The Book of OHDSI

Observational Health Data Science and Informatics

2019-07-04

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

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, 2019) 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.

Two important drivers for open science are the increased public scrutiny and call for transparency for scientific funding, and the crisis of the scientific practice itself. Name * explosion of online available data and knowledge * incentive system that is aligned to publish peer-reviewed articles rather

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

[2] https://www.ehden.eu/webinars/

than seek new insights –> p-value hacking etc. * Reference American Science Councils Consensus Report?

3.2 FAIR Guiding Principles

- Reference book by Barend Mons Mons (2018)?

Options for structuring the expose:

- Open Source / Open Standards / Open Data
- Lifecycle (Design and Planning of Experiment / Data Capture / Data Processing & Integration / Data Analysis & Intepretation / Information and Insight Publishing
- Findable / Accessible / Interoperable / Reusable

Chapter 4

Where to begin

This chapter will discuss where to begin if one is new in OHDSI. For various activities, we can describe how one might get started.

For example, if interested in doing a network study, these are the steps. Same for interests in methods research, grant writing, etc.

Add a diagram that shows what tools are used for which steps?

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).

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:

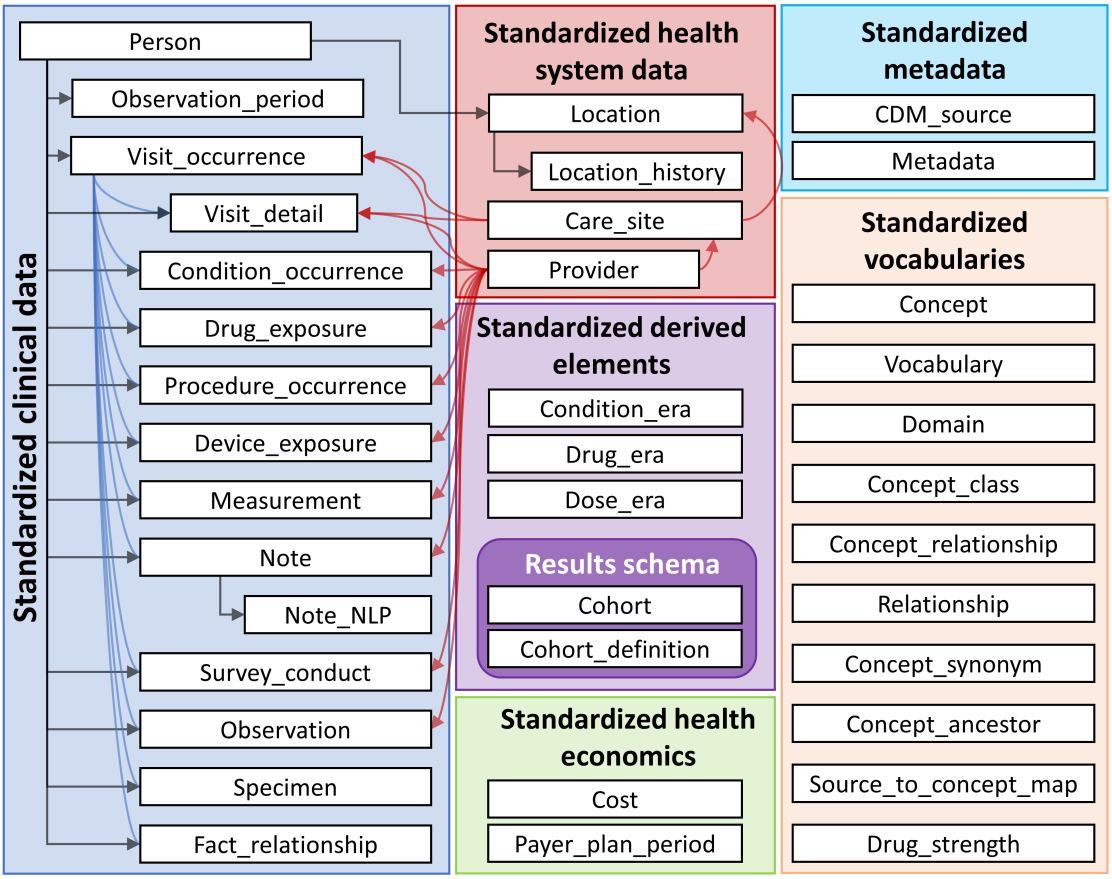


Figure 5.1: Overview of all tables in the CDM version 6.0. Note that not all relationships between tables are shown.

- **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.
- **Rationale for domains**: Domains are identified and separately defined in an entity-relationship model if they have an analysis use case and the domain has specific attributes that are not otherwise applicable. All other data can be preserved as an observation 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 CDMv5, 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, **Todo: add a sentence explaining that these tables describe groups of interest that the user might define, put in links to the later sections** though likely more will be added over the course of v6.0 point releases. 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.

5.2.4 General conventions of fields

Variable names across all tables follow one convention:

Table 5.1: 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.

Notation	Description
[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"

5.2.5 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.

5.2.6 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 this relationship is depicted in the tables is shown in Table 5.6.

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.

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

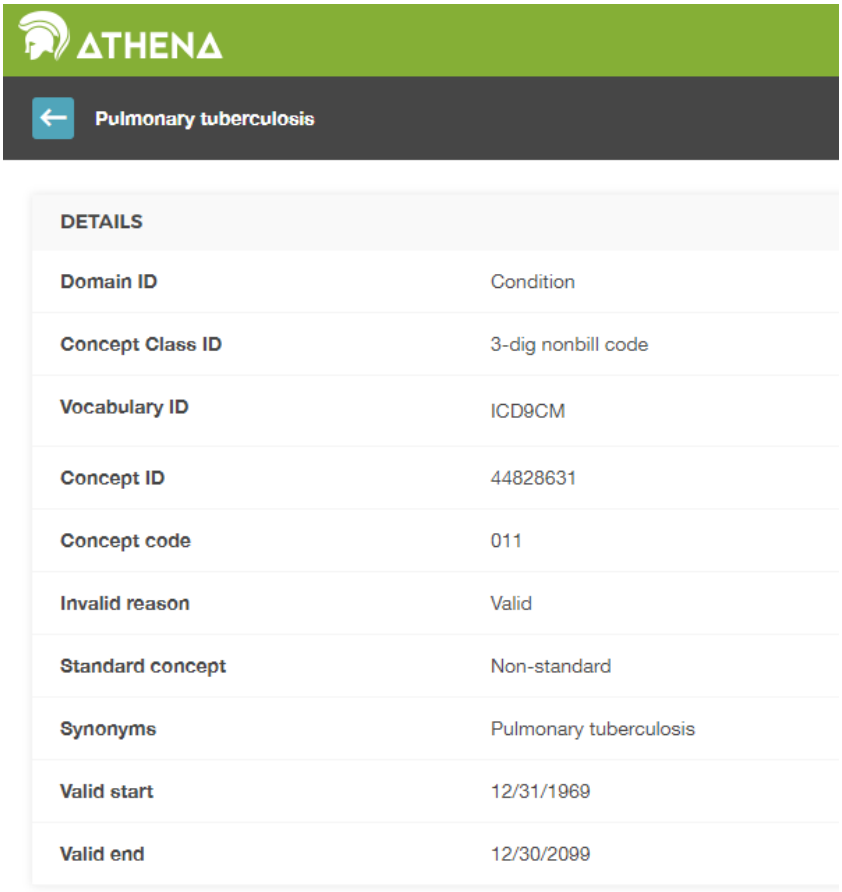


Figure 5.2: ICD9CM code for Pulmonary Tuberculosis

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

5.3.1 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.

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 5.2: 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.
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.

Column name	Value	Explanation
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 5.3 is most similar to EHR data in that we only have records of her encounters from which to determine her observation period.

Table 5.3: Lauren's healthcare encounters.

Encounter ID	Start date	Stop date	Type
70	2010-01-06	2010-01-06	outpatient
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 5.4: 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.

Column name	Value	Explanation
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 5.3 as a record in the VISIT_OCCURRENCE table.

Table 5.5: 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.

Column name	Value	Explanation
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.
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”).

Column name	Value	Explanation
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 5.6 shows how that would be represented in the CONDITION_OCCURRENCE table:

Table 5.6: 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.

Column name	Value	Explanation
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.
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.

Column name	Value	Explanation
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 5.7: 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.

Column name	Value	Explanation
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.
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.

Column name	Value	Explanation
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 5.8: 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.
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.

Column name	Value	Explanation
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 Summary

- TODO: add

5.5 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

Leads: Mui van Zandt & Clair Blacketer

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-

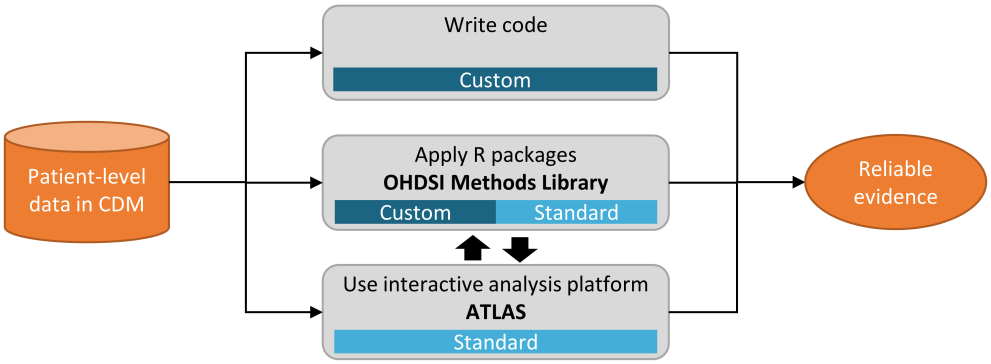


Figure 9.1: Different ways to implement an analysis against data in the CDM.

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.

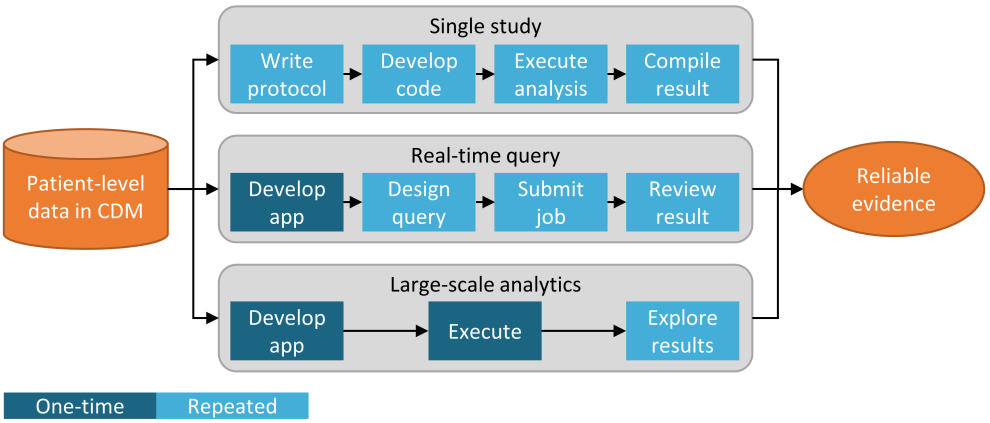


Figure 9.2: Strategies for generating evidence for (clinical) questions.

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

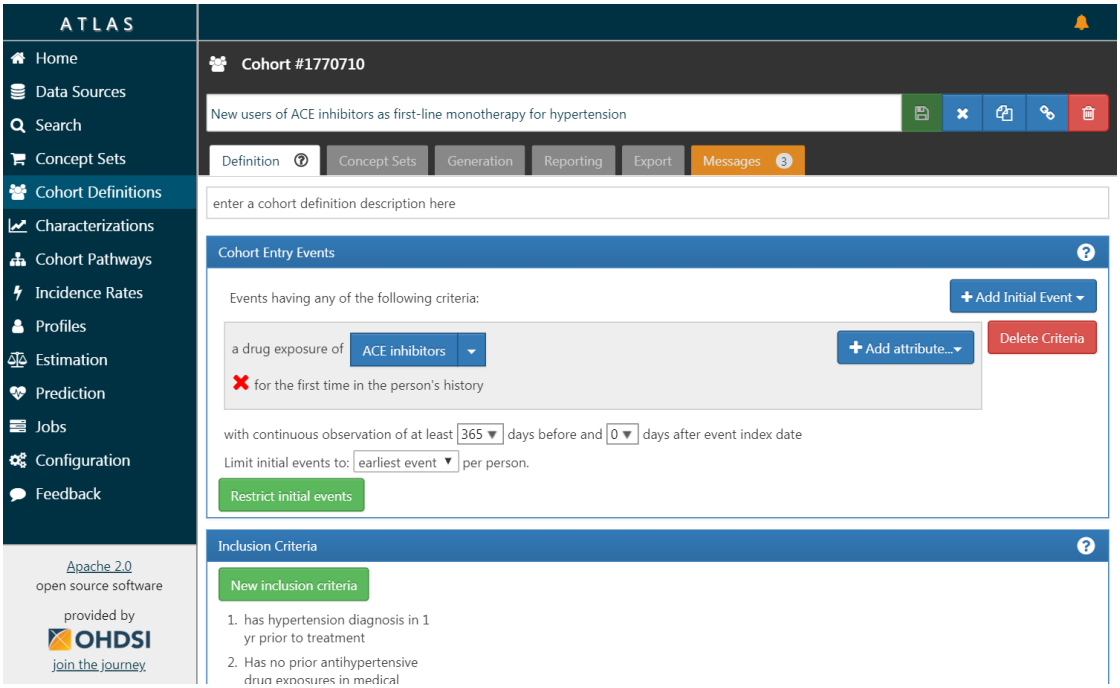


Figure 9.3: ATLAS user interface.

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

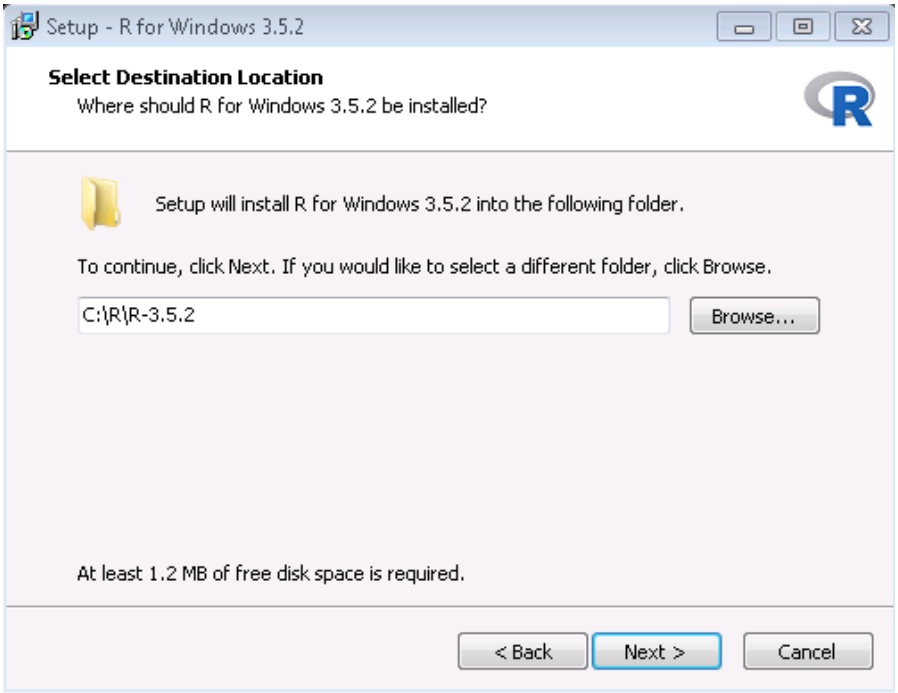


Figure 9.6: Settings the destination folder for R.

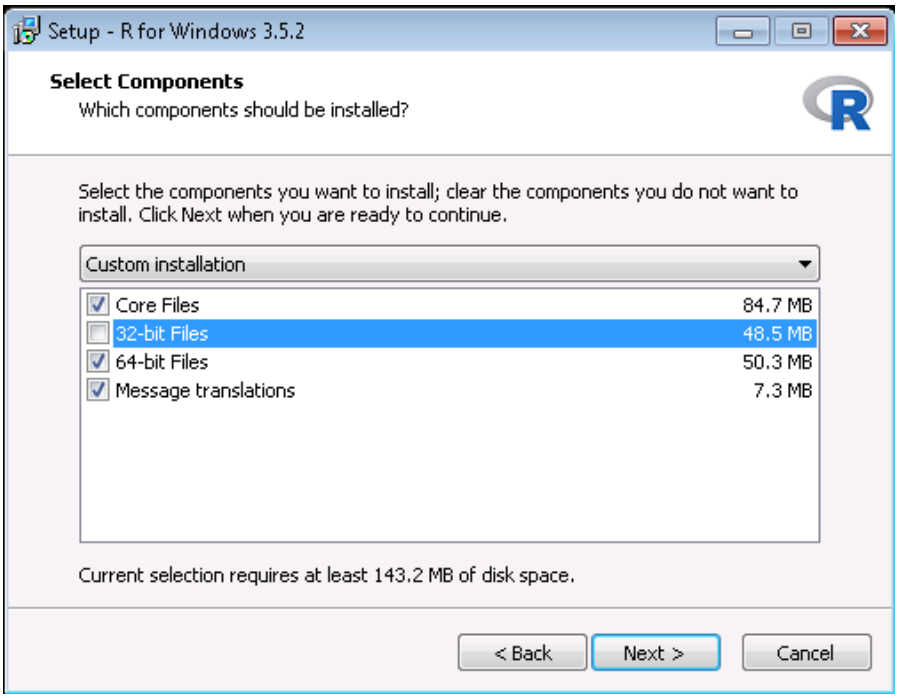


Figure 9.7: Disabling the 32-bit version of R.

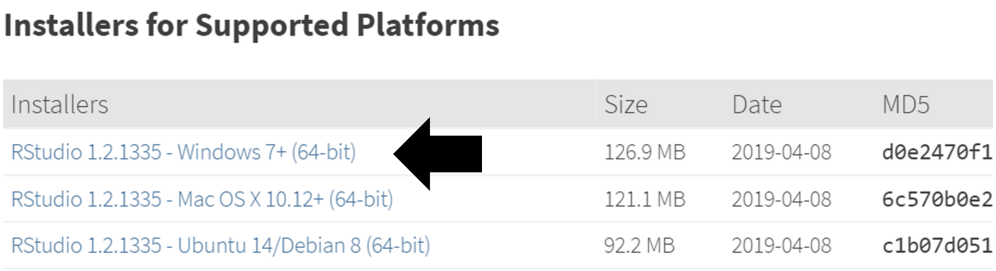


Figure 9.8: Downloading RStudio.

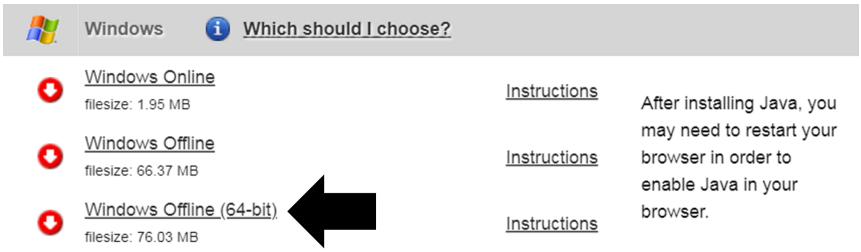


Figure 9.9: Downloading Java.

```
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).

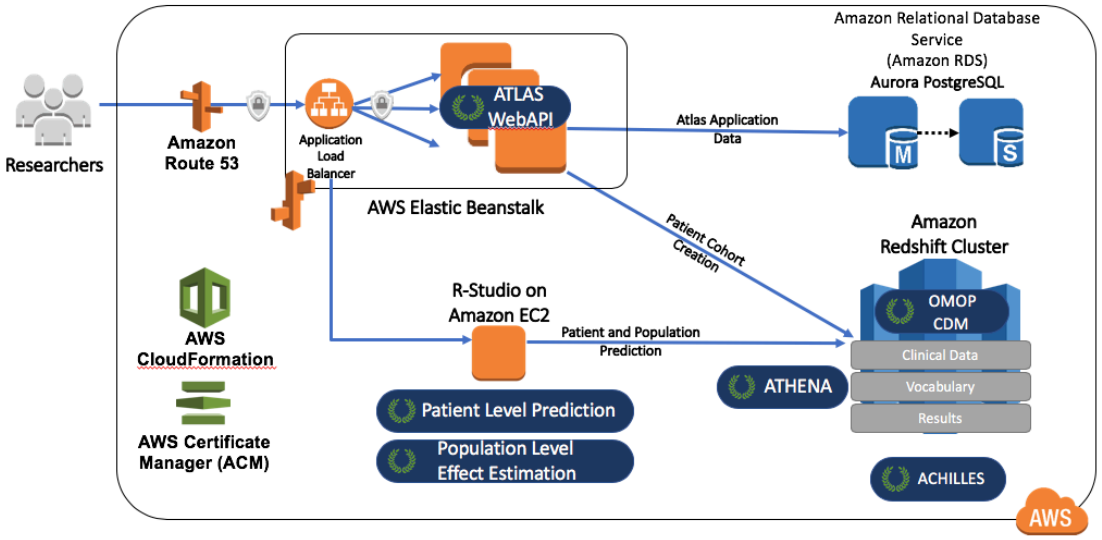


Figure 9.10: The Amazon Web Services architecure for OHDSI-in-a-Box and OHDSIonAWS.

9.6.1 Broadsea

BroadSea uses Docker container technology[5]. 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 …

TODO: Find out if Broadsea is still maintained

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[6] and OHDSIonAWS[7]. OHDSI-in-a-Box is specifically created as a learning environment, and is used in most of the tutorials provided by the OHDSI community. OHDSIonAWS is intended as a starting point to create an enterprise class, multi-user, scalable and fault tolerant environment that can be used by organizations to perform their data analytics. Both follow the architecture depicted in Figure 9.10.

On the back end there is data in the CDM. By default several simulated datasets are provided alongside the Standardized Vocabularies, although of course for OHDSIonAWS it is likely organizations will want to include real healthcare data taht they have access to as well. The data is placed in the Amazon's RedShift database platform, which is supported by the OHDSI tools. Intermediary results

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

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.

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.cowxc37kchildren ;"
```

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/

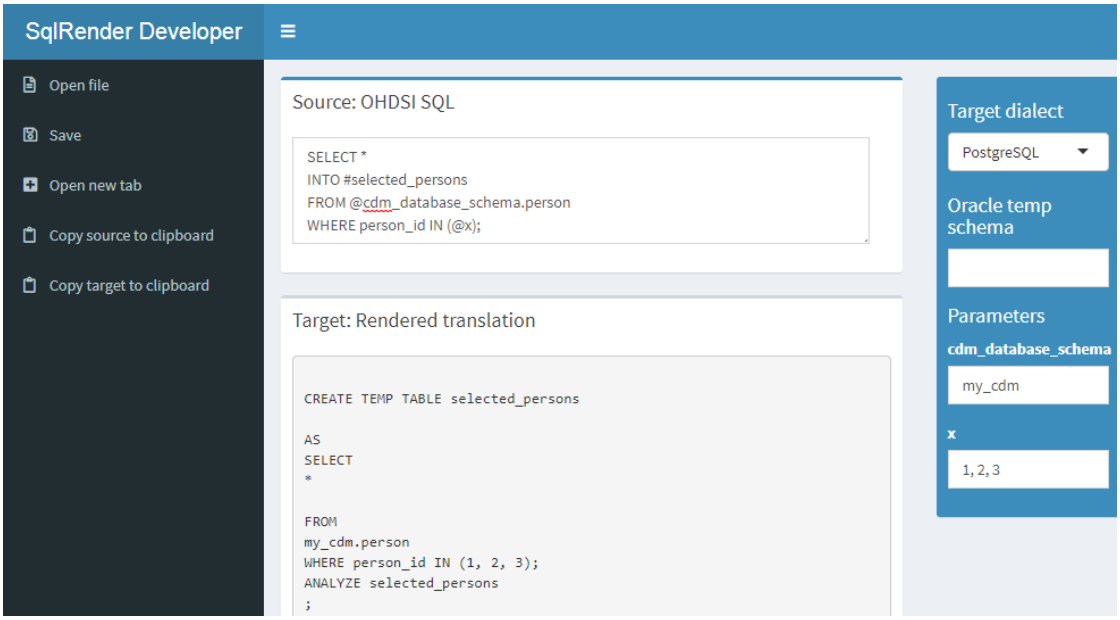


Figure 10.1: The SqlDeveloper Shiny app.

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 ingredient.concept_name = 'Ibuprofen'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S';
```

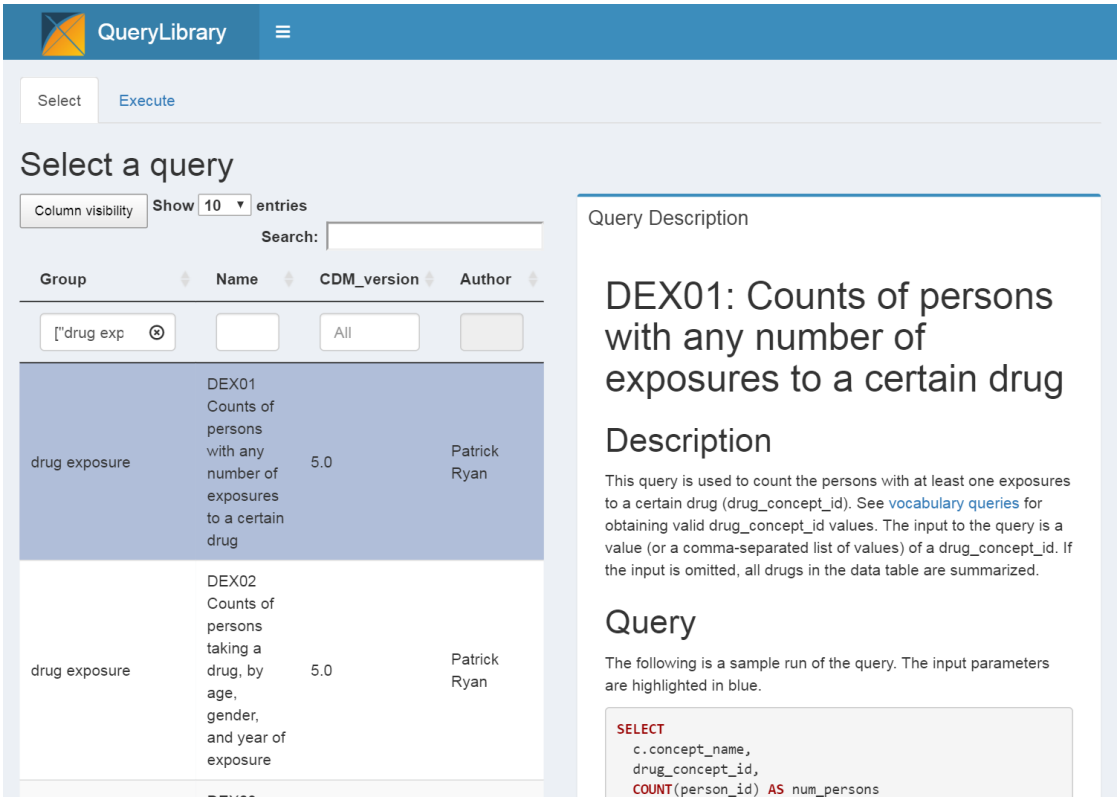


Figure 10.2: QueryLibrary: a library of SQL queries against the CDM.

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.

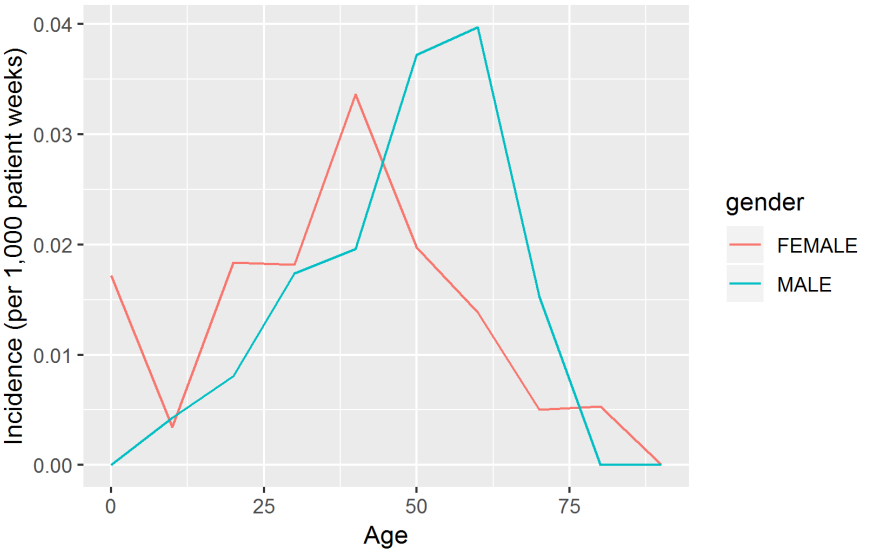
We use `snakeCaseToCamelCase = TRUE` because in SQL we tend to use snake_case for field names (because SQL in case-insensitive), whereas in R we tend to use camelCase (because R is case-sensitive). The `results` data frame column names will now be in camelCase.

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

Chapter 11

Building the building blocks: cohorts

Introduction: a cohort is a group of people that meet a set of criteria for a particular span of time etc. Cohorts are used throughout OHDSIs analytical tools as the primary building blocks.

Using ATLAS: use material from Patrick's tutorial on cohort building

Using SQL: For advanced users, explain how cohorts can be created programmatically.

Probabilistic cohorts: Aphrodite?

Case study: some example cohort definitions

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).

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.

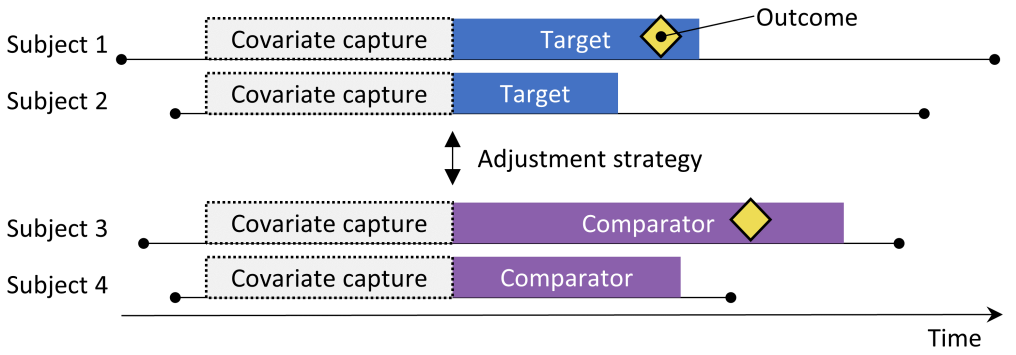


Figure 13.1: The new-user cohort design. Subjects observed to initiate the target treatment are compared to those initiating the comparator treatment. To adjust for differences between the two treatment groups several adjustment strategies can be used, such as stratification, matching, or weighting by 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

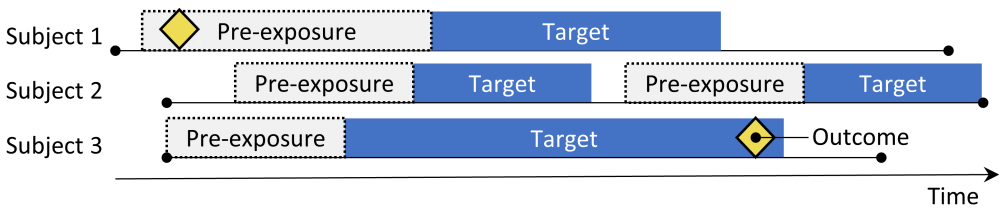


Figure 13.2: The self-controlled cohort design. The rate of outcomes during exposure to the target is compared to the rate of outcomes in the time pre-exposure.

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

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

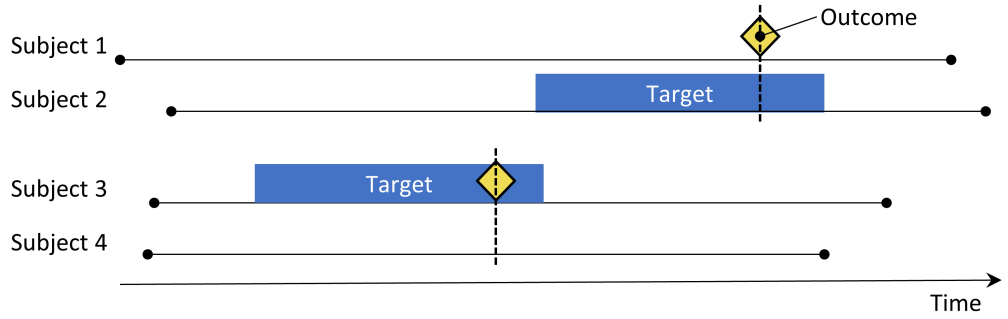


Figure 13.3: The case-control design. Subjects with the outcome ('cases') are compared to subjects without the outcome ('controls') in terms of their exposure status. Often, cases and controls are matched on various characteristics such as age and sex.

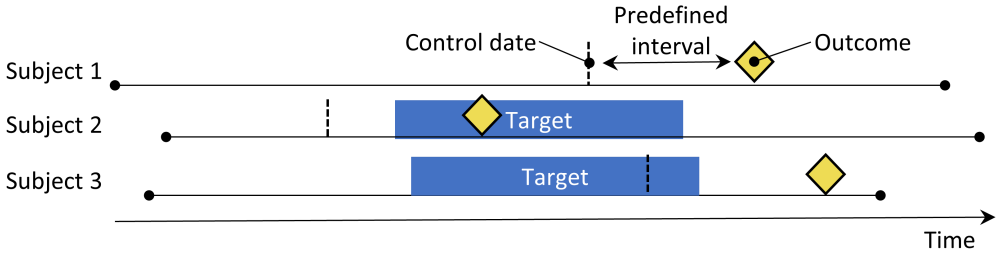


Figure 13.4: The case-crossover design. The time around the outcome is compared to a control date set at a predefined interval prior to the outcome date.

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

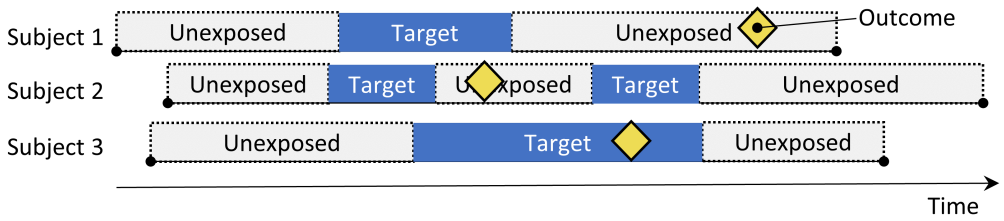


Figure 13.5: The Self-Controlled Case Series design. The rate of outcomes during exposure is compared to the rate of outcomes when not exposed.

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

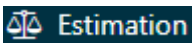
Table 13.6: Main design choices four our comparative cohort study.

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.

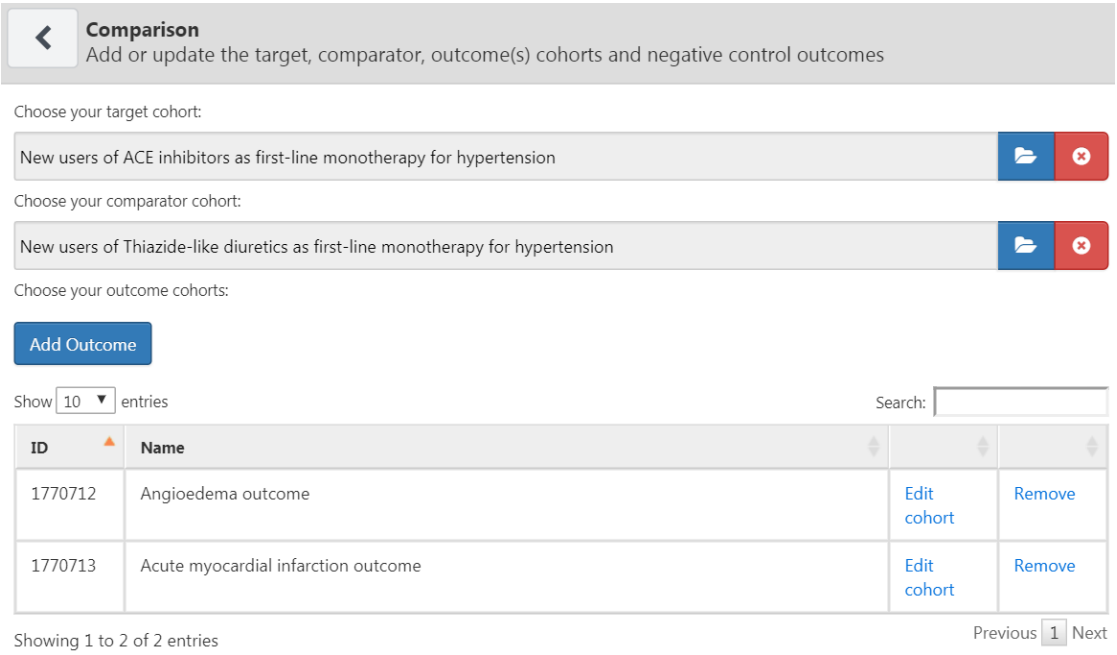


Figure 13.6: The comparison dialog

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).

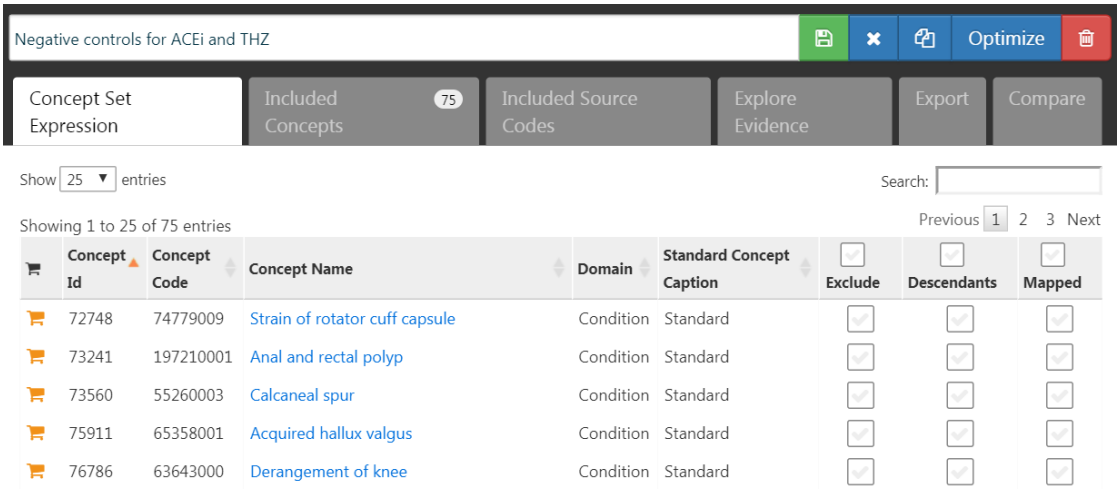


Figure 13.7: Negative Control concept set.

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

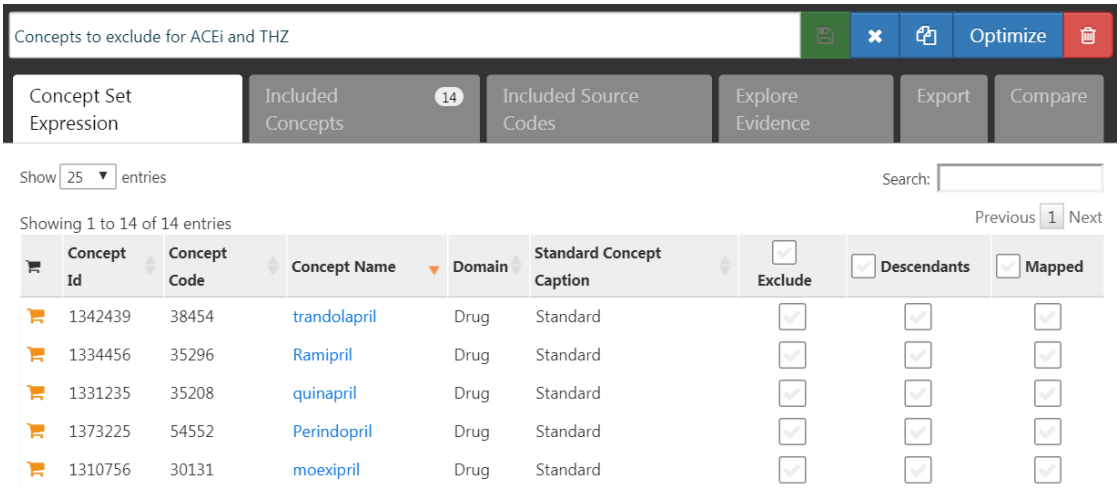


Figure 13.8: The concept set defining the concepts to exclude.

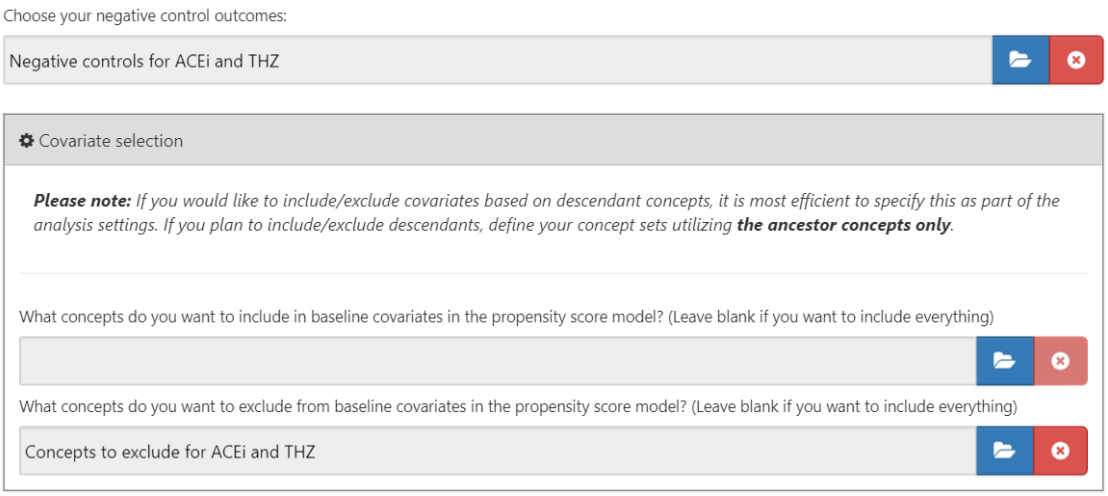


Figure 13.9: The comparison window showing concept sets for negative controls and concepts to exclude.

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**.

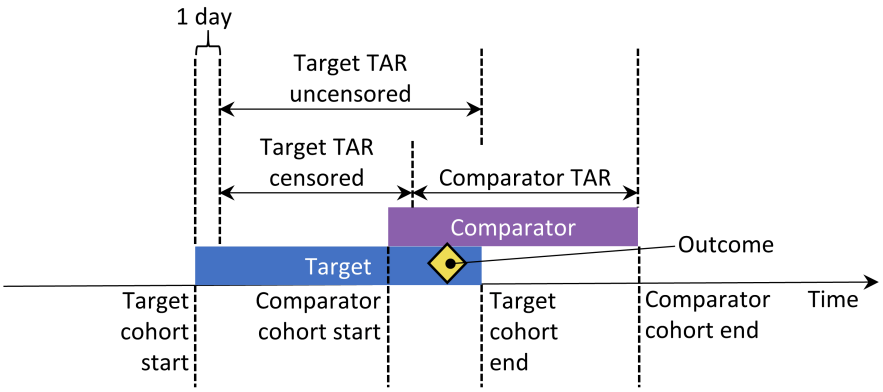


Figure 13.10: Time-at-risk (TAR) for subjects who are in both cohorts, assuming time-at-risk starts the day after treatment initiation, and stops at exposure end.

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

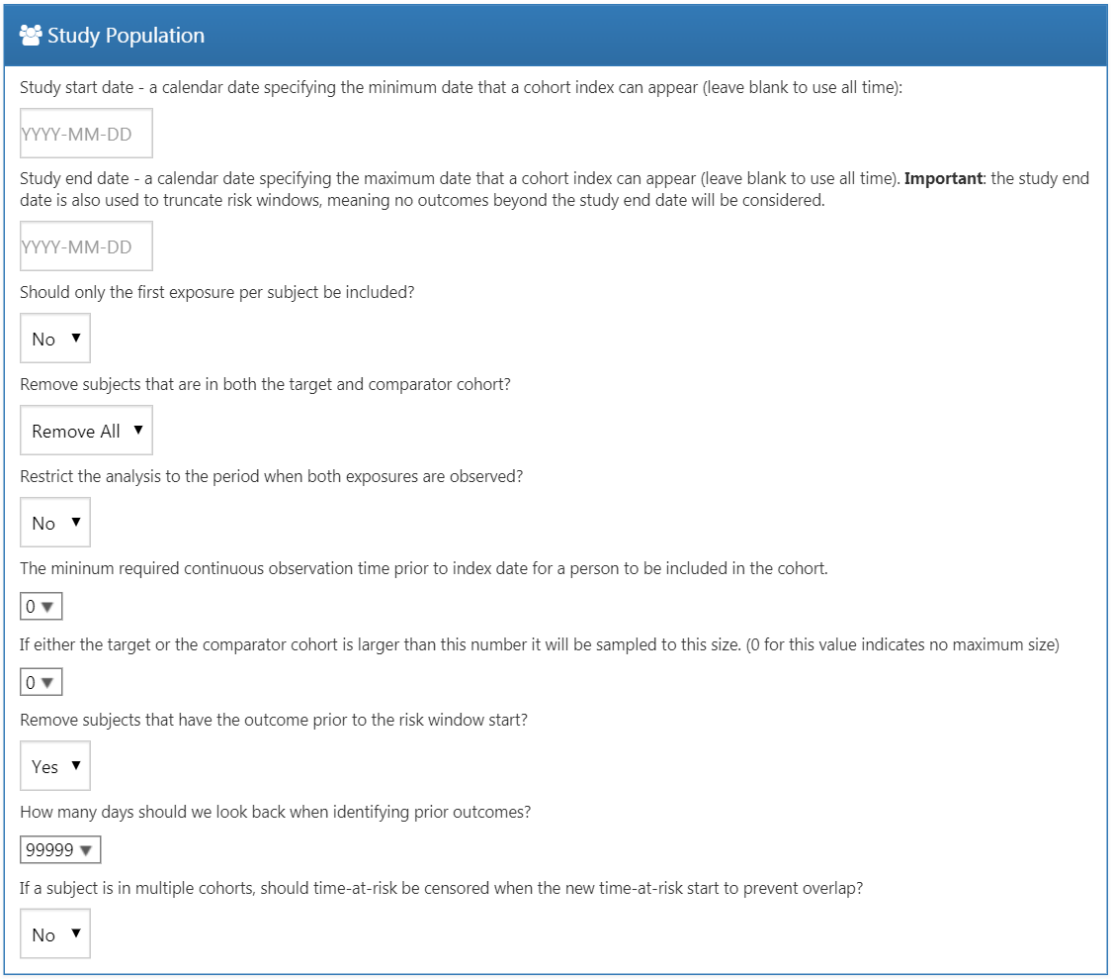


Figure 13.11: Study population settings..

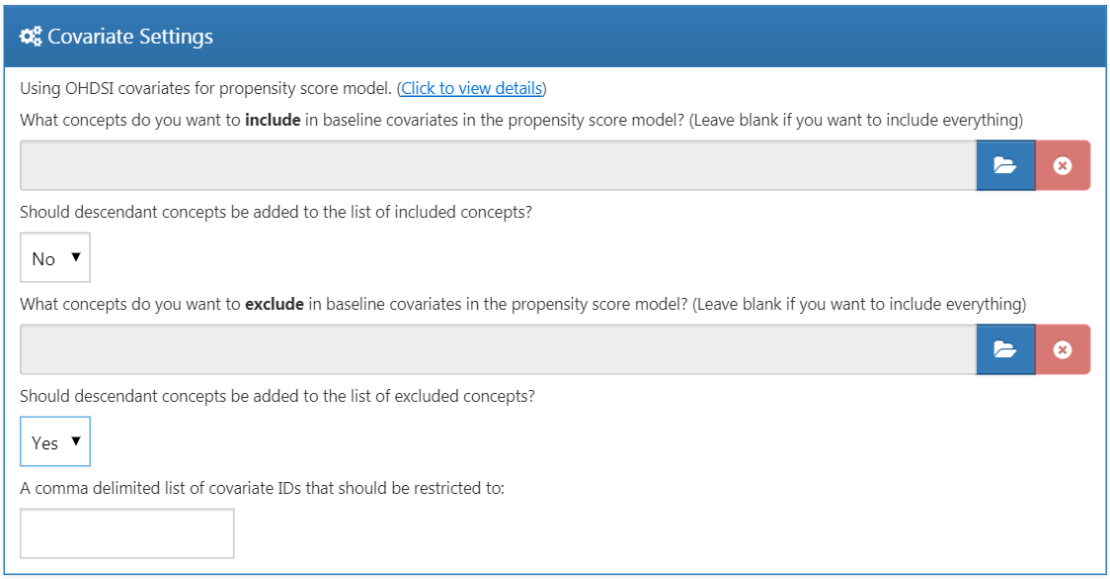


Figure 13.12: Covariate settings.

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,

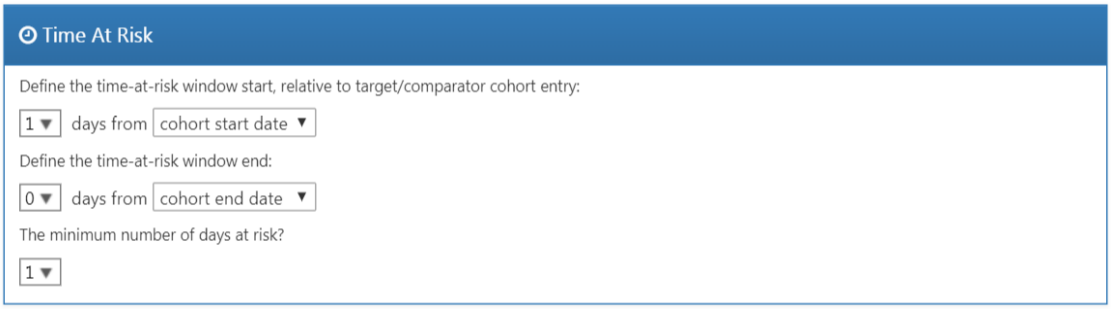


Figure 13.13: Time-at-risk settings.

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-**

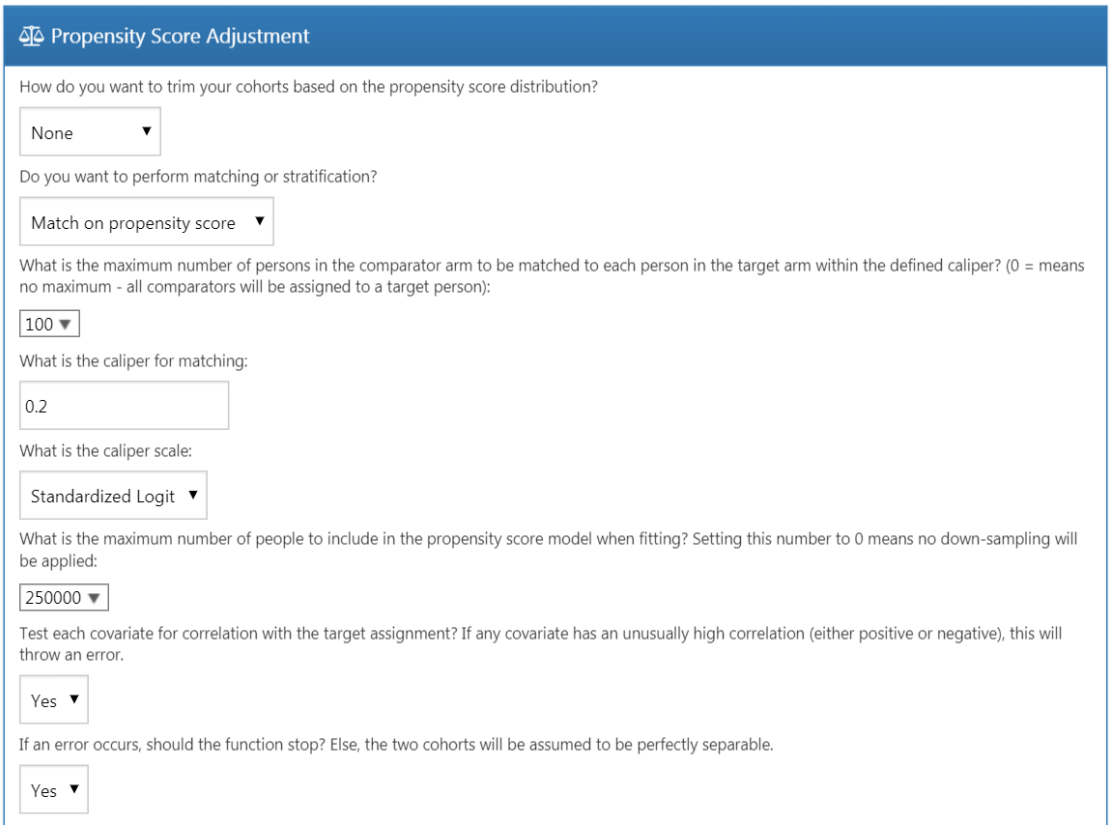


Figure 13.14: Propenstity score adjustment settings.

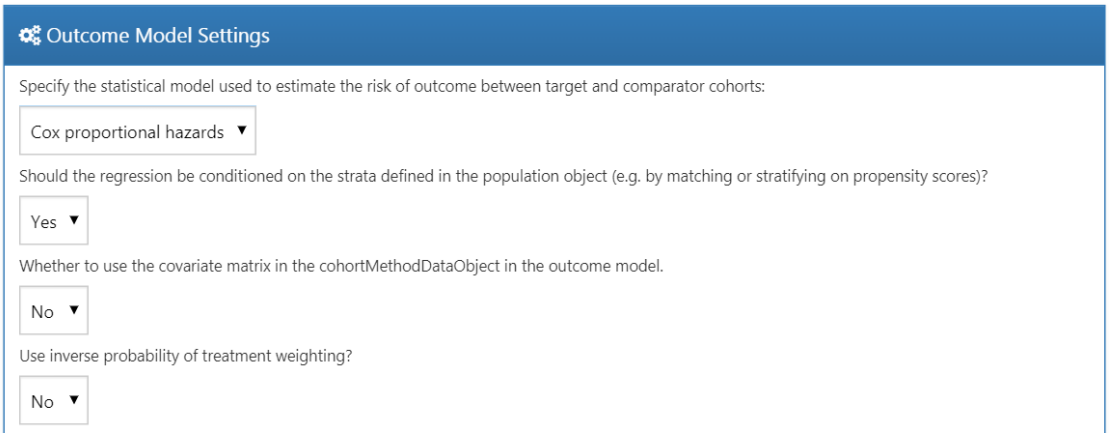


Figure 13.15: Outcome model settings.

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**

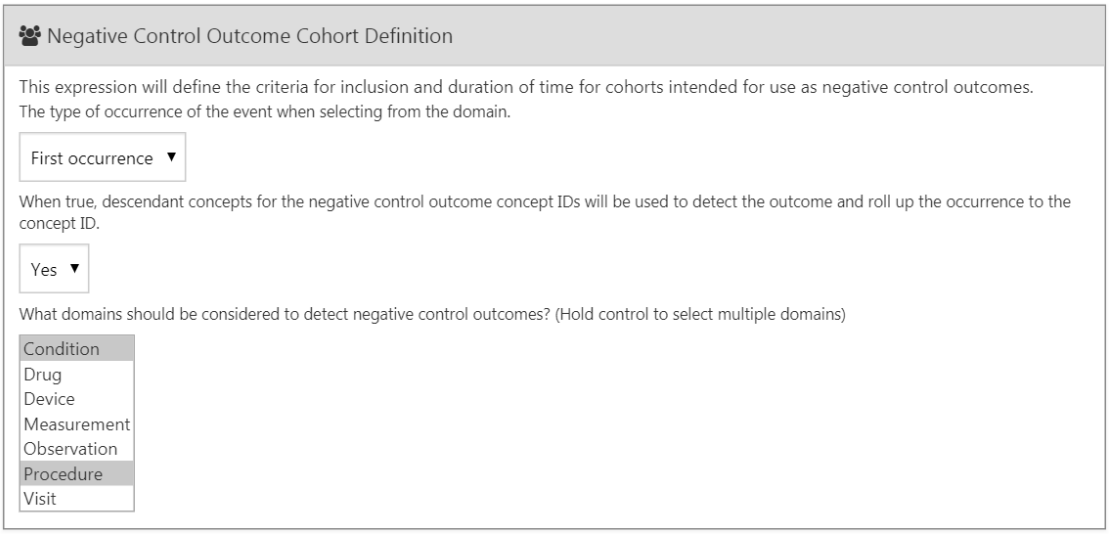


Figure 13.16: Negative control outcome cohort definition settings.

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

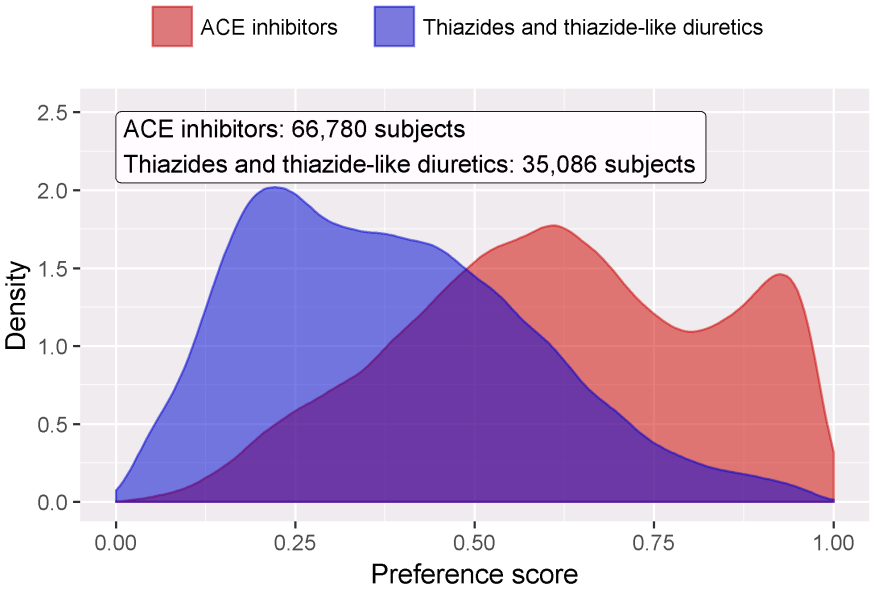


Figure 13.17: Preference score distribution.

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.

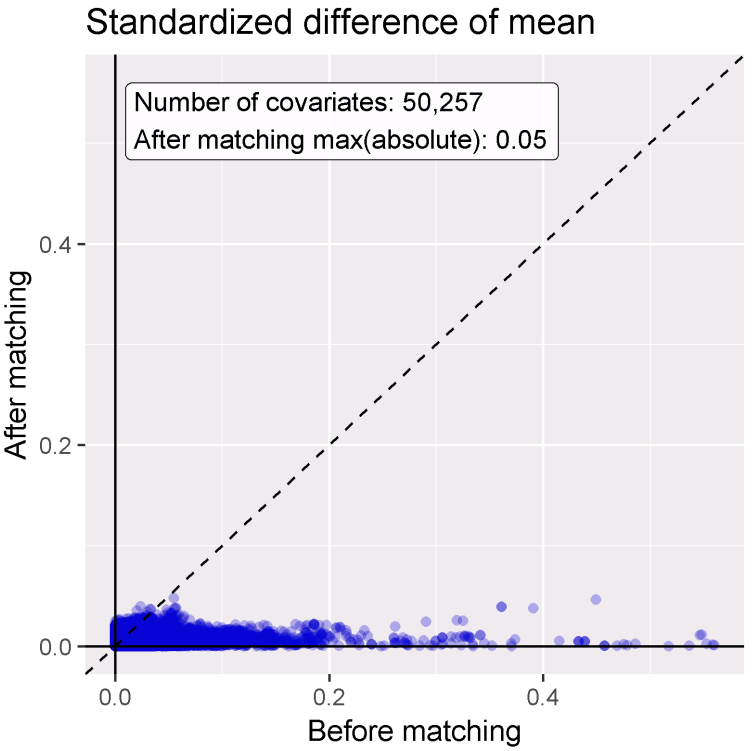


Figure 13.18: Covariate balance, showing the absolute standardized difference of mean before and after propensity score matching. Each blue dot represents a covariate.

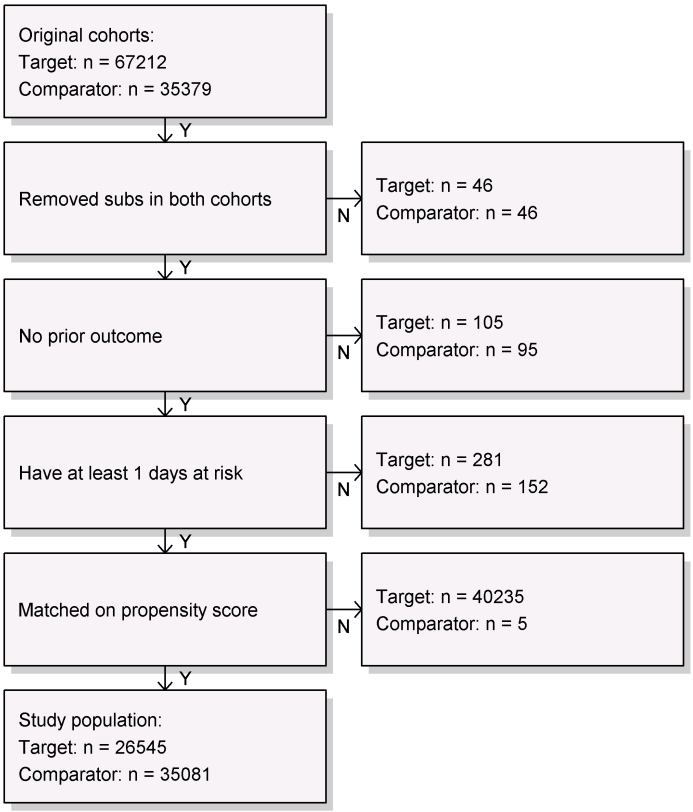


Figure 13.19: Attrition diagram. The counts shown at the top are those that meet our target and comparator cohort definitions. The counts at the bottom are those that enter our outcome model, in this case a Cox regression.

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.

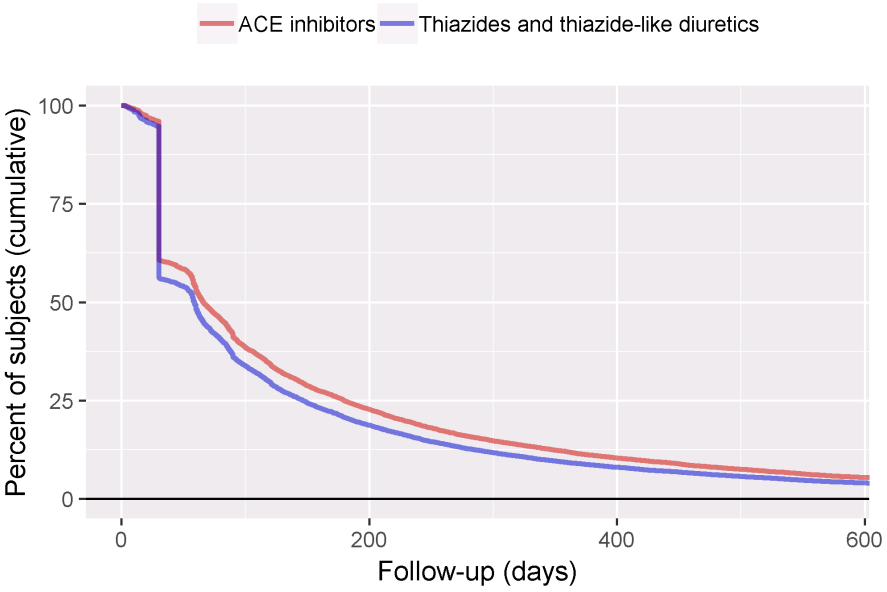


Figure 13.20: Distribution of follow-up time for the target and comparator cohorts.

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.

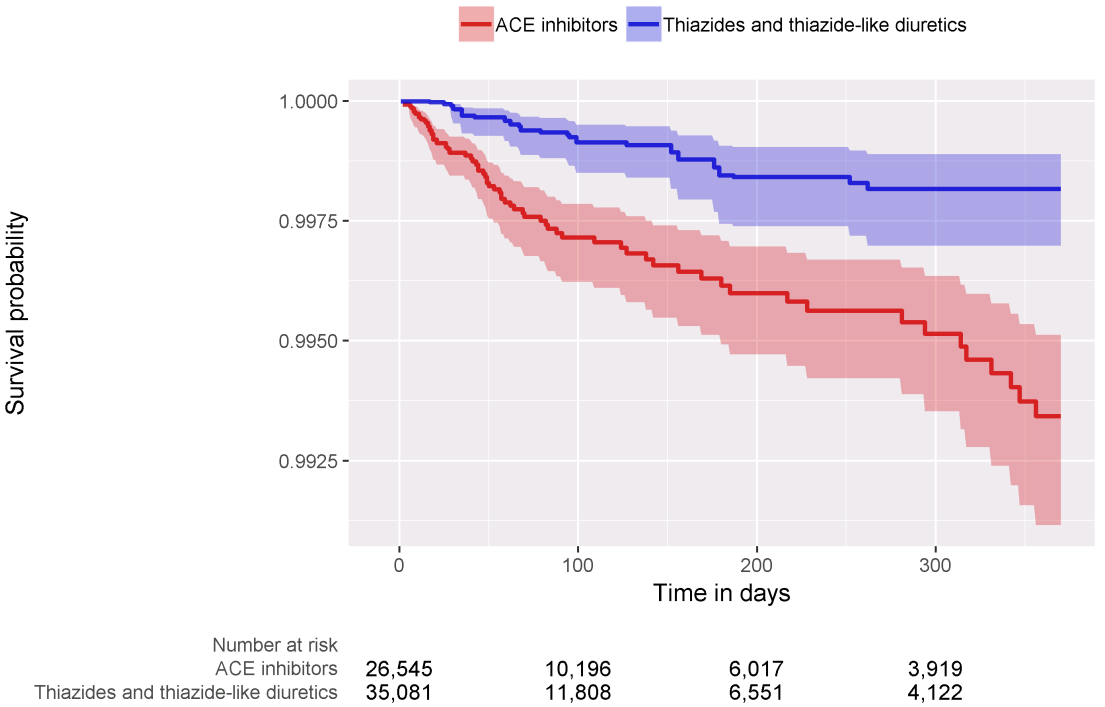


Figure 13.21: Kaplan Meier plot.

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

14.1 Introduction

It is widely excepted in medicine that prevention is better than cure. If we could identify patients who are at a high risk of developing some illness, then it might be possible to intervene and slow down the progression of, or even prevent, the illness. Big observational healthcare data capture rich information across a diverse and often large patient population. These data are collected retrospectively. It may be possible to learn associations from the data that can be used to calculate patient-level predictions for various illnesses or health outcomes. In this chapter we consider predicting the general tasks:

- Disease onset and progression
  - **Structure**: Amongst 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 fibrilation patients, who will go on to have ischemic stroke in the next three years?
- Treatment choice
  - **Structure**: 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).
  - **Example**: Among patients with atrial fibrilation who took either warfarin or rivaroxaban, which patients gets warfarin? (e.g. for a propensity model)
- Treatment response
  - **Structure**: Amongst new users of *[a treatment]*, who will experience *[some effect]* in *[time window]* ?
  - **Example**: Which patients with diabetes who start on metformin stay on metform for three years?
- Treatment safety

  - **Structure**: Amongst new users of *[a treatment]*, who will experience *[adverse event]* in *[time window]*?
  - **Example**: Amongst new users of warfarin, who will have a GI bleed in one year?
- Treatment adherence
  - **Structure**: Amongst 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?

All these examples answer the question, within some target population predict the occurrence of some outcome during some time interval relative to the target population index. Prediction models can be used to find groups of patients that are at a high or low risk of some health outcome or to calculate a patient's personalized risk. However, prediction does not tell us anything about the causality relationship between the predictors and the outcome. If you are interested in causal inference, then you want to read Chapter [add link to Population level estimation].

In this chapter we first describe the theory behind patient-level prediction, including a summary of current progress in patient-level prediction, details about how labeled data are extracted from observational databases, an overview of supervised learning and the various classifiers and how to evaluate the performance of a model. There are two ways to use the OHDSI tools for patient-level prediction. The first approach is to use the atlas interface to design a study and then atlas creates an R package that goes end to end from data extraction to model develoment and evaluation. This requires only basic R knowledge. The second approach is to manually write R code using our PatientLevelPrediction R library. This requires intermediate R knowledge at a minimum, but enables greater flexibility in the prediction analysis study design. In sections - we provide a complete walk-through demonstrating how to correctly specify a patientl-level prediction problem, how to use Atlas and/or R to develop the prediction model and how to evaluate the model.

14.2 Current Progress in Patient-Level Prediction

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. An example is the Garvan model that predicts the 5-years and 10-years fractures risk in any elderly man or woman based on age, fracture history, fall history, bone mass density or weight (Nguyen et al., 2008). Many prediction models have been developed in patient subgroups at higher risk that need more intensive monitoring, e.g. the prediction of 30-day mortality after an acute myocardial described by Lee et al. (1995). Also, many models have been developed for asymptomatic subjects in the population, e.g. the famous Framingham risk functions for cardiovascular disease (Wilson et al., 1998), or the models for breast cancer screening (Engel and Fischer, 2015).

Surprisingly, most currently used models are estimated using small datasets and contain a limited set of patient characteristics. For example, in a review of 102 prognostic models in traumatic brain injury showed that three quarters of the models were based on samples with less than 500 patients (Perel et al., 2006). This low sample size, and thus low statistical power, forces the data analyst to make stronger modelling assumptions. The selection of the often 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 huge amount of patient-specific information contained in the Electronic Health Record (EHR). This includes structured data in the form of diagnose, medication, laboratory test results, and unstructured data contained in clinical narratives. Currently, 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.

Massive-scale, patient-specific predictive modeling has become reality due the OHDSI initiative in which the common data model (CDM) allows for uniform and transparent analysis at an unprecedented scale. These large standardized populations 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. Furthermore, these models could assist in the design and analysis of randomized controlled trials (RCT) by enabling a better patient stratification or can be utilized to adjust for confounding variables in observational research. More accurate prediction models contribute to targeting of treatment and to increasing cost-effectiveness of medical care.

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 that limits the ability of independent researchers to reproduce the models and perform external validation. This makes it hard to fairly evaluate the pre-

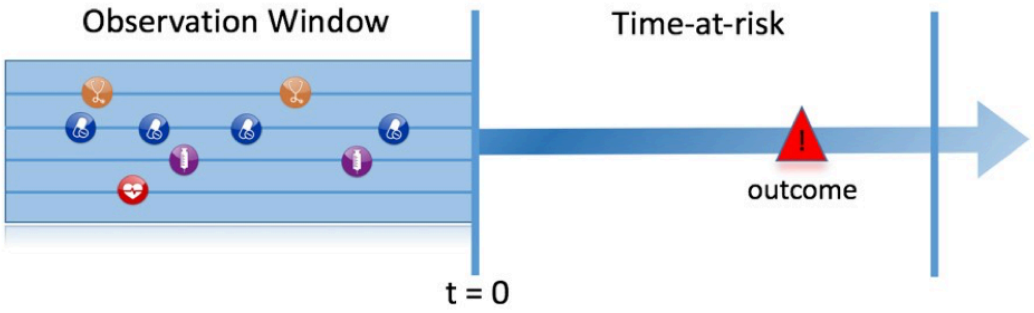


Figure 14.1: The prediction problem.

dictive 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.

The Transparent Reporting of a multivariable prediction model for Individual Prognosis Or Diagnosis (TRIPOD) statement [1] provides clear recommendations for reporting prediction model development and validation and addresses some of the concerns related to transparency. However, data structure heterogeneity and inconsistent terminologies still make collaboration and model sharing difficult as different researchers are often required to write new code to extract the data from their databases and may define variables differently.

In our paper (Reps et al., 2018), we propose a standardised framework for patient-level prediction that utilizes the OMOP Common Data Model (CDM) and standardized vocabularies, and describe the open-source software that we developed implementing the framework's pipeline. The framework is the first to support existing best practice guidelines and will enable open dissemination of models that can be extensively validated across the network of OHDSI collaborators.

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 (T), the outcome we like to predict by an outcome cohort (O), and the time-at-risk (TAR). We define the standard prediction question as:

Amongst [add Target cohort definition], who will go on to have [add outcome definition] within [add 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.

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

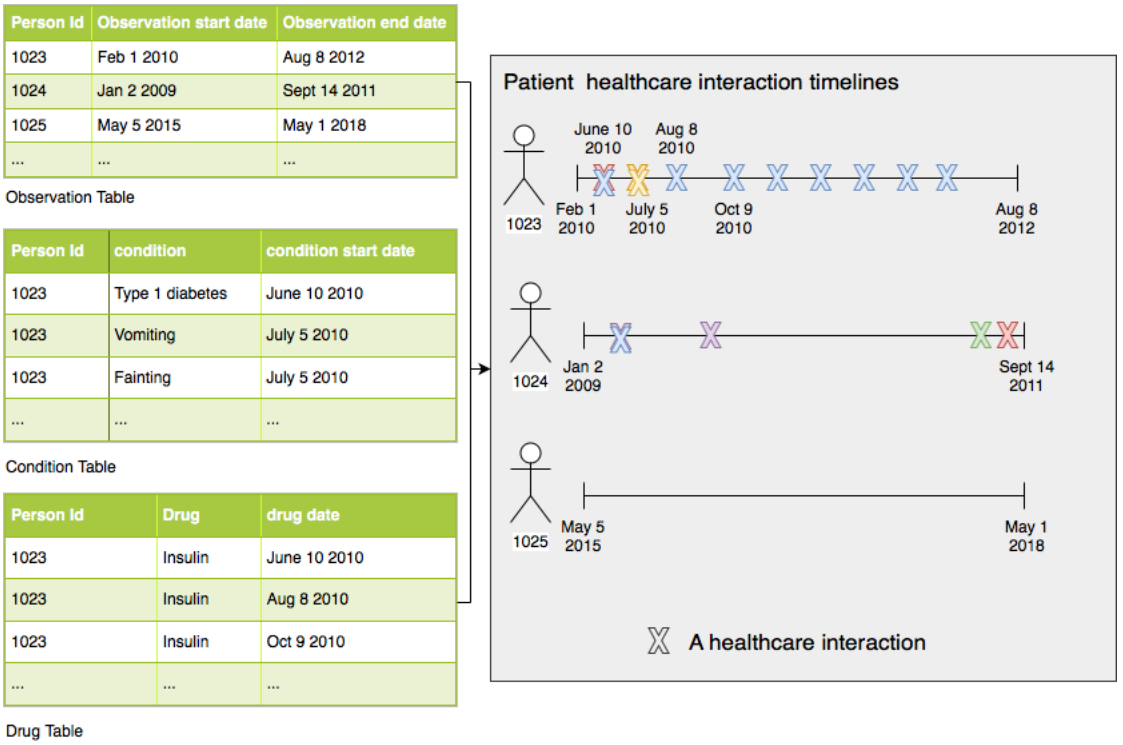


Figure 14.2: Observational data.

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, see [introduction link]

14.3 Creating Labelled Data

The observational datasets we use in OHDSI consist of timestamped records of patient medical interactions. These are represented by tables containing anonymised patient details such as gender and year of birth in addition to tables containing date stamped medical records.

Applying supervised learning techniques for prediction requires having covariate and label pairs for a sufficient number of patients. The covariates (also referred to as features or independant variables)

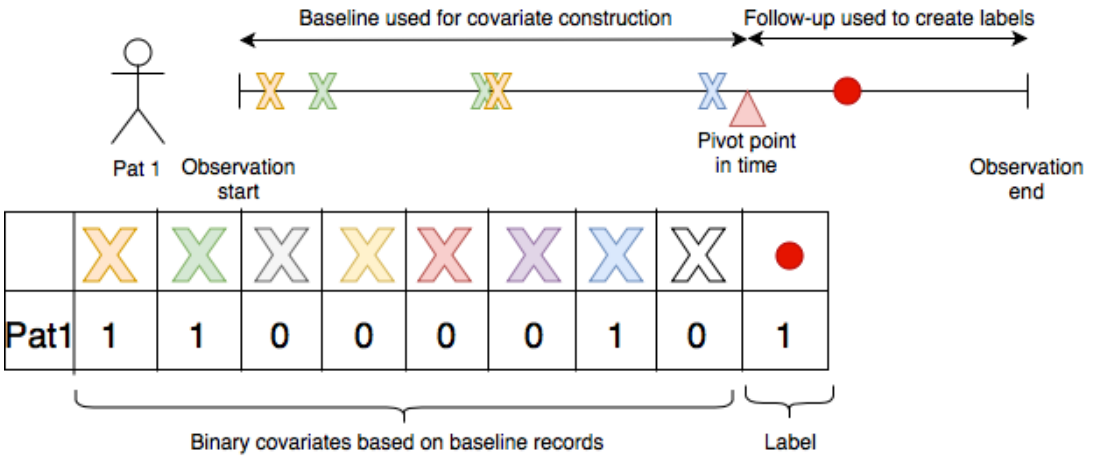


Figure 14.3: Create laelled data from observational data.

describe a patient. Example covariates could be: the patient's gender, age and health state based on the presence or absense of medical conditions. Many of these covariates are time dependant, for example age changes over time, as do some medical conditions. The labels correspond to whether a patient has a outcome of interest during some time interval. The label is also time dependant.

To convert the observational data into labelled data consisting of covariate and label pairs for a set of patients, we need to specify a point in time for each patient that will be used as a pivot. Covariates can be constructed at that pivot point in time (using all records up to that point), and we can can determine whether a patient has the outcome of interest during some time interval relative to the pivot point in time (the time at risk).

This will then provide us with labelled data. The definition of the target cohort population is what we use to define this pivot point in time. For example, if the target cohort was new users of drug A, then the pivot point in time is the date a patient first had drug A recorded in the database. Alternatively, if the target cohort was diagnoses of cardiovascular disease, then the pivot point in time is the date a patient first has a record indicating cardiovascular disease is present. Our prediction specification directly links to how the labelled data are constructed.

14.4 Supervised learning

The idea of supervised learning is to be able to generalise what is observed in the labelled data so that when a new patient's covariates are known but their label is unknown, we can predict their label.

If we consider the situation where we have two covariates, then we can represent each patient as a plot in two dimensional space. The shape/color of a data points corresponds to the patient's label. The idea of supervised learning is to generalise the 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, as seen in Figure 14.5 that aims to minimise the cases where the data point labels do not match the

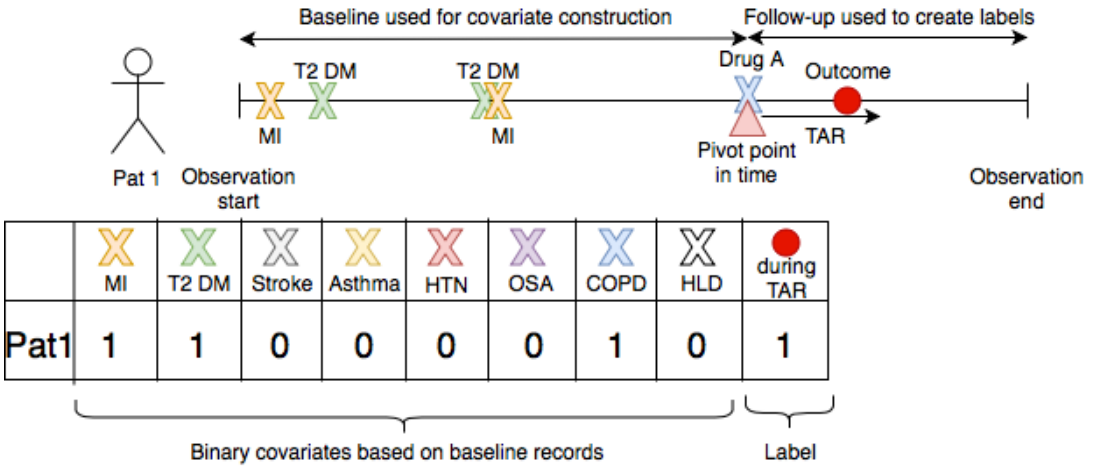


Figure 14.4: Create laelled data from observational data.

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 Figure 14.5 you can see three different decision boundaries. The boundaries are used to infer the class of any new data point. In figure Figure 14.6 the decision boundaries are used to shade the 2 dimentional space into red regions and green regions. If a new data point falls into the green shaded area then the model will predict 'no outcome', otherwise it will predict 'has outcome'.

Ideally a decision boundary should partition the two classes with no error. However, generlizability is an issue, as complex models can 'overfit' where they can correctly partition each data points in the labelled data by using very complex boundaries:

The issue here is that these boundaries may be fit too closely to the labelled data used to learn them and may not work for new data. For example, noise causing incorrectly positioned data points can cause issues. This is shown in Figure Figure 14.6 where the decision boundary goes around a data point that was incorrectly positioned due to noise and this impacts predictions near this region.

Therefore, you want a model that appears to partition the labelled data well but is also as simple as possible. Techniques such as regularization aim to maximise model performance on the labelled data while minimising 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. [add function]. Each supervised learning model 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 when developing patient-level prediction models.

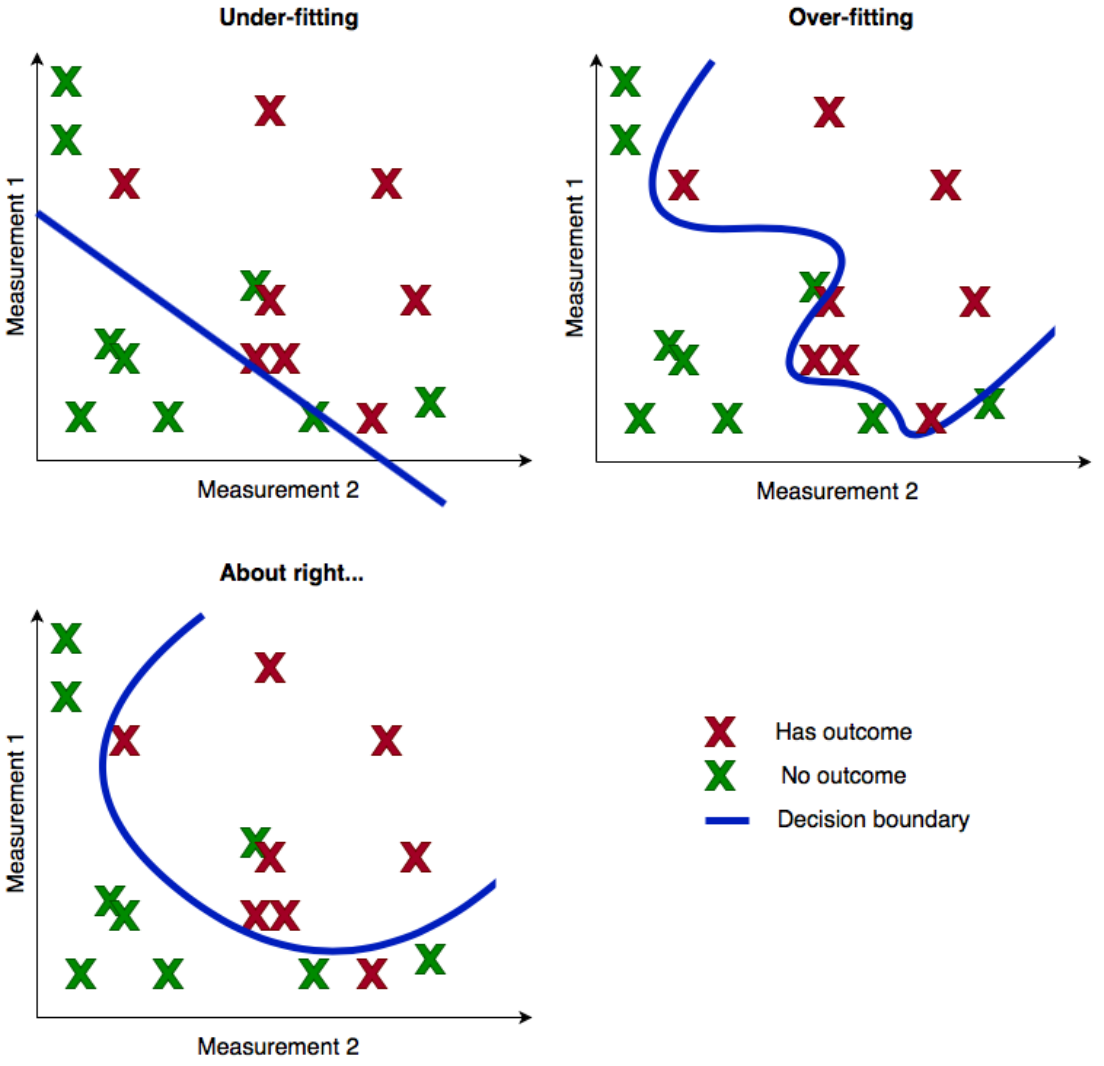


Figure 14.5: Decision boundary.

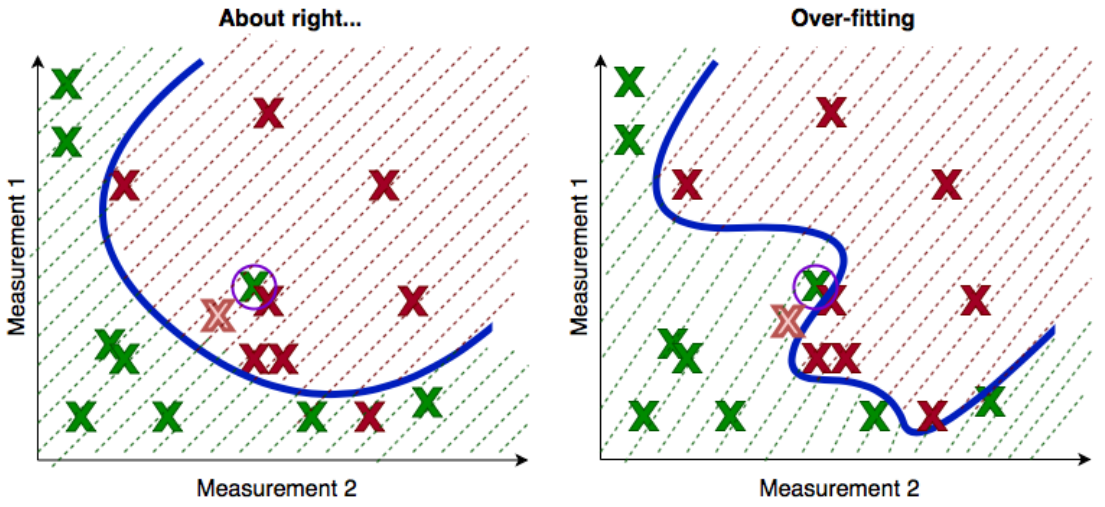


Figure 14.6: Overfitting issues.

14.4.1 Regularized Logistic Regression

Lasso 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. **Hyper-parameters**: var (starting variance), seed.

14.4.2 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. **Hyper-parameters**: ntree (number of trees), max depth (max levels in tree), min rows (minimum data points in in node), learning rate, seed | mtry (number of features in each tree),ntree (number of trees), maxDepth (max levels in tree), minRows (minimum data points in in node),balance (balance class labels), seed.

14.4.3 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 learn implementation of Random Forest in python. **Hyper-parameters**: mtry (number of features in each tree),ntree (number of trees), maxDepth (max levels in tree), minRows (minimum data points in in node),balance (balance class labels), seed.

14.4.4 K-nearest neighbors

K-nearest neighbors (KNN) is an algorithm that uses some metric to find the K closest labelled data-points, given the specified metric, 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 k-nearest neighbor classifier. **Hyper-parameters**: k (number of neighbours), weighted (weight by inverse frequency).

14.4.5 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. **Hyper-parameters**: none.

14.4.6 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. **Hyper-parameters**: 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).

14.4.7 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. **Hyper-parameters**: maxDepth (the maximum depth of the tree), minSamplesSplit,minSamplesLeaf, min-

ImpuritySplit (threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.), seed,classWeight ("Balance"" or "None").

14.4.8 Multilayer Perception

Neural networks containing multiple layers that weight their inputs using a non-linear function. The first layer is the input layer, the last layer is the output layer the between are the hidden layers. Neural networks are generally trained using feed forward back-propagation. This is when you go through the network with a data-point and calculate the error between the true label and predicted label, then go backwards through the network and update the linear function weights based on the error. **Hyper-parameters**: size (the number of hidden nodes), alpha (the l2 regularisation), seed.

14.4.9 Deep Learning

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

14.5 Evaluating Patient-Level Prediction Models

14.5.1 Evaluation Types

There are various ways to evaluate a prediction model: internal validation, external valdiation, temporal validation and spatial validation.

Internal validation is when a prediction model is evaluated using the same dataset used to develop the model. There are three ways to perform internal validation: using a holdout set, using cross validation and using bootstrapping. In the patient-level prediction framework we use a holdout set for internal validation.

A holdout set approach simply 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.

Cross validation is useful when the data are limited. A user needs to specify the number of folds, such as 10, then the data are split into that number of independent sets. For each data split, a model is trained on all other data splits and then applied to the split to obtain the predicted risks for each patient in the split. The model performance can then be estimated using the predicted risk obtained when the patients were held out from model training. A form of cross validation is leave one out validation, where for each patient, the model is trained using all other data except that patient's data and then applied to the patient to obtain their predicted risk. This is repeated for each patient to obtain risks for all the patients which can be used to evaluate the model. In the patient-level prediction framework we use cross validation to pick the optimal hyper-parameters on the train set.

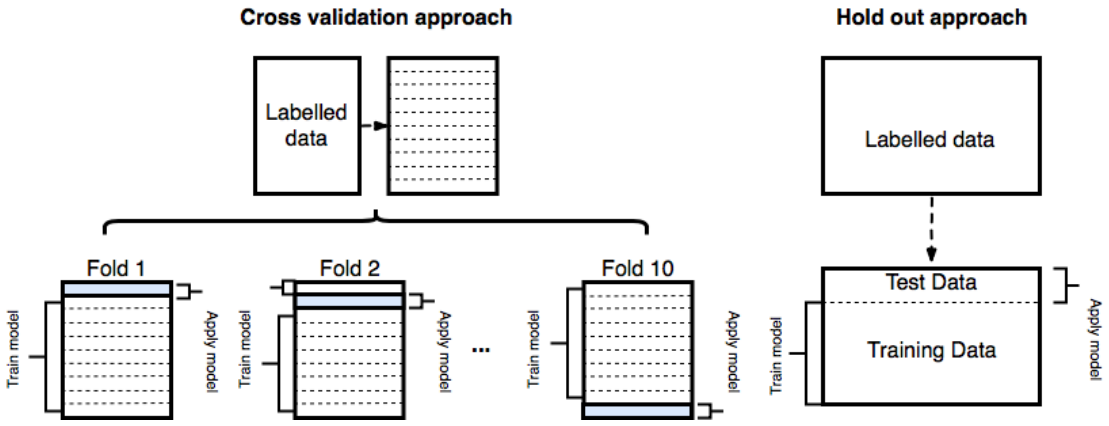


Figure 14.7: Types of internal valdiation.

Bootstrapping is useful when calculating confidence intervals. In bootstrapping multiple sample sets are drawn with replacement from the whole labelled dataset to generate hold out sets, the unsampled patient data are used to develop the models and then evaluated on the sample sets. This gives a range for each metric. We currently do not use bootstrapping in the patient-level prediction framework.

External validation is when a model trained on one dataset is validated on a new dataset or set of patients. This is important as it helps model developers understand which types of patients the model will transport to.

Temporal validation is a type of validation where a model is validated on data that were collected after the data used to develop the model. This can help identify situations where they may be temporal shifts in the data that impact the transportability of the model across time. Another type of validation, spatial validation, is location based where a model is developed on patients for some locations (perhaps certain hospitals or doctor surgeries) and validated on patients at a different location.

14.5.2 Performance Metrics

** Threhold measures ** A prediction model assigned 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, positivity predictive value can be calculated by first specifying a threshold that is used to class patients as having the outcome or not having the outcome during the time at risk. For example, given Table **??**, 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.

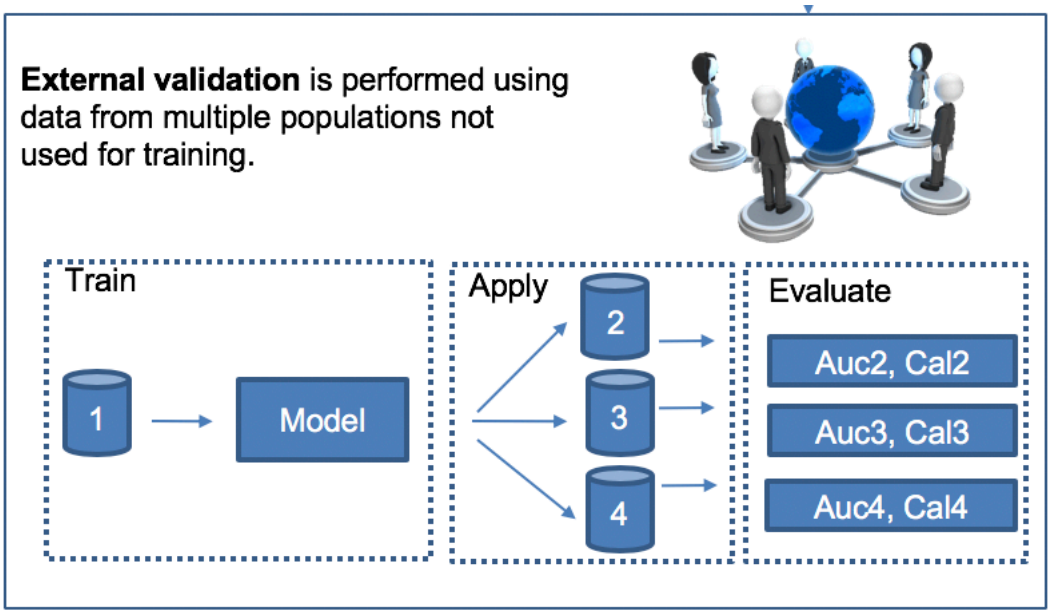


Figure 14.8: Visulisation of external validation.

Patient ID	Predicted risk	Predicted class at 0.5 theshold	Has outcome during TAR	Type
1	0.8	1	1	TP
2	0.1	0	0	TN
3	0.7	1	0	FP
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 TAR then this is called as a true positive (TP). If a patient is predicted to have the outcome but does not have the outcome during TAR then this I called a false positive (FP). If a patient is predicted to not have the outcome and does not have the outcome during TAR then this is called a true negative (TN). Finally, if a patient is predicted to not have the outcome but does have the outcome during TAR 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)

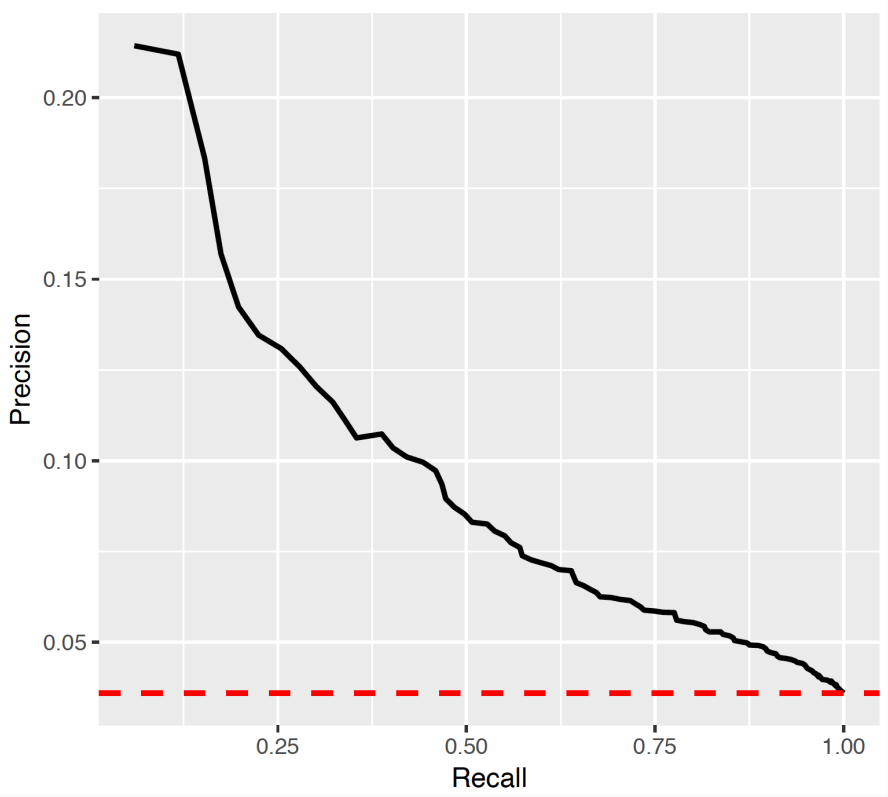


Figure 14.9: Positive predictive value-sensitivy aka Precision-recall plot.

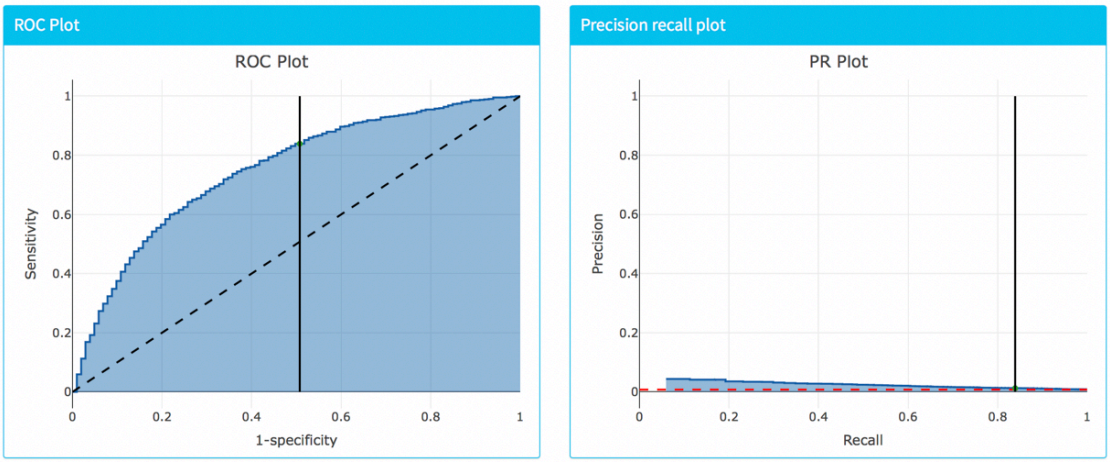


Figure 14.10: Example discrimination plots generated by the patient-level prediction framework.

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, see Figure 14.10. The dashed diagonal line in Figure 14.10 is the performance of a model that randomly assigns predictions. The area under the receiver operating characteristic curve (AUROC) 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. The AUROC is invariant to class imbalance, unlike accuracy, but for rare outcomes even a model with a high AUROC may not be practical. When the outcome is rare another measure known as the area under the precision recall curve (AUPRC) is recommended. A model may obtain a high AUC when the outcome is rare but have a very low positive predictive value. This would mean may false positives. Depending on the severity of the outcome and cost (health risk and/or monetary) of some intervention, a low false positive rate may result in a non-practical model. 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 AUROC 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 AUROC is high, then the distributions will be mostly disjointed, whereas when there is a lot of overlap, the AUROC will be closer to 0.5, see Figure 14.11.

Preference distribution

The preference distribution plot (Figure 14.12) shows the preference score distributions for people in the test set with the outcome (red) without the outcome (blue).

Predicted probability distribution

The prediction distribution box plot shows the predicted risks of the people in the test set with the outcome (blue) and without the outcome (red).

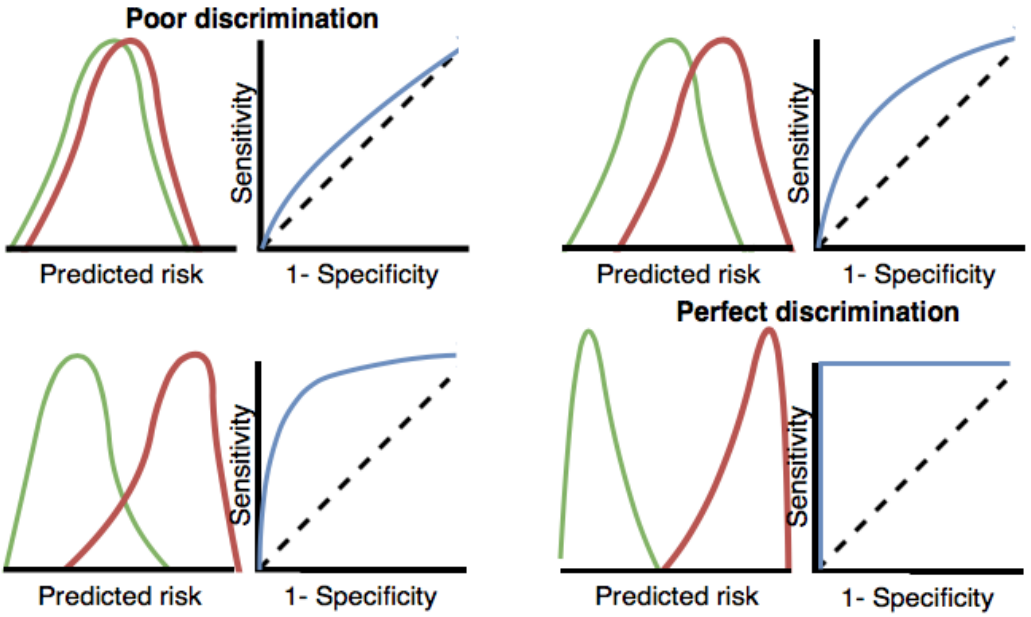


Figure 14.11: How the ROC plots are linked to discrimination.

The box plots in Figure 14.13 show that the predicted probability of the outcome is indeed higher for those with the outcome but there is also overlap between the two distribution which lead to an imperfect discrimination.

** Calibration **

Calibration is the ability of the model to assign a 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. We also fit a linear model using the points to calculate the intercept (which should be close to 0) and the gradient (which should be close to 1). If the gradient is greater than 1 then the model is assigning a higher risk than the true risk and if the gradient is less than 1 the model is assigning a lower risk than the true risk.

AN example calibration plot generated by the Patient-level prediction package is shown in Figure 14.14. The diagonal dashed line thus indicates a perfectly calibrated model. The ten (or fewer) dots represent the mean predicted values for each quantile plotted against the observed fraction of people in that quantile who had the outcome (observed fraction). The straight black line is the linear regression using these 10 plotted quantile mean predicted vs observed fraction points. The straight vertical lines represented the 95% lower and upper confidence intervals of the slope of the fitted line.

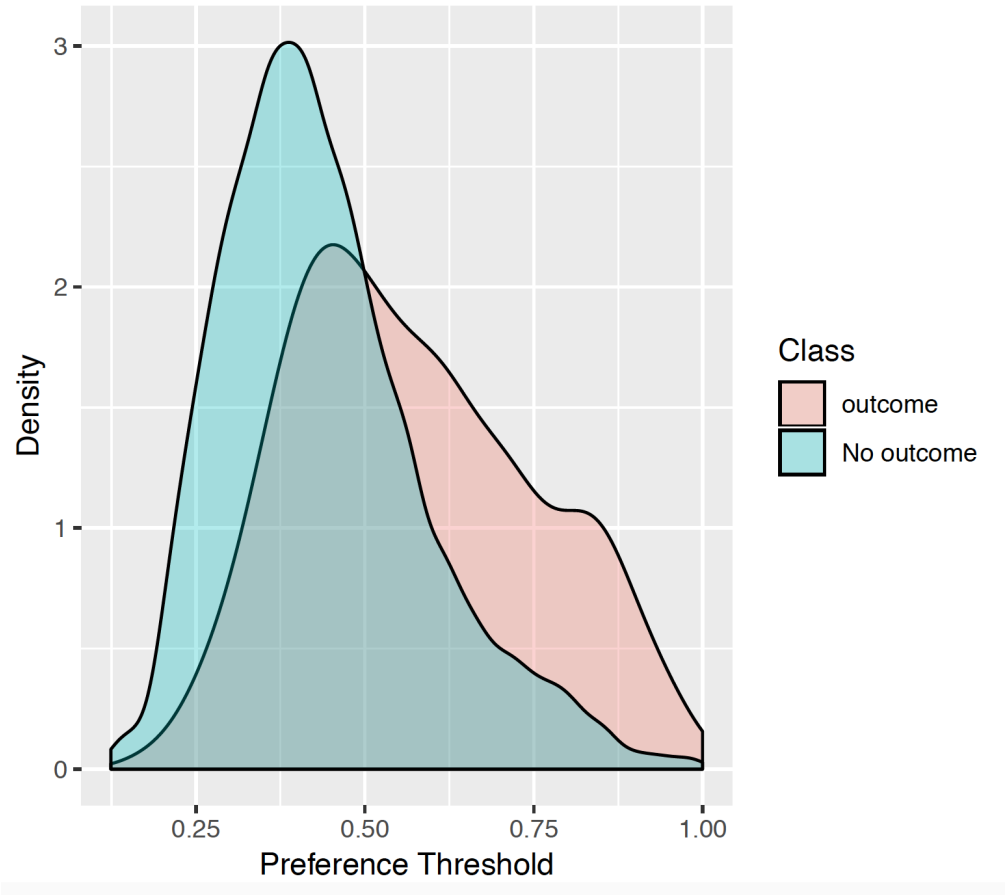


Figure 14.12: Preference distribution plot.

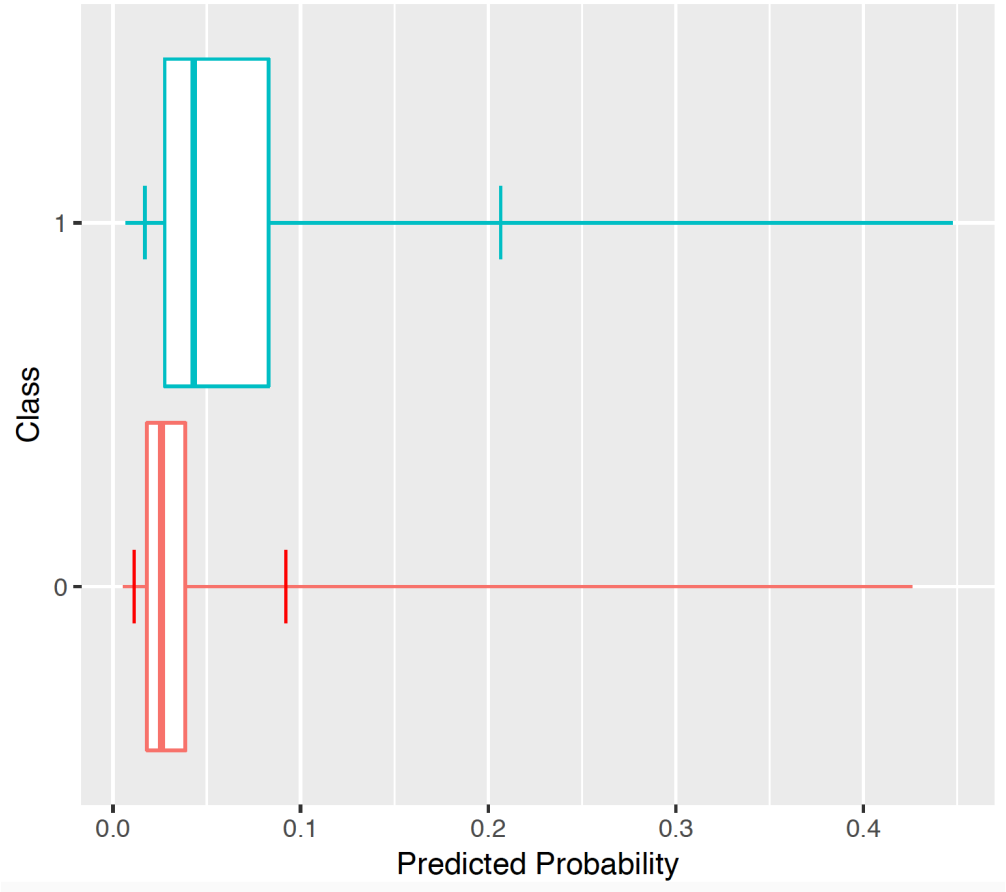


Figure 14.13: Predicted probability distribution.

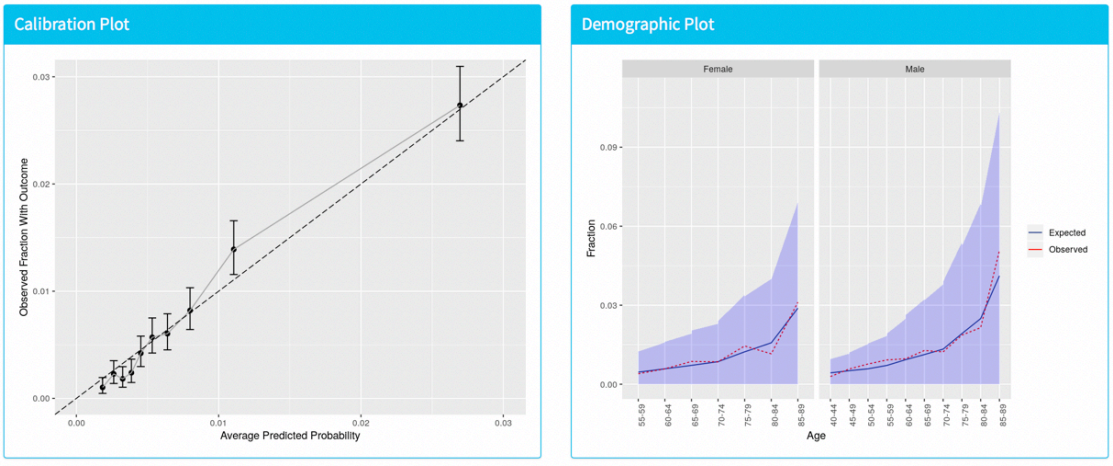


Figure 14.14: Example calibration plots generated by the patient-level prediction framework.

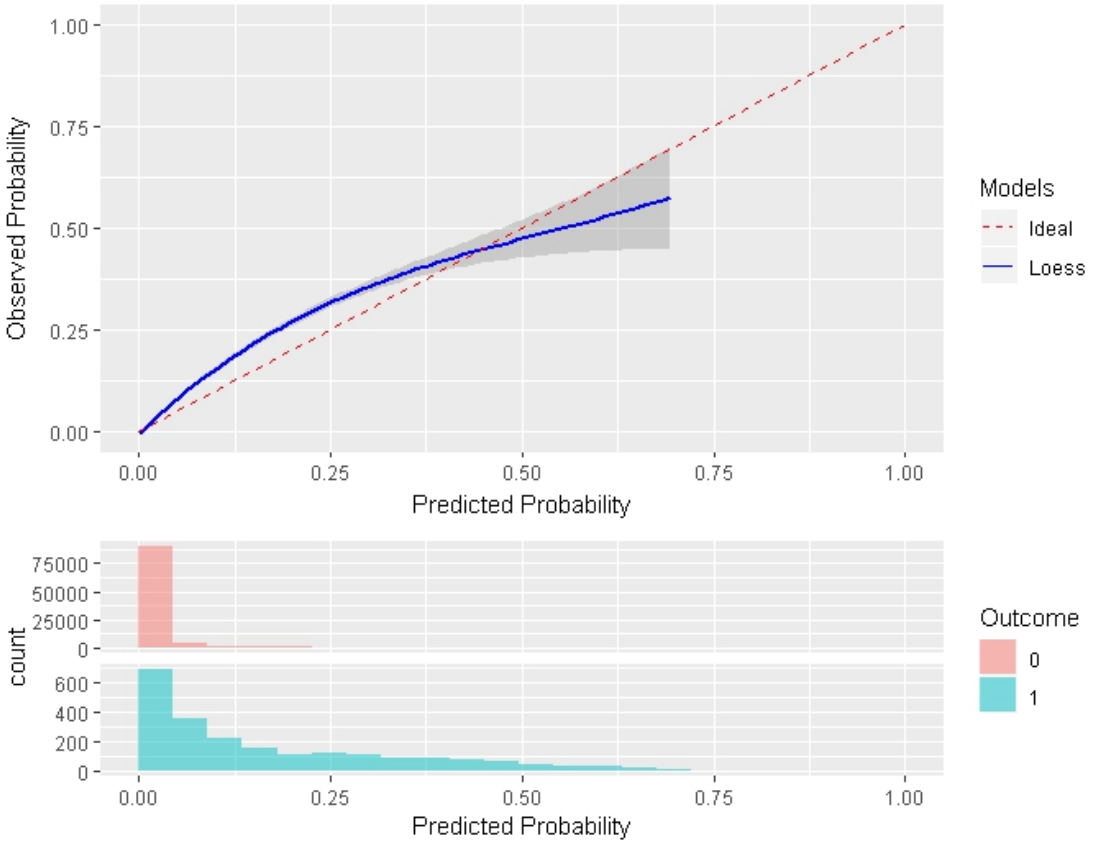


Figure 14.15: Smooth calibration plot.

Smooth Calibration

Similar to the traditional calibration shown above the Smooth Calibration plot shows the relationship between predicted and observed risk. the major difference is that the smooth fit allows for a more fine grained examination of this. Whereas the traditional plot will be heavily influenced by the areas with the highest density of data the smooth plot will provide the same information for this region as well as a more accurate interpretation of areas with lower density. the plot also contains information on the distribution of the outcomes relative to predicted risk.

Figure 14.15 shows an example that better demonstrates the impact of using a smooth calibration plot. The default line fit would not highlight the miss-calibration at the lower predicted probability levels that well.

Demographic summary It can also be useful to determine how well calibrated a model is for different demographics (age and gender groups). This can be calculated by partitioning the patients into groups of similar age and gender and comparing the mean predicted risk within the group with the observed fraction of the patients who experience the outcome during the time at risk. This can identify demographic groups where the model does not perform well when applied. Figure 14.16 shows for females and males the expected and observed risk in different age groups together with a

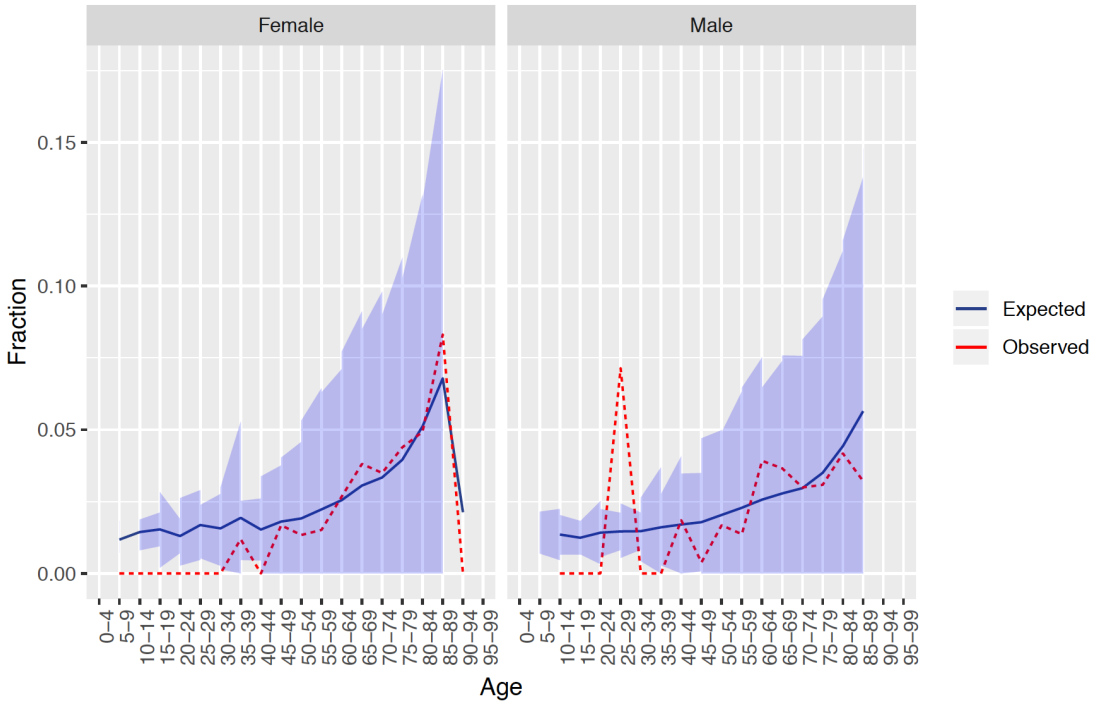


Figure 14.16: Precision-recall plot.

confidence area. This example shows that the model is well calibrated across gender and age groups.

14.5.3 Inspecting the model

Test-Train similarity

The test-train similarity is assessed by plotting the mean covariate values in the train set against those in the test set for people with and without the outcome. This can be a useful way to see how similar the test and train sets are.

The results in Figure 14.17 show that the mean values of the covariates in both the test and train sets are comparable as the points are on the diagonal. If some of the points were off the diagonal then this would tell us the test and train datasets differed.

Variable scatter plot

The variable scatter plot shows the mean covariate value for the people with the outcome against the mean covariate value for the people without the outcome. The color of the dots corresponds to the inclusion (green) or exclusion in the model (blue), respectively. Figure 14.18 shows that the mean of most of the covariates is higher for subjects with the outcome compared to those without. This would indicate that the patients with the outcome are generally sicker.

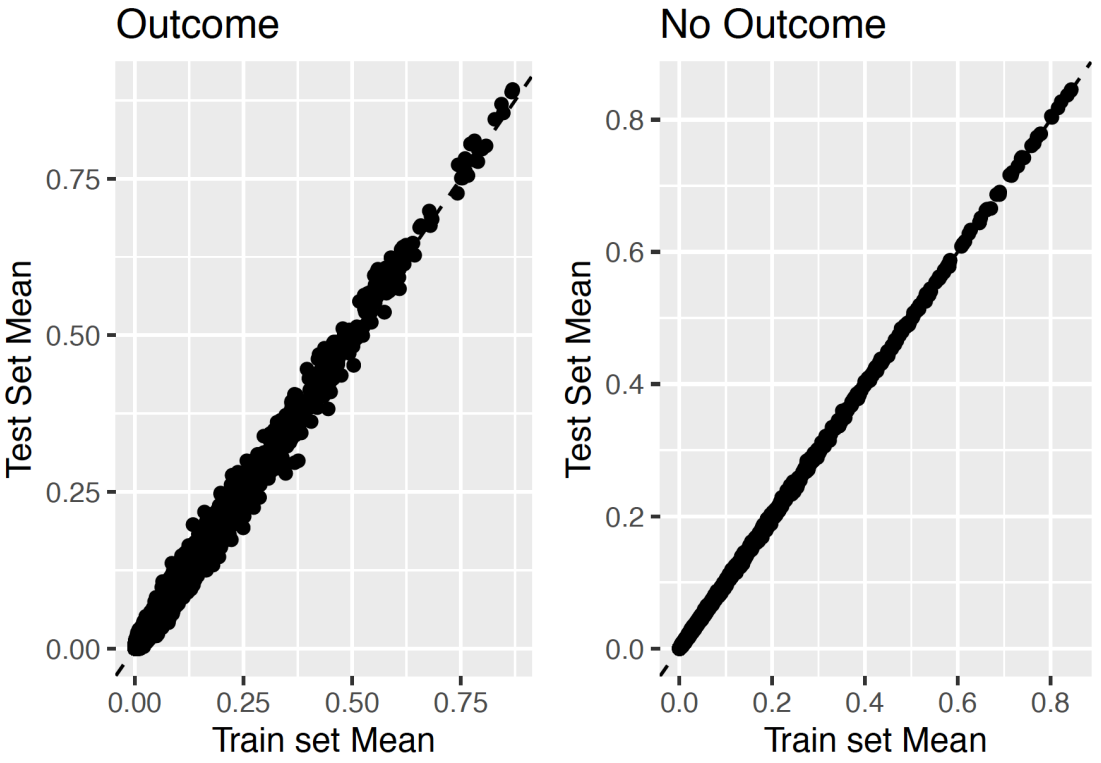


Figure 14.17: Predicted probability distribution.

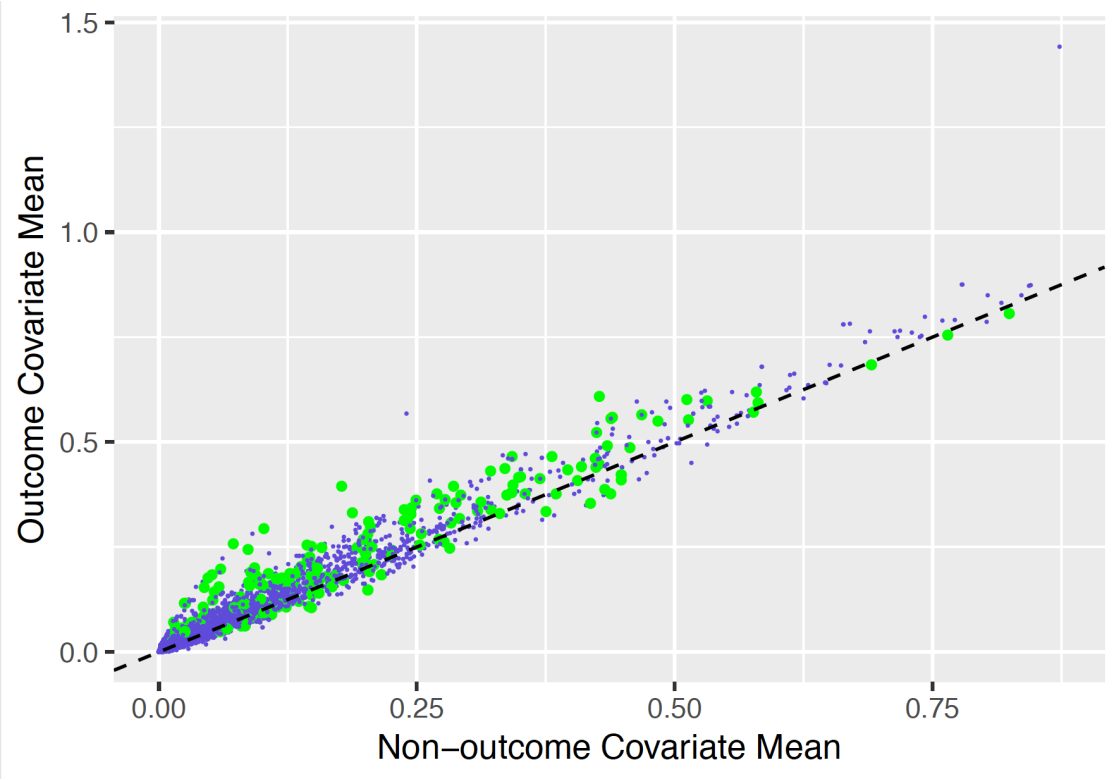


Figure 14.18: Predicted probability distribution.

14.6 Specfifying a Patient-level Prediction Study

In this section we will demonstrate how to define a prediction problem using an example for hypertension.

The first step is to clearly define the prediction problem. Interestingly, in many published papers the prediction problem is poorly defined, e.g. 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 PLP framework we have enforced that we have to define the prediction problem we like to address, in which population we will build the model, which model we will build and how we will evaluate its performance. In this section we will guide you through this process and we will use a "Treatment safety" prediction type as an example.

14.6.1 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 inhibitorin favor of another hypertension drug.

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

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

14.6.2 Study population definition

The final study population in which we will develop our model is often a subset of the target population, because we will e.g. apply criteria that are dependent on T and O or we want to do sensitivity analyses with subpopulations of T. 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 your training data, but also on the time you expect to be available in the data sources you 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 lookback period. For our example, we will use a prior history as lookback 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, a user 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 actually have to make two decisions to answer that 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 you want to avoid outcomes that were entered late in the record that actually occurred before the start of the target cohort or you want to leave a gap where interventions to prevent the outcome could theoretically be implemented. Second, you 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 do not observe them. For our prediction problem we decide to answer this question with Yes, require a mimimum 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.

14.6.3 Model development settings

To develop the 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. you need 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 therefore aim to implement many

algorithms. Furthermore, we made the system modular so you can add your own custom algorithms. This out-of-scope for this chapter but mode details can be found in the *AddingCustomAlgorithms* vignette in the PatientLevelPrediction package.

Our framework currently contains the following algorithms to choose from:

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 also have to specify in which time windows we will look and we decide to look in year before and any time prior.

14.6.4 Model evaluation

Finally, we have to define how we will train and test our model on our data, i.e. how we perform **internal validation**. For this we have to decide how we divide our dataset in a training and testing dataset and how we randomly assign patients to these two sets. Dependent on the size of the training set we can decide how much data we like to use for training, typically this is a 75% - 25% split. If you have very large datasets you can use more data for training. To randomly assign patients to the training and testing set, there are two commonly used approaches:

1. split by person. In this case a random seed is used to assign the patient to either sets.
2. split by time. In this case a time point is used to split the persons, e.g. 75% of the data is before and 25% is after this date. The advantage of this is that you take into consideration that the health care system has changed over time.

For our prediction model we decide to start with a Regularized Logistic Regression and will use the default parameters. We will do a 75%-25% split by person.

14.6.5 Study summary

We now completely defined our study as shown in Table 14.3.

Table 14.3: Main design choices for our study.

Choice	Value
Target cohort	Patients who have just started on an ACE inhibitor for the first time.
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.

We define the target cohort as the first exposure to any ACE inhibitor. Patients are excluded if they have less than 365 days of prior observation time or have prior angioedema.

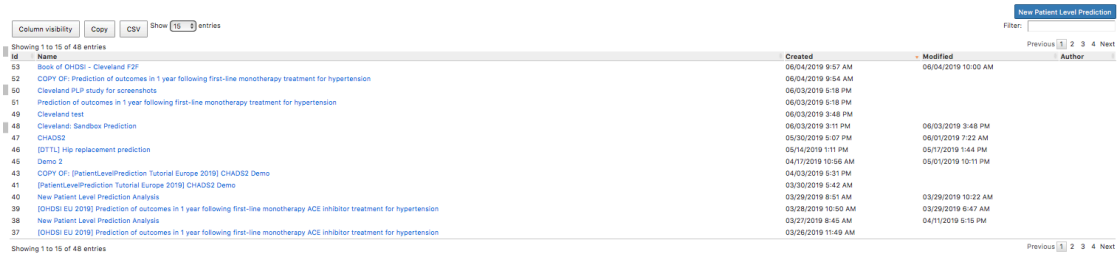


Figure 14.19: The Atlas prediction page.

14.7 Implementing the study in Atlas

14.7.1 Introduction

The atlas interface to patient-level prediction enables a users to design a prediction study analysis containing multiple prediction questions and analyses settings. The atlas interface creates a prediction study R package populated with all the code ready to develop and evaluate the specified models. All a user needs to develop the models is R studio with:

- OHDSI's PatientLevelPrediction R package installed
- devtools R package installed
- connection details for the OMOP CDM databases

The atlas created prediction study R package has additional functionality to:

- Create a study protocol template
- Create a shiny app for interactively exploring the results
- Create a validation study R package that can be shared to externally validate the developed models

In this section we will detail the design choices for the prediction problem specification, the analysis settings and the executing settings. We will then guide the user through the process of reviewing analysis, downloading and running the prediction study package and interpreting the results via the shiny app.

14.7.2 The Atlas layout

The interface for designing a prediction study can be opened by clicking on the 'Prediction' button in the left hand side atlas menu.

Once in the 'Prediction' view you should see Figure 14.19

You can create a new study by clicking on the blue 'New Patient Level Prediction''button or by clicking on a row in the table with the name of the study you want to open. Once inside the prediction study (either by clicking the blue 'New Patient Level Prediction''button or an existing row in the table) you should see a specification options as shown in 14.20-14.21 with the top stating 'Patient

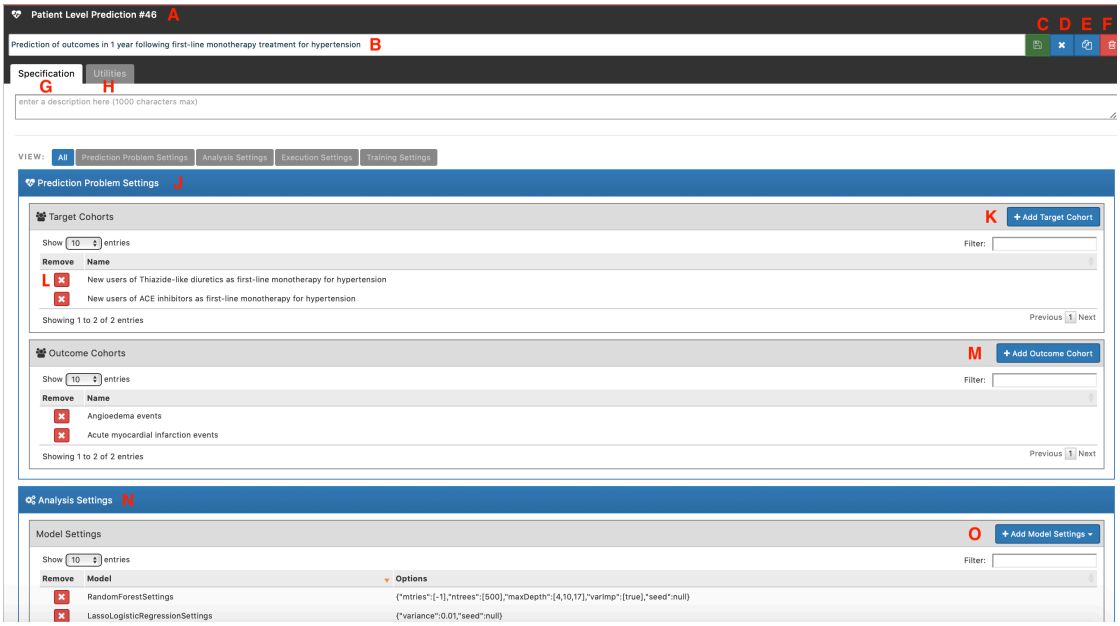


Figure 14.20: The atlas prediction specification part 1.

Level Prediction' with a number such as '#46', as highlighed by the red A. This tells us the cohort definition id is 46.

To the right of the 'Patient Level Prediction #46' there are buttons to save (green button), exit (blue button with x), copy (blue button with double paper) and delete (red button with bin) the current study highlighted by C-F respectively in Figure 14.20.

Below these is a white text form where you can name the study (B in Figure 14.20). The 'Specification' tab (G in Figure 14.20) contains all the settings a user needs to define for the prediction study. The first part is the 'Prediction Problem Settings' (J in Figure 14.20), this is where the user defines the Target cohorts and Outcome cohorts for the prediction analyses. These cohorts need to be created in atlas using the 'Cohort Definition' view and can then be imported into the Prediction study. Instantiating cohorts is described in Chapter 11.

The next part of the 'Specification' is the 'Analysis Settings' (N in Figure 14.20). This is where the user specifies the models to train (classifiers or survival models), the candidate covariates (these are standard OHDSI covariates), the time-at-risk and additional inclusion criteria.

Then the 'Execution Settings' (R in Figure 14.21) define how many patients to extract for the model development, whether to remove rare covariates and whether to normalise the covariates.

Finally, the last part in the 'Specification' is the 'Training Settings' (S in Figure 14.21) which specifies' how to split the labelled data into data used to develop the model (including how many folds you want to use when applying cross validation) and validate the model.

Each of the 'Specification' settings are described in more detail in the following sections. We also describe the 'Utilities' tab (H in Figure 14.20) where a user can review, import/export and download

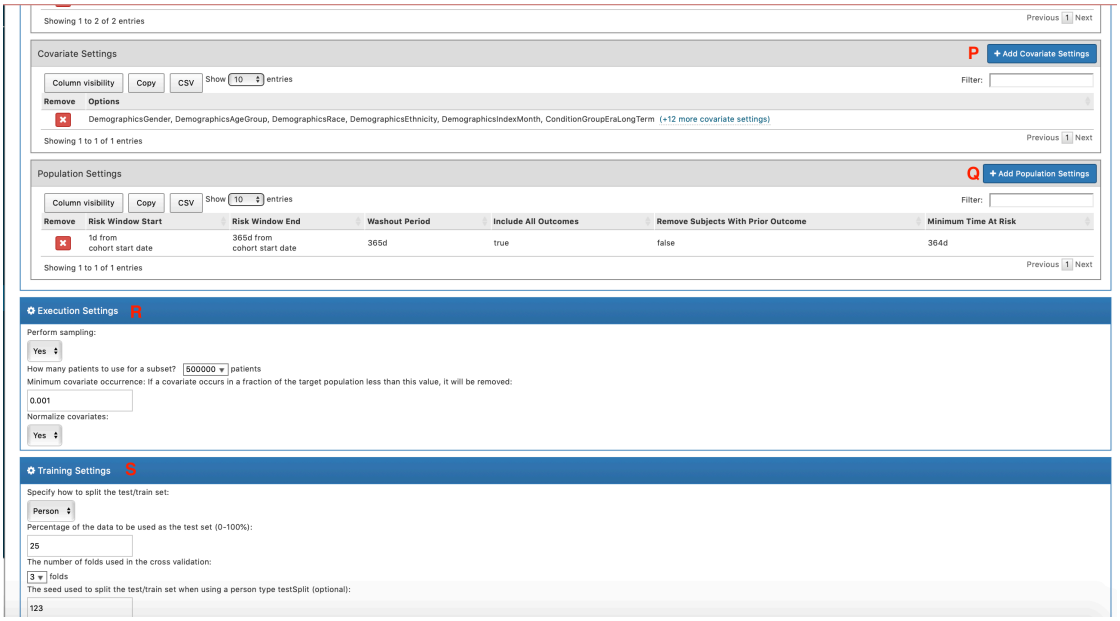


Figure 14.21: The atlas prediction specification part 2

their study as an executional R library.

14.7.3 Atlas Specification Tab

The specification section is where a user can specify her prediction question, covariates, additional study population inclusion criteria, model type and hyper-parameters and execution settings.

14.7.4 Prediction Problem Settings

The prediction problem settings enables you to 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 you specify two target populations:

- 'T1: new users of ACE inhibitors'
- 'T2: new users of ACE inhibitors with no prior anti-hypertensive'

and three outcomes:

- 'O1: angioedema'
- 'O2: stroke'
- 'O3: myocardial infarction'

then six prediction problems will be investigated in the study:

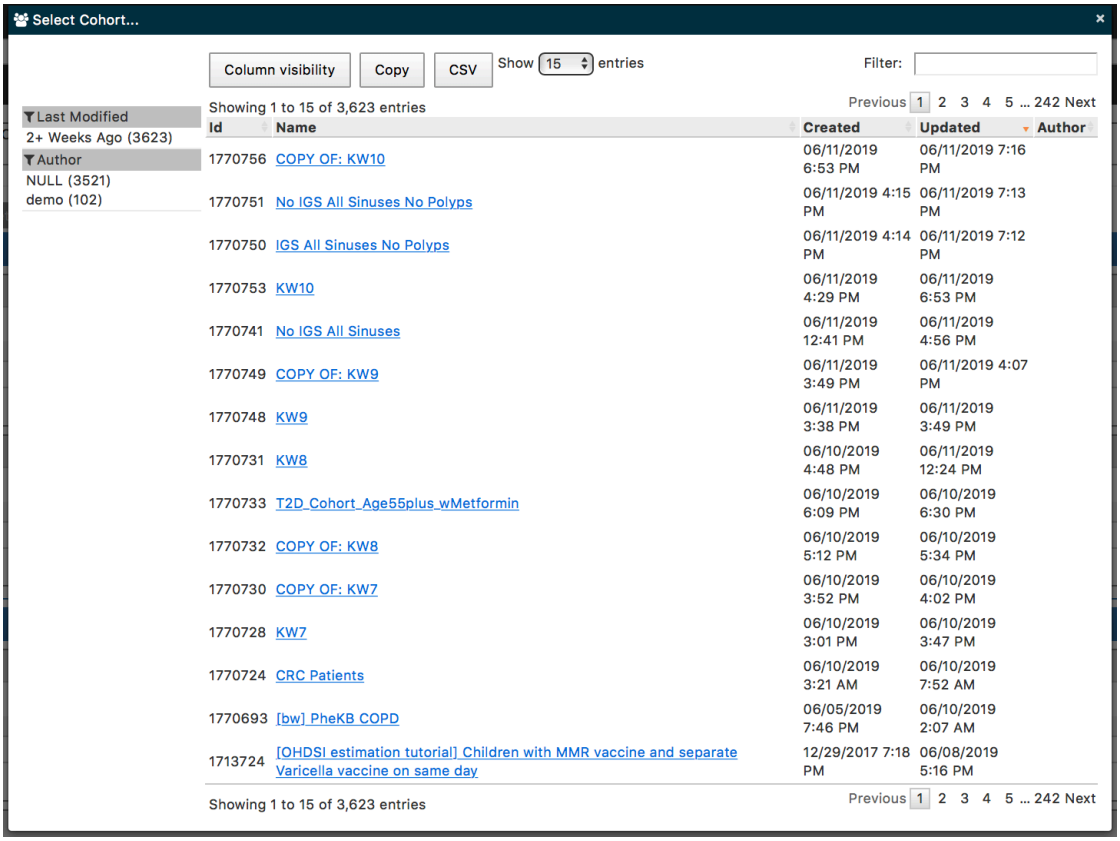


Figure 14.22: The Training Settings area

- 'In T1: new users of ACE inhibitors predict O1: angioedema during TAR'
- 'In T1: new users of ACE inhibitors predict O1: stroke during TAR'
- 'In T1: new users of ACE inhibitors predict O1: myocardial infarction during TAR'
- 'In T2: new users of ACE inhibitors with no prior anti-hypertensive predict O1: angioedema during TAR'
- 'In T2: new users of ACE inhibitors with no prior anti-hypertensive predict O1: stroke during TAR'
- 'In T2: new users of ACE inhibitors with no prior anti-hypertensive predict O1: myocardial infarction during TAR'

To select a target population cohort you 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 you then need to click on the blue '+ Add Target Cohort' button, see K in Figure 14.20.

This will open up a table of cohorts that have been created in atlas, see Figure 14.22.

You can simple click on any row in the table to add that cohort. If you have many cohorts, using the filter option on the top right may help (just make sure to remember the cohort name). We filtered

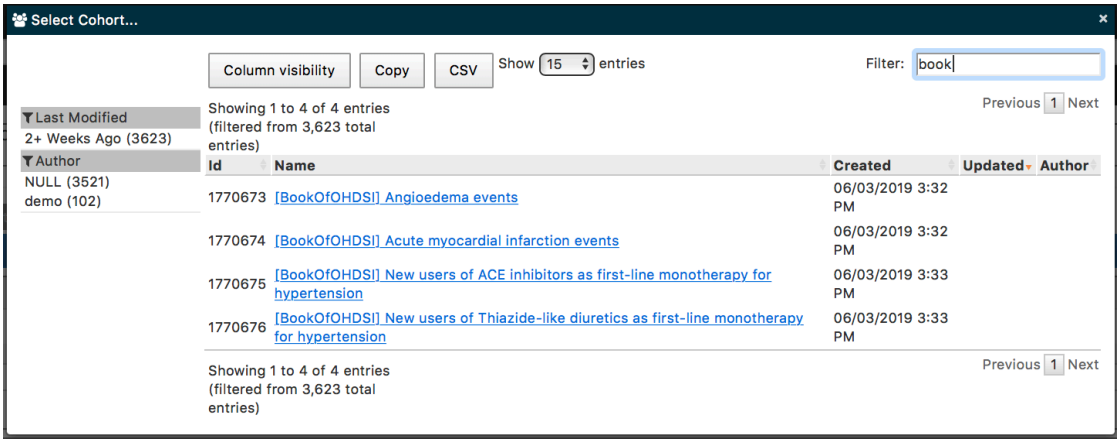


Figure 14.23: The Training Settings area

the book of ohdsi cohorts by adding 'book' to the filter as the cohort names all included the work 'book', see Figure 14.23.

By clicking on the row '[BookOfOHDSI] New users of ACE inhibitors as first-line monotherapy for hypertension' this is now added as a target population cohort in the study. This process can be repeated to add more target population cohorts. Adding outcome cohorts is a similar process, but requires click on the blue '+ Add Outcome Cohort' button (M in Figure 14.20).

You need to specify, at minimum, one target population cohort and one outcome cohort. Once you have added all the target population cohorts and outcome cohorts you are now ready to procede to the analysis settings.

14.7.5 Analysis Settings

The analysis settings enables you to pick the supervised learning models, the covariates and population settings.

14.7.6 Model Settings

You can pick one or more supervised learning models to investigate using for model development. To add a supervised learning model click on the blue '+ Add Model Settings' button (O in Figure 14.20). A dropdown containing all the models currently supported in the Atlas interface will appear (note: more models may be available outside of Atlas).

You can select the supervised learning model you want to include in the study by clicking on the name in the dropdown menu. This will then take you to a view for that specific model and the hyperparameters you can include into a grid search. For example, if I click on 'Lasso Logistic Regression' the following view shown in Figure 14.24 will appear.

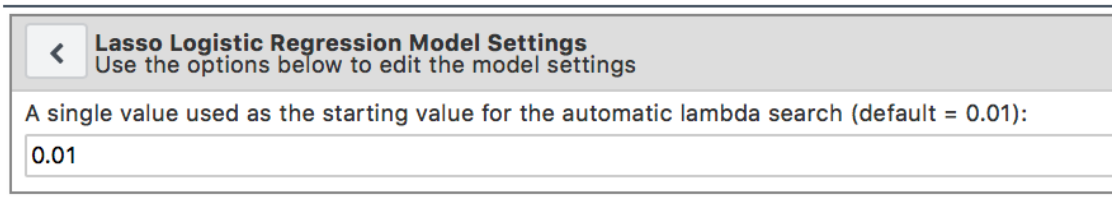


Figure 14.24: The lasso logistic regression view

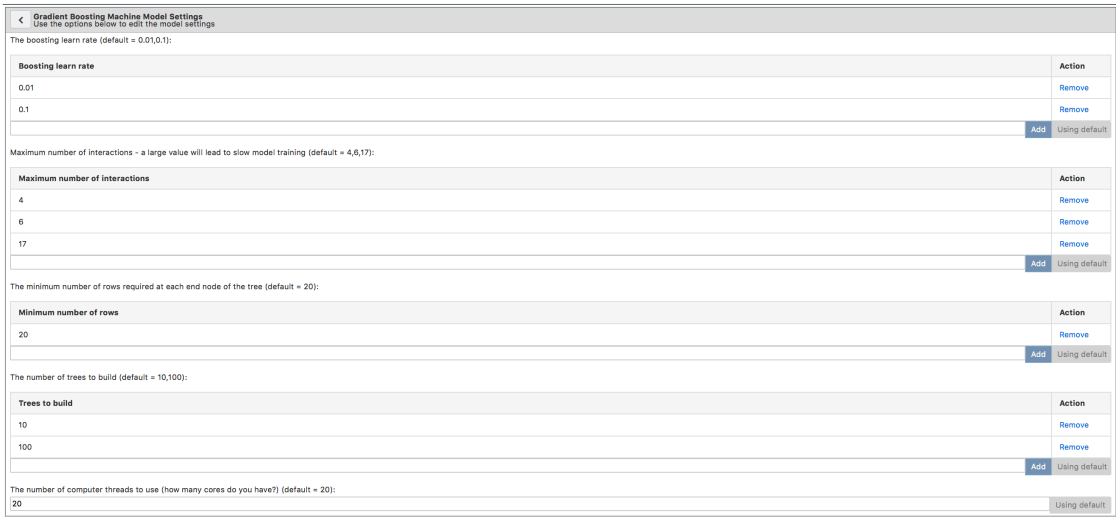


Figure 14.25: The gradient boosting machine view

As the Lasso Logistic Regression model only has one hyper-parameter, we do an automatic search for the optimal value rather than a grid search so a user just needs to specify the starting value, see Figure 14.24. Once you are happy with the hyper-parameter settings you can return to the main settings view by clicking on the grey '<' button.

You will now see your chosen supervised learning model added tosection N in Figure 14.20. To edit the model you added, click on the corresponding row and it will take you back to the model view where you can edit the hyper-parameter settings.

To add a gradient boosting machine model we can follow the same process and click on 'Gradient Boosting Machine' in the drop down menu. This will take us into the gradient boosting machine view:

The gradient boosting machine model has four hyper-parameters you can define a grid search for (boosting learn rate, maximum number of interactions, minimum number of trees and number of trees to build). Initially the default values are shown, but a user can add a new value by typing it into the text field at the bottom of the hyper-parameter box and clicking on the blue 'Add' button.

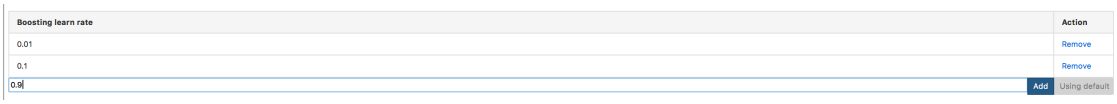


Figure 14.26: Adding a hyper-parameter value into the grid search

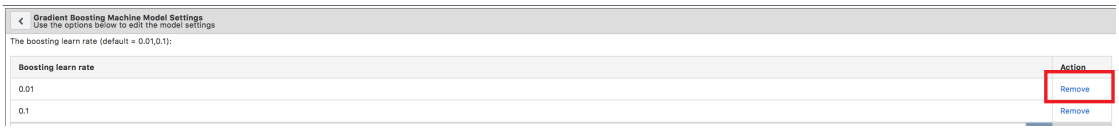


Figure 14.27: Removing a hyper-parameter value into the grid search

It is also possible to remove a hyper-parameter value from the grid search by clicking on 'Remove' for the corresponding row:

Once happy with the hyper-parameters, click on the grey '<' button on the top left to add the model into the prediction study. You will now see you model and hyper-parameter settings in the 'Model Setttings' table. Repeat the process to include all the supervised learning models you want to investigate.

14.7.7 Covariate Settings

We have defined a set of *standard* covariates that can be extracted from the observational data in the OMOP CDM format. In the covariate settings view, it is possible to select which of the standard covariates to include. It is possible to add many different types of covariate settings.

To add a covariate setting into the study, click on the blue '+ Add Covariate Settings' button (P in Figure 14.21). This will take you into the covariate setting view:

The *standard* OHDSI covariates includes indicator covariates corresponding to any concept id that is recorded in the database. The indicator covariates are binary and indicate whether a patient had a concept id recorded during some time interval relative to the target cohort start date. The user can specify up to three time intervals, longterm, mediumterm and shortterm in addition to using anytime prior. There is also the option of whether to include the target cohort start date.

Although the *standard* OHDSI covariates iclude 4 time intervals (all time prior, longterm, mediumterm and short term) and all concept ids, generally only a subset of these covariates will be chosen. The concept ids can be restricted by OHDSI vocabulary domain (condition, drug, procedure, measurement and observation). Generally, a user will select one or two time intervals and some of the domains. For example, if a user selects long term (using default set to 365 days prior) conditions and drugs and anytime prior measurements with end days set to 0, then there could be covariates for any condition or drug concept id record 365 days prior to and up to the cohort start day for any patient in the target cohort and covariates for any measurement concept id recorded on the cohort start day or anytime prior.

Age group and gender are also binary covariates, with age group covariates for every 5 years (0-4,

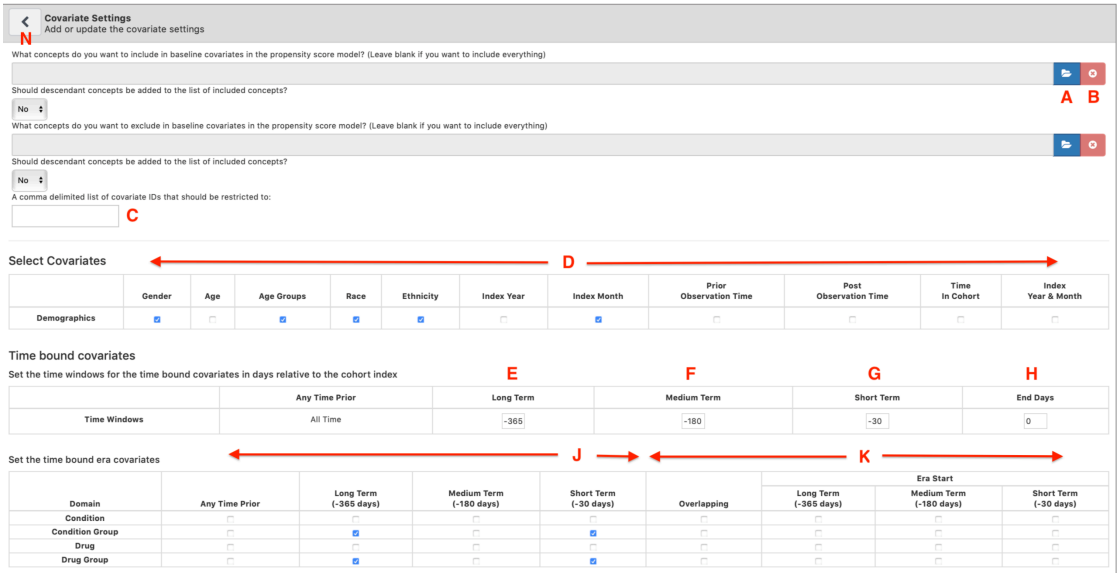


Figure 14.28: The covariate settings view part 1

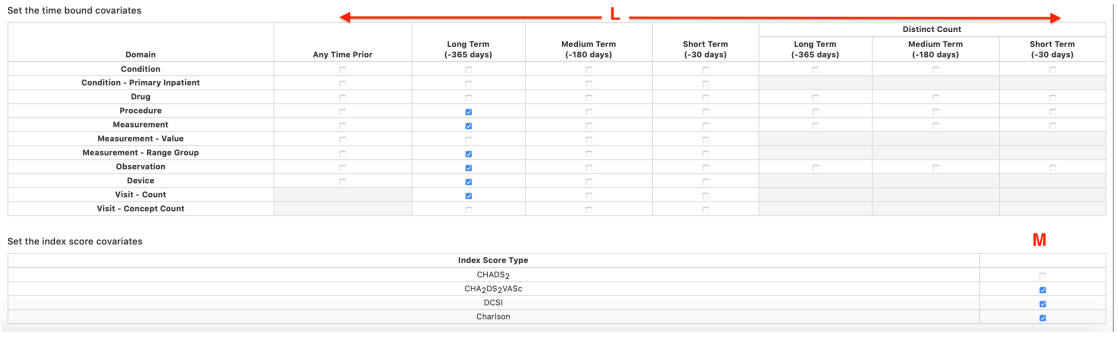


Figure 14.29: The covariate settings view part 2

5-9, 10-14, …, 95+).

Non binary covariates include age, domain counts, such as the number of condition concept ids that were recorded for each time interval per patient or the number of inpatient visits a patient had during the time interval. Measurement covaraites can be binary (indicating a measurement was taken or whether it was abnormal) or non-binary (the value of the measurement). Existing risk scores can also be chosen.

Include/Exclude options

The first part is of the covariate settings is the exclude/include option, see A Figure 14.28. Previously we mentioned that covariates are generally constructed for any concept id in the chosen time intervals and domains. However, you may be in a situation where you only want to include certain concept ids or you may want to exlcude concept ids (e.g., if the concept id 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 the blue button with a folder icon (see A in Figure 14.28). This will then open up a table with all the concept sets, select the one you want. You can include the concept ids in the concept set and all descendants by select 'yes' to the "Should descendant concepts be added to the list of included concepts?" option. This option will mean after you select the covariates you want, only covariates corresponding to these included concept ids will be included.

The same process can be repeated for the "What concepts do you want to exclude in baseline covariates in the patient-level prediction model? (Leave blank if you want to include everything)" but this will mean after you select the covariates you want, any covariates corresponding to these concept ids will be removed.

To remove any include/exclude setting, click on the red button with an X (see B in Figure 14.28).

The final option "A comma delimited list of covariate IDs that should be restricted to:" (see C in Figure 14.28) enables you to add a set of covariate ids (rather than concept ids) comma seperated that will only be included in the model. For example if you wanted covariate ids 340504504 and 8373747504 then you would type "340504504,8373747504" into the text box. You must ensure the domain/time interval corresponding to these covariates are selected below.

Non time bound options

The next section enables the selection of non-time bound variables (see D in Figure 14.28).

- 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: [Not recommended for prediction] a binary variable for each cohort start date year, 1 means that was the patients cohort start date year, 0 otherwise
- Index month - a binary variable for each cohort start date month, 1 means that was the patients cohort start date month, 0 otherwise
- 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

To include any of these variables, click the corresponding unticked box to add a tick (clicking a ticked box will remove the variable).

Time interval options

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]

These settings can be input into the text boxes to update them at E-H in Figure 14.28.

Domain covariates

The next option is the covariates extracted from the era tables (see J in Figure 14.28):

- Condition: Construct covariates for each condition concept id and time interval selected and 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 ths 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 ths drug era table, the covariate value is 1, otherwise 0.

Click on a box with no tick to add a tick and select that covariate into the covariate settings. Clicking

on a box with a tick with untick it and remove that covariate from the covaraite settings.

[need to check this] Overlapping time inverval setting means you want the drug or condition to 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. These options are at K in Figure 14.28.

The domain tables covariates enable you to pick whether to include covariates corresponding to concept ids in each domain for the various time intervals (see L in Figure 14.29):

- 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: ?
- 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: ?
- 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 counds the number of records per domain and time interval [expand].

Risk score covariates

The final option is whether to include commonly used risk scores as covariate, M in Figure 14.29.

Once happy with the covariate settings, click the '<' button (see N Figure 14.28) on the top left corner to return to the main prediction settings. Your covariate options you picked will now show in

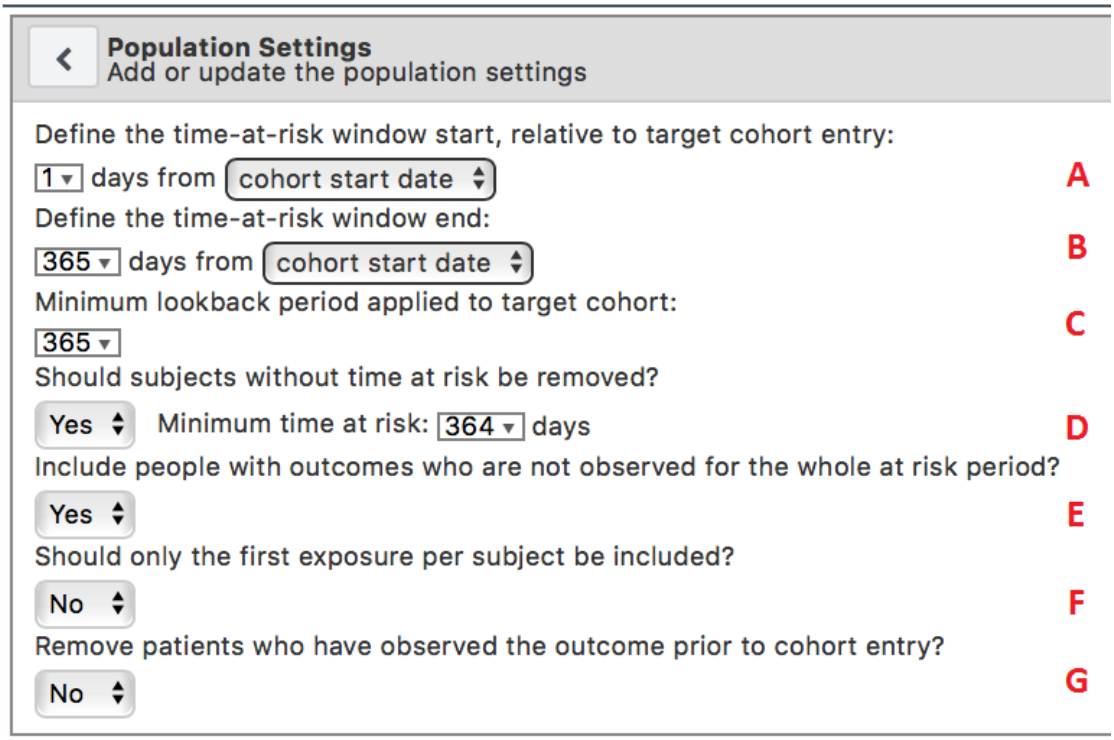


Figure 14.30: The population setting options

the covaraite settings table. You can edit an exisitng setting by clicking on the corresponding row or add more covariate settings by clicking on the blue '+ Add Covariate Settings' button again.

14.7.8 Population Settings

The population settings is where addition inclusion criteria can be applied to the target population (this may be useful for sensitivity investigations) and is also where the time-at-risk is defined. To add a population setting into the study, click on the blue '+ Add Population Settings' button (Q in Figure 14.21).

This will open up the population setting view containing various setting to define, see Figure 14.30.

The first set of options, A and B, enable the user to specify the time-at-risk period. This is a time interval relative to the target cohort dates 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 class them as 'outcome', otherwise they are classed as 'non-outcome'.

The first option labelled with a red A is: "Define the time-at-risk window start, relative to target cohort entry:" - this settings lets you define the start of the time-at-risk. It is relative to the target cohort dates (cohort start date or cohort end date). You can pick an offset corresponding to the number of days and whether it is relative to the target cohort start date or the target cohort end date.

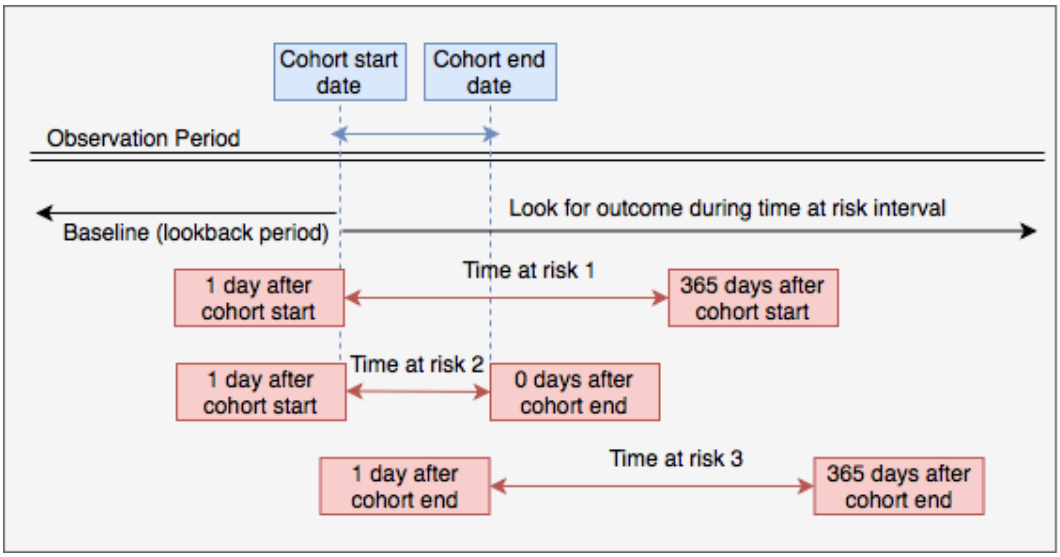


Figure 14.31: How the population setting options define the time-at-risk

The second option labelled with a red B is: "Define the time-at-risk window end:" - this settings lets you define the end of the time-at-risk. It is relative to the target cohort dates (cohort start date or cohort end date). You can pick an offset corresponding to the number of days and whether it is relative to the target cohort start date or the target cohort end date.

See Figure 14.31 for an illustration of how these settings define the time-at-risk period:

The next option, marked by the red C, is "Minimum lookback period applied to target cohort:". This is where you can specify the minimum baseline period, specifically the minimum number of days prior to the cohort start date that a patient has been continuously observed. The default is 365 days. Expanding the minimum lookback will give a more complete picture of a patient (as they must have been observed for longer) but will filter many patienst (who do not have the minimum number of days prior observation).

The option maked by the red D is "Should subjects without time at risk be removed?". If this is set to yes, then a value for "Minimum time at risk:" is also required. This option lets you deal with people who are lost to follow-up (e.g., they leave the database during the time-at-risk period). If you select 'yes' then you need to specify the minimum time a patient needs to be in the time-at-risk period for them to be included in the labelled data (if they do not have the minimum time they are excluded from the population). 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 you only want to include patients who are observed the whole interval, then set the minimum time at risk to be 364. If you are happy as long as people are in the time-at-risk for the first 100 days, then 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 you set "Should subjects without time at risk be removed?" to 'No', then this will keep every patient, even those who drop out from the database during the time-at-risk.

The option E "Include people with outcomes who are not observed for the whole at risk period?" is also linked to D. This option lets you treat people with the outcome who drop out of the database during time-at-risk differently to those who do not have the outcome observed before dropping out. If "Include people with outcomes who are not observed for the whole at risk period?" is set to 'No', then people who are not observed for the whole time-at-risk are include/excluded depending on your settings for D. However, if "Include people with outcomes who are not observed for the whole at risk period?" is set to 'Yes', then this means people who have the outcome recorded during the time-at-risk interval are included in the labelled data even if they drop out from the database before the end of the time-at-risk interval.

The option "Should only the first exposure per subject be included?" labelled in F is only useful if you have a target cohort that contains patients multiple times but with different cohort start dates. In this situation, picking 'yes' for "Should only the first exposure per subject be included?" will result in only keeping the earliest target cohort date per patient in the analysis (i.e., unique patients); otherwise a patient can be in the labbelled dataset multiple times but the covariates and time-at-risk will be at different time points in the patients observation.

The final option G is "Remove patients who have observed the outcome prior to cohort entry?". Selecting 'Yes' to this option will remove patients who have the outcome prior to the time-at-risk start date, so the model is in patients who have never experience the outcome prior. If 'No' is selected, then patients could have had the outcome prior. Generally, having the outcome prior is very predictive of having the outcome during the time-at-risk.

Once you are happy with the population settings, click on the grey '<' button in the top left and this will return you to the main setting view. You will now see your population settings as a new row in the population settings table. To edit the settings click on the corresponding row. This will take you to the population setting view where you can change any of the settings.

To add more population settings, repeat the process detailed in this section.

14.7.9 Execution settings

Execution settings (R in Figure 14.21) determine whether to use sampling, how to manage rare events, and whether to normalize covariates. Sampling can be an efficient means to determine if a model for a large population (i.e. 10 million patients) is accurate, by creating and testing the model with a subgroup of patients (e.g. if AUC is close to 0.5 on your sampling, you might abandon the model). The user specifies the size of the subgroup to be sampled. A minimum threshold value for covariate occurrence is necessary to remove rare events that are not representative of the overall population. Normalization of the covariates is usually necessary for successful implementation of a LASSO model.

There are three options:

- "Perform sampling": here you can choosen whether to perform sampling (default = 'No'). If you set this to 'yes', another option will appear "How many patients to use for a subset?", here you can add the sample size you wish to extract.

- "Minimum covariate occurrence: If a covariate occurs in a fraction of the target population less than this value, it will be removed:": here you can choose then minimum covariate occurrence (default = 0.001)
- "Normalize covariate": here you can choose whether to normalize covariates (default = 'Yes)

14.7.10 Training settings

Training settings (S in Figure 14.21) determine how to distribute the data between training and testing groups. Most of the data will be used to train the model and the rest will be used to test it. The data can be divided by either unit person or time. The percentage of data attributed to training or testing the model is specified by the user. Additionally, the number of folds for cross-validation is specified, which partitions the training data for hyper-parametric analysis. The user has the option of specifying the seed used to split the training and testing data for consistent distribution of the outcomes between the groups. This option is only needed for person based splitting.

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 (default = 3)
- "The seed used to split the test/train set when using a person type testSplit (optional):": Select the seed used to split the train/test set when using a person type test split

14.7.11 Atlas Utilities Tab

The Utilities tab (H in Figure 14.20) is where a user can review the prediction study (once minimum required settings are defined), export/import existing atlas prediction studies and download the prediction study R package.

Review and Download Tab

If you have not completed all pre-requisites needed to run the study, you will see the same as Figure 14.32.

Assuming your study contains all necessary components, you will see Figure 14.33, showing the tabs Full Analysis List, Prediction Problem Settings, and Analysis Settings.

Clicking on the Prediction Problem Settings (see Y in Figure 14.33) will show all the combinations of the Target Cohort and Outcome Cohort names specfied in the analysis.

Finally, clicking on the Analysis Settings tab (see Z in Figure 14.33) shows a table allowing you to review all of the Model Names, Model Settings, Covariate Settings, Risk Window Start and Risk Window End combinations.

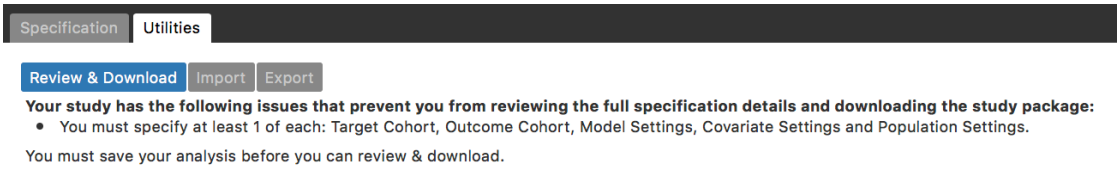


Figure 14.32: Reviewing when insufficient design

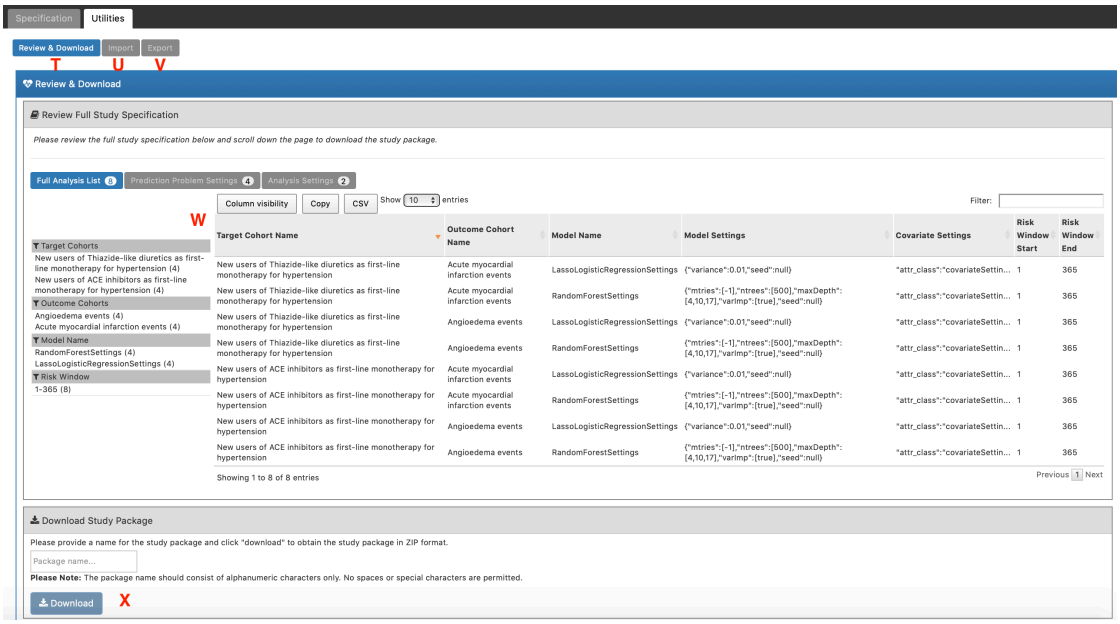


Figure 14.33: The utilities tab reviewing valid study

14.7.12 How to import/export study

To export a study, click on the Export tab under utilities (see V in Figure 14.33). ATLAS will produce JSON file that can be directly copied and pasted into a file that contains all of the data (study name, cohort definitions, models selected, covariates, settings, etc.) needed to run the study. This is displayed in Figure 14.34.

To import a study, first go back to the main ATLAS menu and click on Prediction. Click on the New Patient Level Prediction button, give your study a name, and Save. Next, click on the Utilities tab, then the Import tab (see U in Figure 14.33). Paste the contents of a Patient Level Prediction JSON file into this window, then click on the Import button below the other tab buttons.

14.7.13 How to download package

The Download Study button is available at the bottom of the Utilities screen (see X in Figure 14.33). 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.

ATLAS will generate an R package for the study, see Figure 14.35.

14.7.14 Building Atlas created prediction study R package

Setting up R

To run the atlas generated prediction R package study requires having R studio () installed, the devtools R package [in R run: install.packages('devtools')] and the OHDSI PatientLevelPrediction package installed (see …).

Unzipping atlas compressed folder

Atlas generates a zipped directory containing the R package. This zipped directory needs to be extracted. Once extracted the directory will look like:

Opening package project in R

The easiest way to open the atlas created package in R is to double click on the project file:

This will then open a new R studio session:

Building project

Once R studio has opened the project, you can then build the package by clicking on the 'build' option in the top right hand side:

If you find a message like (but with the text in red matching the name you called your study):

Your package has now been created and will be available to run. If you have a message with an error then there was an issue with building the package and the package did not get built. Common issues

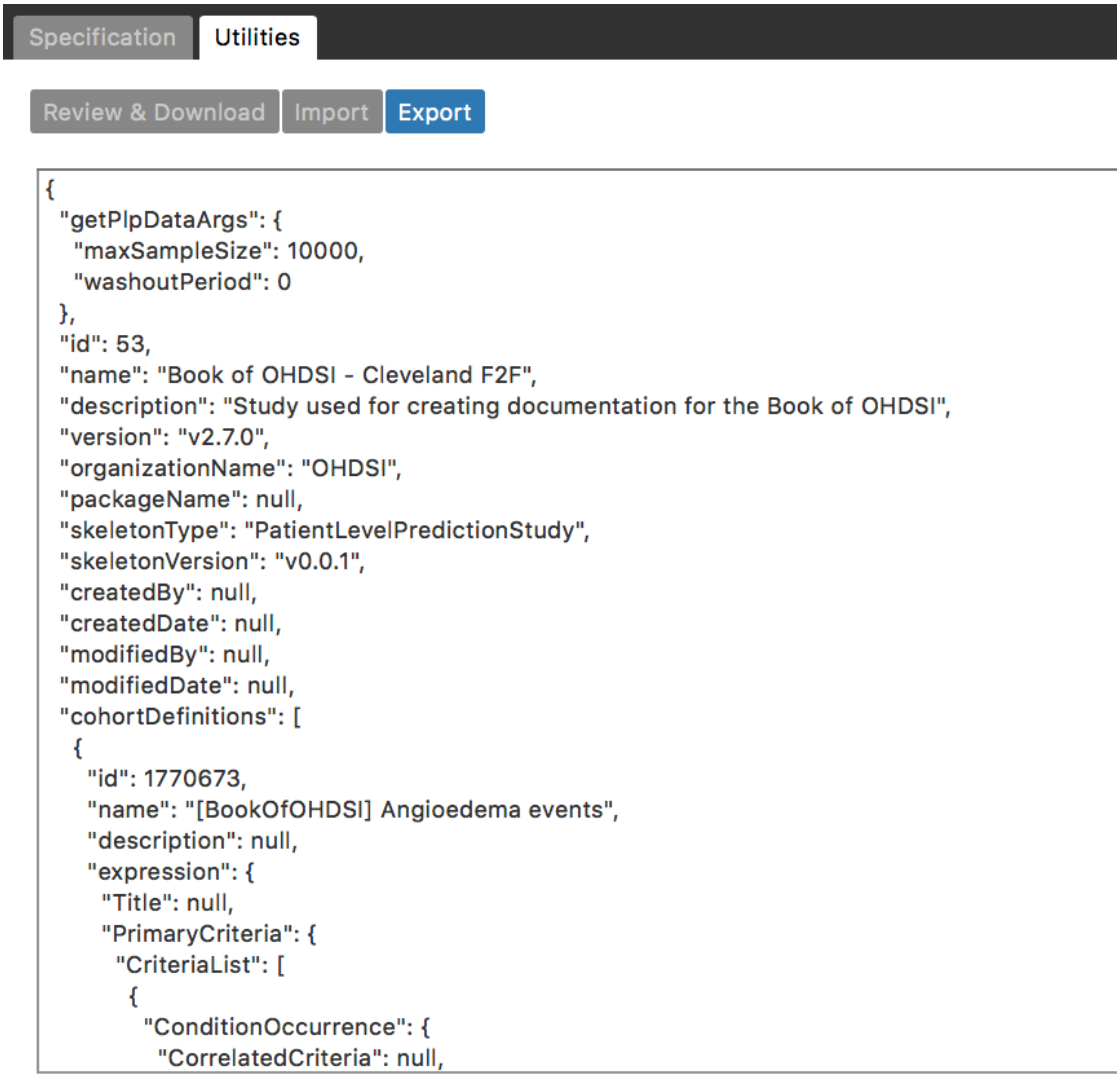


Figure 14.34: Exporting a study design

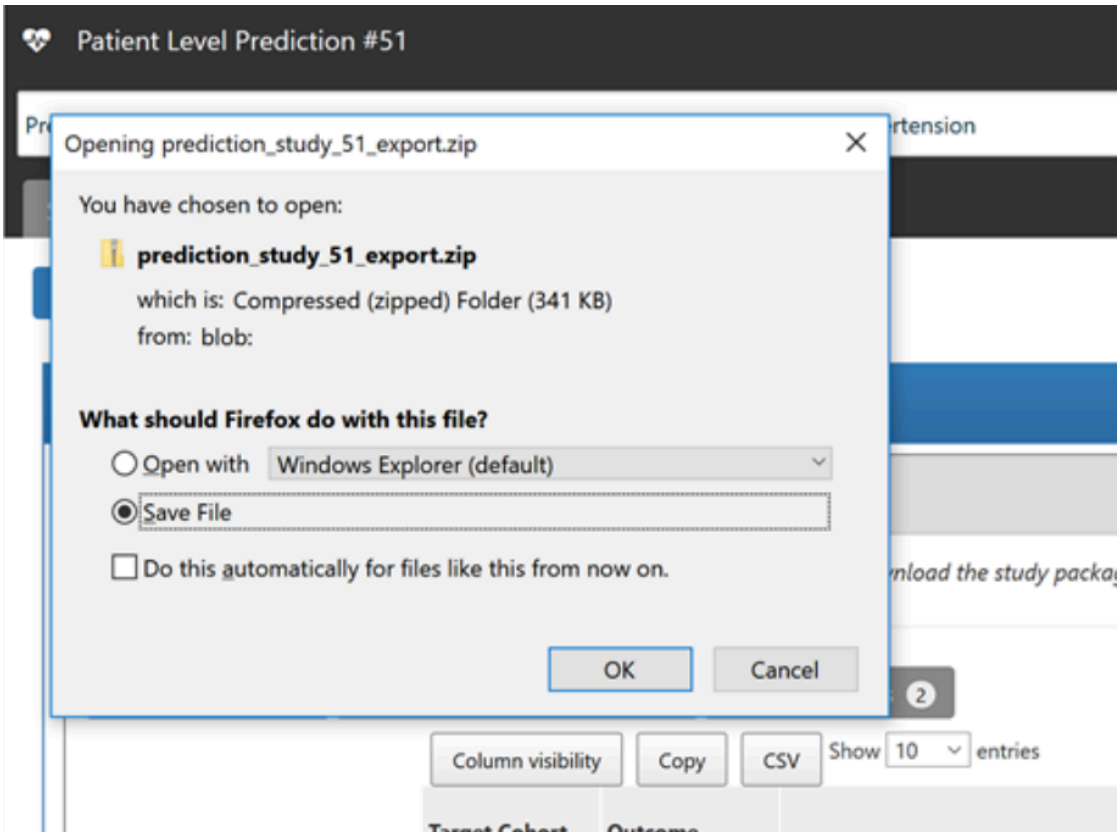


Figure 14.35: The downloaded study design R package

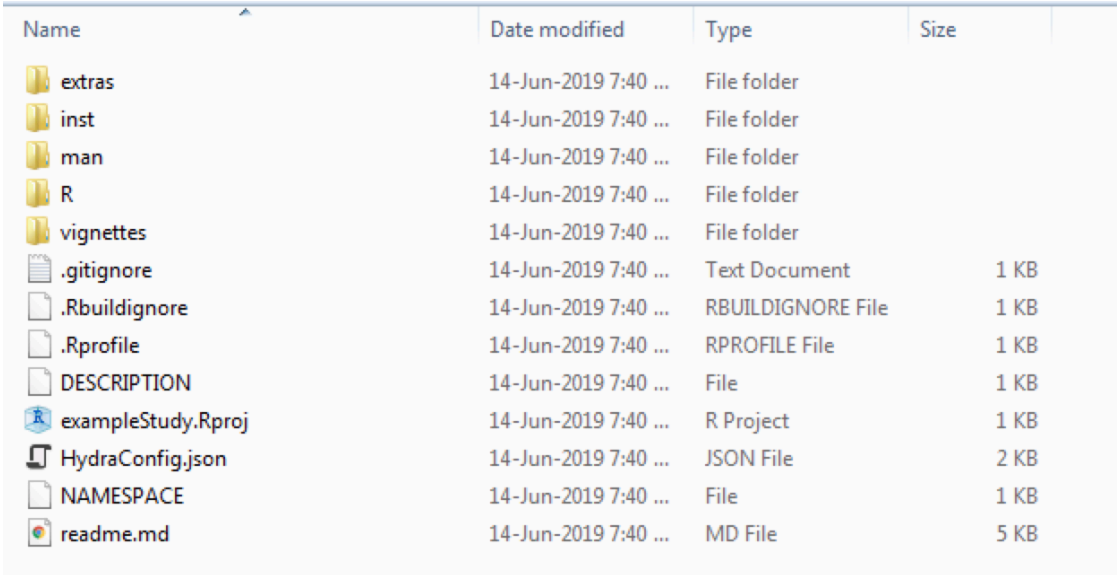


Figure 14.36: The directory of study design R package

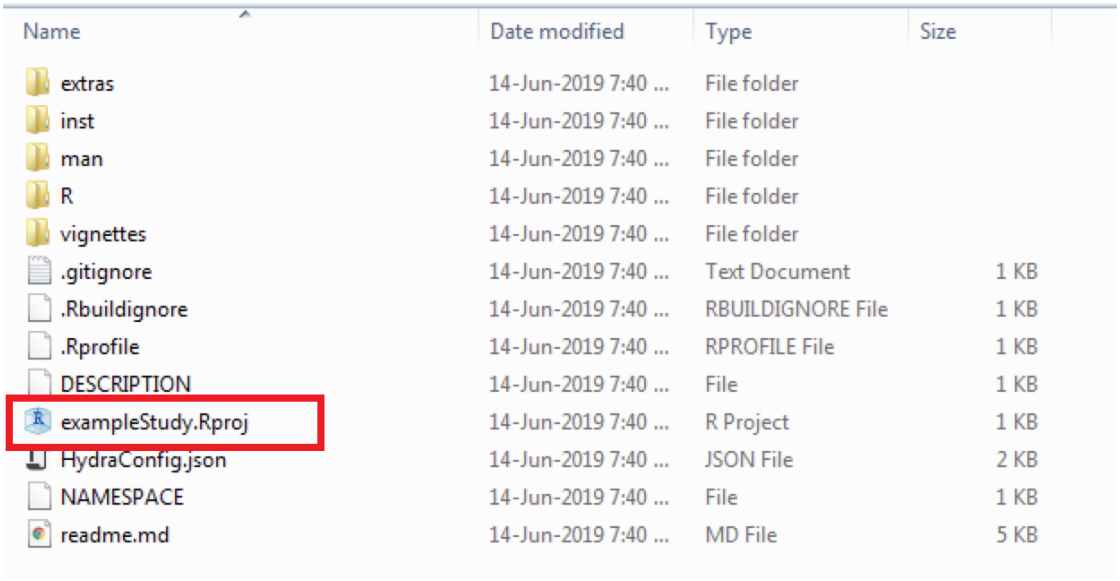


Figure 14.37: Opening the study design R package

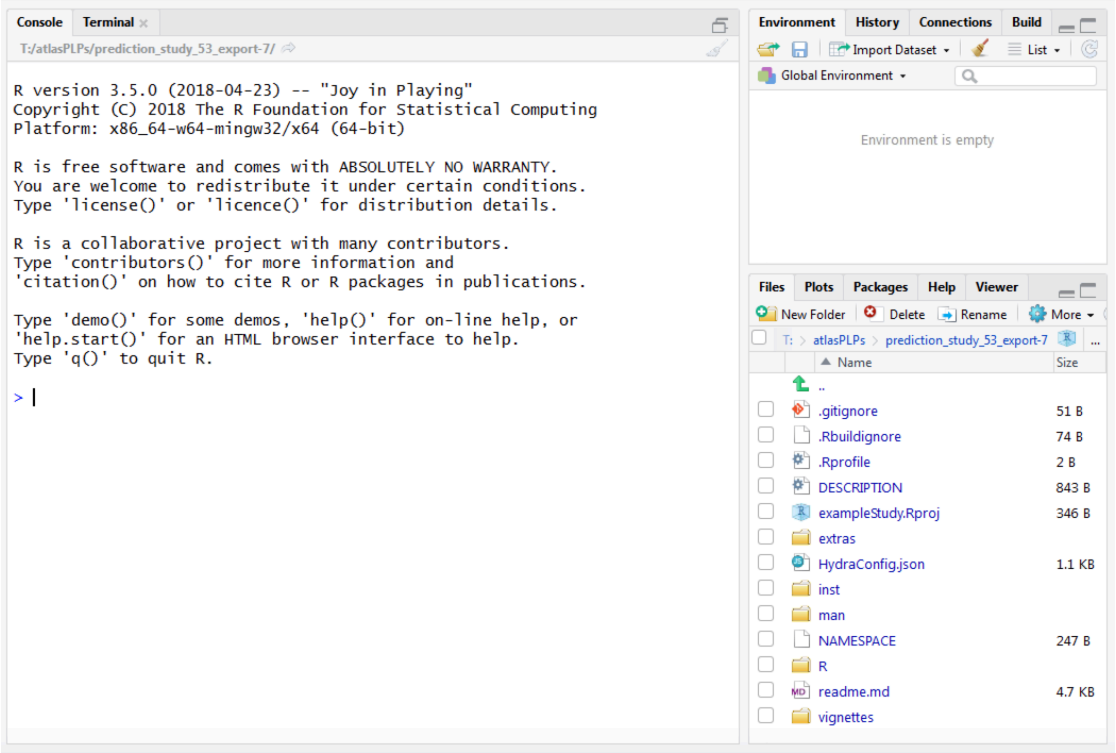


Figure 14.38: Rstudio open with the study design project

Figure 14.39: Building the R project into a local R library

```
** building package indices
** installing vignettes
** testing if installed package can be loaded
* DONE (YourStudyName)
In R CMD INSTALL
```

Figure 14.40: Building the R project completed

causing the build to fail are missing dependencies, to find out the R packages required for your built, open the 'DESCRIPTION' file in the main directory:

This will open up in R studio and show what R packages are required (the packages in the Imports section)

If you do not have any of the packages listed in 'Imports:' then you will need to install them before building the atlas generated package.

14.7.15 Running Study

Readme and extras/codetorun.R

The key file in the atlas generated package directory is the one that contains code for running the study, the CodeToRun.R file found in the extras directory:

We recommend opening the file CodeToRun.R

CodeToRun.R Settings

The final step to running the study is to connect to the database through R and specify where the results should be saved.,

The CodeToRun.R file looks like:

The inputs for the CodeToRun file are:

- outputFolder: This is a string specifying where in your computer to save the results. This location needs to have sufficient space as data will be extracted from the database into this location and the lcoation must have read/write access.
- options(fftempdir = ''): this is a location in your computer that must have read/write access and large amounts of space. It will be used to store temporary data.
- dbms: The database management system you use

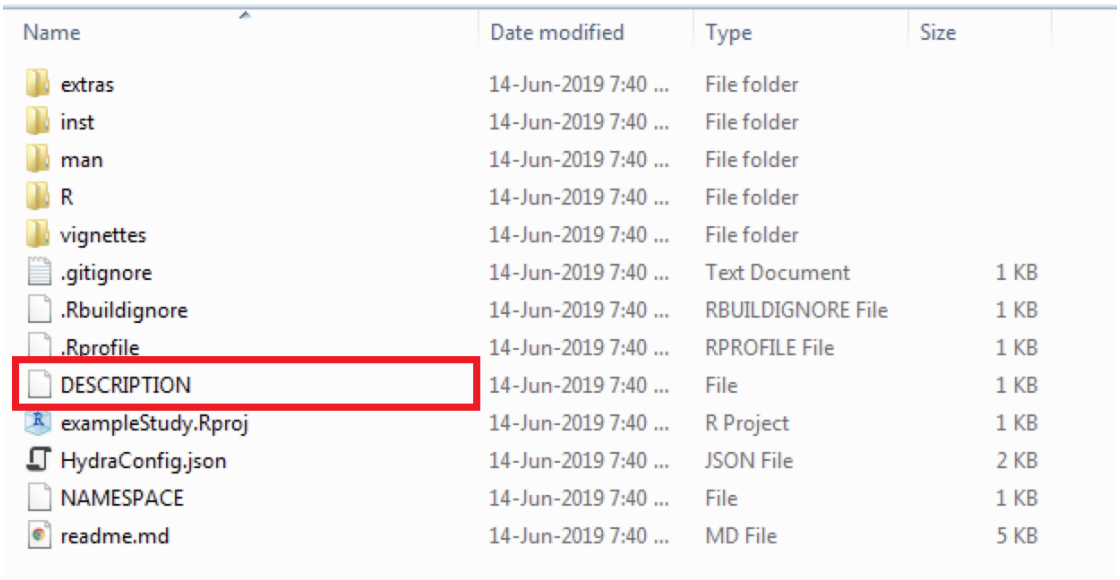


Figure 14.41: Finding the DECSRIPTION file

- user: Your username for the database connection (contact database administrator if unknown)
- pw: Your password for the database connection (contact database administrator if unknown)
- server: a string specifying the database server (contact database administrator if unknown)
- port: (optional) the port number (contact database administrator if unknown)
- cdmDatabaseSchema: a string specifying the database schema containing the OMOP CDM instance
- cohortDatabaseSchema: a string specifying the database schema either containing the cohorts or where to create the cohorts.
- oracleTempSchema: if using oracle, this is your temp database schema
- cohortTable: the name of the cohort table (if using atlas cohorts then this will be 'cohort')

Once the settings are filled out, the final step is to pick what parts of the study to execute:

The following options specify:

- A: createProtocol - set to 'True' if you want to create a word document protocol template that automatically inserts the study design settings. This can be shared if creating a network study.
- B: createCohorts - do you need to create the cohorts for this study? If you are using atlas cohorts you can set this to 'False' otherwise set this to 'True' and the cohorts you picked for the study will all be generated.
- C: runAnalyses - setting this to 'True' will result in models being developed and evaluated for each setting you specified in the study design. This requires cohorts to have been generated (in atlas or using B createCohorts set to 'True').
- D: createResultsDoc - if you set A: createProtocol to 'True' and generated a protocol and also ran the analysis by setting C: runAnalyses to 'True' then you can add the results into the protocol to create a word document with the protocol and results.
- E: packageResults - if you set C: runAnalyses to 'True' and have results, you can set pack-

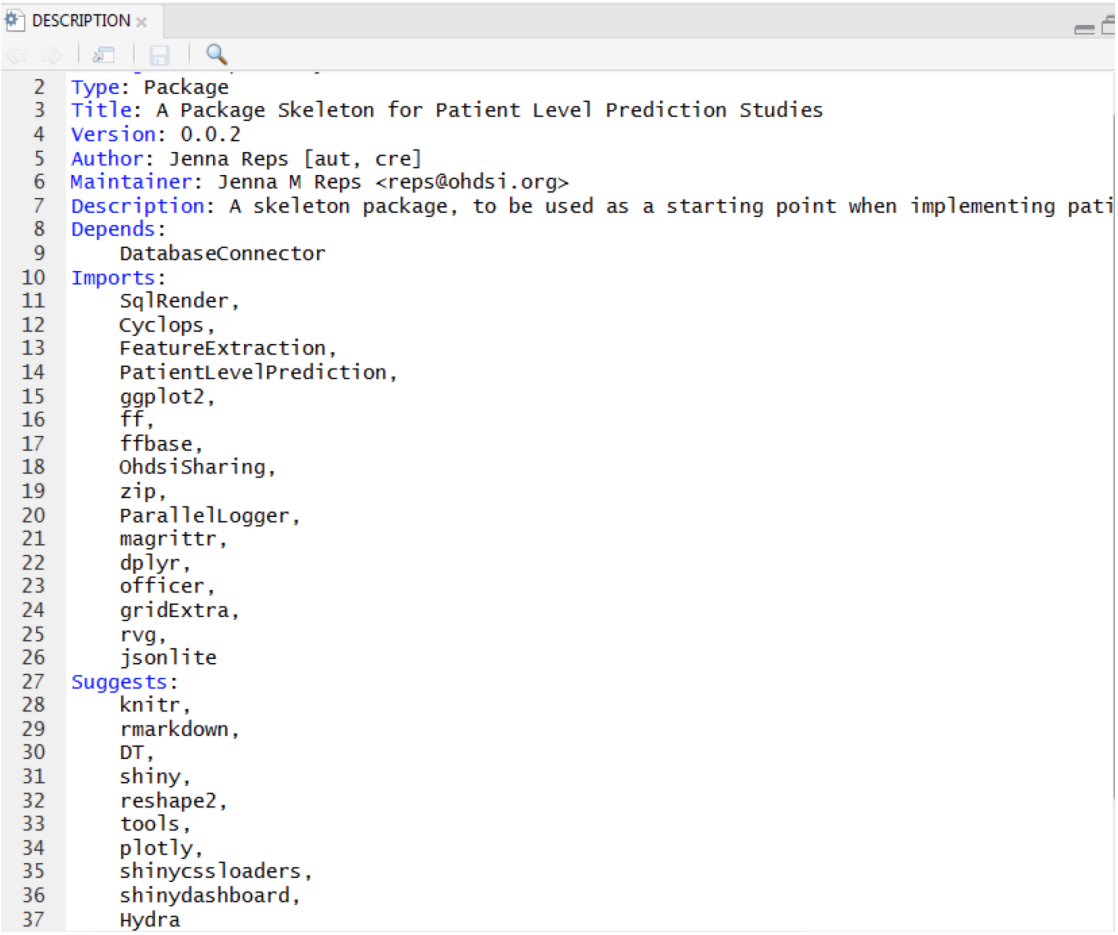


Figure 14.42: The DECSRIPTION file content

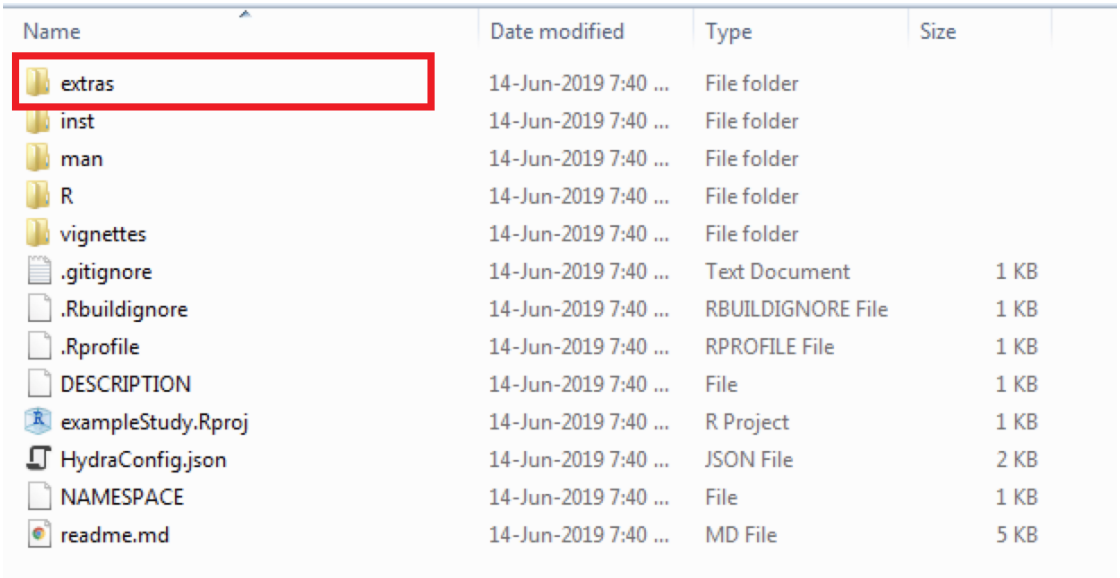


Figure 14.43: The CodeToRun.R file is in the extras folder

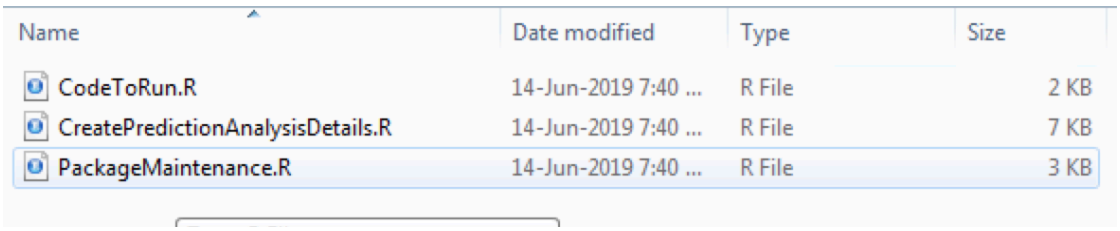


Figure 14.44: The CodeToRun.R file

DESCRIPTION × CodeToRun.R ×

Source on Save Run Source

```r
library(exampleStudy)
# USER INPUTS
#=======================
# The folder where the study intermediate and result files will be written:
outputFolder <- "./exampleStudyResults"

# Specify where the temporary files (used by the ff package) will be created:
options(fftempdir = "location with space to save big data")

# Details for connecting to the server:
dbms <- "you dbms"
user <- 'your username'
pw <- 'your password'
server <- 'your server'
port <- 'your port'

connectionDetails <- DatabaseConnector::createConnectionDetails(dbms = dbms,
                                                                server = server,
                                                                user = user,
                                                                password = pw,
                                                                port = port)

# Add the database containing the OMOP CDM data
cdmDatabaseSchema <- 'cdm database schema'
# Add a database with read/write access as this is where the cohorts will be generated
cohortDatabaseSchema <- 'work database schema'

oracleTempSchema <- NULL

# table name where the cohorts will be generated
cohortTable <- 'exampleStudyCohort'
#=======================

```

Figure 14.45: The CodeToRun.R default setting

```r
execute(connectionDetails = connectionDetails,
        cdmDatabaseSchema = cdmDatabaseSchema,
        cohortDatabaseSchema = cohortDatabaseSchema,
        cohortTable = cohortTable,
        outputFolder = outputFolder,
      A createProtocol = F,
      B createCohorts = F,
      C runAnalyses = F,
      D createResultsDoc = F,
      E packageResults = F,
      F createValidationPackage = F,
      G minCellCount= 5)

```

Figure 14.46: Executing the study

ageResults to 'True' to create a zipped folder containing your results with any sensitive data removed. This can be easily shared with other OHDSI colabortors.

- F: createValidationPackage - if a model seems to do, we can use this option to create a new R package for validating the model. Set to 'True' to create a validation package containing all the models for external validation. In later Atlas versions there is another input where you can specify the analysis id of a model rather than validating all models.
- G: minCellCount - this is linked to E: packageResults and F: createValidationPackage and specifies the minimum cell count for any result to be included when sharing the models. For example, if the minCellCount is 5, then any count with a value less than 5 will be removed.

Viewing the Results

After running the R package analysis you can view the results in an interactive shiny app by running:

```
PatientLevelPrediction::viewMultiplePlp(outputFolder)
```

14.8 Implementing the study in R

Now we have completely designed our study we have to implement the study in R. This will be done using the PatientLevelPrediction package to build patient-level predictive models. The package enables data extraction, model building, and model evaluation using data from databases that are translated into the OMOP CDM.

14.8.1 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.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.

14.8.2 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 washoutPeriod 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.

14.8.3 Additional inclusion criteria

To completely define the prediction problem the final study population is obtained by applying additional constraints on the two earlier defined cohorts, e.g., a minumim 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"
)
```

14.8.4 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 gradientBoostingMachine: 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 bestcross-validation performance will then be chosen for the final model. For our problem we choose to build a logistic regression model with the default hyper-parameters

```
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.

You can save the model using:

```
savePlpModel(gbmResults$model, dirPath = "model")
```

You can load the model using:

```
plpModel <- loadPlpModel("model")
```

You can also save the full results structure using:

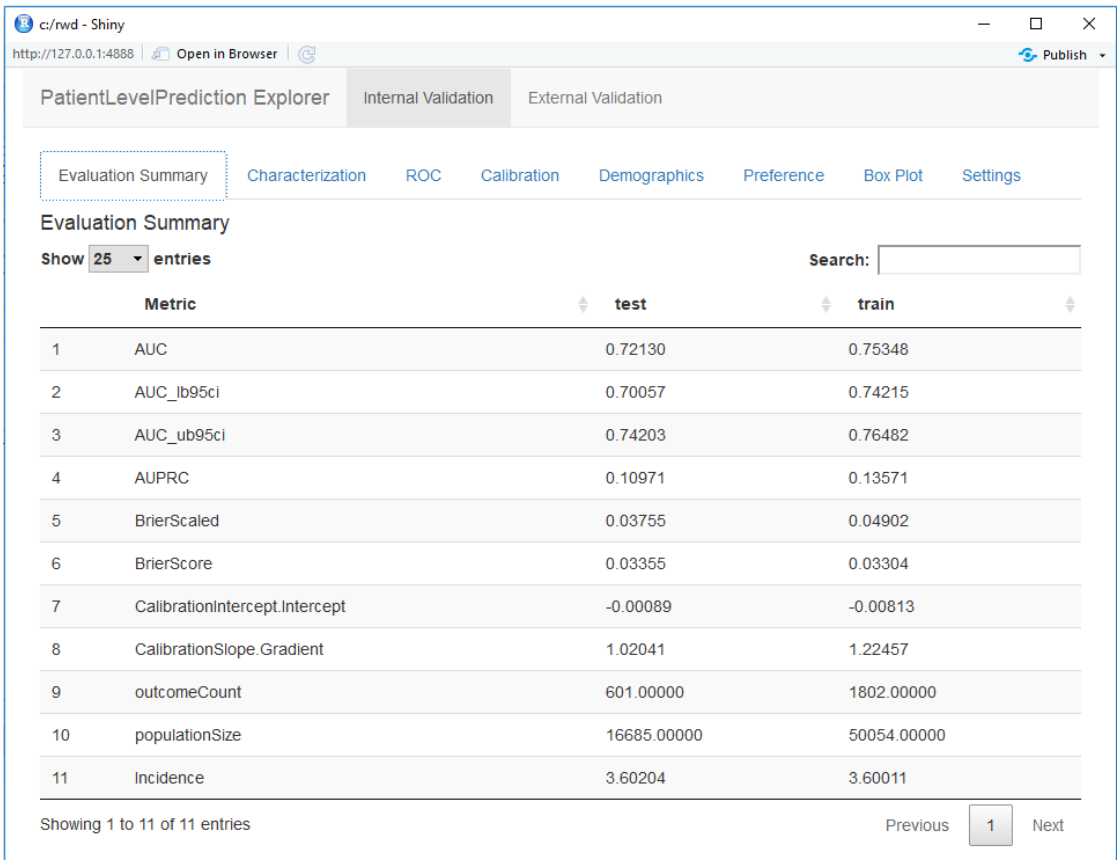
```
savePlpResult(gbmResults, location = "gbmResults")
```

To load the full results structure use:

```
gbmResults <- loadPlpResult("gbmResults")
```

14.8.5 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 your browser in which you can view all performance measures created by the framework, including interactive plots, as shown in Figure **??**.



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 here

14.8.5.0.1 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 peformed 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 you ran `result <- runPlp(...)` then you 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 you have multiple databases to validate the model on then you can run:

```
valResults <- externalValidatePlp(plpResult = result,
                      connectionDetails = connectionDetails,
                      validationSchemaTarget = list('mainschema.dob','difschema.dob', 'ano
                      validationSchemaOutcome = list('mainschema.dob','difschema.dob', 'an
                      validationSchemaCdm = list('cdms1chema.dbo','cdm2schema.dbo','cdm3sc
                      databaseNames = list('new database 1','new database 2','new database
                      validationTableTarget = list('cohort1','cohort2','cohort3'),
                      validationTableOutcome = list('cohort1','cohort2','cohort3'),
                      validationIdTarget = list(1,3,5),
                      validationIdOutcome = list(2,4,6)
                      )
```

14.9 Exploring a single PLP Shiny App

Exploring the performance of a single plp model is easiest with the `viewPlp()` function. This requires a plpResult as the input. If developing models in R you can use the result of `plpResult <- runPLp(...)` as the input. If using the Atlas generated study package, then you need to load one of the models (in this example we will load Analysis_1):

```
plpResult <- loadPlpResult(file.path(outputFolder, 'Analysis_1', 'plpResult'))
```

Analysis_1 corresponds to the lasso logistic regression for the prediction problem: **Withing new users of ACE inhibitors as first-line monotherapy for hypertension who will develop Acute myocardial infarction (AMI) events within a year.**

You can then launch the shiny app by running:

```
viewPlp(plpResult)
```

The shiny launches with a summary of the performance metrics on the test and train sets, see Figure 14.47. The results show that the AUROC on the train set was 0.78 and this dropped to 0.74 on the test set. The test set AUC is the more accurrate measure. Overal, the model appears to be able to discriminate those who will develop AMI in new users of ACE inhibitors but it slightly over fit as the performance on the train set is higher than the test set. The ROC plot is presented in Figure 14.48.

The calibration plot in Figure 14.49 shows that generally the observed risk matches the predicted risk as the dots are around the diagonal line. The demographic calibration plot in Figure 14.50 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 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 14.51. 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.

14.10 Exploring the Atlas PLP Shiny App

To view the atlas generated analysis results via an interactive shiny app, run: `PatientLevelPrediction::vi`
where *outputFolder* is the directory path containing the analysis results (e.g., 'C:/atlasResults/Example'),
it will look like:

The interactive shiny app will start at the summary page:

This summary page table contains:

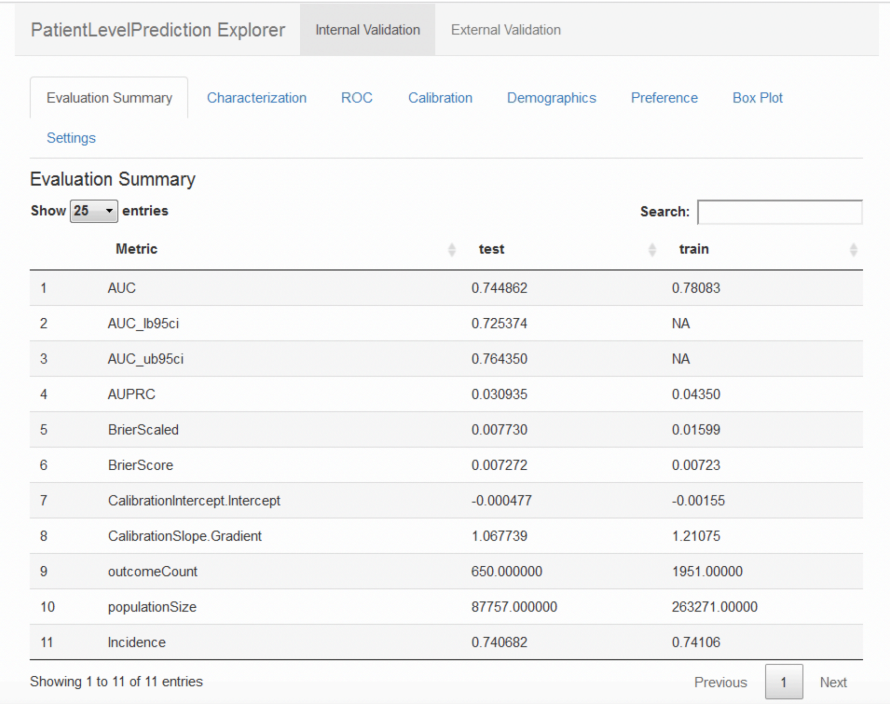


Figure 14.47: The starting summary page of the viewPlp() shiny app

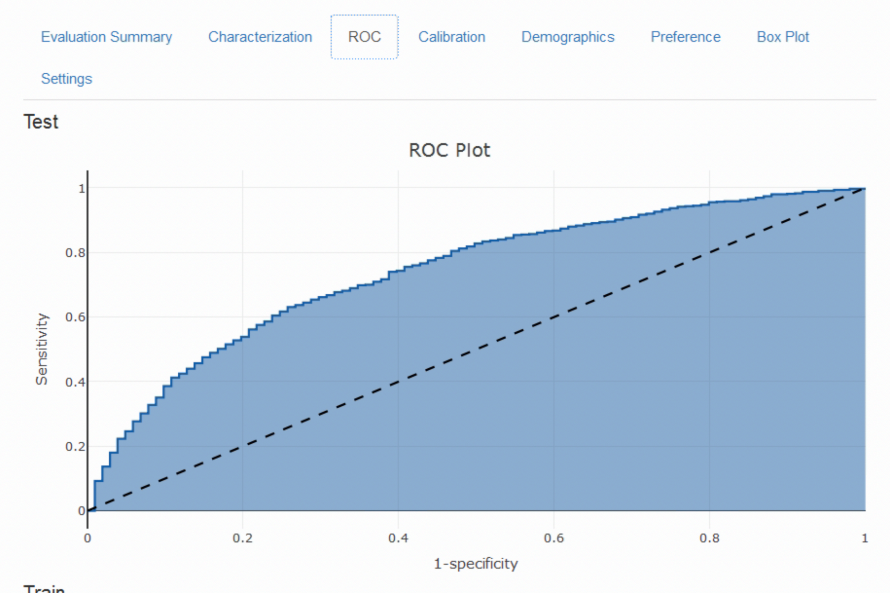


Figure 14.48: The ROC plot for predicting AMI within a year in new users of ACE inhibitors

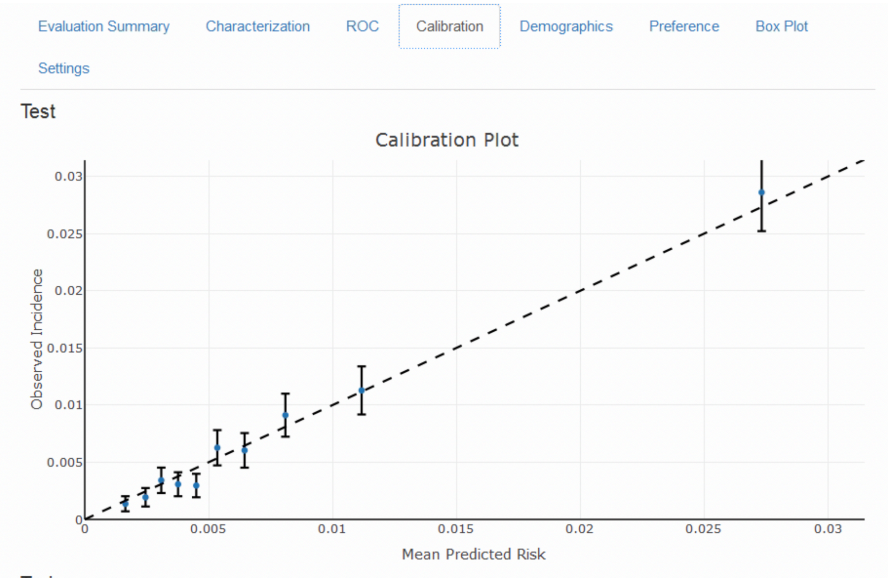


Figure 14.49: The calibration of the model

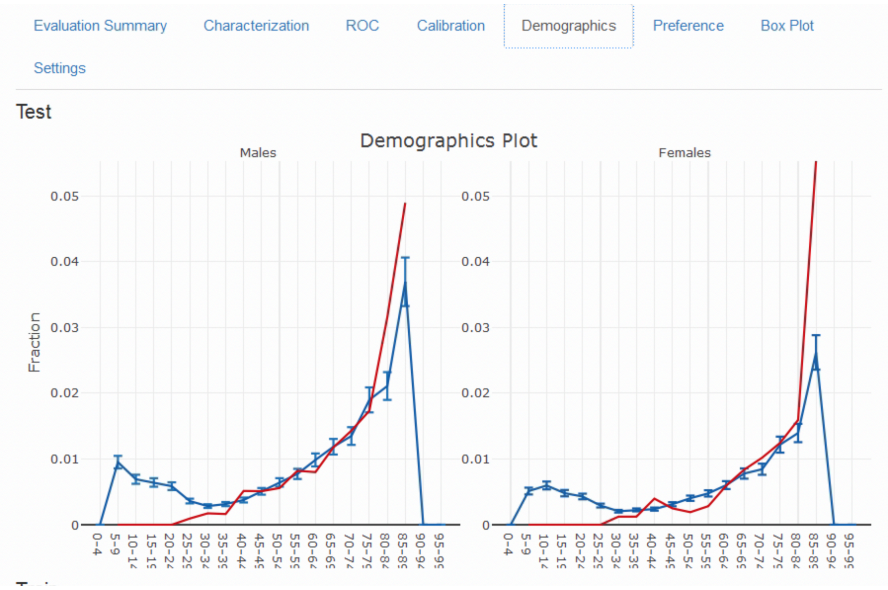


Figure 14.50: The demographic calibration of the model

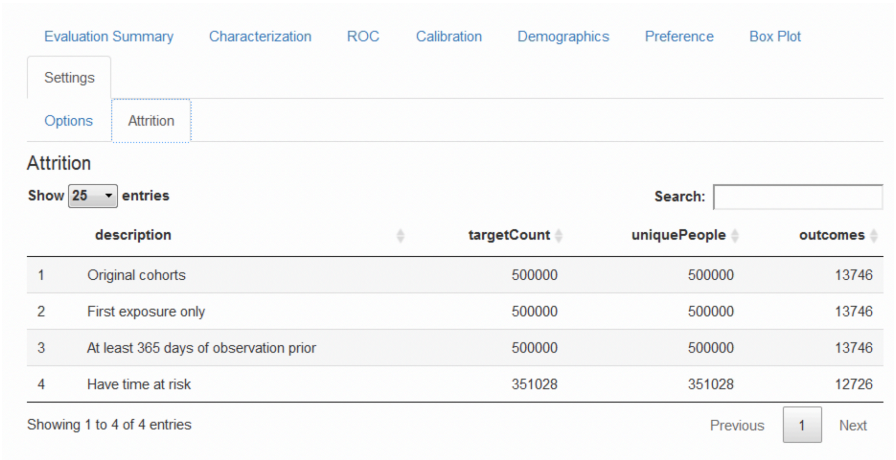


Figure 14.51: The attrition plot for the prediction problem

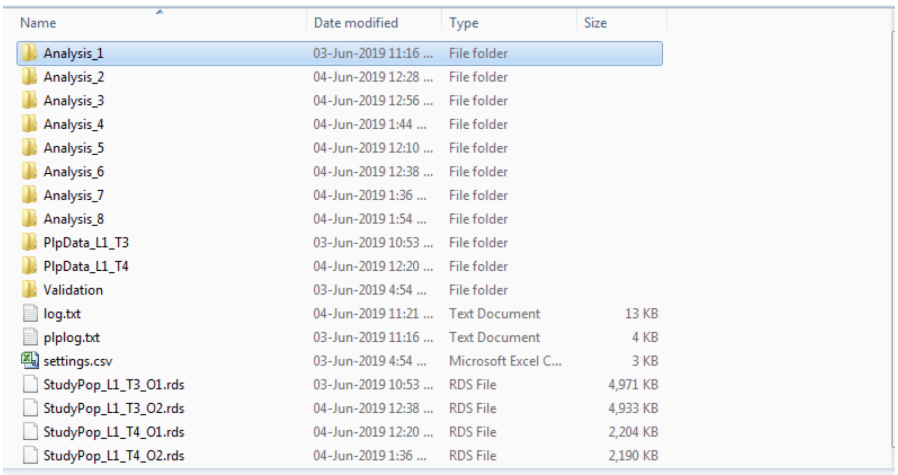


Figure 14.52: The directory where the atlas models and results were saved

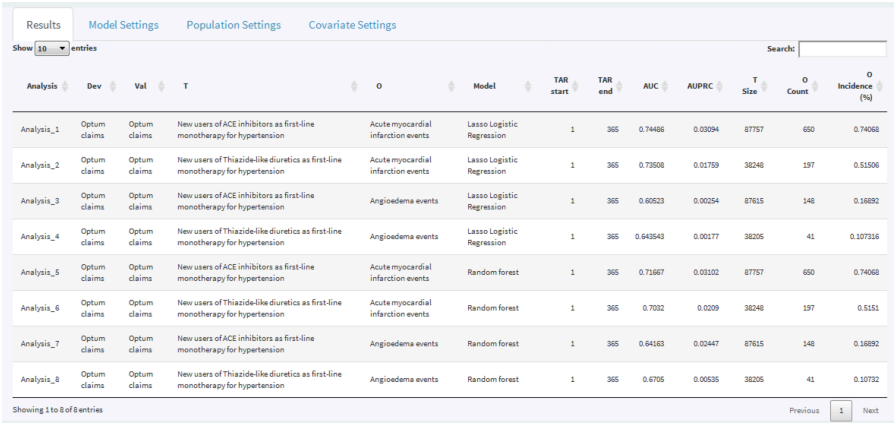


Figure 14.53: The shiny summary page containing key hold out set performance metrics for each model trained

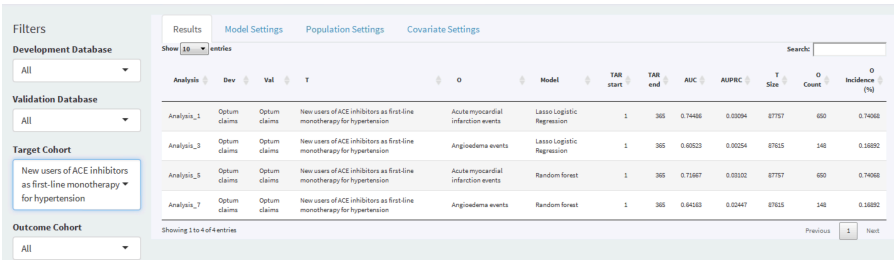


Figure 14.54: Demonstration of the filter option

- 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:

Here a user can specify the development/valdiation 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. To unselect simply click on the selected row again or select a new row.

With a row selected, you can now explore the model settings used when developing the model by clicking on the *Model Settings* tab:

To explore the population settings, click on the *Population Settings* tab to display the settings used when developing the model:

Simialrly, to explore the covariates settings, click on the *Covariate Settings* tab to display which

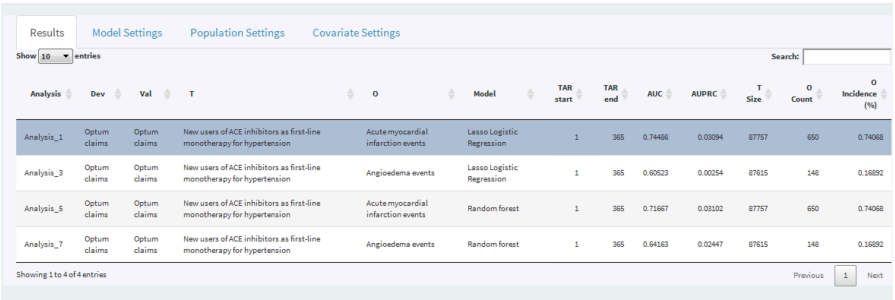


Figure 14.55: The highlighted row shows a selected model. We can then use other tab to explore the settings and results for the selected model

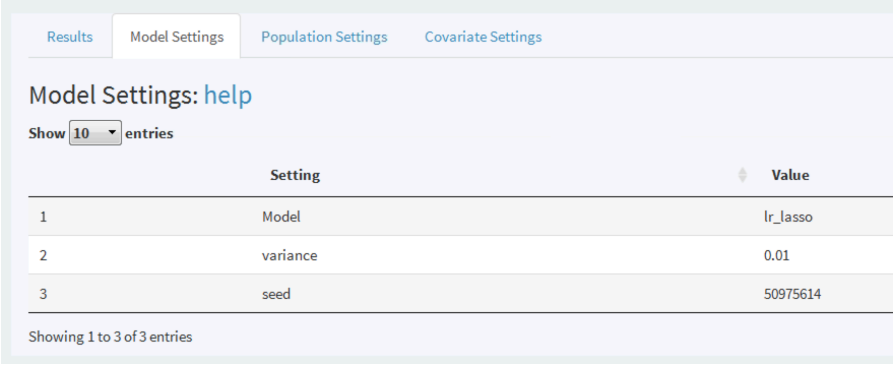


Figure 14.56: To view the model settings used when developing the model.

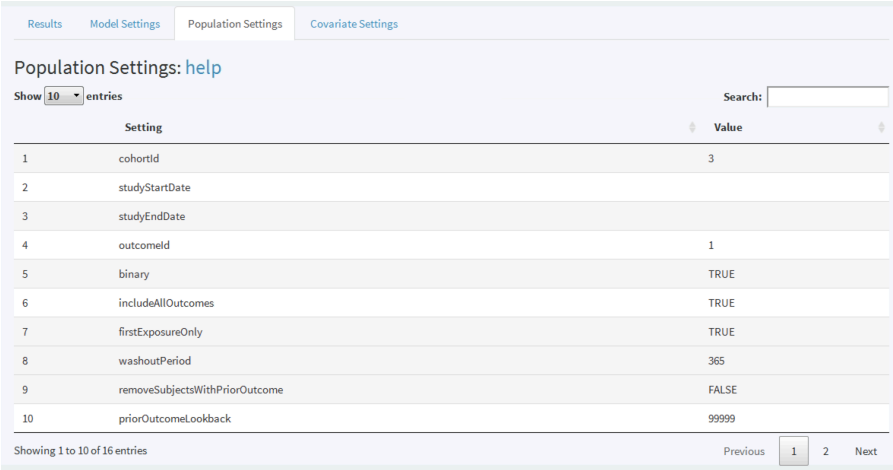


Figure 14.57: To view the model settings used when developing the model.

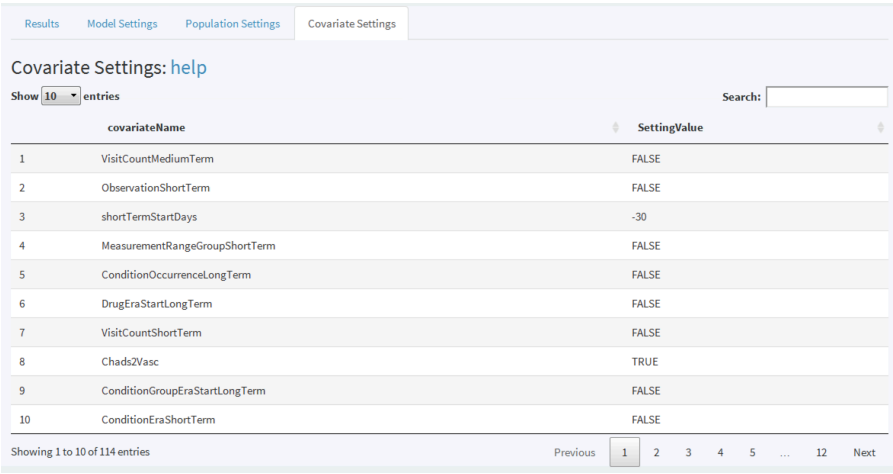


Figure 14.58: To view the covariate settings used when developing the model.

covariates were used as candidate covariates in the model:

The row selection also works for displaying the model performance. To view the performance you need to select 'Performance' from the left menu:

By clicking the 'Performance' option from the menu you will be taken to a threshold performance summary:

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 … for more details about these measurements. In Figure 14.60 we see the selected prediction model is: "within new users of ACE inhibitors as first line monotherapy for hypertension predict who will developed acute myocardial infarction during 1 day after cohort start and 365 days after cohort start". At a threshold of 0.00482 the sensitivity is 83.4% (83.4% of patients with the acute MI 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 an acute MI in the following year). As the incidence of the acute MI 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.

You can adjust the threshold by moving the dot in the *Input* box:

To look at the overal discrimination ability of the model click on the 'Discrimination' tab, this then takes you to a view with the ROC plot, PR plot, and distribution plots (the line on the plots corresponds to the selected threshold point):

We see in Figure 14.62 that the ROC plot shows the model was able to discriminate between those who will have the acute MI within the year and those who will not. However, the performance looks less impressive when we see the PR plot, as the low incidence of the acute MI means there is a high false positive rate.

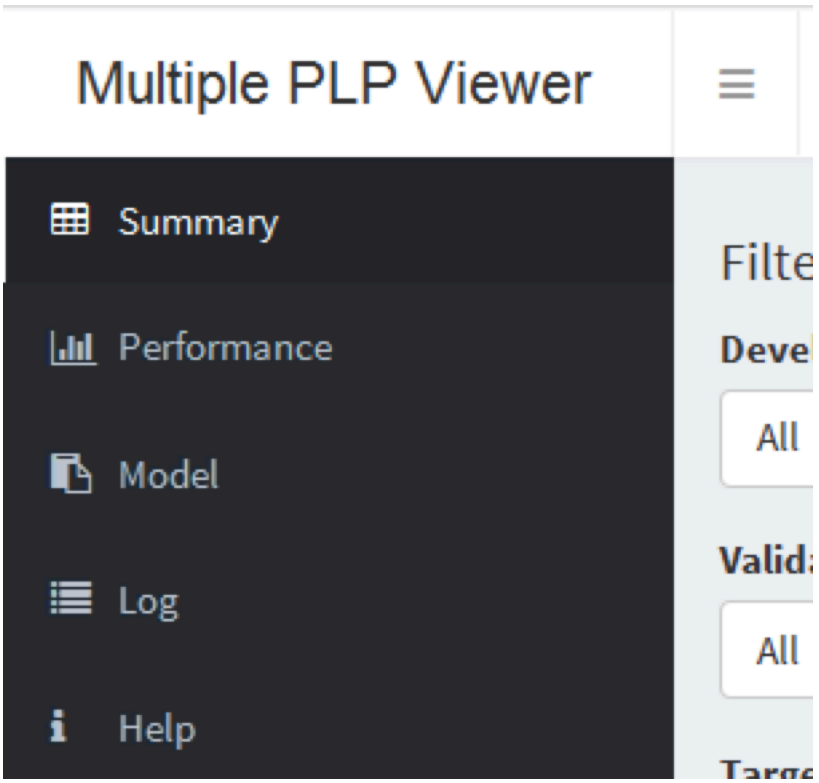


Figure 14.59: The shiny option bar for navigating around the interface.

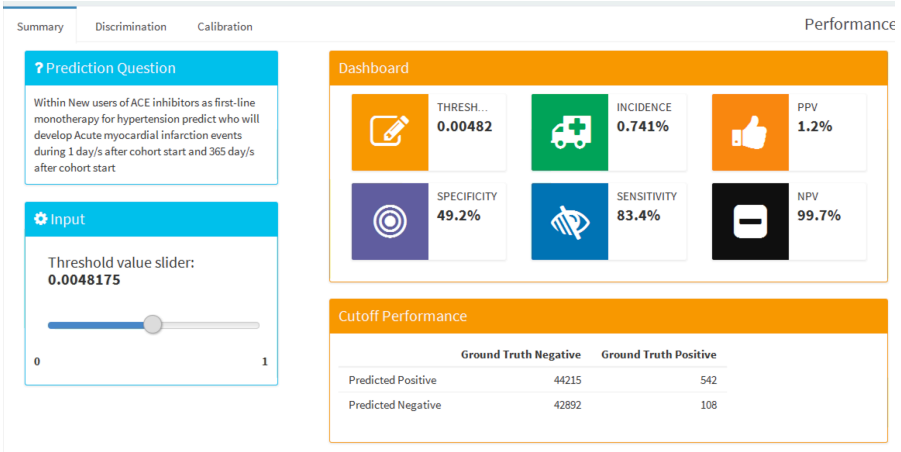


Figure 14.60: The summary performance measures at a set threshold.

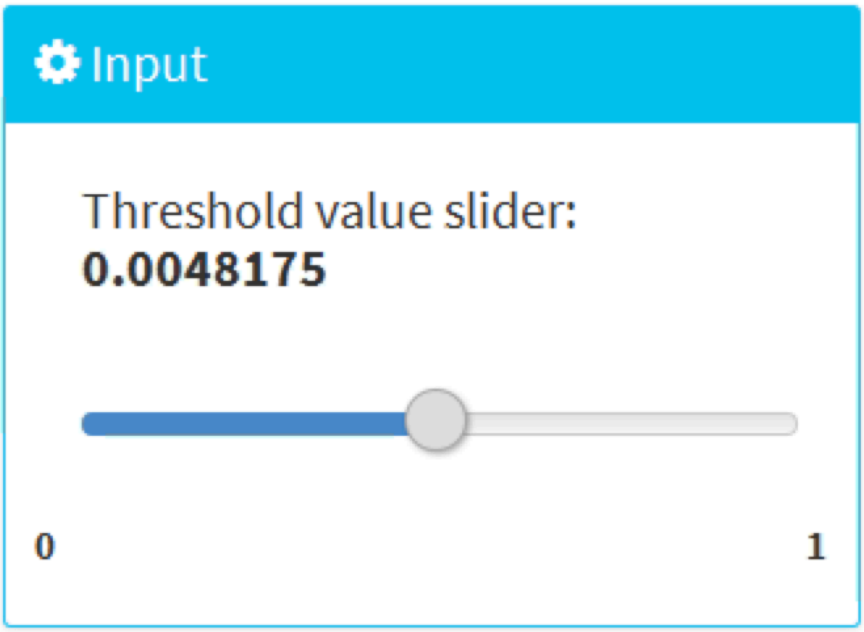


Figure 14.61: Moving this changes the threshold and the values in the Dashboard will update.

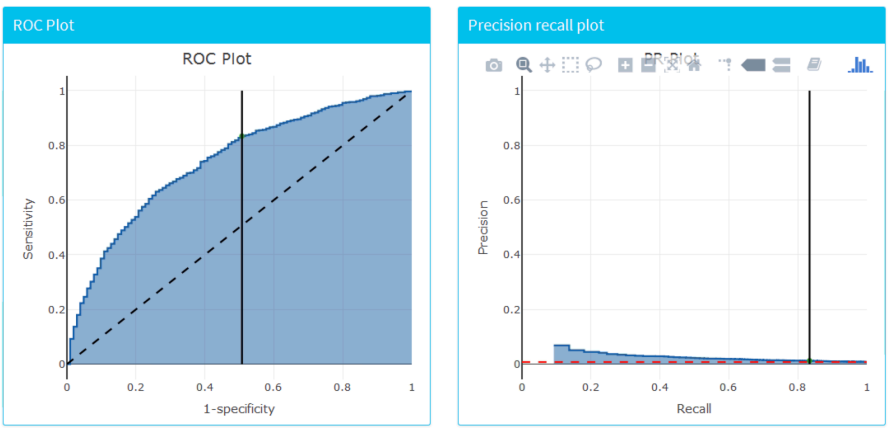


Figure 14.62: The ROC and PR plots used to access the overal discrimination ability of the model.

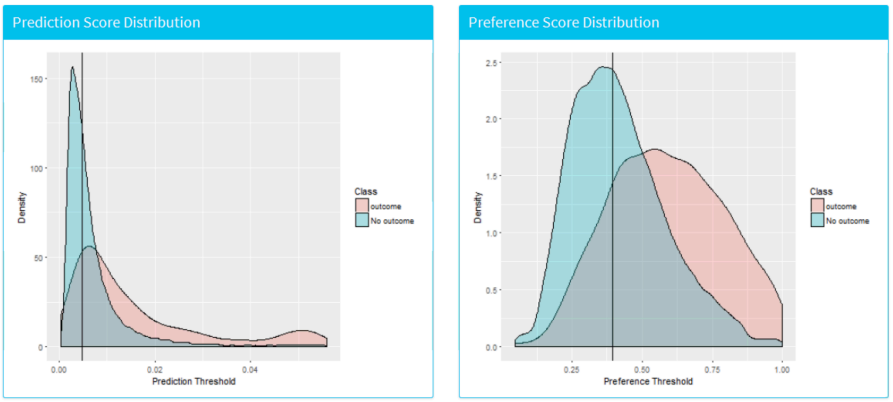


Figure 14.63: The predicted risk distribtion for those with and without the outcome. The more these overlap the worse the discrimination

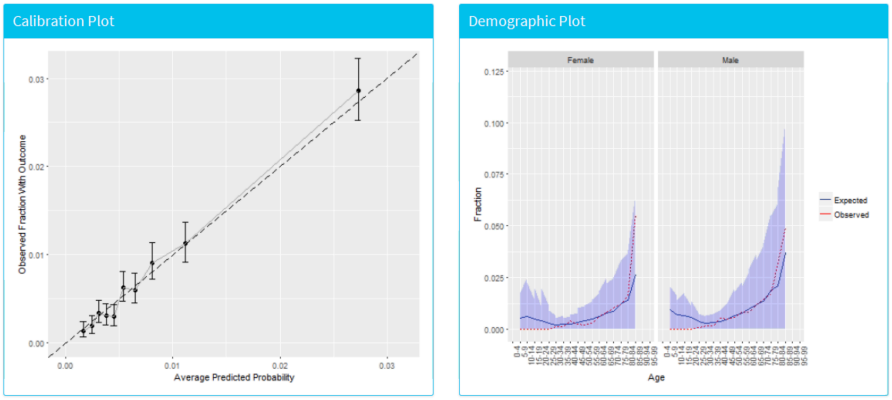


Figure 14.64: The risk stratified calibration and demographic calibration

Finally, you can also inspect the calibration of the model by clicking on the 'Calibration' tab. This displays the calibration plot and the demographic calibration:

Figure 14.64 shows the average predicted risk appears to match the observed fraction who experienced the acute MI 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 seperate models for the younger or older patients.

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 and a table summarising all the candidate covariates. The variable plots are seperated into binary variables and continuous variables. The x-axis is the prevalance/mean in patients without the outcome and the y-axis is the prevalance/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

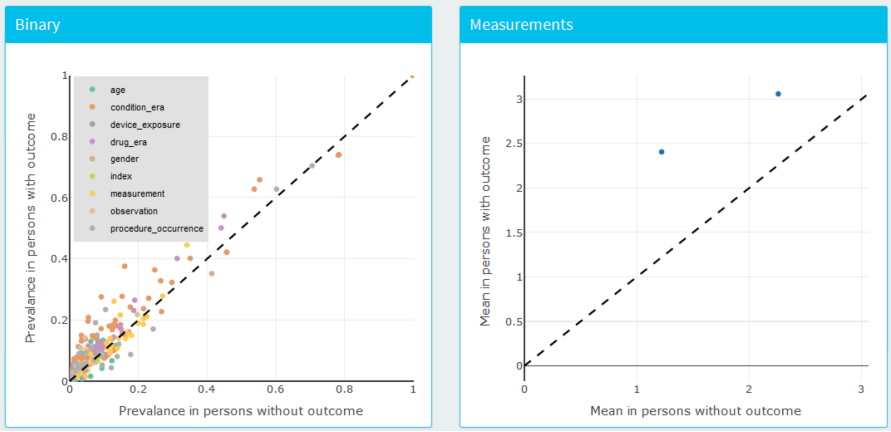


Figure 14.65: Each dot corresponds to a varible included in the model.

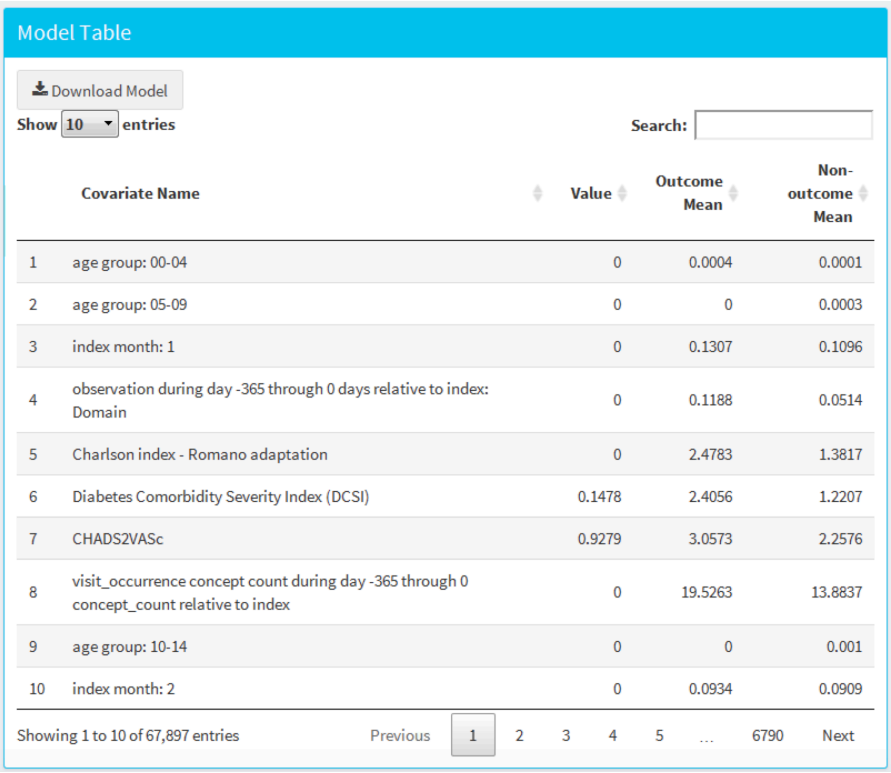
patients with the outcome:

The table below displays the Name, Value (coefficient if using a glm or variable importance otherwise) 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):

You can click on the columns headers to order by the chosen column. For example, to order by Value, click on the 'Value' heading.

The shiny interface also enables you to view the model development and evaluation log file. Click on 'Log' in the left hand option bar:

Finally, for instructions on accessing a youtube video demonstrating how to use the interactive shiny result viewer click on "Help" in the left hand option bar:



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 14.66: Each dot corresponds to a varible included in the model.

```
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          Patient-Level Prediction Package version 3.0.5
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          AnalysisID:         Analysis_1
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          CohortID:           3
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          OutcomeID:          1
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          Cohort size:        500000
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          Covariates:         74348
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          Population size:    351028
2019-06-03 22:53:09     [Main thread]   INFO    PatientLevelPrediction          Cases:              2601
2019-06-03 22:53:10     [Main thread]   INFO    PatientLevelPrediction  personSplitter  Creating a 25% test and 75% train (into
3 folds) stratified split by person
2019-06-03 22:53:10     [Main thread]   INFO    PatientLevelPrediction  personSplitter  Data split into 87757 test cases and 263
271 train cases (87758, 87757, 87756)
2019-06-03 22:53:11     [Main thread]   INFO    PatientLevelPrediction          Training Lasso Logistic Regression model
2019-06-03 22:56:44     [Main thread]   INFO    PatientLevelPrediction  fitGLMModel     Running Cyclops
2019-06-03 23:09:24     [Main thread]   INFO    PatientLevelPrediction  fitGLMModel     Done.
2019-06-03 23:09:24     [Main thread]   INFO    PatientLevelPrediction  fitGLMModel     GLM fit status:  OK
2019-06-03 23:09:24     [Main thread]   INFO    PatientLevelPrediction  fitGLMModel     Fitting model took 13.8 mins
2019-06-03 23:10:00     [Main thread]   INFO    PatientLevelPrediction  predictProbabilities    Prediction took 11.9 secs
2019-06-03 23:10:48     [Main thread]   INFO    PatientLevelPrediction  predictProbabilities    Prediction took 4.04 secs
2019-06-03 23:10:48     [Main thread]   INFO    PatientLevelPrediction          Train set evaluation
2019-06-03 23:10:51     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     AUC:                78.08
2019-06-03 23:10:51     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     AUPRC:              4.35
2019-06-03 23:10:51     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     Brier:              0.01
2019-06-03 23:11:25     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     Calibration gradient: 1.21 intercept:
      -0.00
2019-06-03 23:11:30     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     Average Precision:  0.04
2019-06-03 23:11:30     [Main thread]   INFO    PatientLevelPrediction          Test set evaluation
2019-06-03 23:11:31     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     AUC:                74.49
2019-06-03 23:11:31     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     95% lower AUC:      72.54
2019-06-03 23:11:31     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     95% upper AUC:      76.44
2019-06-03 23:11:31     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     AUPRC:              3.09
2019-06-03 23:11:31     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     Brier:              0.01
2019-06-03 23:12:04     [Main thread]   INFO    PatientLevelPrediction  evaluatePlp     Calibration gradient: 1.07 intercept:
```

Figure 14.67: Example log display.

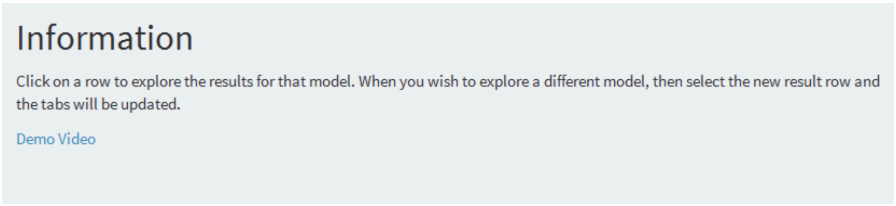


Figure 14.68: Instructions for viewing a demo video.

14.11 Additional Patient-level Prediction Features

14.11.1 Journal paper generation

We have added functionality to automatically generate a word document you can use as start of a journal paper. It contains many of the generated study details and results. If you have performed external validation these results will can be added as well. Optionally, you can add a "Table 1" that contains data on many covariates for the target population. You 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.

14.12 Excercises

Part IV

Evidence Quality

Chapter 15

Introduction to Evidence Quality

Chapter lead: Jon Duke

> Loss of fidelity begins with the movement of data from the doctor's brain to the medical record.

Clem McDonald, MD Director, Lister Hill Center for Biomedical Informatics National Library of Medicine, USA

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. As a community, OHDSI strives to take a holistic view on the subject of quality by focusing more broadly on the question of "evidence quality" rather than data quality alone. Specifically, OHDSI seeks to recognize, evaluate, and optimize the diverse set of processes necessary to achieve the highest quality reproducible evidence from diverse data sources.

As such, we frame the discussion of evidence quality by considering the following four dimensions:

- Data Quality (ie data validity)
- Clinical Validity
- Software Validity
- Method Validity

In this chapter, we will review each of these components and discuss how OHDSI tools, conventions, and community support their evaluation and improvement.

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.

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, 2013)

- 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).

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)

Chapter 17

Clinical Validity

Chapter 18

Software Validity

Chapter lead: Martijn Schuemie

The central question of sofware validity is

> Does the software do what it is expected to do?

In broad strokes there are two approaches to ensure software validity: by using a software development process aimed at creating valid software, and by testing whether the software is valid. Here we focus specifically on the OHDSI Methods Library, the set of R packages used in population-level estimation and patient-level prediction. The OHDSI Population-Level Estimation Workgroup and the OHDSI Patient-Level Prediction Workgroup together are responsible for developing and maintaining the OHDSI Methods Library. The OHDSI Population-Level Estimation Workgroup is headed by Drs. Marc Suchard and Martijn Schuemie. The OHDSI Patient-Level Prediction Workgroup his headed by Drs. Peter Rijnbeek and Jenna Reps.

18.1 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 and the OHDSI Forums. 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 and OHDSI Patient-Level Prediction Workgroup leadership 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 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.

18.1.1 Source Code Management

All of the Methods Library's source code is managed in the source code version control system 'git' publicly assessible 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 madef in terms of compatibility

18.1.2 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 repositories, and as a web page. In addition, many packages also have vignettes that highlight specific use cases of a package. 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.

18.1.3 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.

18.1.4 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.

18.1.5 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.

18.1.6 Physical and Logical Security

The OHDSI Methods Library is hosted on the GitHub 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.

18.1.7 Disaster Recovery

The OHDSI Methods Library is hosted on the GitHub system. GitHub's disaster recovery facilities are described at https://github.com/security.

18.2 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 to determine whether analyses implemented using the Methods Library produce reliable and accurate results (we will call this 'method tests').

18.2.1 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 amongst 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. Appendix A lists the locations of the tests in each package. 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 results of the unit tests can be found in the locations specified in Appendix A. 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 Conclusions

The purpose of this chapter is to document evidence to provide a high degree of assurance that the Methods Library can be used in observational studies to consistently produce reliable and accurate estimates. Both through adoption of best software development practices during the software lifecycle, as well as continuous extensive testing of individual components of the software and the start-to-finish application of the methods library on a gold standard aim to ensure the validity of the Methods Library. However, use of the Methods Library does not guarantee validity of a study, since validity depends on many other components outside of the Methods Library as well, including appropriate study design, exposure and outcome definitions, and data quality. It is important to note that there is a significant obligation on the part of the end-user's organization to define, create, implement and enforce the Method Library installation, validation and utilization related Standard Operating Procedures (SOPs) within the end-user's environment. These SOPs should define appropriate and

reasonable quality control processes to manage end-user related risk within the applicable operating framework. The details and content of any such SOPs are beyond the scope of this document.

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:

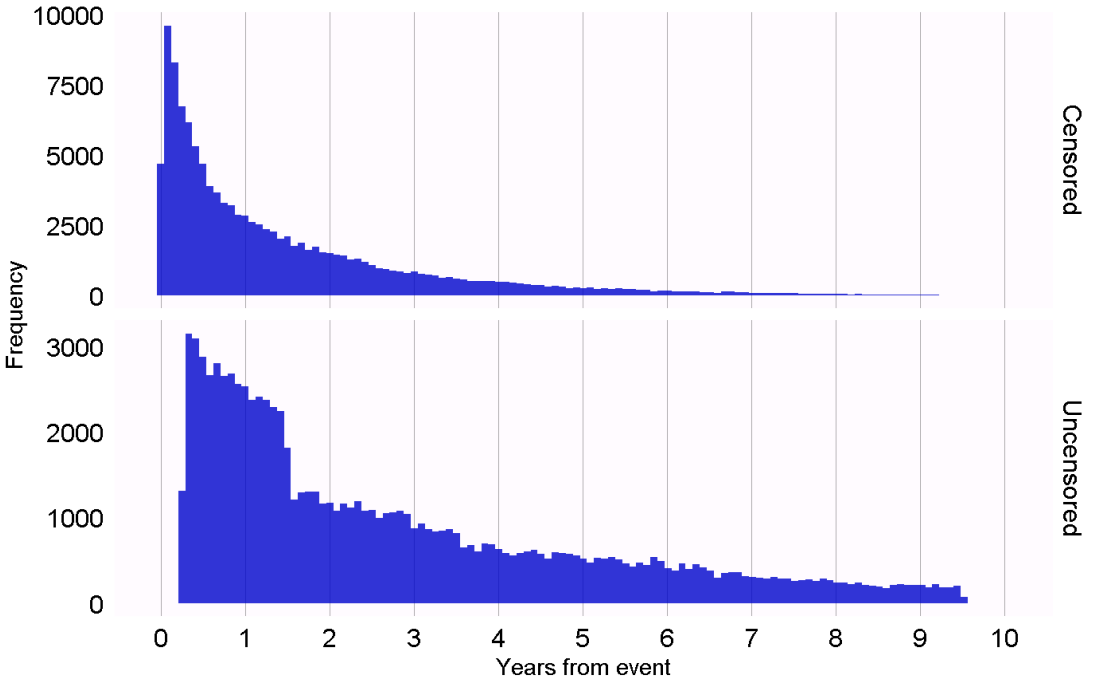


Figure 19.1: Time to observation end for those that are censored, and those that uncensored.

- **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

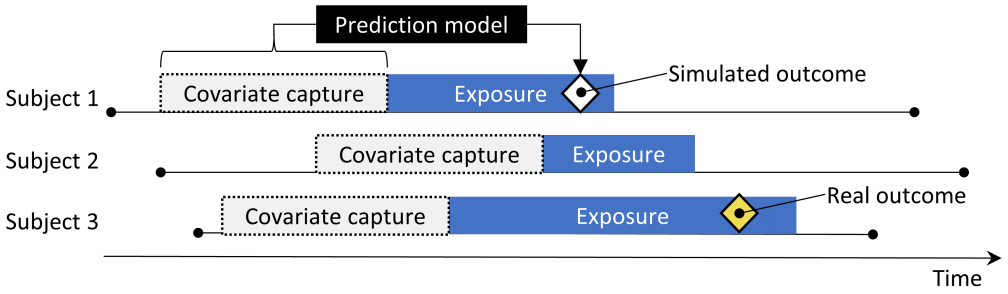


Figure 19.2: Synthesizing positive controls from negative controls.

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 α 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 ith negative control drug–outcome pair, and let $\hat{\tau}_i$ denote the corresponding estimated standard error, $i = 1, \ldots, n$. Let θ_i denote the true log effect size (assumed 0 for negative controls), and let β_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, β_i is always assumed to be equal to zero, but that we assume the β_i's, arise from a normal distribution with mean μ and variance σ^2. This represents the null (bias) distribution. We estimate μ and σ^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 μ and σ^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 β_{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

$$\phi\left(\frac{\theta_{n+1} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \hat{\tau}^2_{n+1}}}\right)$$

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

$$1 - \phi\left(\frac{\theta_{n+1} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \hat{\tau}^2_{n+1}}}\right)$$

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 β_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 β_{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 θ_{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 in Figure 19.3.

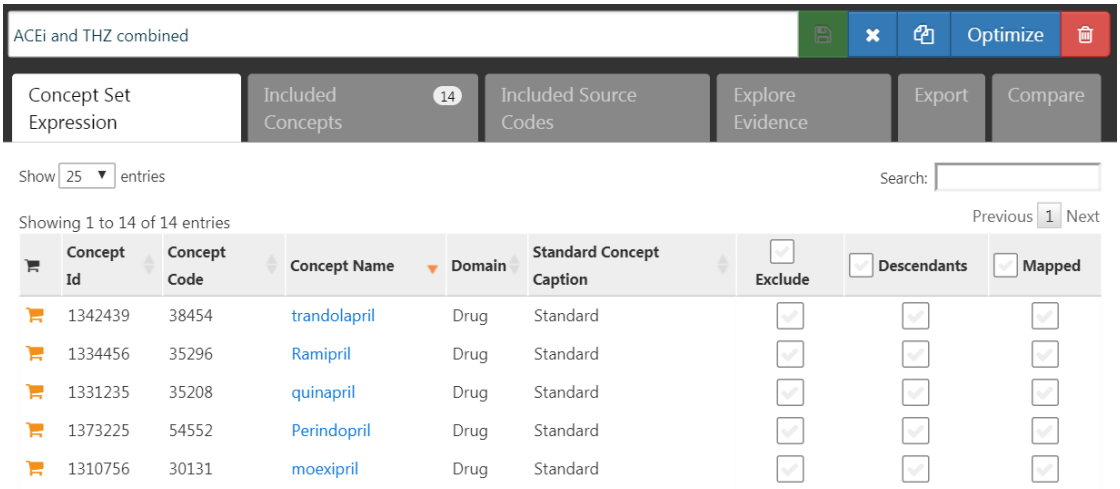


Figure 19.3: A concept set containing the concepts defining the target and comparator exposures.

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 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

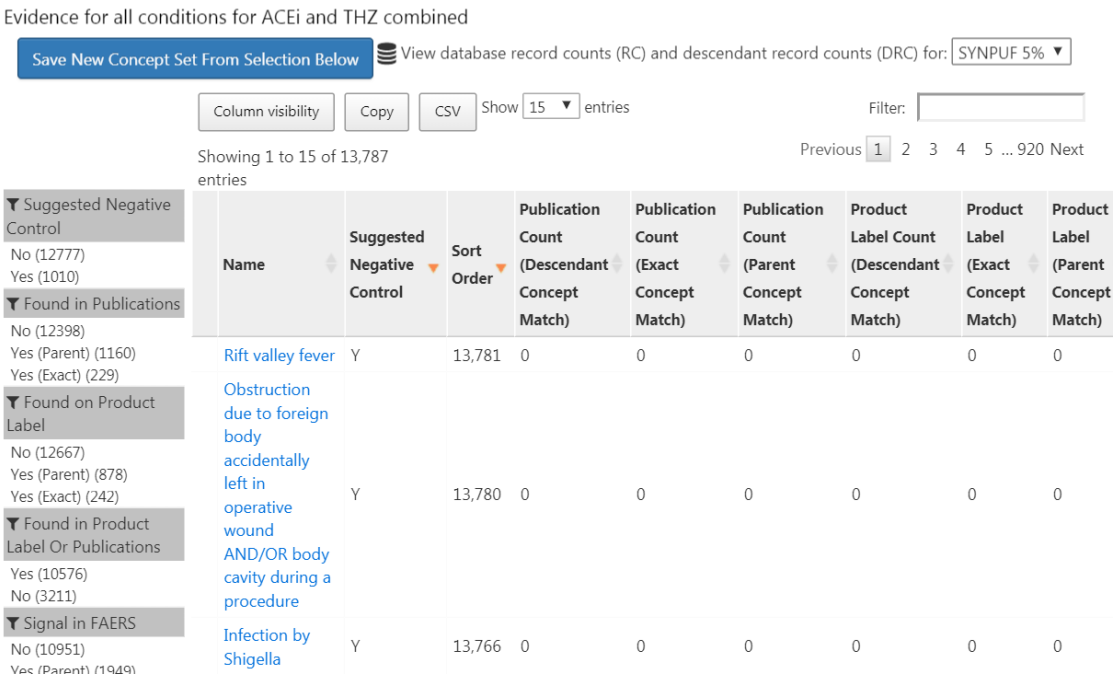


Figure 19.4: Candidate control outcomes with an overview of the evidence found in literature, product labels, and spontaneous reports.

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.

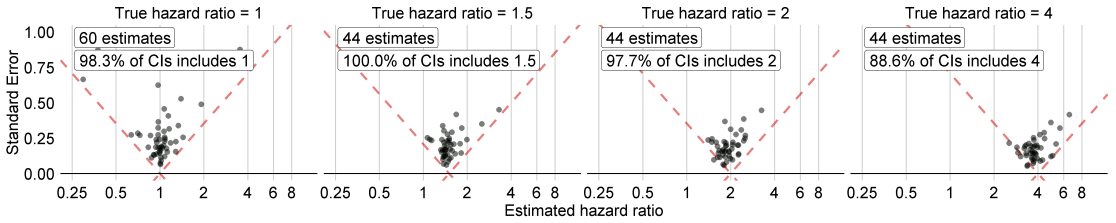


Figure 19.5: Estimates for the negative (true hazard ratio = 1) and positive controls (true hazard ratio > 1). Each dot represents a control. Estimates below the dashed line have a confidence interval that doesn't include the true effect size.

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,
```

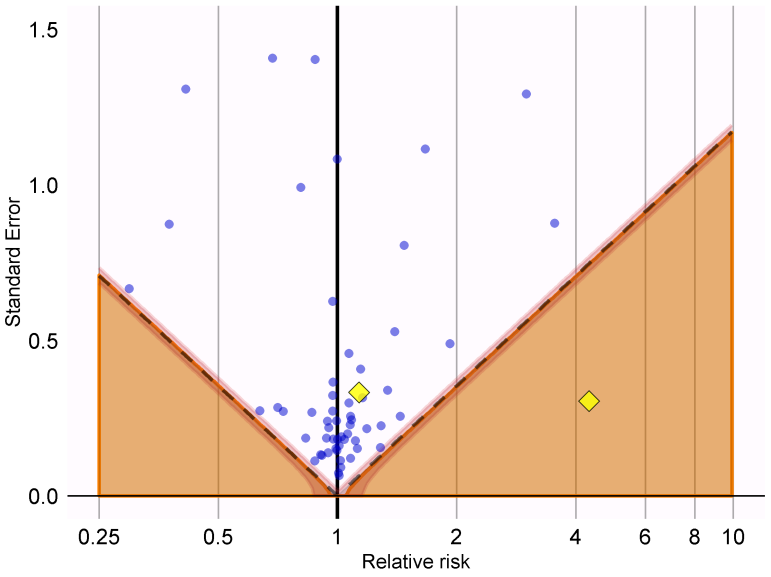


Figure 19.6: P-value calibration: estimates below the dashed line have a conventional p < 0.05. Estimates in the orange area have calibrated p < 0.05. The pink area denotes the 95% credible interval around the edge of the orange area. Blue dots indicate negative controls. Yellow diamonds indicate outcomes 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.

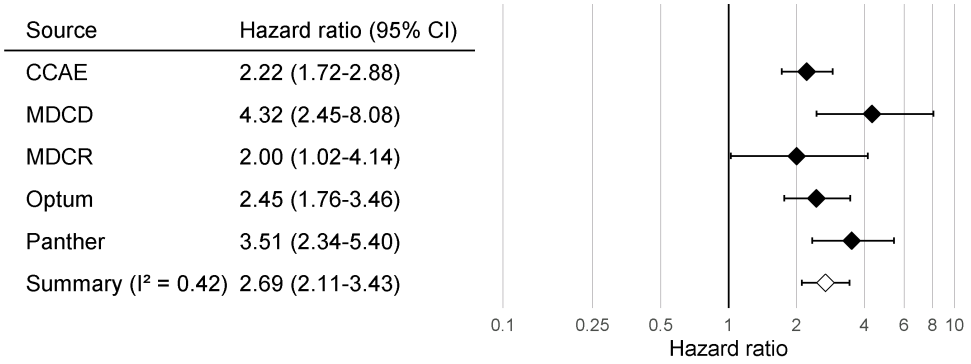


Figure 19.7: Effect size estimates and 95% confidence intervals (CI) from five different databases and a meta-analytic estimatewhen comparing ACE inhibitors to thiazides and thiazide-like diuretics for the risk of angioedema.

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.

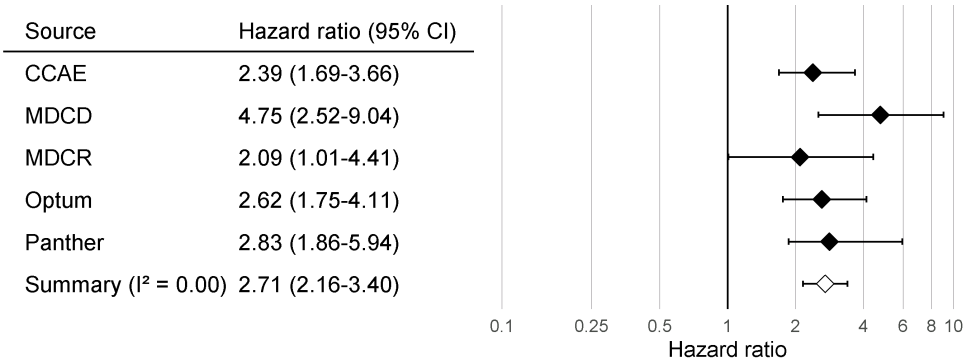


Figure 19.8: Calibrated Effect size estimates and 95% confidence intervals (CI) from five different databases and a meta-analytic estimate for the hazard ratio of angioedema when comparing ACE inhibitors to thiazides and thiazide-like diuretics.

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

long-term or short-term exposures. The results on this benchmark can help demonstrate the overall usefulness of a method, and can be used to form a prior belief about the performance of a method when a context-specific empirical evaluation is not (yet) available. The benchmark consists of 200 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, Vojtech Huser, list others

What is OHDSI Network?

- OHDSI Community and Network Research
- International Open Science Networks
- OHDSI US
- OHDSI EU and EHDEN
- OHDSI APAC

OHDSI Network Study Process

- Goals
- Workflow Overview
- Structure of Studies
- Protocol and IRB issues
- Existing framework (de-identified [time shifted] OMOP dataset under existing IRB protocol
- Overcoming Network Study Challenges
- Data Privacy, Security and Compliance
- Data Quality
- Running OHDSI Methods in Isolated Environment
- OMOP CDM Versioning

Tools, Platforms and Study Automation * OHDSI Methods support for Network Studies * LEGEND (should we have it here?) * OHDSI ARACHNE Network Platform

Opportunities, future trends and Roadmap

21.1 OHDSI Network Study Examples

21.1.1 Endometriosis study

An endometriosis characterization study (available at https://github.com/molliemckillop/Endometriosis-Phenotype-Characterization) works with two cohorts. They are defined in cohorts.csv file (see here https://github.com/molliemckillop/Endometriosis-Phenotype-Characterization/blob/master/inst/settings/cohorts.csv).

After creating a cohort table, it is populated by executing this command here by infering a name of a '.sql' file from the previously defined cohort file. A createCohorts function is executed next. (see https://github.com/molliemckillop/Endometriosis-Phenotype-Characterization/blob/master/R/createCohorts.R). An SQL file that is generated by Atlas populates the cohort table with specific person_ids that fulfill the cohort definition.

21.2 Excercises

21.2.1 Defining a cohort

Q: Study the code for the x study and determine whether the cohort definition is available on the public OHDSI server. It if it, what is the cohort ID there?

A:

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

Bibliography

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*.

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.

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.

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.

Engel, C. and Fischer, C. (2015). Breast cancer risks and risk prediction models. *Breast Care (Basel)*, 10(1):7–12.

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.

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.

Higgins, J. P., Thompson, S. G., Deeks, J. J., and Altman, D. G. (2003). Measuring inconsistency in meta-analyses. *BMJ*, 327(7414):557–560.

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., 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.

Lee, K. L., Woodlief, L. H., Topol, E. J., Weaver, W. D., Betriu, A., Col, J., Simoons, M., Aylward, P., Van de Werf, F., and Califf, R. M. (1995). Predictors of 30-day mortality in the era of reperfusion for acute myocardial infarction. Results from an international trial of 41,021 patients. GUSTO-I Investigators. *Circulation*, 91(6):1659–1668.

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.

Mons, B. (2018). *Data Stewardship for Open Science: Implementing FAIR Principles*. Chapman and Hall/CRC.

Nguyen, N. D., Frost, S. A., Center, J. R., Eisman, J. A., and Nguyen, T. V. (2008). Development of prognostic nomograms for individualizing 5-year and 10-year fracture risks. *Osteoporos Int*, 19(10):1431–1444.

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.

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.

Perel, P., Edwards, P., Wentz, R., and Roberts, I. (2006). Systematic review of prognostic models in traumatic brain injury. *BMC Med Inform Decis Mak*, 6:38.

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.

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).

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.

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. (2013). 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 (2019). Open science — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Open%20science&oldid=900178688. [Online; accessed 24-June-2019].

Wilson, P. W., D'Agostino, R. B., Levy, D., Belanger, A. M., Silbershatz, H., and Kannel, W. B. (1998). Prediction of coronary heart disease using risk factor categories. *Circulation*, 97(18):1837–1847.

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.