or ER visit

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 314666 | Old myocardial infarction | YES | YES | NO |
| 4329847 | Myocardial infarction | NO | YES | NO |

Table B.6: Inpatient or ER visit

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 262 | Emergency Room and Inpatient Visit | NO | YES | NO |
| 9201 | Inpatient Visit | NO | YES | NO |
| 9203 | Emergency Room Visit | NO | YES | NO |

## B.4 Angioedema

**Initial Event Cohort**

People having any of the following:

- a condition occurrence of *Angioedema* (Table B.7)

with continuous observation of at least 0 days prior and 0 days after event index date, and limit initial events to: all events per person.

For people matching the Primary Events, include: Having any of the following criteria:

- at least 1 occurrences of a visit occurrence of *Inpatient or ER visit* (Table B.8) where event starts between all days Before and 0 days After index start date and event ends between 0 days Before and all days After index start date

Limit cohort of initial events to: all events per person.

Limit qualifying cohort to: all events per person.

**End Date Strategy**

This cohort defintion end date will be the index event's start date plus 7 days

**Cohort Collapse Strategy**

Collapse cohort by era with a gap size of 30 days.

**Concept Set Definitions**

Table B.7: Angioedema

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 432791 | Angioedema | NO | YES | NO |

Table B.8: Inpatient or ER visit

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 262 | Emergency Room and Inpatient Visit | NO | YES | NO |
| 9201 | Inpatient Visit | NO | YES | NO |
| 9203 | Emergency Room Visit | NO | YES | NO |

## B.5 New users of Thiazide-like diuretics as first-line monotherapy for hypertension

**Initial Event Cohort**

People having any of the following:

- a drug exposure of *Thiazide or thiazide-like diuretic* (Table B.9) for the first time in the person's history

with continuous observation of at least 365 days prior and 0 days after event index date, and limit initial events to: earliest event per person.

**Inclusion Rules**

Inclusion Criteria #1: has hypertension diagnosis in 1 yr prior to treatment

Having all of the following criteria:

- at least 1 occurrences of a condition occurrence of *Hypertensive disorder* (Table B.10) where event starts between 365 days Before and 0 days After index start date

Inclusion Criteria #2: Has no prior antihypertensive drug exposures in medical history

Having all of the following criteria:

- exactly 0 occurrences of a drug exposure of *Hypertension drugs* (Table B.11) where event starts between all days Before and 1 days Before index start date

Inclusion Criteria #3: Is only taking ACE as monotherapy, with no concomitant combination treatments

Having all of the following criteria:

- exactly 1 distinct occurrences of a drug era of *Hypertension drugs* (Table B.11) where event starts between 0 days Before and 7 days After index start date

Limit qualifying cohort to: earliest event per person.

**End Date Strategy**

Custom Drug Era Exit Criteria. This strategy creates a drug era from the codes found in the specified concept set. If the index event is found within an era, the cohort end date will use the era's end date. Otherwise, it will use the observation period end date that contains the index event.

Use the era end date of *Thiazide or thiazide-like diuretic* (Table B.9)

- allowing 30 days between exposures
- adding 0 days after exposure end

**Cohort Collapse Strategy**

Collapse cohort by era with a gap size of 0 days.

**Concept Set Definitions**

Table B.9: Thiazide or thiazide-like diuretic

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 907013 | Metolazone | NO | YES | NO |
| 974166 | Hydrochlorothiazide | NO | YES | NO |
| 978555 | Indapamide | NO | YES | NO |
| 1395058 | Chlorthalidone | NO | YES | NO |

Table B.10: Hypertensive disorder

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 316866 | Hypertensive disorder | NO | YES | NO |

Table B.11: Hypertension drugs

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 904542 | Triamterene | NO | YES | NO |
| 907013 | Metolazone | NO | YES | NO |
| 932745 | Bumetanide | NO | YES | NO |
| 942350 | torsemide | NO | YES | NO |
| 956874 | Furosemide | NO | YES | NO |
| 970250 | Spironolactone | NO | YES | NO |
| 974166 | Hydrochlorothiazide | NO | YES | NO |
| 978555 | Indapamide | NO | YES | NO |

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 991382 | Amiloride | NO | YES | NO |
| 1305447 | Methyldopa | NO | YES | NO |
| 1307046 | Metoprolol | NO | YES | NO |
| 1307863 | Verapamil | NO | YES | NO |
| 1308216 | Lisinopril | NO | YES | NO |
| 1308842 | valsartan | NO | YES | NO |
| 1309068 | Minoxidil | NO | YES | NO |
| 1309799 | eplerenone | NO | YES | NO |
| 1310756 | moexipril | NO | YES | NO |
| 1313200 | Nadolol | NO | YES | NO |
| 1314002 | Atenolol | NO | YES | NO |
| 1314577 | nebivolol | NO | YES | NO |
| 1317640 | telmisartan | NO | YES | NO |
| 1317967 | aliskiren | NO | YES | NO |
| 1318137 | Nicardipine | NO | YES | NO |
| 1318853 | Nifedipine | NO | YES | NO |
| 1319880 | Nisoldipine | NO | YES | NO |
| 1319998 | Acebutolol | NO | YES | NO |
| 1322081 | Betaxolol | NO | YES | NO |
| 1326012 | Isradipine | NO | YES | NO |
| 1327978 | Penbutolol | NO | YES | NO |
| 1328165 | Diltiazem | NO | YES | NO |
| 1331235 | quinapril | NO | YES | NO |
| 1332418 | Amlodipine | NO | YES | NO |
| 1334456 | Ramipril | NO | YES | NO |
| 1335471 | benazepril | NO | YES | NO |
| 1338005 | Bisoprolol | NO | YES | NO |
| 1340128 | Captopril | NO | YES | NO |
| 1341238 | Terazosin | NO | YES | NO |
| 1341927 | Enalapril | NO | YES | NO |
| 1342439 | trandolapril | NO | YES | NO |
| 1344965 | Guanfacine | NO | YES | NO |
| 1345858 | Pindolol | NO | YES | NO |
| 1346686 | eprosartan | NO | YES | NO |
| 1346823 | carvedilol | NO | YES | NO |
| 1347384 | irbesartan | NO | YES | NO |
| 1350489 | Prazosin | NO | YES | NO |
| 1351557 | candesartan | NO | YES | NO |
| 1353766 | Propranolol | NO | YES | NO |
| 1353776 | Felodipine | NO | YES | NO |
| 1363053 | Doxazosin | NO | YES | NO |
| 1363749 | Fosinopril | NO | YES | NO |
| 1367500 | Losartan | NO | YES | NO |

| Concept Id | Concept Name | Excluded | Descendants | Mapped |
|---|---|---|---|---|
| 1373225 | Perindopril | NO | YES | NO |
| 1373928 | Hydralazine | NO | YES | NO |
| 1386957 | Labetalol | NO | YES | NO |
| 1395058 | Chlorthalidone | NO | YES | NO |
| 1398937 | Clonidine | NO | YES | NO |
| 40226742 | olmesartan | NO | YES | NO |
| 40235485 | azilsartan | NO | YES | NO |

# Appendix C

# Negative controls

This Appendix contains negative controls used in various chapters of the book.

## C.1 ACEi and THZ

Table C.1: Negative control outcomes when comparing ACE inhibitors (ACEi) to thiazides and thiazide-like diuretics (THZ).

| Concept ID | Concept Name |
|---|---|
| 434165 | Abnormal cervical smear |
| 436409 | Abnormal pupil |
| 199192 | Abrasion and/or friction burn of trunk without infection |
| 4088290 | Absence of breast |
| 4092879 | Absent kidney |
| 44783954 | Acid reflux |
| 75911 | Acquired hallux valgus |
| 137951 | Acquired keratoderma |
| 77965 | Acquired trigger finger |
| 376707 | Acute conjunctivitis |
| 4103640 | Amputated foot |
| 73241 | Anal and rectal polyp |
| 133655 | Burn of forearm |
| 73560 | Calcaneal spur |
| 434327 | Cannabis abuse |
| 4213540 | Cervical somatic dysfunction |
| 140842 | Changes in skin texture |
| 81378 | Chondromalacia of patella |
| 432303 | Cocaine abuse |
| 4201390 | Colostomy present |

| Concept ID | Concept Name |
|---|---|
| 46269889 | Complication due to Crohn's disease |
| 134438 | Contact dermatitis |
| 78619 | Contusion of knee |
| 201606 | Crohn's disease |
| 76786 | Derangement of knee |
| 4115402 | Difficulty sleeping |
| 45757370 | Disproportion of reconstructed breast |
| 433111 | Effects of hunger |
| 433527 | Endometriosis |
| 4170770 | Epidermoid cyst |
| 4092896 | Feces contents abnormal |
| 259995 | Foreign body in orifice |
| 40481632 | Ganglion cyst |
| 4166231 | Genetic predisposition |
| 433577 | Hammer toe |
| 4231770 | Hereditary thrombophilia |
| 440329 | Herpes zoster without complication |
| 4012570 | High risk sexual behavior |
| 4012934 | Homocystinuria |
| 441788 | Human papilloma virus infection |
| 4201717 | Ileostomy present |
| 374375 | Impacted cerumen |
| 4344500 | Impingement syndrome of shoulder region |
| 139099 | Ingrowing nail |
| 444132 | Injury of knee |
| 196168 | Irregular periods |
| 432593 | Kwashiorkor |
| 434203 | Late effect of contusion |
| 438329 | Late effect of motor vehicle accident |
| 195873 | Leukorrhea |
| 4083487 | Macular drusen |
| 4103703 | Melena |
| 4209423 | Nicotine dependence |
| 377572 | Noise effects on inner ear |
| 40480893 | Nonspecific tuberculin test reaction |
| 136368 | Non-toxic multinodular goiter |
| 140648 | Onychomycosis due to dermatophyte |
| 438130 | Opioid abuse |
| 4091513 | Passing flatus |
| 4202045 | Postviral fatigue syndrome |
| 373478 | Presbyopia |
| 46286594 | Problem related to lifestyle |
| 439790 | Psychalgia |

| Concept ID | Concept Name |
|---|---|
| 81634 | Ptotic breast |
| 380706 | Regular astigmatism |
| 141932 | Senile hyperkeratosis |
| 36713918 | Somatic dysfunction of lumbar region |
| 443172 | Splinter of face, without major open wound |
| 81151 | Sprain of ankle |
| 72748 | Strain of rotator cuff capsule |
| 378427 | Tear film insufficiency |
| 437264 | Tobacco dependence syndrome |
| 194083 | Vaginitis and vulvovaginitis |
| 140641 | Verruca vulgaris |
| 440193 | Wristdrop |
| 4115367 | Wrist joint pain |

# Appendix D

# Suggested Answers

This Appendix contains suggested answers for the exercises in the book.

## D.1 SQL and R

**Exercise 10.1**

To compute the number of people we can simply query the PERSON table:

```
library(DatabaseConnector)
connection <- connect(connectionDetails)
sql <- "SELECT COUNT(*) AS person_count
FROM @cdm.person;"

renderTranslateQuerySql(connection, sql, cdm = "main")
```

```
##   PERSON_COUNT
## 1         2694
```

**Exercise 10.2**

To compute the number of people with at least one prescription of celecoxib, we can query the DRUG_EXPOSURE table. To find all drugs containing the ingredient celecoxib, we join to the CONCEPT_ANCESTOR and CONCEPT tables:

```
library(DatabaseConnector)
connection <- connect(connectionDetails)
sql <- "SELECT COUNT(DISTINCT(person_id)) AS person_count
FROM @cdm.drug_exposure
INNER JOIN @cdm.concept_ancestor
  ON drug_concept_id = descendant_concept_id
```

```
INNER JOIN @cdm.concept ingredient
  ON ancestor_concept_id = ingredient.concept_id
WHERE LOWER(ingredient.concept_name) = 'celecoxib'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S';"

renderTranslateQuerySql(connection, sql, cdm = "main")
```

```
##   PERSON_COUNT
## 1         1844
```

Note that we use `COUNT(DISTINCT(person_id))` to find the number of distinct persons, considering that a person might have more than one prescription. Also note that we use the `LOWER` function to make our search for "celecoxib" case-insensitive.

Alternatively, we can use the DRUG_ERA table, which is already rolled up to the ingredient level:

```
library(DatabaseConnector)
connection <- connect(connectionDetails)

sql <- "SELECT COUNT(DISTINCT(person_id)) AS person_count
FROM @cdm.drug_era
INNER JOIN @cdm.concept ingredient
  ON drug_concept_id = ingredient.concept_id
WHERE LOWER(ingredient.concept_name) = 'celecoxib'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S';"

renderTranslateQuerySql(connection, sql, cdm = "main")
```

```
##   PERSON_COUNT
## 1         1844
```

**Exercise 10.3**

To compute the number of diagnoses during exposure we extend our previous query by joining to the CONDITION_OCCURRENCE table. We join to the CONCEPT_ANCESTOR table to find all condition concepts that imply a gastrointestinal haemorrhage:

```
library(DatabaseConnector)
connection <- connect(connectionDetails)
sql <- "SELECT COUNT(*) AS diagnose_count
FROM @cdm.drug_era
INNER JOIN @cdm.concept ingredient
  ON drug_concept_id = ingredient.concept_id
INNER JOIN @cdm.condition_occurrence
  ON condition_start_date >= drug_era_start_date
```

```
    AND condition_start_date <= drug_era_end_date
INNER JOIN @cdm.concept_ancestor
  ON condition_concept_id =descendant_concept_id
WHERE LOWER(ingredient.concept_name) = 'celecoxib'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S'
  AND ancestor_concept_id = 192671;"

renderTranslateQuerySql(connection, sql, cdm = "main")
```

```
##   DIAGNOSE_COUNT
## 1             41
```

Note that in this case it is essential to use the DRUG_ERA table instead of the DRUG_EXPOSURE table, because drug exposures with the same ingredient can overlap, but drug eras can. This could lead to double counting. For example, imagine a person received two drug drugs containing celecoxib at the same time. This would be recorded as two drug exposures, so any diagnoses occurring during the exposure would be counted twice. The two exposures will be merged into a single non-overlapping drug era.

# Bibliography


Allison, D. B., Brown, A. W., George, B. J., and Kaiser, K. A. (2016). Reproducibility: A tragedy of errors. *Nature*, 530(7588):27–29.

Arnold, B. F., Ercumen, A., Benjamin-Chung, J., and Colford, J. M. (2016). Brief Report: Negative Controls to Detect Selection Bias and Measurement Bias in Epidemiologic Studies. *Epidemiology*, 27(5):637–641.

Austin, P. C. (2011). Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies. *Pharmaceutical statistics*, 10(2):150–161.

Boland, M. R., Parhi, P., Li, L., Miotto, R., Carroll, R., Iqbal, U., Nguyen, P. A., Schuemie, M., You, S. C., Smith, D., Mooney, S., Ryan, P., Li, Y. J., Park, R. W., Denny, J., Dudley, J. T., Hripcsak, G., Gentine, P., and Tatonetti, N. P. (2017). Uncovering exposures responsible for birth season - disease effects: a global study. *J Am Med Inform Assoc*.

Botsis, T., Hartvigsen, G., Chen, F., and Weng, C. (2010). Secondary use of ehr: data quality issues and informatics opportunities. *Summit on Translational Bioinformatics*, 2010:1.

Byrd, J. B., Adam, A., and Brown, N. J. (2006). Angiotensin-converting enzyme inhibitor-associated angioedema. *Immunol Allergy Clin North Am*, 26(4):725–737.

Callahan, T. J., Bauck, A. E., Bertoch, D., Brown, J., Khare, R., Ryan, P. B., Staab, J., Zozus, M. N., and Kahn, M. G. (2017). A comparison of data quality assessment checks in six data sharing networks. *eGEMs*, 5(1).

Cicardi, M., Zingale, L. C., Bergamaschini, L., and Agostoni, A. (2004). Angioedema associated with angiotensin-converting enzyme inhibitor use: outcome after switching to a different treatment. *Arch. Intern. Med.*, 164(8):910–913.

Cutrona, S. L., Toh, S., Iyer, A., Foy, S., Daniel, G. W., Nair, V. P., Ng, D., Butler, M. G., Boudreau, D., Forrow, S., Goldberg, R., Gore, J., McManus, D., Racoosin, J. A., and Gurwitz, J. H. (2013). Validation of acute myocardial infarction in the Food and Drug Administration's Mini-Sentinel program. *Pharmacoepidemiology and Drug Safety*, 22(1):40–54.

DerSimonian, R. and Laird, N. (1986). Meta-analysis in clinical trials. *Control Clin Trials*, 7(3):177–188.

Duke, J. D., Ryan, P. B., Suchard, M. A., Hripcsak, G., Jin, P., Reich, C., Schwalm, M. S., Khoma, Y., Wu, Y., Xu, H., Shah, N. H., Banda, J. M., and Schuemie, M. J. (2017). Risk of angioedema associated with levetiracetam compared with phenytoin: Findings of the observational health data sciences and informatics research network. *Epilepsia*, 58(8):e101–e106.

Farrington, C. P. (1995). Relative incidence estimation from case series for vaccine safety evaluation. *Biometrics*, 51(1):228–235.

Farrington, C. P., Anaya-Izquierdo, K., Whitaker, H. J., Hocine, M. N., Douglas, I., and Smeeth, L. (2011). Self-controlled case series analysis with event-dependent observation periods. *Journal of the American Statistical Association*, 106(494):417–426.

Garza, M., Del Fiol, G., Tenenbaum, J., Walden, A., and Zozus, M. N. (2016). Evaluating common data models for use with a longitudinal community registry. *J Biomed Inform*, 64:333–341.

Hernan, M. A., Hernandez-Diaz, S., Werler, M. M., and Mitchell, A. A. (2002). Causal knowledge as a prerequisite for confounding evaluation: an application to birth defects epidemiology. *Am. J. Epidemiol.*, 155(2):176–184.

Hernan, M. A. and Robins, J. M. (2016). Using Big Data to Emulate a Target Trial When a Randomized Trial Is Not Available. *Am. J. Epidemiol.*, 183(8):758–764.

Hersh, W. R., Weiner, M. G., Embi, P. J., Logan, J. R., Payne, P. R., Bernstam, E. V., Lehmann, H. P., Hripcsak, G., Hartzog, T. H., Cimino, J. J., et al. (2013). Caveats for the use of operational electronic health record data in comparative effectiveness research. *Medical care*, 51(8 0 3):S30.

Higgins, J. P., Thompson, S. G., Deeks, J. J., and Altman, D. G. (2003). Measuring inconsistency in meta-analyses. *BMJ*, 327(7414):557–560.

Hripcsak, G., Duke, J. D., Shah, N. H., Reich, C. G., Huser, V., Schuemie, M. J., Suchard, M. A., Park, R. W., Wong, I. C. K., Rijnbeek, P. R., van der Lei, J., Pratt, N., Norén, G. N., Li, Y.-C., Stang, P. E., Madigan, D., and Ryan, P. B. (2015). Observational Health Data Sciences and Informatics (OHDSI): Opportunities for Observational Researchers. *Studies in health technology and informatics*, 216:574–578.

Huser, V., DeFalco, F. J., Schuemie, M., Ryan, P. B., Shang, N., Velez, M., Park, R. W., Boyce, R. D., Duke, J., Khare, R., Utidjian, L., and Bailey, C. (2016). Multisite Evaluation of a Data Quality Tool for Patient-Level Clinical Data Sets. *EGEMS (Washington, DC)*, 4(1):1239.

Huser, V., Kahn, M. G., Brown, J. S., and Gouripeddi, R. (2018). Methods for examining data quality in healthcare integrated data repositories. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 23:628–633.

Johnston, S. S., Morton, J. M., Kalsekar, I., Ammann, E. M., Hsiao, C. W., and Reps, J. (2019). Using Machine Learning Applied to Real-World Healthcare Data for Predictive Analytics: An Applied Example in Bariatric Surgery. *Value Health*, 22(5):580–586.

Kahn, M. G., Brown, J. S., Chun, A. T., Davidson, B. N., Meeker, D., Ryan, P. B., Schilling, L. M., Weiskopf, N. G., Williams, A. E., and Zozus, M. N. (2015). Transparent reporting of data quality in distributed data networks. *EGEMS (Washington, DC)*, 3(1):1052.

Kahn, M. G., Callahan, T. J., Barnard, J., Bauck, A. E., Brown, J., Davidson, B. N., Estiri, H., Goerg, C., Holve, E., Johnson, S. G., Liaw, S.-T., Hamilton-Lopez, M., Meeker, D., Ong, T. C., Ryan, P. B., Shang, N., Weiskopf, N. G., Weng, C., Zozus, M. N., and Schilling, L. (2016). A Harmonized Data Quality Assessment Terminology and Framework for the Secondary Use of Electronic Health Record Data. *EGEMS (Washington, DC)*, 4(1):1244.

Kahn, M. G., Raebel, M. A., Glanz, J. M., Riedlinger, K., and Steiner, J. F. (2012). A pragmatic framework for single-site and multisite data quality assessment in electronic health record-based clinical research. *Medical care*, 50.

Liaw, S.-T., Rahimi, A., Ray, P., Taggart, J., Dennis, S., de Lusignan, S., Jalaludin, B., Yeo, A., and Talaei-Khoei, A. (2013). Towards an ontology for data quality in integrated chronic disease management: a realist review of the literature. *International journal of medical informatics*, 82(1):10–24.

Lipsitch, M., Tchetgen Tchetgen, E., and Cohen, T. (2010). Negative controls: a tool for detecting confounding and bias in observational studies. *Epidemiology*, 21(3):383–388.

Maclure, M. (1991). The case-crossover design: a method for studying transient effects on the risk of acute events. *Am. J. Epidemiol.*, 133(2):144–153.

Madigan, D., Ryan, P. B., Schuemie, M., Stang, P. E., Overhage, J. M., Hartzema, A. G., Suchard, M. A., DuMouchel, W., and Berlin, J. A. (2013). Evaluating the impact of database heterogeneity on observational study results. *Am. J. Epidemiol.*, 178(4):645–651.

Magid, D. J., Shetterly, S. M., Margolis, K. L., Tavel, H. M., O'Connor, P. J., Selby, J. V., and Ho, P. M. (2010). Comparative effectiveness of angiotensin-converting enzyme inhibitors versus beta-blockers as second-line therapy for hypertension. *Circ Cardiovasc Qual Outcomes*, 3(5):453–458.

Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition.

Moons, K. G. M., Altman, D. G., Reitsma, J. B., Ioannidis, J. P. A., Macaskill, P., Steyerberg, E. W., Vickers, A. J., Ransohoff, D. F., and Collins, G. S. (2015). Transparent Reporting of a multivariable prediction model for Individual Prognosis or Diagnosis (TRIPOD): explanation and elaboration. *Annals of Internal Medicine*, 162(1):W1–73.

Noren, G. N., Caster, O., Juhlin, K., and Lindquist, M. (2014). Zoo or savannah? Choice of training ground for evidence-based pharmacovigilance. *Drug Saf*, 37(9):655–659.

Norman, J. L., Holmes, W. L., Bell, W. A., and Finks, S. W. (2013). Life-threatening ACE inhibitor-induced angioedema after eleven years on lisinopril. *J Pharm Pract*, 26(4):382–388.

Oliveira, J. L., Trifan, A., and Silva, L. A. B. (2019). EMIF catalogue: A collaborative platform for sharing and reusing biomedical data. *International Journal of Medical Informatics*, 126:35–45.

O'Mara, N. B. and O'Mara, E. M. (1996). Delayed onset of angioedema with angiotensin-converting enzyme inhibitors: case report and review of the literature. *Pharmacotherapy*, 16(4):675–679.

Perkins, N. J., Cole, S. R., Harel, O., Tchetgen Tchetgen, E. J., Sun, B., Mitchell, E. M., and Schisterman, E. F. (2017). Principled approaches to missing data in epidemiologic studies. *American journal of epidemiology*, 187(3):568–575.

Powers, B. J., Coeytaux, R. R., Dolor, R. J., Hasselblad, V., Patel, U. D., Yancy, W. S., Gray, R. N., Irvine, R. J., Kendrick, A. S., and Sanders, G. D. (2012). Updated report on comparative effectiveness of ACE inhibitors, ARBs, and direct renin inhibitors for patients with essential hypertension: much more data, little new information. *J Gen Intern Med*, 27(6):716–729.

Prasad, V. and Jena, A. B. (2013). Prespecified falsification end points: can they validate true observational associations? *JAMA*, 309(3):241–242.

Ramcharran, D., Qiu, H., Schuemie, M. J., and Ryan, P. B. (2017). Atypical Antipsychotics and the Risk of Falls and Fractures Among Older Adults: An Emulation Analysis and an Evaluation of Additional Confounding Control Strategies. *J Clin Psychopharmacol*, 37(2):162–168.

Rassen, J. A., Shelat, A. A., Myers, J., Glynn, R. J., Rothman, K. J., and Schneeweiss, S. (2012). One-to-many propensity score matching in cohort studies. *Pharmacoepidemiol Drug Saf*, 21 Suppl 2:69–80.

Reps, J. M., Rijnbeek, P. R., and Ryan, P. B. (2019). Identifying the DEAD: Development and Validation of a Patient-Level Model to Predict Death Status in Population-Level Claims Data. *Drug Saf*.

Reps, J. M., Schuemie, M. J., Suchard, M. A., Ryan, P. B., and Rijnbeek, P. R. (2018). Design and implementation of a standardized framework to generate and evaluate patient-level prediction models using observational healthcare data. *Journal of the American Medical Informatics Association*, 25(8):969–975.

Rosenbaum, P. (2005). *Sensitivity Analysis in Observational Studies*. American Cancer Society.

Rosenbaum, P. and Rubin, D. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70:41–55.

Rubbo, B., Fitzpatrick, N. K., Denaxas, S., Daskalopoulou, M., Yu, N., Patel, R. S., UK Biobank Follow-up and Outcomes Working Group, and Hemingway, H. (2015). Use of electronic health records to ascertain, validate and phenotype acute myocardial infarction: A systematic review and recommendations. *International Journal of Cardiology*, 187:705–711.

Rubin, D. B. (2001). Using propensity scores to help design observational studies: application to the tobacco litigation. *Health Services and Outcomes Research Methodology*, 2(3-4):169–188.

Ryan, P. B., Buse, J. B., Schuemie, M. J., DeFalco, F., Yuan, Z., Stang, P. E., Berlin, J. A., and Rosenthal, N. (2018). Comparative effectiveness of canagliflozin, SGLT2 inhibitors and non-SGLT2 inhibitors on the risk of hospitalization for heart failure and amputation in patients with type 2 diabetes mellitus: A real-world meta-analysis of 4 observational databases (OBSERVE-4D). *Diabetes Obes Metab*, 20(11):2585–2597.

Ryan, P. B., Schuemie, M. J., and Madigan, D. (2013). Empirical performance of a self-controlled cohort method: lessons for developing a risk identification and analysis system. *Drug Saf*, 36 Suppl 1:95–106.

Ryan, P. B., Schuemie, M. J., Ramcharran, D., and Stang, P. E. (2017). Atypical Antipsychotics and the Risks of Acute Kidney Injury and Related Outcomes Among Older Adults: A Replication Analysis and an Evaluation of Adapted Confounding Control Strategies. *Drugs Aging*, 34(3):211–219.

Sabroe, R. A. and Black, A. K. (1997). Angiotensin-converting enzyme (ACE) inhibitors and angio-oedema. *Br. J. Dermatol.*, 136(2):153–158.

Schneeweiss, S. (2018). Automated data-adaptive analytics for electronic healthcare data to study causal treatment effects. *Clin Epidemiol*, 10:771–788.

Schuemie, M. J., Hripcsak, G., Ryan, P. B., Madigan, D., and Suchard, M. A. (2016). Robust empirical calibration of p-values using observational data. *Stat Med*, 35(22):3883–3888.

Schuemie, M. J., Hripcsak, G., Ryan, P. B., Madigan, D., and Suchard, M. A. (2018a). Empirical confidence interval calibration for population-level effect estimation studies in observational healthcare data. *Proc. Natl. Acad. Sci. U.S.A.*, 115(11):2571–2577.

Schuemie, M. J., Ryan, P. B., DuMouchel, W., Suchard, M. A., and Madigan, D. (2014). Interpreting observational studies: why empirical calibration is needed to correct p-values. *Stat Med*, 33(2):209–218.

Schuemie, M. J., Ryan, P. B., Hripcsak, G., Madigan, D., and Suchard, M. A. (2018b). Improving reproducibility by using high-throughput observational studies with empirical calibration. *Philos Trans A Math Phys Eng Sci*, 376(2128).

Sherman, R. E., Anderson, S. A., Dal Pan, G. J., Gray, G. W., Gross, T., Hunter, N. L., LaVange, L., Marinac-Dabic, D., Marks, P. W., Robb, M. A., et al. (2016). Real-world evidence—what is it and what can it tell us. *N Engl J Med*, 375(23):2293–2297.

Simpson, S. E., Madigan, D., Zorych, I., Schuemie, M. J., Ryan, P. B., and Suchard, M. A. (2013). Multiple self-controlled case series for large-scale longitudinal observational databases. *Biometrics*, 69(4):893–902.

Slater, E. E., Merrill, D. D., Guess, H. A., Roylance, P. J., Cooper, W. D., Inman, W. H. W., and Ewan, P. W. (1988). Clinical Profile of Angioedema Associated With Angiotensin Converting-Enzyme Inhibition. *JAMA*, 260(7):967–970.

Suchard, M. A., Simpson, S. E., Zorych, I., Ryan, P. B., and Madigan, D. (2013). Massive parallelization of serial inference algorithms for a complex generalized linear model. *ACM Trans. Model. Comput. Simul.*, 23(1):10:1–10:17.

Suissa, S. (1995). The case-time-control design. *Epidemiology*, 6(3):248–253.

Thompson, T. and Frable, M. A. (1993). Drug-induced, life-threatening angioedema revisited. *Laryngoscope*, 103(1 Pt 1):10–12.

Tian, Y., Schuemie, M. J., and Suchard, M. A. (2018). Evaluating large-scale propensity score performance through real-world and synthetic data experiments. *Int J Epidemiol*, 47(6):2005–2014.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Toh, S., Reichman, M. E., Houstoun, M., Ross Southworth, M., Ding, X., Hernandez, A. F., Levenson, M., Li, L., McCloskey, C., Shoaibi, A., Wu, E., Zornberg, G., and Hennessy, S. (2012). Comparative risk for angioedema associated with the use of drugs that target the renin-angiotensin-aldosterone system. *Arch. Intern. Med.*, 172(20):1582–1589.

Vandenbroucke, J. P. and Pearce, N. (2012). Case-control studies: basic concepts. *Int J Epidemiol*, 41(5):1480–1489.

Vashisht, R., Jung, K., Schuler, A., Banda, J. M., Park, R. W., Jin, S., Li, L., Dudley, J. T., Johnson, K. W., Shervey, M. M., Xu, H., Wu, Y., Natrajan, K., Hripcsak, G., Jin, P., Van Zandt, M., Reckard, A., Reich, C. G., Weaver, J., Schuemie, M. J., Ryan, P. B., Callahan, A., and Shah, N. H. (2018). Association of Hemoglobin A1c Levels With Use of Sulfonylureas, Dipeptidyl Peptidase 4 Inhibitors, and Thiazolidinediones in Patients With Type 2 Diabetes Treated With Metformin: Analysis From the Observational Health Data Sciences and Informatics Initiative. *JAMA Netw Open*, 1(4):e181755.

Voss, E. A., Boyce, R. D., Ryan, P. B., van der Lei, J., Rijnbeek, P. R., and Schuemie, M. J. (2016). Accuracy of an Automated Knowledge Base for Identifying Drug Adverse Reactions. *J Biomed Inform*.

Walker, A. M., Patrick, A. R., Lauer, M. S., Hornbrook, M. C., Marin, M. G., Platt, R., Roger, V. L., Stang, P., and Schneeweiss, S. (2013). A tool for assessing the feasibility of comparative effectiveness research. *Comp Eff Res*, 3:11–20.

Wang, Y., Desai, M., Ryan, P. B., DeFalco, F. J., Schuemie, M. J., Stang, P. E., Berlin, J. A., and Yuan, Z. (2017). Incidence of diabetic ketoacidosis among patients with type 2 diabetes mellitus treated with SGLT2 inhibitors and other antihyperglycemic agents. *Diabetes Res. Clin. Pract.*, 128:83–90.

Weinstein, R. B., Ryan, P., Berlin, J. A., Matcho, A., Schuemie, M., Swerdel, J., Patel, K., and Fife, D. (2017). Channeling in the Use of Nonprescription Paracetamol and Ibuprofen in an Electronic Medical Records Database: Evidence and Implications. *Drug Saf*, 40(12):1279–1292.

Weiskopf, N. G. and Weng, C. (2013a). Methods and dimensions of electronic health record data quality assessment: enabling reuse for clinical research. *Journal of the American Medical Informatics Association*, 20(1):144–151.

Weiskopf, N. G. and Weng, C. (2013b). Methods and dimensions of electronic health record data quality assessment: enabling reuse for clinical research. *Journal of the American Medical Informatics Association: JAMIA*, 20(1):144–151.

Whelton, P. K., Carey, R. M., Aronow, W. S., Casey, D. E., Collins, K. J., Dennison Himmelfarb, C., DePalma, S. M., Gidding, S., Jamerson, K. A., Jones, D. W., MacLaughlin, E. J., Muntner, P., Ovbiagele, B., Smith, S. C., Spencer, C. C., Stafford, R. S., Taler, S. J., Thomas, R. J., Williams, K. A., Williamson, J. D., and Wright, J. T. (2018). 2017 ACC/AHA/AAPA/ABC/ACPM/AGS/APhA/ASH/ASPC/NMA/PCNA Guideline for the Prevention, Detection, Evaluation, and Management of High Blood Pressure in Adults: Executive Summary: A Report of the American College of Cardiology/American Heart Association Task Force on Clinical Practice Guidelines. *Circulation*, 138(17):e426–e483.

Whitaker, H. J., Farrington, C. P., Spiessens, B., and Musonda, P. (2006). Tutorial in biostatistics: the self-controlled case series method. *Stat Med*, 25(10):1768–1797.

Wickham, H. (2015). *R Packages*. O'Reilly Media, Inc., 1st edition.

Wikipedia (2019a). Open science — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Open%20science&oldid=900178688. [Online; accessed 24-June-2019].

Wikipedia (2019b). Science 2.0 — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Science%202.0&oldid=887565958. [Online; accessed 09-July-2019].

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S. A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data*, 3:160018.

Yoon, D., Ahn, E. K., Park, M. Y., Cho, S. Y., Ryan, P., Schuemie, M. J., Shin, D., Park, H., and Park, R. W. (2016). Conversion and Data Quality Assessment of Electronic Health Record Data at a Korean Tertiary Teaching Hospital to a Common Data Model for Distributed Network Research. *Healthc Inform Res*, 22(1):54–58.

Yuan, Z., DeFalco, F. J., Ryan, P. B., Schuemie, M. J., Stang, P. E., Berlin, J. A., Desai, M., and Rosenthal, N. (2018). Risk of lower extremity amputations in people with type 2 diabetes mellitus treated with sodium-glucose co-transporter-2 inhibitors in the USA: A retrospective cohort study. *Diabetes Obes Metab*, 20(3):582–589.

Zaadstra, B. M., Chorus, A. M., van Buuren, S., Kalsbeek, H., and van Noort, J. M. (2008). Selective association of multiple sclerosis with infectious mononucleosis. *Mult. Scler.*, 14(3):307–313.

Zaman, M. A., Oparil, S., and Calhoun, D. A. (2002). Drugs targeting the renin-angiotensin-aldosterone system. *Nat Rev Drug Discov*, 1(8):621–636.

# Index