The Book of OHDSI

Observational Health Data Science and Informatics

2019-06-17

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 organizes in five major sections: (I) The OHDSI Community, (II) Uniform data representation, (III) Data Analytics, (IV) Evidence Quality, and (V) OHDSI Studies. Each section has multiple chapters, and each chapter aims to follow the following main outline: Introduction, Theory, Practice, Excercises.

Part I

The OHDSI Community

Chapter 1

Mission, vision, values

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

History of OHDSI

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

Governance at local sites

Chapter 3

Open Science

Mention FAIR principles?

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 leads: Clair Blacketer & Mui VanZandt

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.

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:

- **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:

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]_ID	Unique identifiers for key entities, which can serve as foreign keys to establish relationships across entities. For example, PERSON_ID uniquely identifies each individual. VISIT_OCCURRENCE_ID uniquely identifies a PERSON encounter at a point of care.
[entity]_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 might be 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 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 5.1 shows that the ICD9CM code for TB is 011.

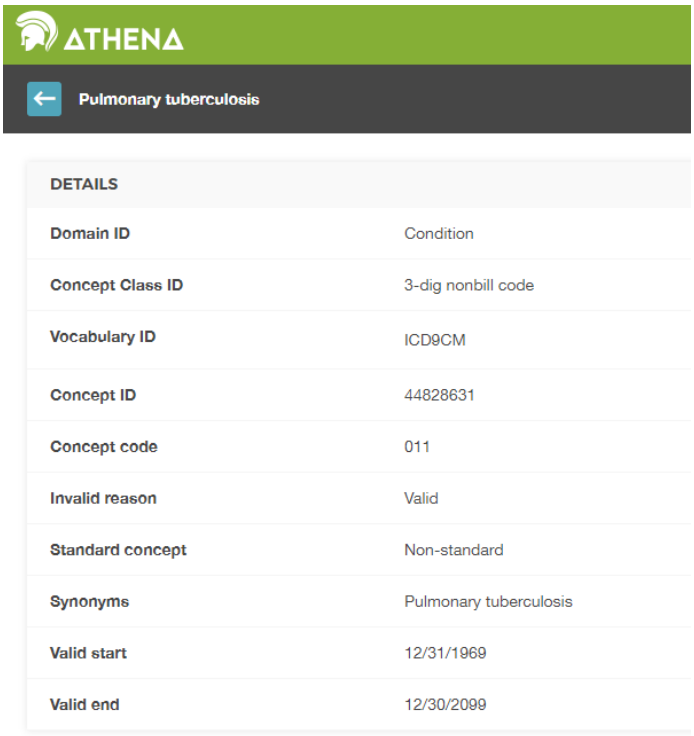
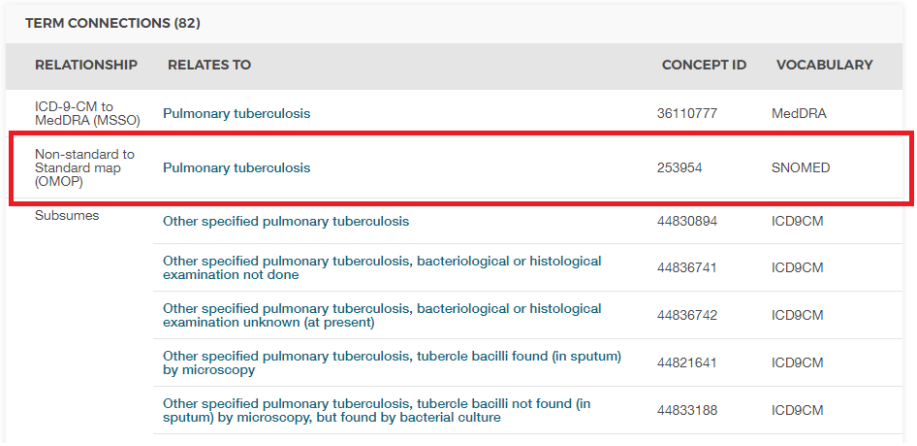


Figure 5.1: ICD9CM code for Pulmonary Tuberculosis



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 5.2: SNOMED code for Pulmonary Tuberculosis

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 5.2 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 5.7.

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: https://github.com/OHDSI/CommonDataModel/wiki.

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

5.3.1 Running Example: Endometriosis

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

> 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.

Column Name	Value	Explanation
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	E	Typically this would be her identifier in the source data, though often is it the same as the PERSON_ID.
GENDER_SOURCE_VALUE	F	The gender value as it appears in the source is stored here.
GENDER_SOURCE_ CONCEPT_ID	0	If the gender value in the source was coded using a vocabulary recognized by OHDSI, that concept ID would go here. For example, if her gender was 'Sex-F' in the source and it was stated to be in the PCORNet vocabulary concept ID 44814665 would go in this field.
RACE_SOURCE_VALUE	white	The race value as it appears in the source is stored here.
RACE_SOURCE_CONCEPT_ID	0	Same principle as GENDER_SOURCE_CONCEPT_ID.
ETHNICITY_SOURCE_VALUE	English	The ethnicity value as it appears in the source is stored here.
ETHNICITY_SOURCE_ CONCEPT_ID	0	Same principle as GENDER_SOURCE_CONCEPT_ID.

5.3.3 OBSERVATION_PERIOD table

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

How can we determine Lauren's observation period?

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

Encounter_ID	Start_Date	Stop_Date	EncounterClass
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.
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.

Column Name	Value	Explanation
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?

Revisting the encounters we used to determine her observation period:

Encounter_ID	Start_Date	Stop_Date	EncounterClass
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**

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

Table 5.6: The VISIT_OCCURRENCE table.

Column Name	Value	Explanation
VISIT_OCCURRENCE_ID	314	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.

Column Name	Value	Explanation
VISIT_START_DATE	2013-01-17	The start date of the visit.
VISIT_START_DATETIME	2013-01-17 00:00:00	The date and time of the visit started. When time is unknown midnight is used.
VISIT_END_DATE	2013-01-24	The end date of the visit. If this is a one-day visit the end date should match the start date.
VISIT_END_DATETIME	2013-01-24 00:00:00	The date and time of the visit end. If time is unknown midnight is used.
VISIT_TYPE_CONCEPT_ID	32034	This column is intended to provide information about the provenance of the visit record, i.e. does it come from an insurance claim, hospital billing record, EHR record, etc. For this example the concept ID 32035 is used as the encounters are similar to electronic health records
PROVIDER_ID*	NULL	If the encounter record has a provider associated, the id for that provider goes in this field. This should be the PROVIDER_ID from the PROVIDER table that represents the provider on the encounter.
CARE_SITE_ID	NULL	If the encounter record has a care site associated, the id for that care site goes in this field. This should be the CARE_SITE_ID from the CARE_SITE table that codes for the care site on the encounter.
VISIT_SOURCE_VALUE	Inpatient	The visit value as it appears in the source goes here. In this context 'visit' means outpatient, inpatient, emergency, etc.
VISIT_SOURCE_CONCEPT_ID	0	If the visit value from the source is coded using a vocabulary that is recognized by OHDSI, the concept ID that represents the visit source value would go here.

Column Name	Value	Explanation
ADMITTED_FROM_CONCEPT_ID	0	If known, this is the concept ID that represents where the patient was admitted from. This concept should have the concept class 'Place of Service' and the domain 'Visit'. For example, if a patient was admitted to the hospital from home, the concept ID would be 8536.
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.
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 on the CDM wiki.

5.3.5 CONDITION_OCCURRENCE

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

What are Lauren's conditions?

Revisiting her account she says:

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

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

Table 5.7: The CONDITION_OCCURRENCE table.

Column	Value	Explanation
CONDITION_OCCURRENCE_ID	961	This is typically an autogenerated field that creates a unique id number for each condition on the person's record in the converted CDM database.
PERSON_ID	1	This comes from the PERSON table and links PERSON and CONDITION_OCCURRENCE.
CONDITION_CONCEPT_ID	194696	The concept ID that represents the SNOMED code 266599000 is 194696
CONDITION_START_DATE	2010-01-06	The date when the instance of the Condition is recorded.
CONDITION_START_DATETIME	2010-01-06 00:00:00	The date and time when the instance of the Condition is recorded. Midnight is used when the time is unknown
CONDITION_END_DATE	NULL	If known, this is the date when the instance of the Condition is considered to have ended.
CONDITION_END_DATETIME	NULL	If known, this is the date and time when the instance of the Condition is considered to have ended.
CONDITION_TYPE_CONCEPT_ID	32020	This column is intended to provide information about the provenance of the condition, i.e. does it come from an insurance claim, hospital billing record, EHR record, etc. For this example the concept ID 32020 is used as the encounters are similar to electronic health records. Concept IDs in this field should be in the 'Condition Type' vocabulary.

Column	Value	Explanation
CONDITION_STATUS_CONCEPT_ID	0	If known, the CONDITION_STATUS_CONCEPT_ID 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	5091	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	266569009	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	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 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 Exposure is inferred from clinical events associated with orders, prescriptions written, pharmacy dispensings, procedural administrations, and other patient-reported information.

What are Lauren's drug exposures?

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

Table 5.8: The DRUG_EXPOSURE table.

Column	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.

Column	Value	Explanation
PERSON_ID	1	This comes from the PERSON table and links PERSON and DRUG_EXPOSURE.
DRUG_CONCEPT_ID	1127433	The NDC code for acetaminophen maps to the RxNorm code 313782 which is represented by the concept ID 1127433.
DRUG_EXPOSURE_START_DATE	2010-01-06	The start date of the drug exposure
DRUG_EXPOSURE_START_DATETIME	2010-01-06 00:00:00	The start date and time of the drug exposure. Midnight is used when the time is not known.
DRUG_EXPOSURE_END_DATE	2010-02-05	The end date of the drug exposure. Depending on different sources, it could be a known or an inferred date and denotes the last day at which the patient was still exposed to the drug. In this case the end is inferred since we know Lauren had a 30 days supply.
DRUG_EXPOSURE_END_DATETIME	2010-02-05 00:00:00	The end date and time of the drug exposure. Similar rules apply as to DRUG_EXPOSURE_END_DATE. Midnight is used when time is unknown
VERBATIM_END_DATE	NULL	If the source provides an end date rather than just days supply that date goes here.
DRUG_TYPE_CONCEPT_ID	38000177	This column is intended to provide information about the provenance of the drug, i.e. does it come from an insurance claim, prescription record, etc. For this example the concept ID 38000177 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.

Column	Value	Explanation
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 is used.
LOT_NUMBER	NULL	An identifier assigned to a particular quantity or lot of Drug product from the manufacturer.
PROVIDER_ID	NULL	If the drug record has a prescribing provider listed, the id for that provider goes in this field. This should be the PROVIDER_ID from the PROVIDER table that represents the provider on the encounter.
VISIT_OCCURRENCE_ID	5109	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.

Column	Value	Explanation
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.
ROUTE_SOURCE_VALUE	NULL	The information about the route of administration as detailed in the source.
DOSE_UNIT_ SOURCE_VALUE	NULL	The information about the dose unit as detailed in the source.

5.3.7 PROCEDURE_OCCURRENCE

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

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

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

Table 5.9: The PROCEDURE_OCCURRENCE table.

Column	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

Column	Value	Explanation
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 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		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 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	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	3044350	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.

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 buckets:

- 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 Chapters X, Y, and Z.

8.1 Characterization

We can use the data to provide answers to questions about the characteristics of the patients in each 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?

8.2 Population-level estimation

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

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.

- 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?

These tasks 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 are sparsely recorded if at all. These include mortality, behavioral outcomes, lifestyle, and socioeconomic status.
- Since patients tend to encounter the healthcare system 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.

Chapter 9

OHDSI Analytics Tools

ATLAS: http://www.ohdsi.org/web/wiki/doku.php?id=documentation:software:atlas

ARACHNE: Network Research

Methods Library: https://ohdsi.github.io/MethodsLibrary/

Best practices enforced in all OHDSI methods.

Ethical consideration: e.g. should always communicate uncertainty. Prespecification of research questions, etc.

Analytic use cases

What is the difference between characterization, population-level estimation, patient-level prediction?

Case study: Perhaps on how to install the tools?

Chapter 10

SQL and R

Chapter lead: Martijn Schuemie & Peter Rijnbeek

The Common Data Model is a relational database model, which means that the data will 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 risks, 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, a single R package that can connect to a wide range of 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 first review how to use qlRender and DatabaseConnector to perform database operations in R in a way that allows the same code to be executed across a range of database platforms. 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) can be used to query the CDM. The following section highlight how to query the OHDSI Standardized Vocabulary when querying the CDM.

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 vignettes 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.

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
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 things 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.cr9buun4children ;"
```

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 PosgreSQL 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.

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")
```

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. In this case one 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 convenient and faster 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 examples below 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. For example the maximum age at observation end can be computed by joining the PERSON table to the OBSERVATION_PERIOD 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, and then order the result set to find the min, max, median, and interquartile range:

```
WITH age
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 age
CROSS JOIN (
    SELECT COUNT(*) AS n
    FROM age
    ) 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, sq, 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 not very efficient.

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

CONDITION_SOURCE_VALUE	CODE_COUNT
319	25798284
31401	22547122
317	22453999
311	19626574
496	19570098
I10	19453451
3180	18973883

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';
```

PRESCRIPTION_COUNT
26871214

10.5 Exercices

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 medical interventions 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**: estimation the effect of one 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 availalbe in OHDSI's standardized tools. We then detail the design of

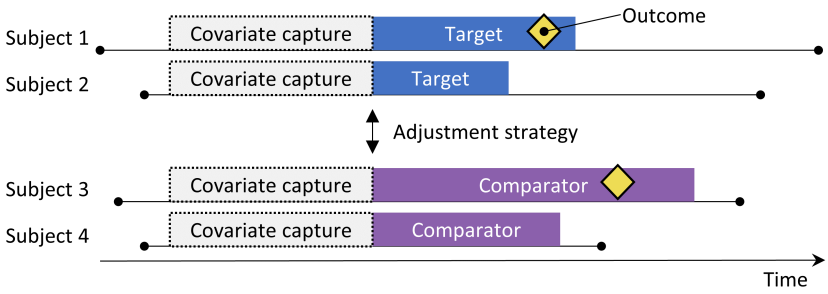


Figure 13.1: The new-user cohort design. Subjects observed to initiate the target treatment are compared to those initiating the comparator treatment. To adjust for differences between the two treatment groups several adjustment strategies can be used, such as stratification, matching, or weighting by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

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.

13.1 Study designs

Several different study designs can be used to estimate treatment effects. The main difference between these is how they construct the (unobserved) counterfactual. Below is a brief discussion of the most commonly used designs, all of which are implemented as R packages in the OHDSI Methods Library.

13.1.1 Cohort method

The new-user 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.

Choice	Description

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 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.

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 is to select a comparator treatment for the same indication as the exposure of interest, where the comparator treatment is believed to have no effect on the outcome. Unfortunately, such a comparator might not always be available.

A key concern is that the patients receiving the target treatment may systematically differ from those receiving the comparator treatment. For example, suppose the target cohort is, on average, 60 years old, whereas the comparator cohort is on average 40 years old. Comparing target to comparator with respect to any age-related health outcome (e.g. stroke) might then show substantial differences. An uninformed investigator might reach the conclusion there is a causal association between the target treatment and stroke as compared to the comparator. More prosaically or commonplace, the investigator might conclude that there exist target patients that experienced stroke that would not

have done so had they received the comparator. This conclusion could well be entirely incorrect! Maybe those target patients disproportionately experienced stroke simply because they are older; 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".

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 that each patient received the target treatment. 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.

For example, suppose we use 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. Unfortunately this is not a testable assumption. See Chapter 19 on Method Validity for further discussion of this issue.

Variable selection

In the past, PS were computed based on manually selected characteristics, and although the OHDSI tools can support such practices, we prefer the use of large-scale regularized regression using many generic characteristics. (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 weight 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.

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) to within some tolerance known as a "caliper." Following Austin (2011), we use a default of caliper of 0.2 standard deviations on the logit scale.

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, 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

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

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

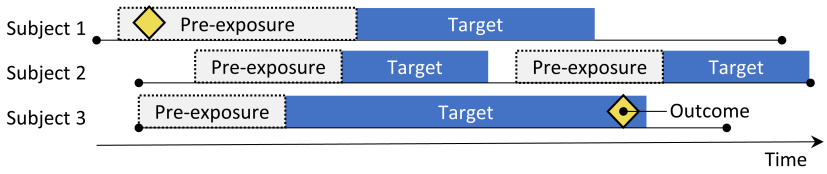


Figure 13.2: The self-controlled cohort design. The rate of outcomes during exposure to the target is compared to the rate of outcomes in the time pre-exposure.

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

Balance

Good practice always checks that the PS adjustment succeeded in creating balanced groups of patients. Figure13.18 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).

13.1.2 Self-controlled cohort

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 to other differences, such as differences between different time periods.

13.1.3 Case-control

Case-control (Vandenbroucke and Pearce, 2012) studies 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.

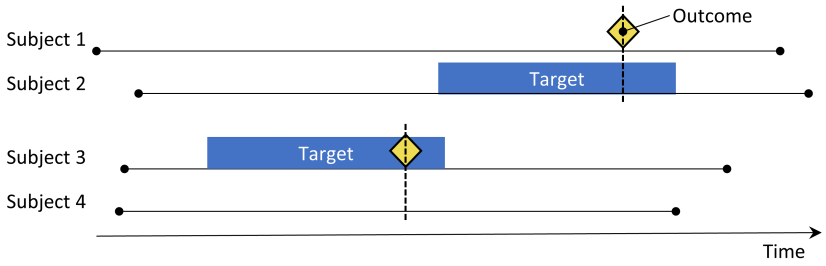


Figure 13.3: The case-control design. Subjects with the outcome ('cases') are compared to subjects without the outcome ('controls') in terms of their exposure status. Often, cases and controls are matched on various characteristics such as age and sex.

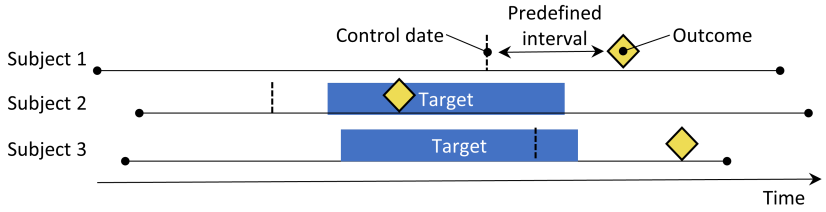


Figure 13.4: The case-crossover design. The time around the outcome is compared to a control date set at a predefined interval prior to the outcome date.

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 selection	A strategy for selecting controls and their index date
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 matches controls to cases based on characteristics such as age and sex to make them more comparable. Another widespread practice is to nest the analysis within a specific subgroup of people, for example people that have all been diagnosed with one of the indications of the exposure of interest.

13.1.4 Case-crossover

The case-crossover (Maclure, 1991) design evaluates whether the rate of exposure is different at the time of the outcome than at some predefined number of days prior to the outcome. It is trying to 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

Choice	Description
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 matched controls to the case-crossover design to adjust for exposure trends.

13.1.5 Self-controlled case series

The Self-Controlled Case Series (SCCS) design (Farrington, 1995, whitaker_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.

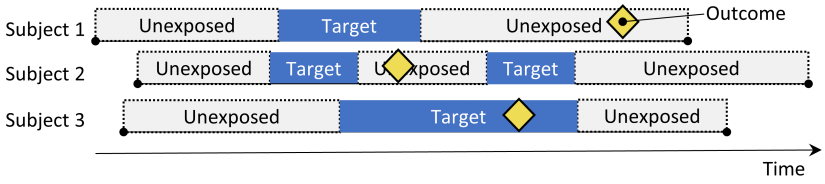


Figure 13.5: The Self-Controlled Case Series design. The rate of outcomes during exposure is compared to the rate of outcomes when not exposed.

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)

13.2 Designing a hypertension study

13.2.1 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).

We will apply our population-level estimation framework to observational healthcare data to address the following comparative estimation question:

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

Since this is a comparative effect estimation question we will apply the cohort method as described in Section 13.1.1.

13.2.2 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 hypertension 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.

13.2.3 Outcome

We define angioedema as any occurrence of an angioedema diagnose code during an inpatient or 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 diagnose code during an inpatient or ER visit, and require there to be no AMI diagnosis record in the 180 days prior.

13.2.4 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 prescriptions.

13.2.5 Model

We fit a PS model using the default set of covariates, which includes demographics, conditions, drugs, procedures, measurements, observations, and several comorbidity scores. We exclude ACEi and THZ from the covariates. We perform variable-ratio matching (Rassen et al., 2012) and condition the Cox regression on the matched sets.

13.2.6 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.

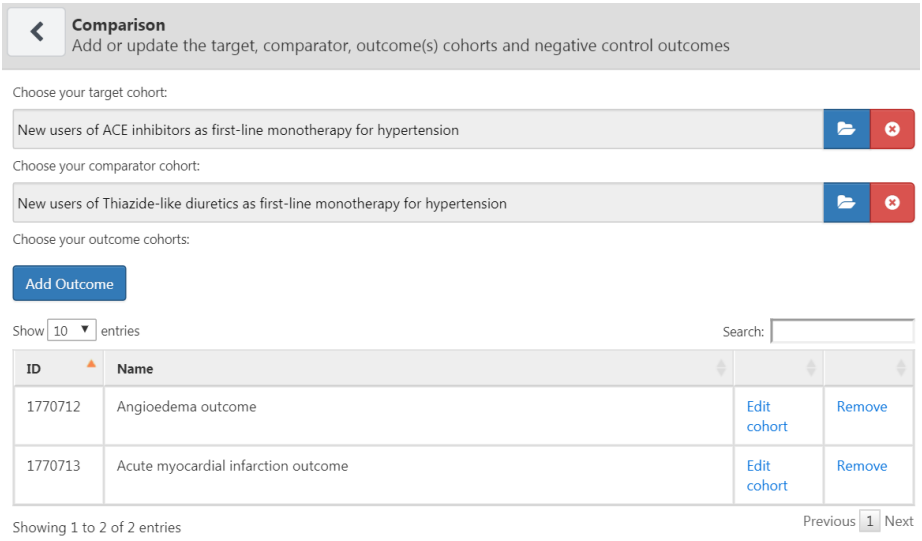


Figure 13.6: The comparison dialog

13.3 Implementation the study using ATLAS

Here we demonstrate how this study can be implemented using the Estimation tool in ATLAS. Click on Estimation in the left bar of ATLAS, and create a new estimation study. Make sure to give the study an easy-to-recognize name. you can save the study design at any time by clicking the button.

In the Estimation design tool, 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:

13.3.1 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.

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

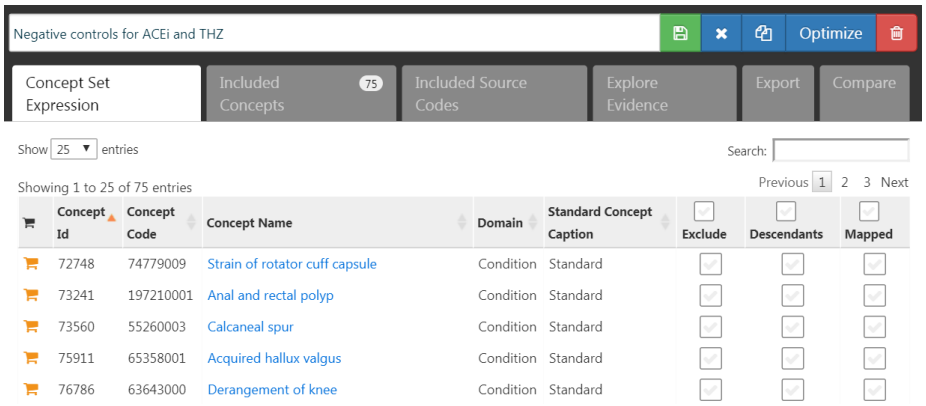


Figure 13.7: Negative Control concept set.

analysis.

Negative control outcomes

We should also include a set of negative control outcomes. These 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. Negative controls are discussed in more detail in Chapter 19. Here we assume a concept set has already been created 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

When selecting concept to include, you can specify which covariates you would like to generate, for example to use in your 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. You might want to specify particular covariates when you are replicating an existing study that manually picked a small number of covariates. These inclusions can be specified in this comparison section or in the analysis section. The option in the analysis section of whether or not to include descendants will apply also here in the comparison section.

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,

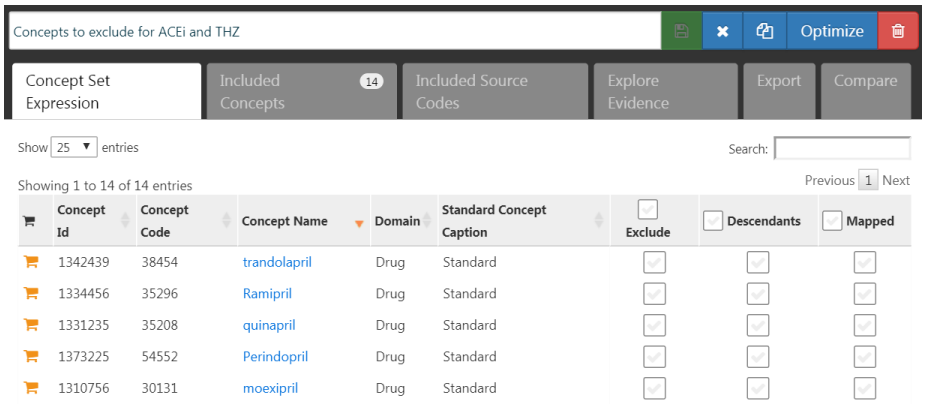


Figure 13.8: The concept set defining the concepts to exclude. Note that no descendantsa have been included. We will specify to include these at analysis time in the Analysis settings.

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.

Often we want to include or exclude concepts **and their descendants**. We could specify the concept set to include descendant concepts. However, for various reasons it might be more efficient to not include descendant in the concept set, but rather automatically add them by setting the 'Should descendant concepts be added' option to *yes* in the Covariate Settings section of the Analysis settings that will be discussed later.

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

13.3.2 Effect estimation analysis settings

After closing the comparisons dialog you can click on 'Add Analysis Settings'. In the box labeled 'Analysis Name', 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

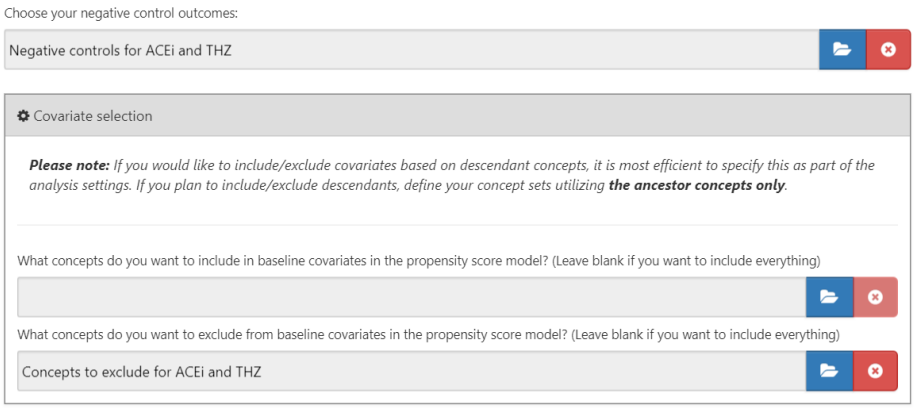


Figure 13.9: The comparison window showing concept sets for negative controls and concepts to exclude.

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 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. Adjusting for this can also 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

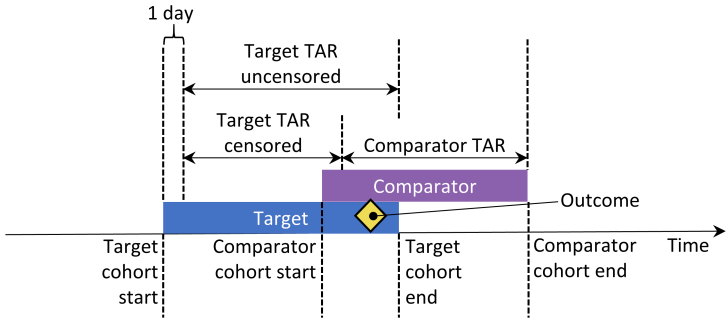


Figure 13.10: Time-at-risk (TAR) for subjects who are in both cohorts, assuming time-at-risk starts the day after treatment initiation, and stops at exposure end.

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 In this case, without censoring the time-at-risk for the two cohorts 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 more likely, and it is likely that the heart failure 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 do choose to remove people that had the outcome before, we can select **how many days we should look back when identifying prior outcomes**.

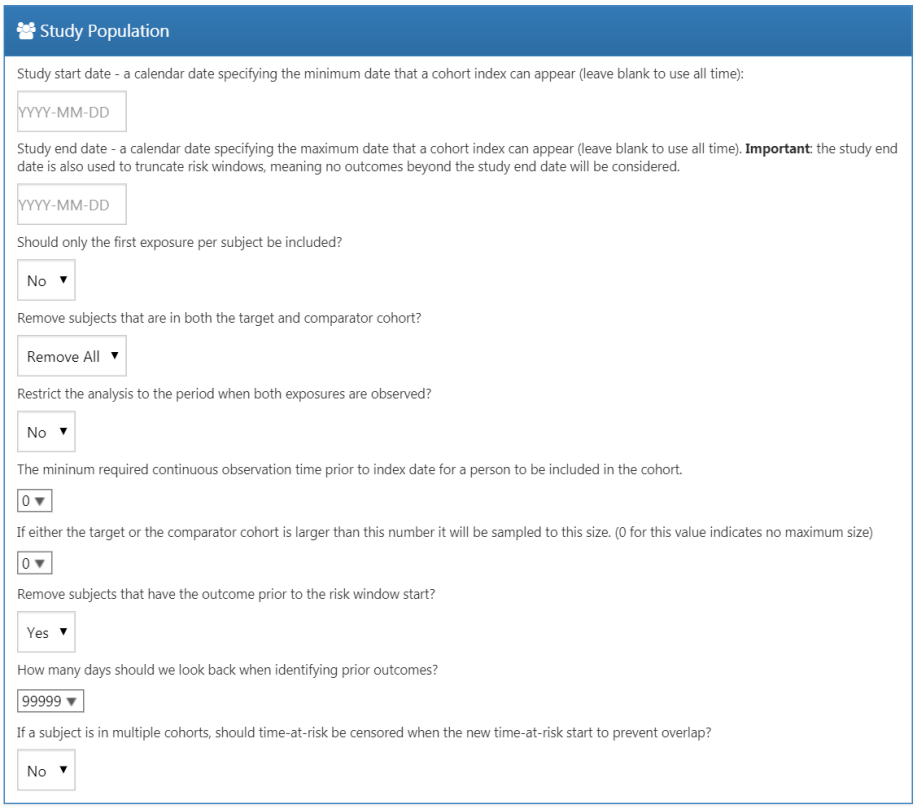


Figure 13.11: Study population settings..

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. 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 13.3.1 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

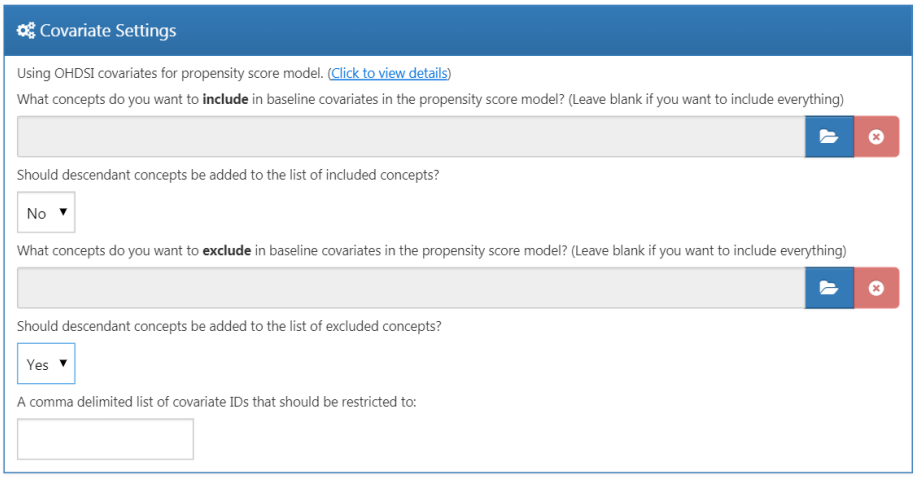


Figure 13.12: Covariate settings.

because we wish to exclude the drugs we are comparing, and sometimes the settings are related to a specific analysis, for example when we wish to use the same covariates used in another study we are trying to replicate. 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 1 day after cohort start, so 1 day after treatment initiation. A reason to set the time-at-risk start to be later than the cohort start is because we may want to exclude outcome events that occur on the day of treatment initiation as 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

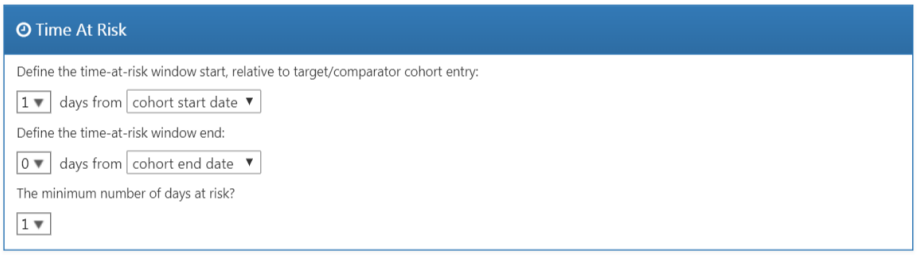


Figure 13.13: Time-at-risk settings.

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 0 days at risk adds no information, so the **minimum days at risk** is normally set at 1 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 (Walker et al., 2013) falls outside the range we prespecify. 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, 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.

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 risk of outcome between target and comparator cohorts**. We can choose between Cox, Poisson, and logistic regression, as discussed briefly in Section 13.1.1. 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 reggression should be condition on the strata**. One way to understand conditioning is to imagine a separate estimate is produced in each strata, 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 to keep 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

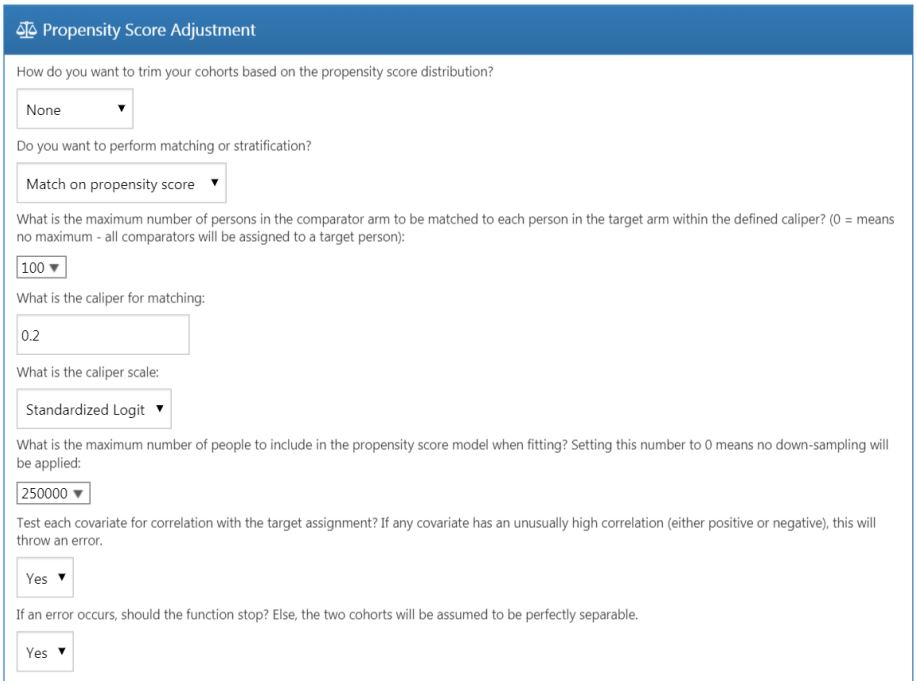


Figure 13.14: Propenstity score adjustment settings.

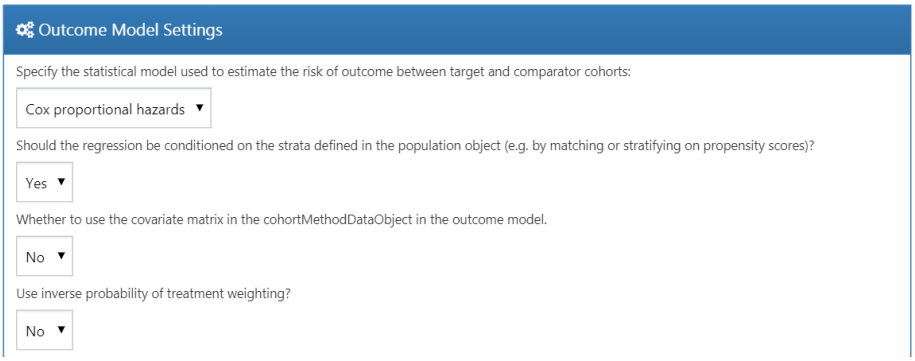


Figure 13.15: Outcome model settings.

to **use inverse probability of treatment weighting** (IPTW). If weighting is used it is often recommended to use some for 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 (ie. the matched sets).

13.3.3 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 13.3.1 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 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

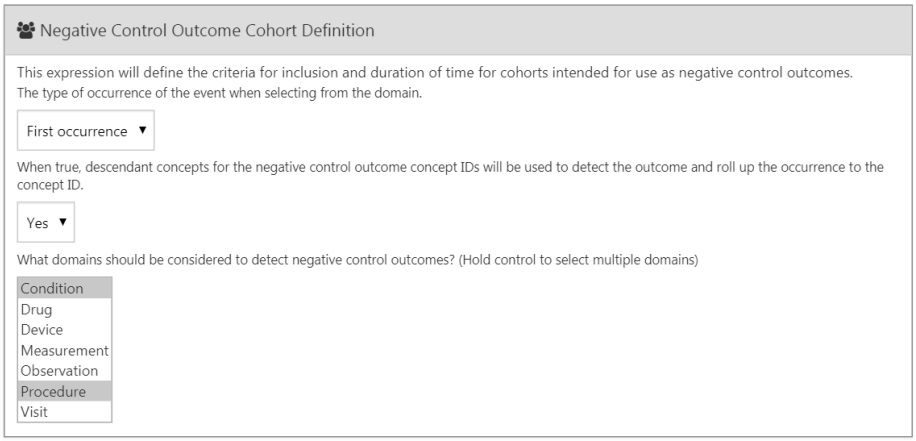


Figure 13.16: Negative control outcome cohort definition settings.

described in Chapter 19. Positive control synthesis is an advanced topic that we will skip for now.

13.3.4 Running the study package

Now that we have fully defined our study, we can export it as a fully executable R package. This package contains everything that is needed to execute the study at a site that has data in the 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 Studio project with a README file that describes the steps needed to execute the analysis.

13.4 Implementation 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 will only consider angioedema as outcome, and leave implementation for AMI and the negative controls as an exercise for the reader.

13.4.1 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 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.

13.4.2 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"
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"`.

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 = cohortsDbSchema,
                                exposureTable = cohortsDbTable,
                                outcomeDatabaseSchema = cohortsDbSchema,
                                outcomeTable = cohortsDbTable,
                                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 13.1.1 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.

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 make sure 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 for example 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.

13.4.3 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 = 30` 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          ...
```

13.4.4 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` 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.

13.4.5 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 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
```

13.4.6 Running multiple analyses

Here we describe performing the analysis for one target, comparator and outcome, using one set of analysis settings. However, often we want to perform more analyses, for example for multiple outcomes including negative controls. The CohortMethod offers functions for performing such studies efficiently. This is described in the package vignette on running multiple analyses.

13.5 Study outputs

Our estimate is 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 by specific R functions.

13.5.1 Propensity score 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

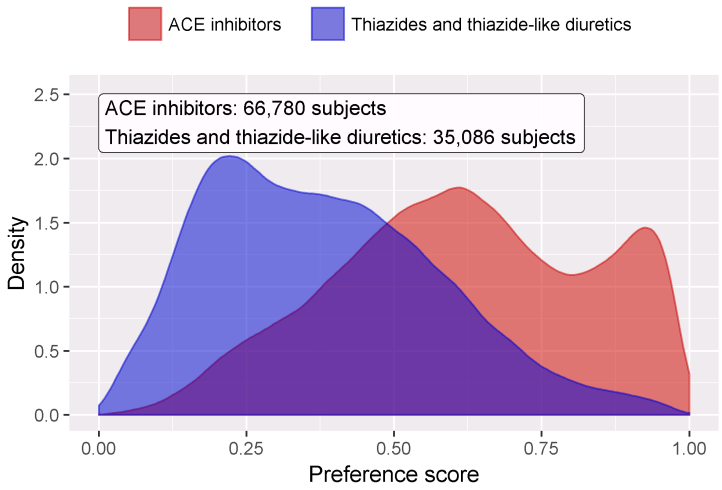


Figure 13.17: Preference score distribution.

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 make the groups comparable.

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 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

Beta	Covariate
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 and should be removed from the model, or else we must conclude that the two populations are truly incomparible, and the analysis must be stopped.

13.5.2 Covariate balance

The goal of using PS is to make the two groups comparable. We must verify whether this is achieved, for example by checked whether the baseline covariates are indeed balanced after adjustment. We can use the `computeCovariateBalance` and `plotCovariateBalanceScatterPlot` functions to generate Figure13.18. One rule-of-thumb to use is that no covariate may have an absolute standardized difference of means greater than 0.1 after propensity score adjustment. Here we see that although there was substantial imbalance before matching, after matching we meet this criterion.

13.5.3 Follow up and power

Before fitting an outcome model, we might be interested to know whether we have sufficient power to detect a particular effect size. It makes sense to perform these power calculations once the study population has been fully defined, so taking into account loss to the various inclusion and exclusion criteria (such as no prior outcomes), and loss due to matching and/or trimming. We can view the attrition of subjects in our study using the `drawAttritionDiagram` function as shown in Figure13.19.

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

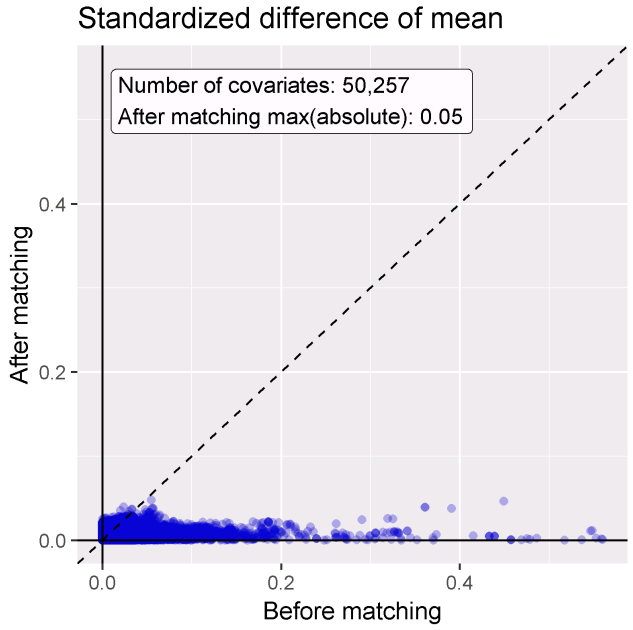


Figure 13.18: Covariate balance, showing the absolute standardized difference of mean before and after propensity score matching. Each blue dot represents a covariate.

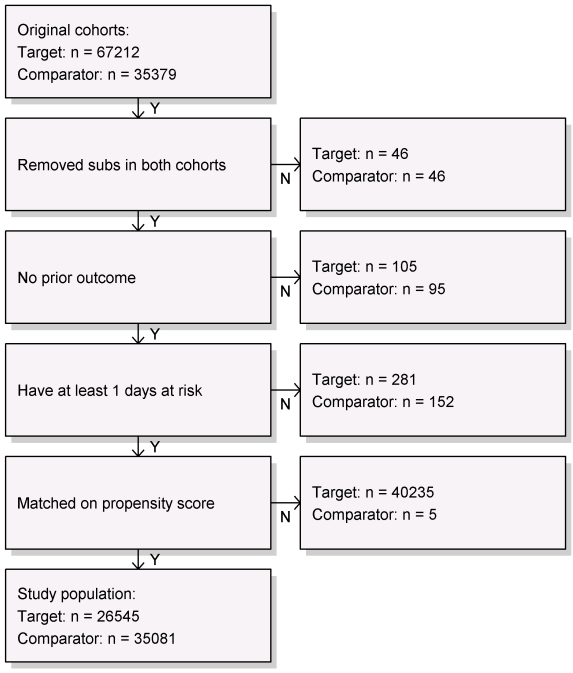


Figure 13.19: Attrition diagram. The counts shown at the top are those that meet our target and comparator cohort definitions. The counts at the bottom are those that enter our outcome model, in this case a Cox regression.

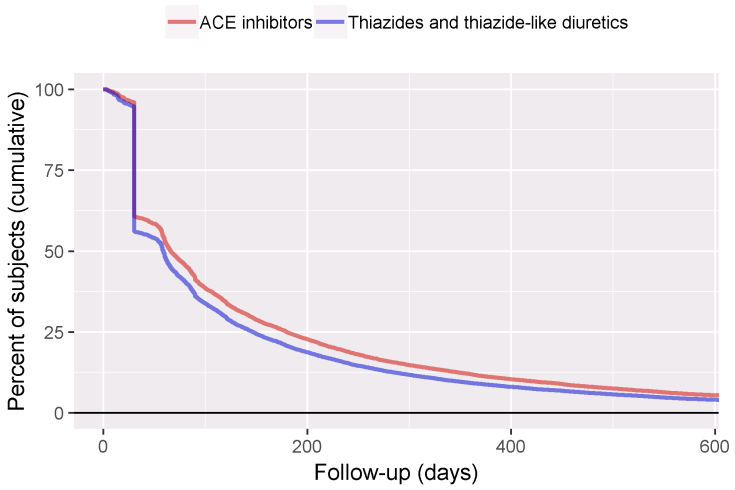


Figure 13.20: Distribution of follow-up time for the target and comparator cohorts.

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.

13.5.4 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.

13.5.5 Effect size estimate

We observe a hazard ratio of 4.32 (95% confidence interval: 2.45 - 8.08), which tells us that ACEi appear to increase the risk of angioedema compared to THZ. Our diagnostics, as reviewed earlier, do not show any major concerns, but ultimately the quality of this evidence, and whether we choose to trust it, depends on many factors, as described in Chapter 15.

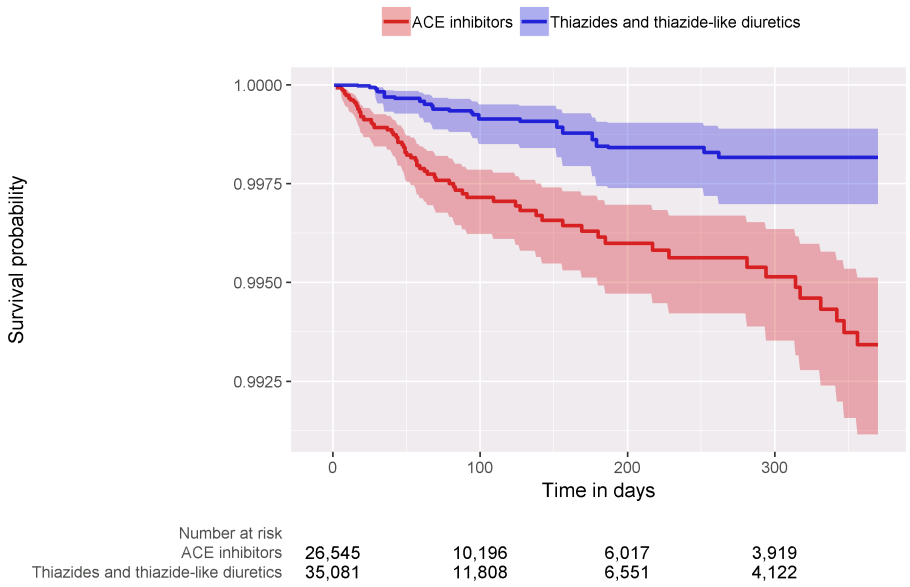


Figure 13.21: Kaplan Meier plot.

13.6 Excercises

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

Chapter 14

Patient Level Prediction

Chapter leads: Peter Rijnbeek & Jenna Reps

Clinical decision making is a complicated task in which the clinician has to infer a diagnosis or treatment pathway based on the available medical history of the patient and the current clinical guidelines. Clinical prediction models have been developed to support this decision making process and are used in clinical practice in a wide spectrum of specialties. These models predict a diagnostic or prognostic outcome based on a combination of patient characteristics, e.g. demographic information, disease history, treatment history. The number of publications describing clinical prediction models has increased strongly over the last 10 years. 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 predictive performance of the models and reduces the likelihood of the model being used appropriately in clinical practice. To improve standards, several papers have been written detailing guidelines for best practices in developing and reporting prediction models.

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

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

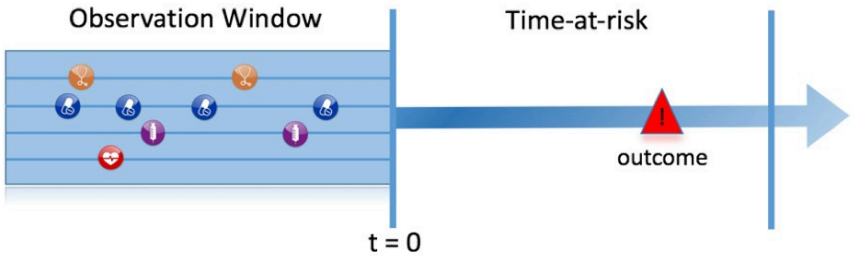


Figure 14.1: The prediction problem.

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

Table 14.1: Main design choices in a prediction design.

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

This conceptual framework works for all type of prediction problems:

- 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

  - **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?

In the next sections we will explain the best practices for model specification, implementation, and evaluation using OHDSI's Patient-Level Prediction (PLP) framework as guidance.

14.1 Designing a hypertension study

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.

14.1.1 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?

14.1.2 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.

14.1.3 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:

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.

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.

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.

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).

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.

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).

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, minImpuritySplit (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").

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.

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.

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.

14.1.4 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.

14.1.5 Study summary

We now completely defined our study as shown in Table 14.2.

Table 14.2: 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.

14.2 Implementing the study in R

Now we have completely designed our study we have to implement the study. 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.

14.2.1 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.

14.2.2 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.

14.2.3 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"
)
```

14.2.4 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:

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

14.3 Implementing the study in ATLAS

The script we created manually above can also be automatically created using a powerful feature in ATLAS. By creating a new prediction study (left menu) you can select the Target and Outcome as created in ATLAS, set all the study parameters, and then you can download a R package that you can use to execute your study. What is really powerful is that you can add multiple Ts, Os, covariate settings etc. The package will then run all the combinations of automatically as separate analyses. The screenshots below explain this process.

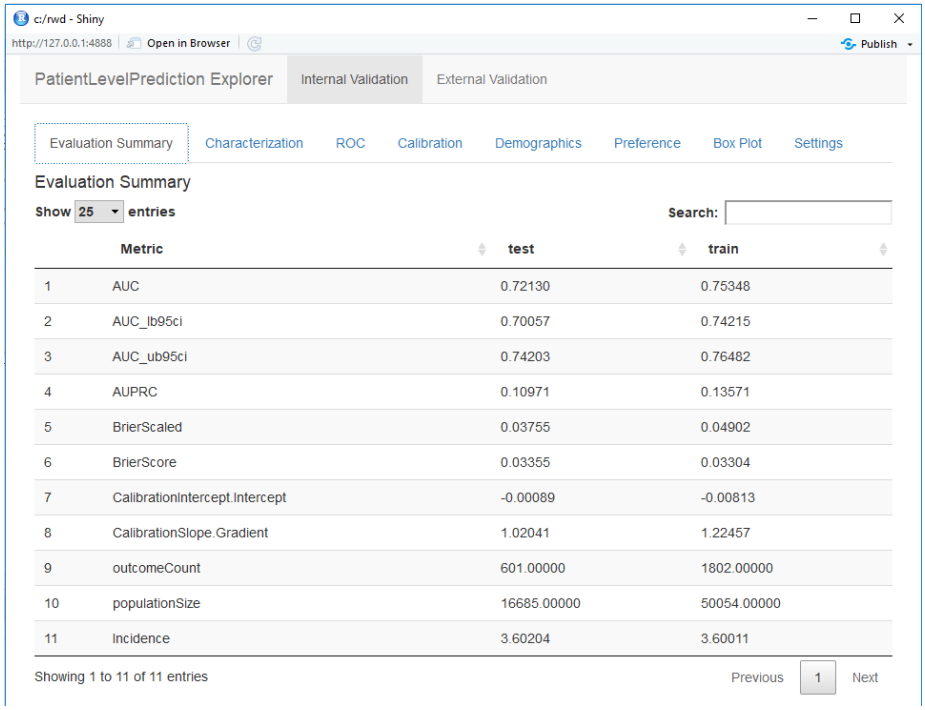
Todo: add description of how to implement study using ATLAS

By opening the R package in R studio and building the package you can run the study using the `execute` function. Theres is also an example `CodeToRun.R` script available in the `extras` folder of the package with extra instructions.

14.4 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 **??**.

Todo: update Shiny app screenshot with hypertension example



To generate and save all the evaluation plots to a folder run the following code:

```
plotPlp(gbmResults, "plots")
```

The plots are described in more detail in the next sections.

14.4.1 Discrimination

The Receiver Operating Characteristics (ROC) plot shows the sensitivity against 1-specificity on the test set. The plot illustrates how well the model is able to discriminate between the people with the outcome and those without. The dashed diagonal line is the performance of a model that randomly assigns predictions. The higher the area under the ROC plot the better the discrimination of the model. Figure 14.2 is created by changing the probability threshold to assign the positive class.

Todo: update plots with hypertension example

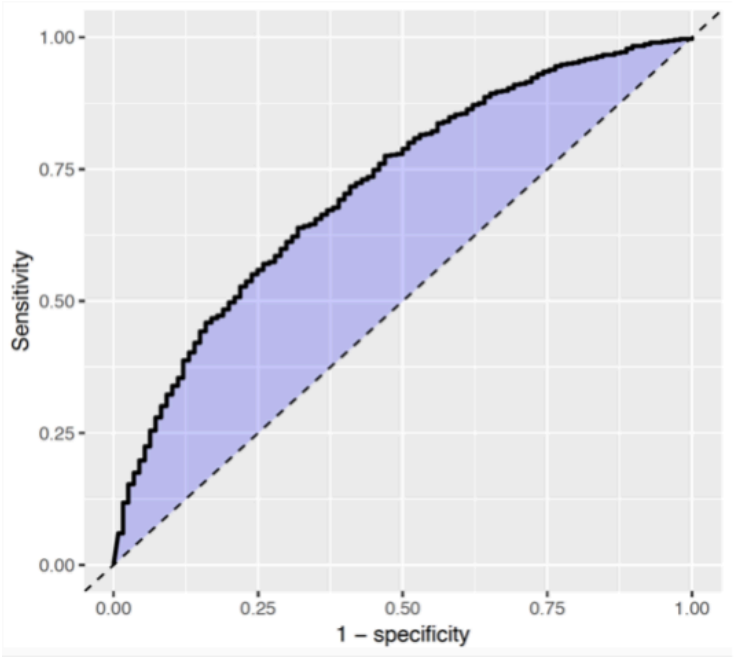


Figure 14.2: The Receiver Operating Characteristics (ROC) curve.

14.4.2 Calibration

The calibration plot (Figure 14.3) shows how close the predicted risk is to the observed risk. 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.

14.4.3 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

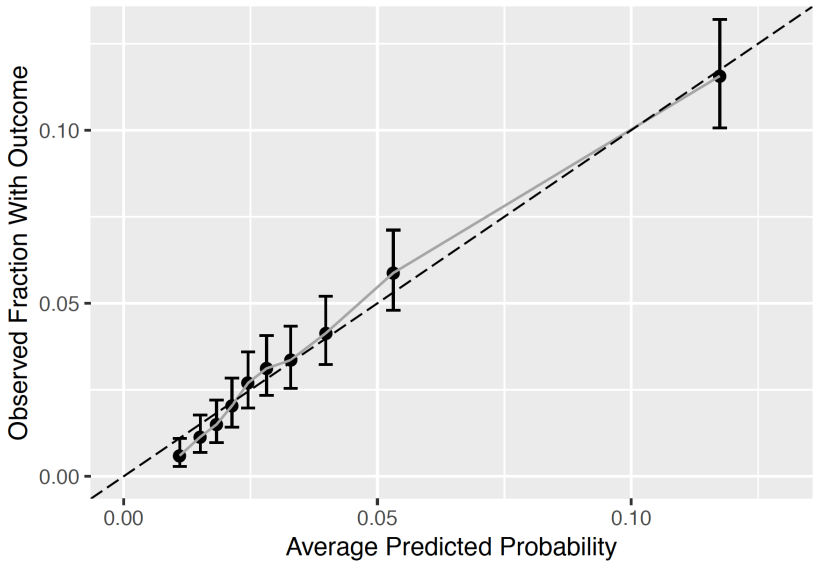


Figure 14.3: Calibration plot.

to predicted risk.

However, the increased information gain comes at a computational cost. It is recommended to use the traditional plot for examination and then to produce the smooth plot for final versions. To create the smooth calibarion plot you have to run the follow command:

```
plotSmoothCalibration(gbmResults)
```

See the help function for more information, on how to set the smoothing method etc.

Figure 14.4 shows an example from another study 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.

14.4.4 Preference distribution

The preference distribution plot (Figure 14.5) shows the preference score distributions for people in the test set with the outcome (red) without the outcome (blue).

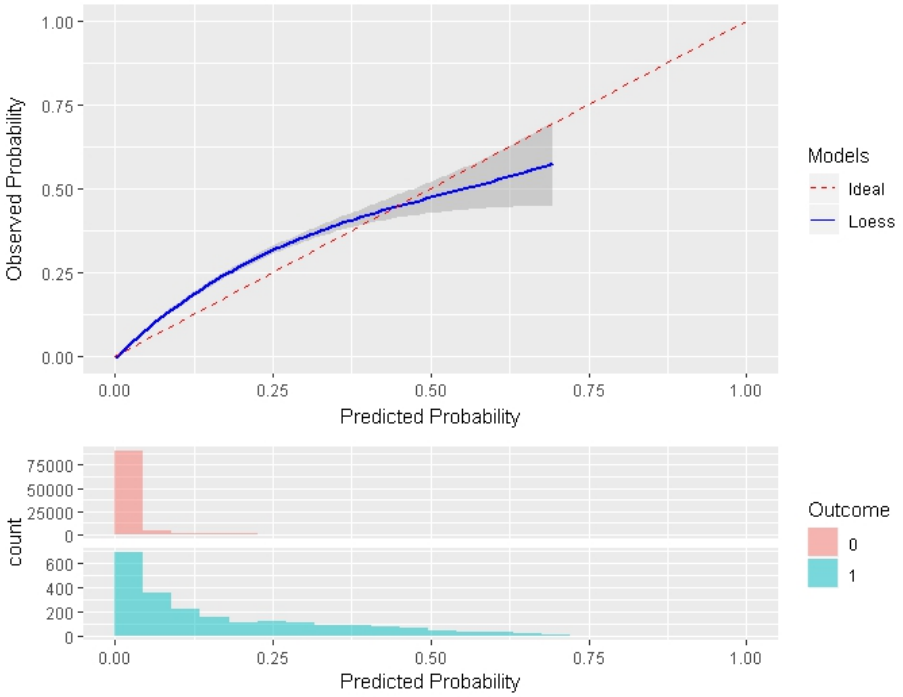


Figure 14.4: Smooth calibration plot.

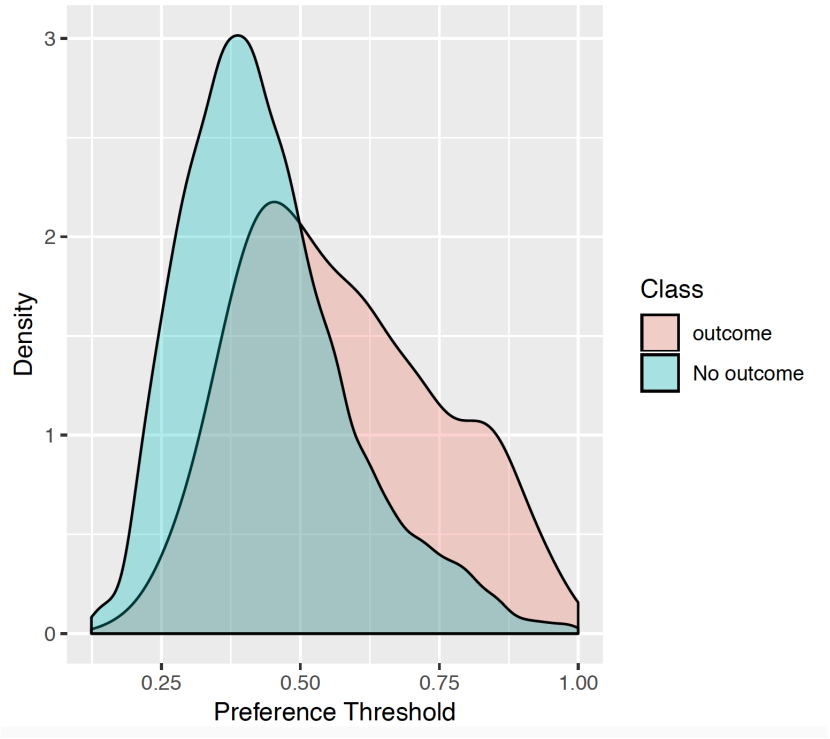


Figure 14.5: Preference distribution plot.

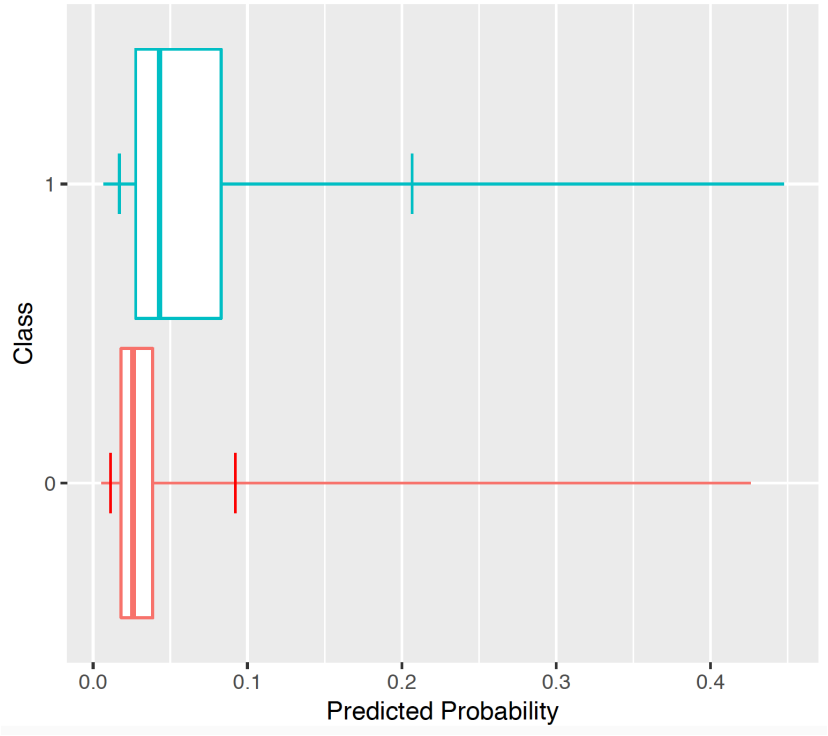


Figure 14.6: Predicted probability distribution.

14.4.5 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).

The box plots in Figure 14.6 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.

14.4.6 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.

The results in Figure 14.7 for our example look very promising since the mean values of the covariates are on the diagonal.

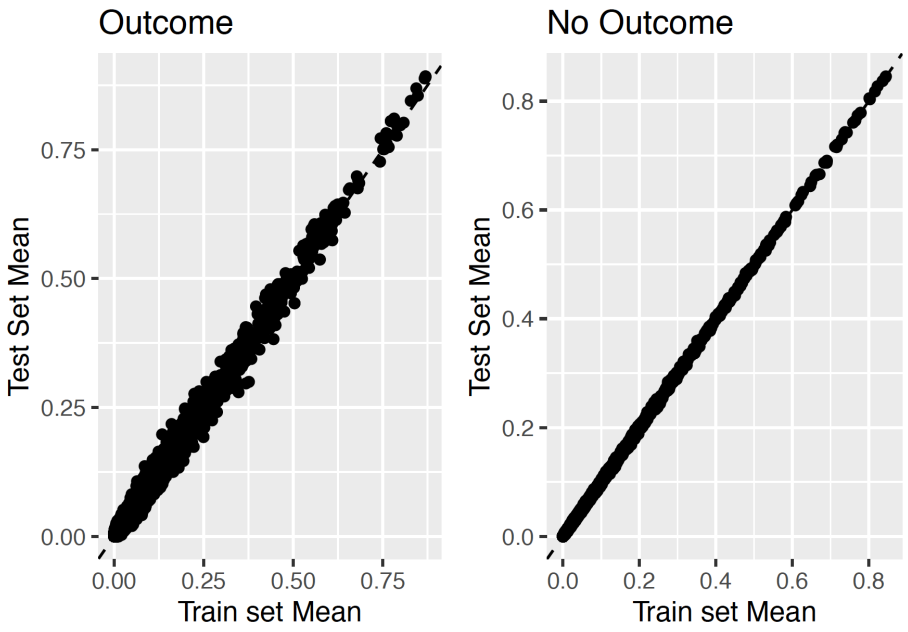


Figure 14.7: Predicted probability distribution.

14.4.7 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. It is highly recommended to use the Shiny App since this allows you to hoover over a covariate to show more details (name, value etc).

Figure 14.8 shows that the mean of most of the covariates is higher for subjects with the outcome compared to those without.

14.4.8 Precision recall

Precision (P) is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP):

$$P = \frac{TP}{TP + FP}$$

Recall (R) is defined as the number of true positives over the number of true positives plus the number of false negatives (FN):

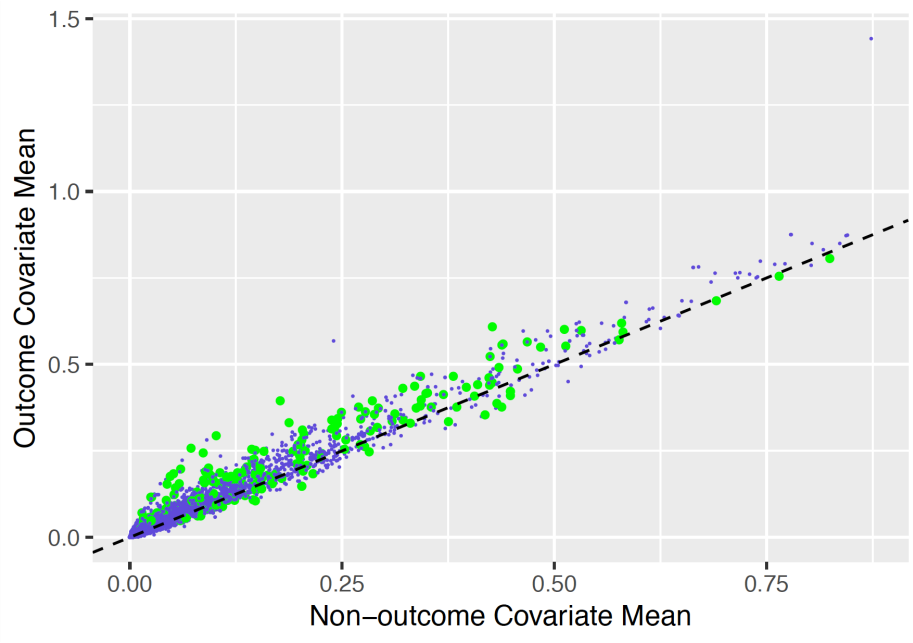


Figure 14.8: Predicted probability distribution.

$$R = \frac{TP}{TP + FN}$$

These quantities are also related to the (F1) score, which is defined as the harmonic mean of precision and recall.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

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

Figure 14.9 shows the tradeoff between precision and recall.

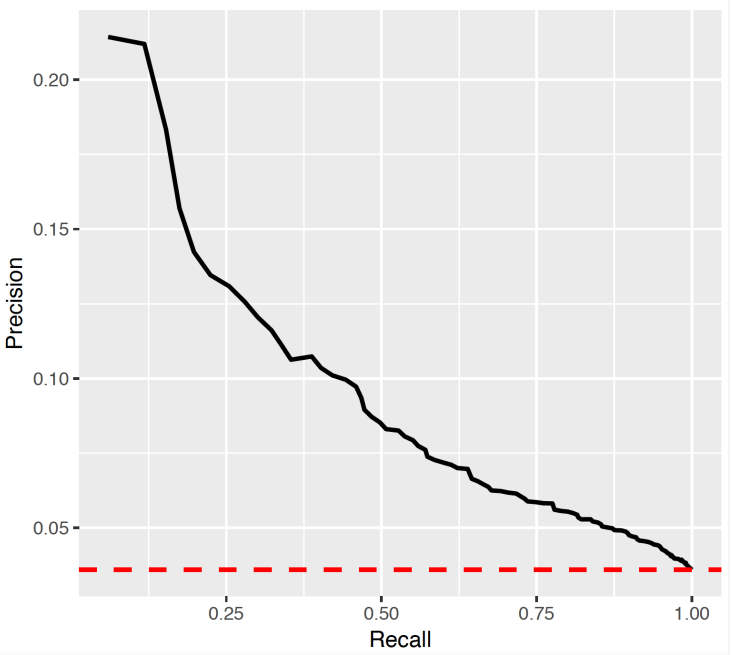


Figure 14.9: Precision-recall plot.

14.4.9 Demographic summary

Figure 14.10 shows for females and males the expected and observed risk in different age groups together with a confidence area. The results show that our model is well calibrated across gender and age groups.

14.5 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")
```

Figure 14.10: Precision-recall plot.

```
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. This function is described in the package manual.

14.6 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.

14.7 Excercises

Part IV

Evidence Quality

Chapter 15

Evidence Quality

> 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

OHDSI views validation as a holistic set of processes necessary to achieve the highest quality reproducible evidence from diverse data sources.

Four components: - Data quality (data validation) - Clinical validity - Software validity - Method validity

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 als 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 a wide range of standardized diagnostics that should be generated and evaluated whenever an analysis is performed. Some of these diagnostics are specific to certain study designs, whereas others are more generic.

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 **??**studyOutputs) lists a wide range of diagnostics generated by the CohortMethod package, including:

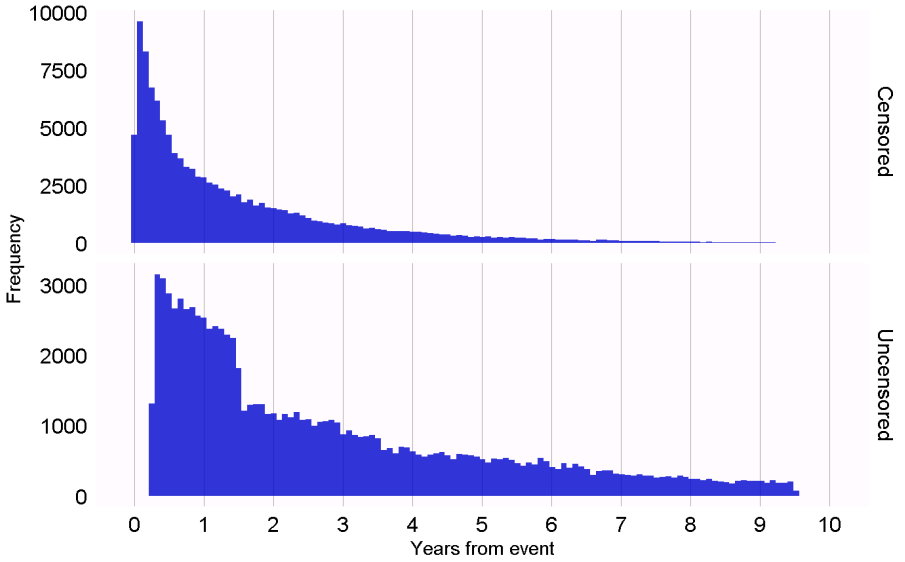


Figure 19.1: Time to observation end for those that are censored, and those that uncensored.

- **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, 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.

Similarly, for the self-controlled case series (SCCS) design we may check other assumptions that are relevant for that design. On such assumptionis 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 a histograms of the time to obsevation 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

No matter what estimation design we use, we should always asses performance using control hypotheses, research questions where the answer is already known. 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) investigate 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). In OHDSI 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 outcomes with no known links with any hypertension treatment. We rank-order this list by prevalence in an observational database and manually review these in order.

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).

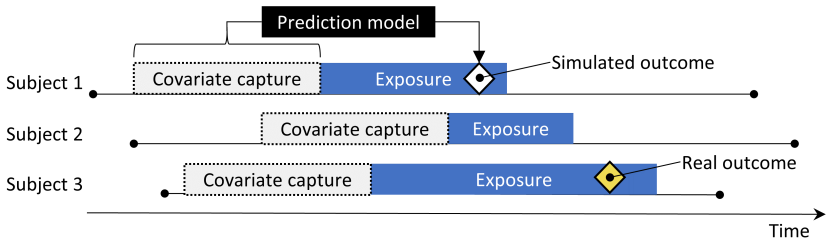


Figure 19.2: Synthesizing positive controls from negative controls.

In OHDSI we therefore use synthetic positive controls (Schuemie et al., 2018), created by modifying a negative control through injection of additional, simulated occurrences of the outcome during the time at risk of the exposure. 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 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 Metrics

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 can 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 I and type II error, or if we are willing to modify our alpha threshold, we may inspect the AUC instead.

19.2.4 P-value calibration

Often the type I error (at alpha = 0.05) is larger than 5%. OHDSI has developed a process for calibrating p-values to restore the type I 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. 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:

$$\theta_i \sim N(\mu, \sigma^2) \text{ , and} \hat{\theta}_i \sim N(\theta_i + \beta_i, \tau_i^2)$$

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 p-value for the new pair is then

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

where $\phi(.)$ denotes the cumulative distribution function of the standard normal distribution. When $\hat{\theta}_{n+1}$ is bigger than $\hat{\mu}$, the one-sided p-value is then

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

19.2.5 Confidence interval calibration

For confidence inteval calibration (Schuemie et al., 2018) we estimate a systematic error distribution, which we assume is Gaussian with a mean and standard

deviation linearly related to the logarithm of the true effect size. Using the estimated distribution, we then generate calibrated confidence intervals considering both random and systematic error. 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.

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 can come from executing the study across several different databases that possibly represent different populations, different health care systems, and 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 the design does not adequately address the different biases found in the different databases. However, not observing heterogeneity of effects 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 matchign 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 Diagnostics for all prediction

Todo

19.4 Method validation in practice

Example: risk of angioedema and AMI in new users of ACE inhibitors compared to new users of thiazide and thiazide-like diuretics

How to select negative controls using ATLAS

- Create a concept set containing both target and comparator exposure concepts.
- Go to the 'Explore evidence' tab and click 'Generate'
- Manually review negative controls, considering
- Does the drug not cause the outcome?
- Does the drug not prevent / treat the outcome?
- Does the negative control appear in the data?

Include negative and positive controls.

Compute metrics

- Need to add functions to MethodEvaluation

Generate calibration plots

Calibrate CI and p-value

- Use EmpiricalCalibration package

19.5 Advanced: OHDSI Methods Benchmark

Todo: add text on OHDSI Methods Benchmark

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
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

Concept Id	Concept Name	Excluded	Descendants	Mapped
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
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

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
1351557	candesartan	NO	YES	NO

Concept Id	Concept Name	Excluded	Descendants	Mapped
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

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.

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.

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. 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.

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.

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.

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.

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., 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., 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.

Sabroe, R. A. and Black, A. K. (1997). Angiotensin-converting enzyme (ACE) inhibitors and angio-oedema. *Br. J. Dermatol.*, 136(2):153–158.

Schuemie, M. J., Hripcsak, G., Ryan, P. B., Madigan, D., and Suchard, M. A. (2018). 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.

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.

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.

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.

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.

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.

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.