python-ohdsi

F.C.Martin

Feb 15, 2024

CONTENTS

Warning: Please note that this python package is not endorsed or affiliated with the official OHDSI community. It is a project intended for enabling vantage6 to connect to an OMOP CDM database.

This package includes python wrappers for the following OHDSI libraries:

- CirceR
- SqlRender
- Cohort Generator
- Database Connector
- Feature Extraction

This packages contains, besides the python interfaces for the OHDSI libraries, a small RestAPI to interact with the libraries.

There are three use-cases to use this package:

1. Use it as a Python interface to interact with an OMOP data source.
2. Let vantage6 connect with an OMOP data source through an SQL connection
3. Let vantage6 connect through a HTTP connection with an OMOP data source

CHAPTER

ONE

CASE 1: PYTHON OMOP INTERFACE

You can use the packages to interact with the OMOP data source from your Python environment. Note that all function are formatted in snake case (instead of the camel-case used in the R packages). For example to execute a simple query:

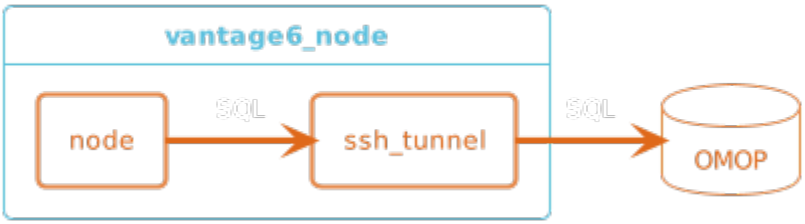
```
from ohdsi import database_connector

connection_details = database_connector.create_connection_details(
    "postgresql",
    server="localhost/postgres",
    user="postgres",
    password="some-password",
    port=5432
)
con = database_connector.connect(connection_details)

database_connector.execute_sql(con, "SELECT * FROM omopcdm.person LIMIT 3")
```

CHAPTER

TWO

CASE 2: VANTAGE6 SQL INTERFACE

The vantage6 algorithm wrapper can directly connect with an OMOP instance through a SQL connection. Important to note that the algorithm container of vantage6 is not able to reach anything outside of its own Docker network. It is required to setup an SSH Tunnel to the machine that hosts the OMOP database. An SSH tunnel brings additional risks, therefore using the API model *vantage6-http-interface* is preferred.

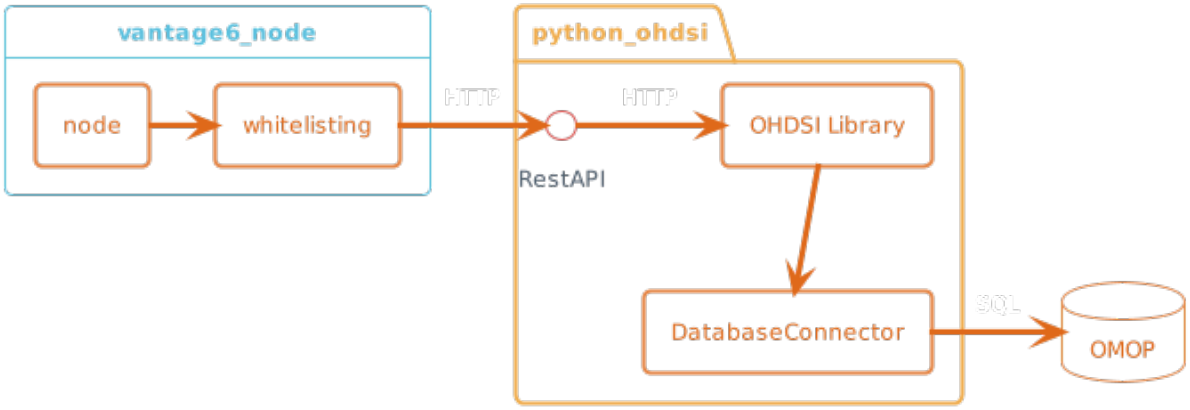


CHAPTER

THREE

CASE 3: VANTAGE6 HTTP INTERFACE

The vantage6 wrapper can also use the RestAPI (included in this package) to retrieve data from the OMOP source. Important to note that the algorithm container of vantage6 is not able to reach anything outside of its own Docker network. It is required to setup Whitelisting to the IP/hostname and port of the machine that hosts the RestAPI.



CHAPTER

FOUR

TABLE OF CONTENTS

4.1 Changelog

4.1.1 0.3.0

15 August 2023 hi .. - **Change** .. - . . .

4.2 Circe

The section bellow is adjusted from the Circe docs.

4.2.1 Introduction

A python-wrapper for Circe, a library for creating queries for the OMOP Common Data Model. These queries are used in cohort definitions (CohortExpression) as well as custom features (CriteriaFeature). This package provides convenient wrappers for Circe functions, and includes the necessary Java dependencies.

4.2.2 Features

- Convert a JSON cohort expression into a markdown print-friendly presentation.
- Convert a JSON cohort expression into SQL.

4.2.3 Installation

Install the python wrapper with pip:

```
pip install ohdsi-circe
```

Since this package only wraps the CirceR both Java and R are required. Full instructions on how to configure your R and Java environment can be found in the OHDSI documentation: Setting up the R environment.

On debian-based systems, install the dependencies boils down to:

```
sudo apt-get update
sudo apt-get install r-base openjdk-17-jdk openjdk-17-jre
```

```
install.packages("remotes")
remotes::install_github("OHDSI/CirceR")
```

4.2.4 Function Reference

build_cohort_query(*cohort_expression*, *options=None*)

Build Cohort SQL

Generates the OMOP CDM Sql to generate the cohort expression.

Wraps the R `CirceR::buildCohortQuery` function defined in `CirceR/R/CohortSqlBuilder.R`.

Parameters

- **cohort_expression** (*RS4 | str*) – An R object or a JSON string containing the cohort expression
- **options** (*RS4, optional*) – The options object from `create_generate_options`, by default None

Returns
contains the SQL statements

Return type
StrVector

build_concept_set_query(*concept_set_expression*)

Generates the OMOP CDM Sql to resolve the concept set expression

Wraps the R `CirceR::buildConceptSetQuery` function defined in `CirceR/R/ConceptSetSqlBuilder.R`.

Parameters
concept_set_expression (*str | dict*) – a string containing the JSON for the conceptset expression.

Returns
OHDSI Sql for the conceptset expression

Return type
StrVector

cohort_expression_from_json(*expression_json*)

Render read JSON into a R CohortExpression instance.

Reads a String (json) and deserializes it into a `CohortExpression`.

Wraps the R `CirceR::cohortExpressionFromJson `` function defined in ``CirceR/R/CohortExpression.R`.

Parameters
expression_json (*str | dict*) – A JSON `str` or a `dict` representing a cohort expression

Returns
A wrapped cohort expression R object

Return type
RS4

Examples

```
>>> from importlib.resources import files
>>> cohort_str = files('ohdsi.circe.data').joinpath('simpleCohort.json')
...     .read_text()
>>> cohort = CohortExpression.cohort_expression_from_json(cohort_str)
```

cohort_print_friendly(*expression*)

Create a print friendly version of a cohort expression.

Wraps the R `CirceR::cohortPrintFriendly` function defined in `CirceR/R/PrintFriendly.R`.

Parameters
expression (*RS4 | dict | str*) – A str, dict or result of `cohort_expression_from_json` containing the cohort expression.

Returns
A character vector containing the print friendly version of the cohort expression.

Return type
StrVector

concept_set_expression_from_json(*expression_json*)

Read JSON into a ConceptSetExpression instance.

Reads a String (json) and deserializes it into a `ConceptSetExpression` as the R library is a wrapper around the Java library.

Wraps the R `CirceR::conceptSetExpressionFromJson` function defined in `CirceR/R/ConceptSetExpression.R`.

Parameters
concept_set (*str | dict*) – A JSON `str` or a `dict` representing a concept set expression

Returns
A wrapped concept set expression R object

Return type
RS4

concept_set_list_print_friendly(*concept_set_list*)

Render conceptSet array for print-friendly

Generates a print-friendly (human-readable) representation of an array of concept sets. This can for example be used in a study protocol.

Wraps the R `CirceR::conceptSetListPrintFriendly` function defined in `CirceR/R/PrintFriendly.R`.

Parameters
expression (*str | dict*) – A JSON `str` or a `dict` representing a concept set list

Returns
A character vector containing the print friendly version of the concept set expression.

Return type
StrVector

concept_set_print_friendly(*concept_set*)

Create a print friendly version of a concept set expression

Wraps the R `CirceR::conceptSetPrintFriendly` function defined in `CirceR/R/PrintFriendly.R`.

Parameters
expression (*str | json*) – A JSON `str` or a `dict` representing a concept set

Returns
A character vector containing the print friendly version of the concept set expression.

Return type
StrVector

`create_generate_options`(*cohort_id_field_name=None*, *cohort_id=None*, *cdm_schema=None*, *target_table=None*, *result_schema=None*, *vocabulary_schema=None*, *generate_stats=None*)

Create Generation Options.

Creates the generation options object for use in `build_cohort_query`

Wraps the R `CirceR::createGenerateOptions` function defined in `CirceR/R/CohortSqlBuilder.R`.

Parameters

- **cohort_id_field_name** (*str, optional*) – The field that contains the cohortId in the cohort table, by default None
- **cohort_id** (*int, optional*) – The generated cohort ID, by default None
- **cdm_schema** (*str, optional*) – The value of the CDM schema, by default None
- **target_table** (*str, optional*) – The cohort table name, by default None
- **result_schema** (*str, optional*) – The schema the cohort table belongs to, by default None
- **vocabulary_schema** (*str, optional*) – The schema of the vocabulary tables (defaults to cdmSchema), by default None
- **generate_stats** (*bool, optional*) – A boolean representing if the query should include inclusion rule statistics calculation, by default None

Returns
A wrapped generation options R object

Return type
RS4

Examples

```
>>> options = create_generate_options()
>>> options = create_generate_options(
...     cohort_id_field_name='cohort_definition_id',
...     cohort_id=1,
...     cdm_schema='cdm',
...     target_table='cohort',
...     result_schema='results',
...     vocabulary_schema='cdm',
...     generate_stats=True
... )
```

4.3 SQL Render

get_temp_table_prefix()

Get the temporary table prefix

Used for emulated temp tables for DBMSs that do not support temp tables (e.g. Oracle, BigQuery).

Wraps the R `SqlRender::getTempTablePrefix` function defined in `SqlRender/R/RenderSql.R`.

Returns
The prefix used for emulated temp tables

Return type
str

Examples

```
>>> get_temp_table_prefix()
```

list_supported_dialects()

List the supported dialects

Wraps the R `SqlRender::listSupportedDialects` function defined in `SqlRender/R/HelperFunctions.R`.

Returns
A dataframe with the supported dialects

Return type
DataFrame

load_render_translate_sql(*sql_file*, *package_name*, *dbms*, *temp_emulation_schema=None*, *warn_on_missing_parameters=True*)

Load, render, and translate a SQL file in a package

`loadRenderTranslateSql` Loads a SQL file contained in a package, renders it and translates it to the specified dialect.

Wraps the R `SqlRender::loadRenderTranslateSql` function defined in `SqlRender/R/HelperFunctions.R`.

Parameters

- **sql_file** (*str | Path*) – The source SQL file
- **package_name** (*str*) – The package name
- **dbms** (*str*) – The target dialect. Currently "oracle", "postgresql", "pdw", "impala", "sqlite", "sqlite extended", "netezza", "bigquery", "snowflake", "synapse", "spark", and "redshift" are supported. Use `list_supported_dialects` to get the list of supported dialects.
- **temp_emulation_schema** (*str | None*) – Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
- **warn_on_missing_parameters** (*bool, optional*) – If `True`, a warning is issued if a parameter is not found in the parameter list, by default `True`

Return type
`StrVector`

read_sql(*source_file*)

loads SQL from a file

Wraps the R `SqlRender::readSql` function defined in `SqlRender/R/HelperFunctions.R`.

Parameters
source_file (*str | Path*) – The source SQL file

Returns
The SQL from the file

Return type
StrVector

render(*sql*, *warn_on_missing_parameters=True*, ***kwargs*)

Renders SQL code based on parameterized SQL and parameter values

This function takes parameterized SQL and a list of parameter values and renders the SQL that can be send to the server.

Wraps the R `SqlRender::render` function defined in `SqlRender/R/RenderSql.R`.

Parameters

- **sql** (*str*) – The parameterized SQL.
- **warnOnMissingParameters** (*bool*) – Should a warning be raised when parameters provided to this function do not appear in the parameterized SQL that is being rendered? By default, this is True
- **kwargs** (*dict*) – The parameter values.

  param sql
  The parameterized SQL.

  type sql
  str

Return type
`StrVector`

Examples

```
>>> sql = "SELECT * FROM table WHERE id = @id"
>>> render("SELECT * FROM @a;", a = "myTable")
```

render_sql_file(*source_file*, *target_file*, *warnOnMissingParameters=True*, ***kwargs*)

Render a SQL file

Renders SQL code in a file based on parameterized SQL and parameter values, and writes it to another file.

Wraps the R `SqlRender::renderSqlFile` function defined in `SqlRender/R/HelperFunctions.R`.

Parameters

- **source_file** (*str | Path*) – The source SQL file
- **target_file** (*str | Path*) – The target SQL file
- **warnOnMissingParameters** (*bool, optional*) – If `True`, a warning is issued if a parameter is not found in the parameter list, by default `True`
- **kwargs** – The parameters to use for rendering

Return type
None

Examples

```
>>> renderSqlFile(
>>>     "myParamStatement.sql",
>>>     "myRenderedStatement.sql",
>>>     a = "myTable"
>>> )
```

spark_handle_insert(*sql*, *connection*)

Handles Spark Inserts

This function is for Spark connections only, it handles insert commands, as Spark cannot handle inserts with aliased or subset columns.

Wraps the R `SqlRender::sparkHandleInsert` function defined in `SqlRender/R/SparkSql.R`.

Parameters

- **sql** (*str*) – The SQL to be translated.
- **connection** (*str*) – The connection to the database server.

Returns
A sql string with INSERT command modified to contain the full column list, padded with NULLS as needed.

Return type
StrVector

split_sql(*sql*)

Split a single SQL string into one or more SQL statements

`splitSql` splits a string containing multiple SQL statements into a vector of SQL statements. This function is needed because some DBMSs (like ORACLE) do not accepts multiple SQL statements being sent as one execution.

Wraps the R `SqlRender::splitSql` function defined in `SqlRender/R/RenderSql.R`.

Parameters
sql (*str*) – The SQL string to split into separate statements

Returns
A vector of strings, one for each SQL statement

Return type
str

Examples

```
>>> split_sql("SELECT * INTO a FROM b; USE x; DROP TABLE c;")
```

translate(*sql*, *target_dialect*, *temp_emulation_schema=None*)

Translates SQL from one dialect to another

Wraps the R `SqlRender::translate` function defined in `SqlRender/R/RenderSql.R`.

Parameters

- **sql** (*str*) – The SQL to be translated
- **target_dialect** (*str*) – The target dialect. Currently "oracle", "postgresql", "pdw", "impala", "sqlite", "sqlite extended", "netezza", "bigquery", "snowflake", "synapse", "spark", and "redshift" are supported. Use `list_supported_dialects` to get the list of supported dialects.
- **temp_emulation_schema** (*str | None*) – Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

Returns
The translated SQL.

Return type
StrVector

translate_single_statement(*sql*, *target_dialect*, *temp_emulation_schema=None*)

Translates a single SQL statement from one dialect to another

This function takes SQL in one dialect and translates it into another. It uses simple pattern replacement, so its functionality is limited. This removes any trailing semicolon as required by Oracle when sending through JDBC. An error is thrown if more than one statement is encountered in the SQL.

Wraps the R `SqlRender::translateSingleStatement` function defined in `SqlRender/R/RenderSql.R`.

Parameters

- **sql** (*str*) – The SQL to be translated
- **target_dialect** (*str*) – The target dialect. Currently "oracle", "postgresql", "pdw", "impala", "sqlite", "sqlite extended", "netezza", "bigquery", "snowflake", "synapse", "spark", and "redshift" are supported. Use `list_supported_dialects` to get the list of supported dialects.
- **temp_emulation_schema** (*str | None*) – Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

Returns
The translated SQL.

Return type
str

Examples

```
>>> translate_single_statement(
>>>     "USE my_schema;",
>>>     targetDialect = "oracle"
>>> )
```

translate_sql_file(*source_file*, *target_file*, *target_dialect*, *temp_emulation_schema=None*)

Translate a SQL file

This function takes SQL and translates it to a different dialect.

Wraps the R `SqlRender::translateSqlFile` function defined in `SqlRender/R/HelperFunctions.R`.

Parameters

- **source_file** (*str | Path*) – The source SQL file
- **target_file** (*str | Path*) – The target SQL file
- **target_dialect** (*str*) – The target dialect. Currently "oracle", "postgresql", "pdw", "impala", "sqlite", "sqlite extended", "netezza", "bigquery", "snowflake", "synapse", "spark", and "redshift" are supported. Use `list_supported_dialects` to get the list of supported dialects.
- **temp_emulation_schema** (*str | None*) – Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

Return type

None

Examples

```
>>> translateSqlFile(
>>>     "mySql.sql",
>>>     "myTranslatedSql.sql",
>>>     targetDialect = "oracle"
>>> )
```

write_sql(*sql*, *file*)

Write SQL to a SQL (text) file

Wraps the R `SqlRender::writeSql` function defined in `SqlRender/R/HelperFunctions.R`.

Parameters

- **sql** (*str*) – The SQL to write
- **file** (*str | Path*) – The target SQL file

Return type

None

Examples

```
>>> write_sql("SELECT * FROM table;", "my_sql.sql")
```

4.4 Cohort generator

create_cohort_tables(*cohort_database_schema*, *connection_details=None*, *connection=None*, *cohort_table_names=None*, *incremental=False*)

Create cohort tables

Creates the cohort tables in the database.

Wraps the R `CohortGenerator::createCohortTables` function defined in `CohortGenerator/R/CohortTables.R`.

Parameters

- **cohort_database_schema** (*str*) – The schema to create the cohort tables in
- **connection_details** (*ListVector, optional*) – The connection details, by default None
- **connection** (*RS4, optional*) – The connection, by default None
- **cohort_table_names** (*ListVector, optional*) – The names of the cohort tables, by default None
- **incremental** (*bool, optional*) – A boolean representing if the tables should be created incrementally, by default False

Returns
A wrapped cohort generation R object

Return type
RS4

create_empty_cohort_definition_set(*verbose=False*)

Create an empty CohortDefinitionSet object

Wraps the R `CohortGenerator::createEmptyCohortDefinitionSet` function defined in `CohortGenerator/R/CohortDefinitionSet.R`.

Parameters
verbose (*bool, optional*) – When True, descriptions of each field in the data frame are returned. By default False.

Return type
RS4

export_cohort_stats_tables(*cohort_database_schema*, *cohort_statistics_folder*, *connection_details=None*, *connection=None*, *cohort_table_names=None*, *snake_case_to_camel_case=True*, *file_names_in_snake_case=False*, *incremental=False*, *database_id=None*)

Export the cohort statistics tables to the file system

This function retrieves the data from the cohort statistics tables and writes them to the inclusion statistics folder specified in the function call.

Parameters

- **cohort_database_schema** (*str*) – The schema to create the cohort tables in
- **cohort_statistics_folder** (*str*) – The path to the folder where the cohort statistics folder where the results will be written
- **connection_details** (*ListVector, optional*) – The connection details, by default None
- **connection** (*RS4, optional*) – The connection, by default None
- **cohort_table_names** (*ListVector, optional*) – The names of the cohort tables, by default None
- **snake_case_to_camel_case** (*bool, optional*) – Should column names in the exported files convert from snake_case to camelCase? Default is FALSE
- **file_names_in_snake_case** (*bool, optional*) – Should the exported files use snake_case? Default is FALSE
- **incremental** (*bool, optional*) – A boolean representing if the tables should be created incrementally, by default False
- **database_id** (*str, optional*) – When specified, the databaseId will be added to the exported results

generate_cohort_set(*cdm_database_schema*, *cohort_definition_set*, *connection_details=None*, *connection=None*, *temp_emulation_schema=None*, *cohort_database_schema=None*, *cohort_table_names=None*, *stop_on_error=True*, *incremental=False*, *incremental_folder=None*)

Generate a cohort set

This function generates a set of cohorts in the cohort table.

Wraps the R `CohortGenerator::generateCohortSet` function defined in `CohortGenerator/R/CohortConstruction.R`.

Parameters

- **cdm_database_schema** (*str*) – The schema containing the CDM
- **connection_details** (*ListVector | None, optional*) – The connection details obtained using `Connect.create_connection_details(...)`, by default None
- **connection** (*None, optional*) – The connection object obtained from `Connect.connect(...)`, by default None
- **temp_emulation_schema** (*None, optional*) – The schema to use for temp tables, by default None

Return type
RS4

get_cohort_counts(*cohort_database_schema*, *connection_details=None*, *connection=None*, *cohort_table='cohort'*, *cohort_ids=[]*, *cohort_definition_set=None*, *database_id=None*)

Get cohort counts.

Gets the counts for the specified cohort ids.

Wraps the R `CohortGenerator::getCohortCounts` function defined in `CohortGenerator/R/CohortCount.R`.

Parameters

- **cohort_database_schema** (*str*) – The schema containing the cohort tables

- **connection_details** (*ListVector, optional*) – The connection details, by default None
- **connection** (*RS4, optional*) – The connection, by default None
- **cohort_table** (*str, optional*) – The name of the cohort table, by default "cohort"
- **cohort_ids** (*list[int], optional*) – The cohort ids to get the counts for, by default []
- **cohort_definition_set** (*RS4, optional*) – The cohort definition set, by default None
- **database_id** (*str, optional*) – The database id, by default None

Returns
A wrapped cohort counts R object

Return type
RS4

get_cohort_table_names(*cohort_table='cohort'*, *cohort_inclusion_table=None*, *cohort_inclusion_result_table=None*, *cohort_inclusion_stats_table=None*, *cohort_summary_stats_table=None*, *cohort_censor_stats_table=None*)

Get the names of the cohort tables

Wraps the R `CohortGenerator::getCohortTableNames` function defined in `CohortGenerator/R/CohortTables.R`.

Parameters

- **cohort_table** (*str, optional*) – The name of the cohort table, by default "cohort"
- **cohort_inclusion_table** (*str, optional*) – The name of the cohort inclusion table, by default None
- **cohort_inclusion_result_table** (*str, optional*) – The name of the cohort inclusion result table, by default None
- **cohort_inclusion_stats_table** (*str, optional*) – The name of the cohort inclusion stats table, by default None
- **cohort_summary_stats_table** (*str, optional*) – The name of the cohort summary stats table, by default None
- **cohort_censor_stats_table** (*str, optional*) – The name of the cohort censor stats table, by default None

Returns
A list of cohort table names

Return type
ListVector

save_cohort_definition_set(*cohort_definition_set*, *settings_file_name='inst/cohorts.csv'*, *json_folder='inst/cohorts'*, *sql_folder='inst/sql/sql_server'*, *cohort_file_name_format='%s'*, *cohort_file_name_value=['cohort_id']*, *subset_json_folder='inst/cohort_subset_definitions/'*, *verbose=False*)

Save the cohort definition set to the file system

This function saves a cohort_definition_set to the file system and provides options for specifying where to write the individual elements: the settings file will contain the cohort information as a CSV specified by the settingsFileName, the cohort JSON is written to the jsonFolder and the SQL is written to the sqlFolder. We also provide a way to specify the json/sql file name format using the cohort_file_name_format and cohort_file_name_value parameters.

Wraps the R `CohortGenerator::saveCohortDefinitionSet` function defined in `CohortGenerator/R/CohortDefinitionSet.R`.

Parameters

- **cohort_definition_set** (*RS4*) – A CohortDefinitionSet object
- **settings_file_name** (*str, optional*) – The name of the CSV file that will hold the cohort information including the cohortId and cohortName
- **json_folder** (*str, optional*) – The name of the folder that will hold the JSON representation of the cohort if it is available in the cohortDefinitionSet
- **sql_folder** (*str, optional*) – The name of the folder that will hold the SQL representation of the cohort
- **cohort_file_name_format** (*str, optional*) – Defines the format string for naming the cohort JSON and SQL files. The format string follows the standard defined in the base `sprintf` function.
- **cohort_file_name_value** (*list[str], optional*) – Defines the columns in the cohortDefinitionSet to use in conjunction with the cohortFileNameFormat parameter
- **subset_json_folder** (*str, optional*) – Defines the folder to store the subset JSON
- **verbose** (*bool, optional*) – When TRUE, logging messages are emitted to indicate export progress. By default False.

Return type
None

4.5 Database Connector

connect(*connection_details=None*, *dbms=None*, *user=None*, *password=None*, *server=None*, *port=None*, *extra_settings=None*, *oracle_driver='thin'*, *connection_string=None*, *path_to_driver=None*)

Connect to a OMOP CDM database

Connects to a database using the provided connection details, created by `create_connection_details` or from the arguments provided.

Wraps the R `DatabaseConnector::connect` function defined in `DatabaseConnector/R/Connect.R`.

Parameters

- **connection_details** (*ListVector*) – The connection details.
- **dbms** (*str | None, optional*) – The DBMS type.
- **user** (*str | None, optional*) – The database user name. Required if `connection_string` is not provided.
- **password** (*str | None, optional*) – The database password. Required if `connection_string` is not provided.
- **server** (*str | None, optional*) – The database server name. Required if `connection_string` is not provided.
- **port** (*int | None, optional*) – The database server port. Required if `connection_string` is not provided.
- **extra_settings** (*str | None, optional*) – Additional database connection settings.

- **oracle_driver** (*str, optional*) – The Oracle driver to use. Defaults to `thin`.
- **connection_string** (*str | None, optional*) – The database connection string. `user`, `password`, `server`, and `port` are ignored if this is provided.
- **path_to_driver** (*str | None, optional*) – The path to the database driver jar file.

Returns
The database connection.

Return type
RS4

Examples

```
>>> connect(connection_details)
```

create_connection_details(*dbms*, *user=None*, *password=None*, *server=None*, *port=None*, *extra_settings=None*, *oracle_driver='thin'*, *connection_string=None*, *path_to_driver=None*)

Create Connection Details.

Creates a list containing all details needed to connect to a database. There are three ways to call this function:

- `create_connection_details(dbms, user, password, server, port, extraSettings, oracleDriver, pathToDriver)`
- `create_connection_details(dbms, connectionString, pathToDriver)`
- `create_connection_details(dbms, connectionString, user, password, pathToDriver)`

Wraps the R `DatabaseConnector::createConnectionDetails` function defined in `DatabaseConnector/R/Connect.R`.

Parameters

- **dbms** (*str*) – The DBMS type.
- **user** (*str | None, optional*) – The database user name. Required if `connection_string` is not provided.
- **password** (*str | None, optional*) – The database password. Required if `connection_string` is not provided.
- **server** (*str | None, optional*) – The database server name. Required if `connection_string` is not provided.
- **port** (*int | None, optional*) – The database server port. Required if `connection_string` is not provided.
- **extra_settings** (*str | None, optional*) – Additional database connection settings.
- **oracle_driver** (*str, optional*) – The Oracle driver to use. Defaults to `thin`.
- **connection_string** (*str | None, optional*) – The database connection string. `user`, `password`, `server`, and `port` are ignored if this is provided.
- **path_to_driver** (*str | None, optional*) – The path to the database driver jar file.

Returns
The connection details.

Return type
ListVector

Examples

```
>>> create_connection_details(
...     dbms="postgresql", user="user", password="password",
...     server="localhost/postgres", port=5432
... )
```

disconnect(*connection*)

Disconnect from a database

Wraps the R `DatabaseConnector::disconnect` function defined in `DatabaseConnector/R/Connect.R`.

Parameters
connection (*RS4*) – The database connection.

Return type
None

Examples

```
>>> disconnect(connection)
```

execute_sql(*connection*, *sql*)

Execute a SQL statement

Wraps the R `DatabaseConnector::executeSql` function defined in `DatabaseConnector/R/Sql.R`.

Parameters

- **connection** (*RS4*) – The database connection.
- **sql** (*str*) – The SQL statement.

Return type
None

Examples

```
>>> execute_sql(connection, sql)
>>> execute_sql(connection, "DROP TABLE IF EXISTS person")
>>> execute_sql(
...     conn, "CREATE TABLE x (k INT); CREATE TABLE y (k INT);"
... )
```

query_sql(*connection*, *sql*)

Query a database

Wraps the R `DatabaseConnector::querySql` function defined in `DatabaseConnector/R/Sql.R`.

Parameters

- **connection** (*RS4*) – The database connection.

- **sql** (*str*) – The SQL query.

Returns
The query result.

Return type
RS4

Examples

```
>>> query_sql(connection, sql)
>>> query_sql(connection, "SELECT COUNT(*) FROM person")
```

4.6 Feature Extraction

aggregate_covariates(*covariate_data*)

Aggregate covariate data

Wraps the R `FeatureExtraction::aggregateCovariates` function defined in `FeatureExtraction/R/Aggregation.R`.

Parameters
covariate_data (*RS4*) – An object of type `covariateData` as generated using `getDbCovariateData`.

Returns
An object of class `covariateData`.

Return type
RS4

Examples

```
>>> covariate_data = create_empty_covariate_data(
...     cohort_id = 1,
...     aggregated = False,
...     temporal = False
... )
... aggregated_covariate_data = aggregate_covariates(covariate_data)
```

compute_standardized_difference(*covariate_data1*, *covariate_data2*, *cohort_id1=None*, *cohort_id2=None*)

Compute standardized difference of mean for all covariates.

Computes the standardized difference for all covariates between two cohorts. The standardized difference is defined as the difference between the mean divided by the overall standard deviation.

Wraps the R `FeatureExtraction::computeStandardizedDifference` function defined in `FeatureExtraction/R/CompareCohorts.R`.

Parameters

- **covariate_data1** (*RS4*) – The covariate data of the first cohort. Needs to be in aggregated format.

- **covariate_data2** (*RS4*) – The covariate data of the second cohort. Needs to be in aggregated format.
- **cohort_id1** (*int | None*) – If provided, `covariateData1` will be restricted to this cohort. If not provided, `covariateData1` is assumed to contain data on only 1 cohort.
- **cohort_id2** (*int | None*) – If provided, `covariateData2` will be restricted to this cohort. If not provided, `covariateData2` is assumed to contain data on only 1 cohort.

Returns
A data frame with means and standard deviations per cohort as well as the standardized difference of mean.

Return type
DataFrame

Examples

```
>>> cov_data_diff = compute_standardized_difference(
...     covariate_data1,
...     covariate_data2,
...     cohort_id1 = 1,
...     cohort_id2 = 2
... )
```

convert_prespec_settings_to_detailed_settings(*covariate_settings*)

Convert pre-specified covariate settings to detailed covariate settings

Wraps the R `FeatureExtraction::convertPrespecSettingsToDetailedSettings` function defined in `FeatureExtraction/R/DetailedCovariateSettings.R`.

Parameters
covariate_settings (*ListVector | ListVectorExtended*) – An object of type `covariateSettings`, to be used in other functions.

Returns
An object of type `covariateSettings`, to be used in other functions.

Return type
ListVector

Examples

```
>>> cov_settings = create_default_covariate_settings(
...     included_covariate_concept_ids = [1],
...     add_descendants_to_include = False,
...     excluded_covariate_concept_ids = [2],
...     add_descendants_to_exclude = False,
...     included_covariate_ids = [1]
... )
>>> cov_settings = convert_prespec_settings_to_detailed_settings(
...     cov_settings
... )
```

create_analysis_details(*analysis_id*, *sql_file_name*, *parameters*, *included_covariate_concept_ids=[]*, *add_descendants_to_include=False*, *excluded_covariate_concept_ids=[]*, *add_descendants_to_exclude=False*, *included_covariate_ids=[]*)

Create detailed covariate settings

Creates an object specifying in detail how covariates should be constructed from data in the CDM model. Warning: this function is for advanced users only.

Wraps the R `FeatureExtraction::createAnalysisDetails` function defined in `FeatureExtraction/R/DetailedCovariateSettings.R`.

Parameters

- **analysis_id** (*int*) – An integer between 0 and 999 that uniquely identifies this analysis.
- **sql_file_name** (*str*) – The name of the parameterized SQL file embedded in the `featureExtraction` package.
- **parameters** (*dict*) – The list of parameter values used to render the template SQL.
- **included_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should be used to construct covariates.
- **add_descendants_to_include** (*bool*) – Should descendant concept IDs be added to the list of concepts to include?
- **excluded_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should NOT be used to construct covariates.
- **add_descendants_to_exclude** (*bool*) – Should descendant concept IDs be added to the list of concepts to exclude?
- **included_covariate_ids** (*list[int]*) – A list of covariate IDs that should be restricted to.

Returns

An object of type `analysisDetails`, to be used in other functions.

Return type

ListVectorExtended

Examples

```
>>> analysis_details = create_analysis_details(
...     analysis_id = 1,
...     sql_file_name = "DemographicsGender.sql",
...     parameters = {
...         analysis_id: 1,
...         analysis_name: "Gender",
...         domain_id: "Demographics",
...     },
...     included_covariate_concept_ids = [],
...     add_descendants_to_include = False,
...     excluded_covariate_concept_ids = [],
...     add_descendants_to_exclude = False,
...     included_covariate_ids = []
... )
```

create_covariate_settings(*use_demographics_gender=False*, *use_demographics_age=False*,
use_demographics_age_group=False, *use_demographics_race=False*,
use_demographics_ethnicity=False, *use_demographics_index_year=False*,
use_demographics_index_month=False,
use_demographics_prior_observation_time=False,
use_demographics_post_observation_time=False,
use_demographics_time_in_cohort=False,
use_demographics_index_year_month=False, *use_care_site_id=False*,
use_condition_occurrence_any_time_prior=False,
use_condition_occurrence_long_term=False,
use_condition_occurrence_medium_term=False,
use_condition_occurrence_short_term=False,
use_condition_occurrence_primary_inpatient_any_time_prior=False,
use_condition_occurrence_primary_inpatient_long_term=False,
use_condition_occurrence_primary_inpatient_medium_term=False,
use_condition_occurrence_primary_inpatient_short_term=False,
use_condition_era_any_time_prior=False, *use_condition_era_long_term=False*,
use_condition_era_medium_term=False, *use_condition_era_short_term=False*,
use_condition_era_overlapping=False,
use_condition_era_start_long_term=False,
use_condition_era_start_medium_term=False,
use_condition_era_start_short_term=False,
use_condition_group_era_any_time_prior=False,
use_condition_group_era_long_term=False,
use_condition_group_era_medium_term=False,
use_condition_group_era_short_term=False,
use_condition_group_era_overlapping=False,
use_condition_group_era_start_long_term=False,
use_condition_group_era_start_medium_term=False,
use_condition_group_era_start_short_term=False,
use_drug_exposure_any_time_prior=False,
use_drug_exposure_long_term=False, *use_drug_exposure_medium_term=False*,
use_drug_exposure_short_term=False, *use_drug_era_any_time_prior=False*,
use_drug_era_long_term=False, *use_drug_era_medium_term=False*,
use_drug_era_short_term=False, *use_drug_era_overlapping=False*,
use_drug_era_start_long_term=False, *use_drug_era_start_medium_term=False*,
use_drug_era_start_short_term=False,
use_drug_group_era_any_time_prior=False,
use_drug_group_era_long_term=False,
use_drug_group_era_medium_term=False,
use_drug_group_era_short_term=False, *use_drug_group_era_overlapping=False*,
use_drug_group_era_start_long_term=False,
use_drug_group_era_start_medium_term=False,
use_drug_group_era_start_short_term=False,
use_procedure_occurrence_any_time_prior=False,
use_procedure_occurrence_long_term=False,
use_procedure_occurrence_medium_term=False,
use_procedure_occurrence_short_term=False,
use_device_exposure_any_time_prior=False,
use_device_exposure_long_term=False,
use_device_exposure_medium_term=False,
use_device_exposure_short_term=False,
use_measurement_any_time_prior=False, *use_measurement_long_term=False*,
use_measurement_medium_term=False, *use_measurement_short_term=False*,
use_measurement_value_any_time_prior=False,
use_measurement_value_long_term=False,
use_measurement_value_medium_term=False,
use_measurement_value_short_term=False,
use_measurement_range_group_any_time_prior=False,
use_measurement_range_group_long_term=False,

Create covariate settings

Creates an object specifying how covariates should be constructed from data in the CDM model.

Wraps the R `FeatureExtraction::createCovariateSettings` function defined in `FeatureExtraction/R/DefaultCovariateSettings.R`.

Parameters

- **use_demographics_gender** (*bool*) – Gender of the subject. (analysis ID 1)
- **use_demographics_age** (*bool*) – Age of the subject in years at the index date. (analysis ID 2)
- **use_demographics_age_group** (*bool*) – Age group of the subject at the index date. (analysis ID 3)
- **use_demographics_race** (bool) – Race of the subject. (analysis ID 4)
- **use_demographics_ethnicity** (bool) – Ethnicity of the subject. (analysis ID 5)
- **use_demographics_index_year** (bool) – Year of the index date. (analysis ID 6)
- **use_demographics_index_month** (bool) – Month of the index date. (analysis ID 7)
- **use_demographics_prior_observation_time** (bool) – Number of continuous days of observation time preceding the index date. (analysis ID 8)
- **use_demographics_post_observation_time** (bool) – Number of continuous days of observation time following the index date. (analysis ID 9)
- **use_demographics_time_in_cohort** (bool) – Number of days of observation time during cohort period. (analysis ID 10)
- **use_demographics_index_year_month** (bool) – Both calendar year and month of the index date in a single variable. (analysis ID 11)
- **use_care_site_id** (bool) – Care site associated with the cohort start, pulled from the visit_detail, visit_occurrence, or person table, in that order. (analysis ID 12)
- **use_condition_occurrence_any_time_prior** (bool) – One covariate per condition in the condition_occurrence table starting any time prior to index. (analysis ID 101)
- **use_condition_occurrence_long_term** (bool) – One covariate per condition in the condition_occurrence table starting in the long term window. (analysis ID 102)
- **use_condition_occurrence_medium_term** (bool) – One covariate per condition in the condition_occurrence table starting in the medium term window. (analysis ID 103)
- **use_condition_occurrence_short_term** (bool) – One covariate per condition in the condition_occurrence table starting in the short term window. (analysis ID 104)
- **use_condition_occurrence_primary_inpatient_any_time_prior** (bool) – One covariate per condition observed as a primary diagnosis in an inpatient setting in the condition_occurrence table starting any time prior to index. (analysis ID 105)
- **use_condition_occurrence_primary_inpatient_long_term** (bool) – One covariate per condition observed as a primary diagnosis in an inpatient setting in the condition_occurrence table starting in the long term window. (analysis ID 106)
- **use_condition_occurrence_primary_inpatient_medium_term** (bool) – One covariate per condition observed as a primary diagnosis in an inpatient setting in the condition_occurrence table starting in the medium term window. (analysis ID 107)

- **use_condition_occurrence_primary_inpatient_short_term** (bool) – One covariate per condition observed as a primary diagnosis in an inpatient setting in the condition_occurrence table starting in the short term window. (analysis ID 108)
- **use_condition_era_any_time_prior** (bool) – One covariate per condition in the condition_era table overlapping with any time prior to index. (analysis ID 201)
- **use_condition_era_long_term** (bool) – One covariate per condition in the condition_era table overlapping with any part of the long term window. (analysis ID 202)
- **use_condition_era_medium_term** (bool) – One covariate per condition in the condition_era table overlapping with any part of the medium term window. (analysis ID 203)
- **use_condition_era_short_term** (bool) – One covariate per condition in the condition_era table overlapping with any part of the short term window. (analysis ID 204)
- **use_condition_era_overlapping** (bool) – One covariate per condition in the condition_era table overlapping with the end of the risk window. (analysis ID 205)
- **use_condition_era_start_long_term** (bool) – One covariate per condition in the condition_era table starting in the long term window. (analysis ID 206)
- **use_condition_era_start_medium_term** (bool) – One covariate per condition in the condition_era table starting in the medium term window. (analysis ID 207)
- **use_condition_era_start_short_term** (bool) – One covariate per condition in the condition_era table starting in the short term window. (analysis ID 208)
- **use_condition_group_era_any_time_prior** (bool) – One covariate per condition era rolled up to groups in the condition_era table overlapping with any time prior to index. (analysis ID 209)
- **use_condition_group_era_long_term** (bool) – One covariate per condition era rolled up to groups in the condition_era table overlapping with any part of the long term window. (analysis ID 210)
- **use_condition_group_era_medium_term** (bool) – One covariate per condition era rolled up to groups in the condition_era table overlapping with any part of the medium term window. (analysis ID 211)
- **use_condition_group_era_short_term** (bool) – One covariate per condition era rolled up to groups in the condition_era table overlapping with any part of the short term window. (analysis ID 212)
- **use_condition_group_era_overlapping** (bool) – One covariate per condition era rolled up to groups in the condition_era table overlapping with the end of the risk window. (analysis ID 213)
- **use_condition_group_era_start_long_term** (bool) – One covariate per condition era rolled up to groups in the condition_era table starting in the long term window. (analysis ID 214)
- **use_condition_group_era_start_medium_term** (bool) – One covariate per condition era rolled up to groups in the condition_era table starting in the medium term window. (analysis ID 215)
- **use_condition_group_era_start_short_term** (bool) – One covariate per condition era rolled up to groups in the condition_era table starting in the short term window. (analysis ID 216)
- **use_drug_exposure_any_time_prior** (bool) – One covariate per drug in the drug_exposure table starting any time prior to index. (analysis ID 301)

- **use_drug_exposure_long_term** (bool) – One covariate per drug in the drug_exposure table starting in the long term window. (analysis ID 302)
- **use_drug_exposure_medium_term** (bool) – One covariate per drug in the drug_exposure table starting in the medium term window. (analysis ID 303)
- **use_drug_exposure_short_term** (bool) – One covariate per drug in the drug_exposure table starting in the short term window. (analysis ID 304)
- **use_drug_era_any_time_prior** (bool) – One covariate per drug in the drug_era table overlapping with any time prior to index. (analysis ID 401)
- **use_drug_era_long_term** (bool) – One covariate per drug in the drug_era table overlapping with any part of the long term window. (analysis ID 402)
- **use_drug_era_medium_term** (bool) – One covariate per drug in the drug_era table overlapping with any part of the medium term window. (analysis ID 403)
- **use_drug_era_short_term** (bool) – One covariate per drug in the drug_era table overlapping with any part of the short window. (analysis ID 404)
- **use_drug_era_overlapping** (bool) – One covariate per drug in the drug_era table overlapping with the end of the risk window. (analysis ID 405)
- **use_drug_era_start_long_term** (bool) – One covariate per drug in the drug_era table starting in the long term window. (analysis ID 406)
- **use_drug_era_start_medium_term** (bool) – One covariate per drug in the drug_era table starting in the medium term window. (analysis ID 407)
- **use_drug_era_start_short_term** (bool) – One covariate per drug in the drug_era table starting in the long short window. (analysis ID 408)
- **use_drug_group_era_any_time_prior** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table overlapping with any time prior to index. (analysis ID 409)
- **use_drug_group_era_long_term** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table overlapping with any part of the long term window. (analysis ID 410)
- **use_drug_group_era_medium_term** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table overlapping with any part of the medium term window. (analysis ID 411)
- **use_drug_group_era_short_term** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table overlapping with any part of the short term window. (analysis ID 412)
- **use_drug_group_era_overlapping** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table overlapping with the end of the risk window. (analysis ID 413)
- **use_drug_group_era_start_long_term** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table starting in the long term window. (analysis ID 414)
- **use_drug_group_era_start_medium_term** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table starting in the medium term window. (analysis ID 415)
- **use_drug_group_era_start_short_term** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table starting in the short term window. (analysis ID 416)
- **use_procedure_occurrence_any_time_prior** (bool) – One covariate per procedure in the procedure_occurrence table any time prior to index. (analysis ID 501)

- **use_procedure_occurrence_long_term** (bool) – One covariate per procedure in the procedure_occurrence table in the long term window. (analysis ID 502)
- **use_procedure_occurrence_medium_term** (bool) – One covariate per procedure in the procedure_occurrence table in the medium term window. (analysis ID 503)
- **use_procedure_occurrence_short_term** (bool) – One covariate per procedure in the procedure_occurrence table in the short term window. (analysis ID 504)
- **use_device_exposure_any_time_prior** (bool) – One covariate per device in the device exposure table starting any time prior to index. (analysis ID 601)
- **use_device_exposure_long_term** (bool) – One covariate per device in the device exposure table starting in the long term window. (analysis ID 602)
- **use_device_exposure_medium_term** (bool) – One covariate per device in the device exposure table starting in the medium term window. (analysis ID 603)
- **use_device_exposure_short_term** (bool) – One covariate per device in the device exposure table starting in the short term window. (analysis ID 604)
- **use_measurement_any_time_prior** (bool) – One covariate per measurement in the measurement table any time prior to index. (analysis ID 701)
- **use_measurement_long_term** (bool) – One covariate per measurement in the measurement table in the long term window. (analysis ID 702)
- **use_measurement_medium_term** (bool) – One covariate per measurement in the measurement table in the medium term window. (analysis ID 703)
- **use_measurement_short_term** (bool) – One covariate per measurement in the measurement table in the short term window. (analysis ID 704)
- **use_measurement_value_any_time_prior** (bool) – One covariate containing the value per measurement-unit combination any time prior to index. (analysis ID 705)
- **use_measurement_value_long_term** (bool) – One covariate containing the value per measurement-unit combination in the long term window. (analysis ID 706)
- **use_measurement_value_medium_term** (bool) – One covariate containing the value per measurement-unit combination in the medium term window. (analysis ID 707)
- **use_measurement_value_short_term** (bool) – One covariate containing the value per measurement-unit combination in the short term window. (analysis ID 708)
- **use_measurement_range_group_any_time_prior** (bool) – Covariates indicating whether measurements are below, within, or above normal range any time prior to index. (analysis ID 709)
- **use_measurement_range_group_long_term** (bool) – Covariates indicating whether measurements are below, within, or above normal range in the long term window. (analysis ID 710)
- **use_measurement_range_group_medium_term** (bool) – Covariates indicating whether measurements are below, within, or above normal range in the medium term window. (analysis ID 711)
- **use_measurement_range_group_short_term** (bool) – Covariates indicating whether measurements are below, within, or above normal range in the short term window. (analysis ID 712)
- **use_observation_any_time_prior** (bool) – One covariate per observation in the observation table any time prior to index. (analysis ID 801)

- **use_observation_long_term** (bool) – One covariate per observation in the observation table in the long term window. (analysis ID 802)
- **use_observation_medium_term** (bool) – One covariate per observation in the observation table in the medium term window. (analysis ID 803)
- **use_observation_short_term** (bool) – One covariate per observation in the observation table in the short term window. (analysis ID 804)
- **use_charlson_index** (bool) – The Charlson comorbidity index (Romano adaptation) using all conditions prior to the window end. (analysis ID 901)
- **use_dcsi** (bool) – The Diabetes Comorbidity Severity Index (DCSI) using all conditions prior to the window end. (analysis ID 902)
- **use_chads2** (bool) – The CHADS2 score using all conditions prior to the window end. (analysis ID 903)
- **use_chads2_vasc** (bool) – The CHADS2VASc score using all conditions prior to the window end. (analysis ID 904)
- **use_hfrs** (bool) – The Hospital Frailty Risk Score score using all conditions prior to the window end. (analysis ID 926)
- **use_distinct_condition_count_long_term** (bool) – The number of distinct condition concepts observed in the long term window. (analysis ID 905)
- **use_distinct_condition_count_medium_term** (bool) – The number of distinct condition concepts observed in the medium term window. (analysis ID 906)
- **use_distinct_condition_count_short_term** (bool) – The number of distinct condition concepts observed in the short term window. (analysis ID 907)
- **use_distinct_ingredient_count_long_term** (bool) – The number of distinct ingredients observed in the long term window. (analysis ID 908)
- **use_distinct_ingredient_count_medium_term** (bool) – The number of distinct ingredients observed in the medium term window. (analysis ID 909)
- **use_distinct_ingredient_count_short_term** (bool) – The number of distinct ingredients observed in the short term window. (analysis ID 910)
- **use_distinct_procedure_count_long_term** (bool) – The number of distinct procedures observed in the long term window. (analysis ID 911)
- **use_distinct_procedure_count_medium_term** (bool) – The number of distinct procedures observed in the medium term window. (analysis ID 912)
- **use_distinct_procedure_count_short_term** (bool) – The number of distinct procedures observed in the short term window. (analysis ID 913)
- **use_distinct_measurement_count_long_term** (bool) – The number of distinct measurements observed in the long term window. (analysis ID 914)
- **use_distinct_measurement_count_medium_term** (bool) – The number of distinct measurements observed in the medium term window. (analysis ID 915)
- **use_distinct_measurement_count_short_term** (bool) – The number of distinct measurements observed in the short term window. (analysis ID 916)
- **use_distinct_observation_count_long_term** (bool) – The number of distinct observations observed in the long term window. (analysis ID 917)

- **use_distinct_observation_count_medium_term** (bool) – The number of distinct observations observed in the medium term window. (analysis ID 918)
- **use_distinct_observation_count_short_term** (bool) – The number of distinct observations observed in the short term window. (analysis ID 919)
- **use_visit_count_long_term** (bool) – The number of visits observed in the long term window. (analysis ID 920)
- **use_visit_count_medium_term** (bool) – The number of visits observed in the medium term window. (analysis ID 921)
- **use_visit_count_short_term** (bool) – The number of visits observed in the short term window. (analysis ID 922)
- **use_visit_concept_count_long_term** (bool) – The number of visits observed in the long term window, stratified by visit concept ID. (analysis ID 923)
- **use_visit_concept_count_medium_term** (bool) – The number of visits observed in the medium term window, stratified by visit concept ID. (analysis ID 924)
- **use_visit_concept_count_short_term** (bool) – The number of visits observed in the short term window, stratified by visit concept ID. (analysis ID 925)
- **long_term_start_days** (int) – What is the start day (relative to the index date) of the long-term window?
- **medium_term_start_days** (int) – What is the start day (relative to the index date) of the medium-term window?
- **short_term_start_days** (int) – What is the start day (relative to the index date) of the short-term window?
- **end_days** (int) – What is the end day (relative to the index date) of the window?
- **included_covariate_concept_ids** (list) – A list of concept IDs that should be used to construct covariates.
- **add_descendants_to_include** (bool) – Should descendant concept IDs be added to the list of concepts to include?
- **excluded_covariate_concept_ids** (list) – A list of concept IDs that should NOT be used to construct covariates.
- **add_descendants_to_exclude** (bool) – Should descendant concept IDs be added to the list of concepts to exclude?
- **included_covariate_ids** (list) – A list of covariate IDs that should be restricted to.

Returns
An object of type covariateSettings, to be used in other functions.

Return type
ListVectorExtended

Examples

```
>>> settings = create_covariate_settings(
...     use_demographics_gender=True,
...     use_demographics_age_group=True,
...     use_condition_occurrence_any_time_prior=True
... )
```

create_default_covariate_settings(*included_covariate_concept_ids=[]*, *add_descendants_to_include=False*, *excluded_covariate_concept_ids=[]*, *add_descendants_to_exclude=False*, *included_covariate_ids=[]*)

Create default covariate settings

Wraps the R `FeatureExtraction::createDefaultCovariateSettings` function defined in `FeatureExtraction/R/DetailedCovariateSettings.R`.

Parameters

- **included_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should be used to construct covariates.
- **add_descendants_to_include** (*bool*) – Should descendant concept IDs be added to the list of concepts to include?
- **excluded_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should NOT be used to construct covariates.
- **add_descendants_to_exclude** (*bool*) – Should descendant concept IDs be added to the list of concepts to exclude?
- **included_covariate_ids** (*list[int]*) – A list of covariate IDs that should be restricted to.

Returns

An object of type `covariateSettings`, to be used in other functions.

Return type

ListVectorExtended

Examples

```
>>> cov_settings = create_default_covariate_settings(
...     included_covariate_concept_ids = [1],
...     add_descendants_to_include = False,
...     excluded_covariate_concept_ids = [2],
...     add_descendants_to_exclude = False,
...     included_covariate_ids = [1]
... )
```

create_default_temporal_covariate_settings(*included_covariate_concept_ids=[]*, *add_descendants_to_include=False*, *excluded_covariate_concept_ids=[]*, *add_descendants_to_exclude=False*, *included_covariate_ids=[]*)

Create default temporal covariate settings

Creates an object specifying in detail how covariates should be constructed from data in the CDM model. Warning: this function is for advanced users only.

Wraps the R `FeatureExtraction::createDefaultTemporalCovariateSettings` function defined in `FeatureExtraction/R/DetailedCovariateSettings.R`.

Parameters

- **included_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should be used to construct covariates.
- **add_descendants_to_include** (*bool*) – Should descendant concept IDs be added to the list of concepts to include?
- **excluded_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should NOT be used to construct covariates.
- **add_descendants_to_exclude** (*bool*) – Should descendant concept IDs be added to the list of concepts to exclude?
- **included_covariate_ids** (*list[int]*) – A list of covariate IDs that should be restricted to.

Returns

An object of type `covariateSettings`, to be used in other functions.

Return type

ListVectorExtended

Examples

```
>>> cov_settings = create_default_temporal_covariate_settings(
...     included_covariate_concept_ids = [1],
...     add_descendants_to_include = False,
...     excluded_covariate_concept_ids = [2],
...     add_descendants_to_exclude = False,
...     included_covariate_ids = [1]
... )
```

create_detailed_covariate_settings(*analyses=[]*)

Create detailed covariate settings

Creates an object specifying in detail how covariates should be constructed from data in the CDM model. Warning: this function is for advanced users only.

Wraps the R `FeatureExtraction::createDetailedCovariateSettings` function defined in `FeatureExtraction/R/DetailedCovariateSettings.R`.

Parameters

analyses (*list*) – A list of analysis detail objects as created using `createAnalysisDetails`.

Returns

An object of type `covariateSettings`, to be used in other functions.

Return type

ListVectorExtended

Examples

```
>>> analysis_details = create_analysis_details(
...     analysis_id = 1,
...     sql_file_name = "DemographicsGender.sql",
...     parameters = {
...         analysis_id: 1,
...         analysis_name: "Gender",
...         domain_id: "Demographics",
...     },
...     included_covariate_concept_ids = [],
...     add_descendants_to_include = False,
...     excluded_covariate_concept_ids = [],
...     add_descendants_to_exclude = False,
...     included_covariate_ids = []
... )
>>> cov_settings = create_detailed_covariate_settings(analysis_details)
```

create_detailed_temporal_covariate_settings(*analyses=[]*, *temporal_start_days=[-365, -364, -363, -362, -361, -360, -359, -358, -357, -356, -355, -354, -353, -352, -351, -350, -349, -348, -347, -346, -345, -344, -343, -342, -341, -340, -339, -338, -337, -336, -335, -334, -333, -332, -331, -330, -329, -328, -327, -326, -325, -324, -323, -322, -321, -320, -319, -318, -317, -316, -315, -314, -313, -312, -311, -310, -309, -308, -307, -306, -305, -304, -303, -302, -301, -300, -299, -298, -297, -296, -295, -294, -293, -292, -291, -290, -289, -288, -287, -286, -285, -284, -283, -282, -281, -280, -279, -278, -277, -276, -275, -274, -273, -272, -271, -270, -269, -268, -267, -266, -265, -264, -263, -262, -261, -260, -259, -258, -257, -256, -255, -254, -253, -252, -251, -250, -249, -248, -247, -246, -245, -244, -243, -242, -241, -240, -239, -238, -237, -236, -235, -234, -233, -232, -231, -230, -229, -228, -227, -226, -225, -224, -223, -222, -221, -220, -219, -218, -217, -216, -215, -214, -213, -212, -211, -210, -209, -208, -207, -206, -205, -204, -203, -202, -201, -200, -199, -198, -197, -196, -195, -194, -193, -192, -191, -190, -189, -188, -187, -186, -185, -184, -183, -182, -181, -180, -179, -178, -177, -176, -175, -174, -173, -172, -171, -170, -169, -168, -167, -166, -165, -164, -163, -162, -161, -160, -159, -158, -157, -156, -155, -154, -153, -152, -151, -150, -149, -148, -147, -146, -145, -144, -143, -142, -141, -140, -139, -138, -137, -136, -135, -134, -133, -132, -131, -130, -129, -128, -127, -126, -125, -124, -123, -122, -121, -120, -119, -118, -117, -116, -115, -114, -113, -112, -111, -110, -109, -108, -107, -106, -105, -104, -103, -102, -101, -100, -99, -98, -97, -96, -95, -94, -93, -92, -91, -90, -89, -88, -87, -86, -85, -84, -83, -82, -81, -80, -79, -78, -77, -76, -75, -74, -73, -72, -71, -70, -69, -68, -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54, -53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38, -37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1]*, *temporal_end_days=[-365, -364, -363, -362, -361, -360, -359, -358, -357, -356, -355, -354, -353, -352, -351, -350, -349, -348, -347, -346, -345, -344, -343, -342, -341, -340, -339, -338, -337, -336, -335, -334, -333, -332, -331, -330, -329, -328, -327, -326, -325, -324, -323, -322, -321, -320, -319, -318, -317, -316, -315, -314, -313, -312, -311, -310, -309, -308, -307, -306, -305, -304, -303, -302, -301, -300, -299, -298, -297, -296, -295, -294, -293, -292, -291, -290, -289, -288, -287, -286, -285, -284, -283, -282, -281, -280, -279, -278, -277, -276, -275, -274, -273, -272, -271, -270, -269, -268, -267, -266, -265, -264, -263, -262, -261, -260, -259, -258, -257, -256, -255, -254, -253, -252, -251, -250, -249, -248, -247, -246, -245, -244, -243, -242, -241, -240, -239, -238, -237, -236, -235, -234, -233, -232, -231, -230, -229, -228, -227, -226, -225, -224, -223, -222, -221, -220, -219, -218, -217, -216, -215, -214, -213, -212, -211, -210, -209, -208, -207, -206, -205, -204, -203, -202, -201, -200, -199, -198, -197, -196, -195, -194, -193, -192, -191, -190, -189, -188, -187, -186, -185, -184, -183, -182, -181, -180, -179, -178, -177, -176, -175, -174, -173, -172, -171, -170, -169, -168, -167, -166, -165, -164, -163, -162, -161, -160, -159, -158, -157, -156, -155, -154, -153, -152, -151, -150, -149, -148, -147, -146, -145, -144, -143, -142, -141, -140, -139, -138, -137, -136, -135, -134, -133, -132, -131, -130, -129, -128, -127, -126, -125, -124, -123, -122, -121, -120,

Create detailed temporal covariate settings

Creates an object specifying in detail how temporal covariates should be constructed from data in the CDM model. Warning: this function is for advanced users only.

Wraps the R `FeatureExtraction::createDetailedTemporalCovariateSettings` function defined in `FeatureExtraction/R/DetailedCovariateSettings.R`.

Parameters

- **analyses** (*list, optional*) – A list of analysis detail objects as created using `createAnalysisDetails`, by default []
- **temporal_start_days** (*list[int], optional*) – A list of integers representing the start of a time period, relative to the index date. 0 indicates the index date, -1 indicates the day before the index date, etc. The start day is included in the time period., by default range(-365,-1, 1)
- **temporal_end_days** (*list[int], optional*) – A list of integers representing the end of a time period, relative to the index date. 0 indicates the index date, -1 indicates the day before the index date, etc. The end day is included in the time period., by default range(-365, -1, 1)

Returns
An object of type `covariateSettings`, to be used in other functions.

Return type
ListVector

Examples

```
>>> cov_settings = create_detailed_temporal_covariate_settings(
...     analyses = analysis_details,
...     temporal_start_days = range(-365, 0, 1),
...     temporal_end_days = range(-365, 0, 1)
... )
```

create_empty_covariate_data(*cohort_id=1*, *aggregated=False*, *temporal=False*)

Creates an empty covariate data object

Wraps the R `FeatureExtraction:::createEmptyCovariateData` function defined in `FeatureExtraction/R/CovariateData.R`.

Parameters

- **cohort_id** (`int`) – cohort number
- **aggregated** (`bool`) – if the data should be aggregated
- **temporal** (`bool`) – if the data is temporal

Returns
An object of type `CovariateData`.

Return type
CovariateData

Examples

```
>>> covariate_data = create_empty_covariate_data(
...     cohort_id = 1,
...     aggregated = False,
...     temporal = False
... )
```

create_table1(*covariate_data1*, *covariate_data2=None*, *cohort_id1=None*, *cohort_id2=None*, *specifications=None*, *output='two columns'*, *show_counts=False*, *show_percent=True*, *percent_digits=1*, *value_digits=1*, *std_diff_digits=2*)

Create a table 1

Creates a formatted table of cohort characteristics, to be included in publications or reports. Allows for creating a table describing a single cohort, or a table comparing two cohorts.

Wraps the R `FeatureExtraction::createTable1` function defined in `FeatureExtraction/R/Table1.R`.

Parameters

- **covariate_data1** (*RS4*) – The covariate data of the cohort to be included in the table.
- **covariate_data2** (*RS4*) – The covariate data of the cohort to also be included, when comparing two cohorts.
- **cohort_id1** (*int*) – If provided, `covariateData1` will be restricted to this cohort. If not provided, `covariateData1` is assumed to contain data on only 1 cohort.
- **cohort_id2** (*int*) – If provided, `covariateData2` will be restricted to this cohort. If not provided, `covariateData2` is assumed to contain data on only 1 cohort.
- **specifications** (*DataFrame*) – Specifications of which covariates to display, and how.
- **output** (*str*) – The output format for the table. Options are: `output = "two columns"`, `output = "one column"`, or `output = "list"`
- **sho_counts** (*bool*) – Show the number of cohort entries having the binary covariate?
- **show_percent** (*bool*) – Show the percentage of cohort entries having the binary covariate?
- **percent_digits** (*int*) – Number of digits to be used for percentages.
- **std_diff_digits** (*int*) – Number of digits to be used for the standardized differences.
- **value_digits** (*int*) – Number of digits to be used for the values of continuous variables.

Returns

A data frame, or, when `output = "list"` a list of two data frames.

Return type

DataFrame

Examples

```
>>> cov_data1 = get_db_covariate_data(
...     connection_details = connection_details,
...     cdm_database_schema = "main",
...     cohort_table = "cohorts_of_interest",
...     cohort_database_schema = "results",
...     cohort_id = 1,
...     covariate_settings = covariate_settings,
...     aggregated = True
... )
```

```
>>> cov_data2 = get_db_covariate_data(
...     connection_details = connection_details,
...     cdm_database_schema = "main",
...     cohort_table = "cohorts_of_interest",
...     cohort_database_schema = "results",
...     cohort_id = 2,
...     covariate_settings = covariate_settings,
...     aggregated = True
... )
```

```
>>> table1 = create_table1(
...     covariate_data1 = cov_data1,
...     covariate_data2 = cov_data2,
...     cohort_id1 = 1,
...     cohort_id2 = 2,
...     specifications = Table1.get_default_table1_specifications(),
...     output = "one column",
...     show_counts = False,
...     show_percent = TRUE,
...     percent_digits = 1,
...     value_digits = 1,
...     std_diff_digits = 2
... )
```

create_table1_covariate_settings(*specifications=None*, *covariate_settings=None*, *included_covariate_concept_ids=[]*, *add_descendants_to_include=False*, *excluded_covariate_concept_ids=[]*, *add_descendants_to_exclude=False*, *included_covariate_ids=[]*)

Create covariate settings for a table 1

Creates a covariate settings object for generating only those covariates that will be included in a table 1. This function works by filtering the `covariateSettings` object for the covariates in `specifications` object.

Wraps the R `FeatureExtraction::createTable1CovariateSettings` function defined in `FeatureExtraction/R/Table1.R`.

Parameters

- **specifications** (*DataFrame*) – A specifications object for generating a table using the `createTable1` function.
- **covariate_settings** (*ListVector*) – The covariate settings object to use as the basis for the filtered covariate settings.

- **included_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should be used to construct covariates.
- **add_descendants_to_include** (*bool*) – Should descendant concept IDs be added to the list of concepts to include?
- **excluded_covariate_concept_ids** (*list[int]*) – A list of concept IDs that should NOT be used to construct covariates.
- **add_descendants_to_exclude** (*bool*) – Should descendant concept IDs be added to the list of concepts to exclude?
- **included_covariate_ids** (*list[int]*) – A list of covariate IDs that should be restricted to.

Returns

A covariate settings object, for example to be used when calling the `getDbCovariateData` function.

Return type

ListVector

Examples

```
>>> table1_cov_settings = Table1.create_table1_covariate_settings(
...     specifications = Table1.get_default_table1_specifications(),
...     included_covariate_concept_ids = [],
...     add_descendants_to_include = False,
...     excluded_covariate_concept_ids = [],
...     add_descendants_to_exclude = False,
...     included_covariate_ids = []
... )
```

create_temporal_covariate_settings(*use_demographics_gender=False*, *use_demographics_age=False*, *use_demographics_age_group=False*, *use_demographics_race=False*, *use_demographics_ethnicity=False*, *use_demographics_index_year=False*, *use_demographics_index_month=False*, *use_demographics_prior_observation_time=False*, *use_demographics_post_observation_time=False*, *use_demographics_time_in_cohort=False*, *use_demographics_index_year_month=False*, *use_care_site_id=False*, *use_condition_occurrence=False*, *use_condition_occurrence_primary_inpatient=False*, *use_condition_era_start=False*, *use_condition_era_overlap=False*, *use_condition_era_group_start=False*, *use_condition_era_group_overlap=False*, *use_drug_exposure=False*, *use_drug_era_start=False*, *use_drug_era_overlap=False*, *use_drug_era_group_start=False*, *use_drug_era_group_overlap=False*, *use_procedure_occurrence=False*, *use_device_exposure=False*, *use_measurement=False*, *use_measurement_value=False*, *use_measurement_range_group=False*, *use_observation=False*, *use_charlson_index=False*, *use_dcsi=False*, *use_chads2=False*, *use_chads2_vasc=False*, *use_hfrs=False*, *use_distinct_condition_count=False*, *use_distinct_ingredient_count=False*, *use_distinct_procedure_count=False*, *use_distinct_measurement_count=False*, *use_distinct_observation_count=False*, *use_visit_count=False*, *use_visit_concept_count=False*, *temporal_start_days=[-365, -364, -363, -362, -361, -360, -359, -358, -357, -356, -355, -354, -353, -352, -351, -350, -349, -348, -347, -346, -345, -344, -343, -342, -341, -340, -339, -338, -337, -336, -335, -334, -333, -332, -331, -330, -329, -328, -327, -326, -325, -324, -323, -322, -321, -320, -319, -318, -317, -316, -315, -314, -313, -312, -311, -310, -309, -308, -307, -306, -305, -304, -303, -302, -301, -300, -299, -298, -297, -296, -295, -294, -293, -292, -291, -290, -289, -288, -287, -286, -285, -284, -283, -282, -281, -280, -279, -278, -277, -276, -275, -274, -273, -272, -271, -270, -269, -268, -267, -266, -265, -264, -263, -262, -261, -260, -259, -258, -257, -256, -255, -254, -253, -252, -251, -250, -249, -248, -247, -246, -245, -244, -243, -242, -241, -240, -239, -238, -237, -236, -235, -234, -233, -232, -231, -230, -229, -228, -227, -226, -225, -224, -223, -222, -221, -220, -219, -218, -217, -216, -215, -214, -213, -212, -211, -210, -209, -208, -207, -206, -205, -204, -203, -202, -201, -200, -199, -198, -197, -196, -195, -194, -193, -192, -191, -190, -189, -188, -187, -186, -185, -184, -183, -182, -181, -180, -179, -178, -177, -176, -175, -174, -173, -172, -171, -170, -169, -168, -167, -166, -165, -164, -163, -162, -161, -160, -159, -158, -157, -156, -155, -154, -153, -152, -151, -150, -149, -148, -147, -146, -145, -144, -143, -142, -141, -140, -139, -138, -137, -136, -135, -134, -133, -132, -131, -130, -129, -128, -127, -126, -125, -124, -123, -122, -121, -120, -119, -118, -117, -116, -115, -114, -113, -112, -111, -110, -109, -108, -107, -106, -105, -104, -103, -102, -101, -100, -99, -98, -97, -96, -95, -94, -93, -92, -91, -90, -89, -88, -87, -86, -85, -84, -83, -82, -81, -80, -79, -78, -77, -76, -75, -74, -73, -72, -71, -70, -69, -68, -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54, -53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38, -37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1]*, *temporal_end_days=[-365, -364, -363, -362, -361, -360, -359, -358, -357, -356, -355, -354, -353, -352, -351, -350, -349, -348, -347, -346, -345, -344, -343, -342, -341, -340, -339, -338, -337, -336, -335, -334, -333, -332, -331, -330, -329, -328, -327,*

Create covariate settings

Creates an object specifying how covariates should be constructed from data in the CDM model.

Wraps the R``FeatureExtraction::createTemporalCovariateSettings`` function defined in `FeatureExtraction/R/DefaultTemporalCovariateSettings.R`

Parameters

- **use_demographics_gender** (bool) – Gender of the subject. (analysis ID 1)
- **use_demographics_age** (bool) – Age of the subject on the index date (in years). (analysis ID 2)
- **use_demographics_age_group** (bool) – Age of the subject on the index date (in 5 year age groups) (analysis ID 3)
- **use_demographics_race** (bool) – Race of the subject. (analysis ID 4)
- **use_demographics_ethnicity** (bool) – Ethnicity of the subject. (analysis ID 5)
- **use_demographics_index_year** (bool) – Year of the index date. (analysis ID 6)
- **use_demographics_index_month** (bool) – Month of the index date. (analysis ID 7)
- **use_demographics_prior_observation_time** (bool) – Number of days of observation time preceding the index date. (analysis ID 8)
- **use_demographics_post_observation_time** (bool) – Number of days of observation time preceding the index date. (analysis ID 9)
- **use_demographics_time_in_cohort** (bool) – Number of days of observation time preceding the index date. (analysis ID 10)
- **use_demographics_index_year_month** (bool) – Calendar month of the index date. (analysis ID 11)
- **use_care_site_id** (bool) – Care site associated with the cohort start, pulled from the visit_detail, visit_occurrence, or person table, in that order. (analysis ID 12)
- **use_condition_occurrence** (bool) – One covariate per condition in the condition_occurrence table starting in the time window. (analysis ID 101)
- **use_condition_occurrence_primary_inpatient** (bool) – One covariate per condition observed as a primary diagnosis in an inpatient setting in the condition_occurrence table starting in the time window. (analysis ID 102)
- **use_condition_era_start** (bool) – One covariate per condition in the condition_era table starting in the time window. (analysis ID 201)
- **use_condition_era_overlap** (bool) – One covariate per condition in the condition_era table overlapping with any part of the time window. (analysis ID 202)
- **use_condition_era_group_start** (bool) – One covariate per condition era rolled up to SNOMED groups in the condition_era table starting in the time window. (analysis ID 203)
- **use_condition_era_group_overlap** (bool) – One covariate per condition era rolled up to SNOMED groups in the condition_era table overlapping with any part of the time window. (analysis ID 204)
- **use_drug_exposure** (bool) – One covariate per drug in the drug_exposure table starting in the time window. (analysis ID 301)

- **use_drug_era_start** (bool) – One covariate per drug in the drug_era table starting in the time window. (analysis ID 401)
- **use_drug_era_overlap** (bool) – One covariate per drug in the drug_era table overlapping with any part of the time window. (analysis ID 402)
- **use_drug_era_group_start** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table starting in the time window. (analysis ID 403)
- **use_drug_era_group_overlap** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table overlapping with any part of the time window. (analysis ID 404)
- **use_procedure_occurrence** (bool) – One covariate per procedure in the procedure_occurrence table in the time window. (analysis ID 501)
- **use_device_exposure** (bool) – One covariate per device in the device exposure table starting in the timewindow. (analysis ID 601)
- **use_measurement** (bool) – One covariate per measurement in the measurement table in the time window. (analysis ID 701)
- **use_measurement_value** (bool) – One covariate containing the value per measurement-unit combination in the time window. If multiple values are found, the last is taken. (analysis ID 702)
- **use_measurement_range_group** (bool) – Covariates indicating whether measurements are below, within, or above normal range within the time period. (analysis ID 703)
- **use_observation** (bool) – One covariate per observation in the observation table in the time window. (analysis ID 801)
- **use_charlson_index** (bool) – The Charlson comorbidity index (Romano adaptation) using all conditions prior to the window end. (analysis ID 901)
- **use_dcsi** (bool) – The Diabetes Comorbidity Severity Index (DCSI) using all conditions prior to the window end. (analysis ID 902)
- **use_chads2** (bool) – The CHADS2 score using all conditions prior to the window end. (analysis ID 903)
- **use_chads2_vasc** (bool) – The CHADS2VASc score using all conditions prior to the window end. (analysis ID 904)
- **use_hfrs** (bool) – The Hospital Frailty Risk Score score using all conditions prior to the window end. (analysis ID 926)
- **use_distinct_condition_count** (bool) – The number of distinct condition concepts observed in the time window. (analysis ID 905)
- **use_distinct_ingredient_count** (bool) – The number of distinct ingredients observed in the time window. (analysis ID 906)
- **use_distinct_procedure_count** (bool) – The number of distinct procedures observed in the time window. (analysis ID 907)
- **use_distinct_measurement_count** (bool) – The number of distinct measurements observed in the time window. (analysis ID 908)
- **use_distinct_observation_count** (bool) – The number of distinct observations in the time window. (analysis ID 909)
- **use_visit_count** (bool) – The number of visits observed in the time window. (analysis ID 910)

- **use_visit_concept_count** (bool) – The number of visits observed in the time window, stratified by visit concept ID. (analysis ID 911)
- **temporal_start_days** (list[int]) – A list of integers representing the start of a time period, relative to the index date. 0 indicates the index date, -1 indicates the day before the index date, etc. The start day is included in the time period.
- **temporal_end_days** (list[int]) – A list of integers representing the end of a time period, relative to the index date. 0 indicates the index date, -1 indicates the day before the index date, etc. The end day is included in the time period.
- **included_covariate_concept_ids** (list) – A list of concept IDs that should be used to construct covariates.
- **add_descendants_to_include** (bool) – Should descendant concept IDs be added to the list of concepts to include?
- **excluded_covariate_concept_ids** (list) – A list of concept IDs that should NOT be used to construct covariates.
- **add_descendants_to_exclude** (bool) – Should descendant concept IDs be added to the list of concepts to exclude?
- **included_covariate_ids** (list) – A list of covariate IDs that should be restricted to.

Returns
An object of type covariateSettings, to be used in other functions.

Return type
ListVectorExtended

Examples

```
>>> settings = create_temporal_covariate_settings(
...     use_demographics_gender = True,
...     use_demographics_age = True,
... )
```

create_temporal_sequence_covariate_settings(*use_demographics_gender=False*, *use_demographics_age=False*, *use_demographics_age_group=False*, *use_demographics_race=False*, *use_demographics_ethnicity=False*, *use_demographics_index_year=False*, *use_demographics_index_month=False*, *use_condition_occurrence=False*, *use_condition_occurrence_primary_inpatient=False*, *use_condition_era_start=False*, *use_condition_era_group_start=False*, *use_drug_exposure=False*, *use_drug_era_start=False*, *use_drug_era_group_start=False*, *use_procedure_occurrence=False*, *use_device_exposure=False*, *use_measurement=False*, *use_measurement_value=False*, *use_observation=False*, *time_part='month'*, *time_interval=1*, *sequence_end_day=-1*, *sequence_start_day=-730*, *included_covariate_concept_ids=[]*, *add_descendants_to_include=False*, *excluded_covariate_concept_ids=[]*, *add_descendants_to_exclude=False*, *included_covariate_ids=[]*)

Create covariate settings

This function creates an object specifying how covariates should be constructed from data in the CDM model.

Wraps the R `FeatureExtraction::createTemporalSequenceCovariateSettings` function defined in `FeatureExtraction/R/DefaultTemporalCovariateSettings.R`.

Parameters

- **use_demographics_gender** (bool) – Gender of the subject. (analysis ID 1)
- **use_demographics_age** (bool) – Age of the subject on the index date (in years). (analysis ID 2)
- **use_demographics_age_group** (bool) – Age of the subject on the index date (in 5 year age groups) (analysis ID 3)
- **use_demographics_race** (bool) – Race of the subject. (analysis ID 4)
- **use_demographics_ethnicity** (bool) – Ethnicity of the subject. (analysis ID 5)
- **use_demographics_index_year** (bool) – Year of the index date. (analysis ID 6)
- **use_demographics_index_month** (bool) – Month of the index date. (analysis ID 7)
- **use_condition_occurrence** (bool) – One covariate per condition in the condition_occurrence table starting in the time window. (analysis ID 101)
- **use_condition_occurrence_primary_inpatient** (bool) – One covariate per condition observed as a primary diagnosis in an inpatient setting in the condition_occurrence table starting in the time window. (analysis ID 102)
- **use_condition_era_start** (bool) – One covariate per condition in the condition_era table starting in the time window. (analysis ID 201)
- **use_condition_era_group_start** (bool) – One covariate per condition era rolled up to SNOMED groups in the condition_era table starting in the time window. (analysis ID 203)

- **use_drug_exposure** (bool) – One covariate per drug in the drug_exposure table starting in the time window. (analysis ID 301)
- **use_drug_era_start** (bool) – One covariate per drug in the drug_era table starting in the time window. (analysis ID 401)
- **use_drug_era_group_start** (bool) – One covariate per drug rolled up to ATC groups in the drug_era table starting in the time window. (analysis ID 403)
- **use_procedure_occurrence** (bool) – One covariate per procedure in the procedure_occurrence table in the time window. (analysis ID 501)
- **use_device_exposure** (bool) – One covariate per device in the device exposure table starting in the time window. (analysis ID 601)
- **use_measurement** (bool) – One covariate per measurement in the measurement table in the time window. (analysis ID 701)
- **use_measurement_value** (bool) – One covariate containing the value per measurement-unit combination in the time window. If multiple values are found, the last is taken. (analysis ID 702)
- **use_observation** (bool) – One covariate per observation in the observation table in the time window. (analysis ID 801)
- **time_part** (str) – The interval scale ('DAY', 'MONTH', 'YEAR')
- **time_interval** (int) – Fixed interval length for timeId using the 'timePart' scale. For example, a 'timePart' of DAY with 'timeInterval' 30 has timeIds where timeId 1 is day 0 to day 29, timeId 2 is day 30 to day 59, etc.
- **sequence_end_day** (int) – What is the end day (relative to the index date) of the data extraction?
- **sequence_start_day** (int) – What is the start day (relative to the index date) of the data extraction?
- **included_covariate_concept_ids** (list) – A list of concept IDs that should be used to construct covariates.
- **add_descendants_to_include** (bool) – Should descendant concept IDs be added to the list of concepts to include?
- **excluded_covariate_concept_ids** (list) – A list of concept IDs that should NOT be used to construct covariates.
- **add_descendants_to_exclude** (bool) – Should descendant concept IDs be added to the list of concepts to exclude?
- **included_covariate_ids** (list) – A list of covariate IDs that should be restricted to.

Returns
An object of type covariateSettings, to be used in other functions.

Return type
ListVectorExtended

Examples

```
>>> settings = create_temporal_sequence_covariate_settings(
...     use_demographics_gender = True,
...     use_demographics_age = False,
...     use_demographics_age_group = True,
...     use_demographics_race = True,
...     use_demographics_ethnicity = True,
...     use_demographics_index_year = True,
...     use_demographics_index_month = True,
...     use_condition_occurrence = False,
...     use_condition_occurrence_primary_inpatient = False,
...     use_condition_era_start = False,
...     use_condition_era_group_start = False,
...     use_drug_exposure = False,
...     use_drug_era_start = False,
...     use_drug_era_group_start = False,
...     use_procedure_occurrence = True,
...     use_device_exposure = True,
...     use_measurement = True,
...     use_measurement_value = False,
...     use_observation = True,
...     time_part = "DAY",
...     time_interval = 1,
...     sequence_end_day = -1,
...     sequence_start_day = -730,
...     included_covariate_concept_ids = [],
...     add_descendants_to_include = False,
...     excluded_covariate_concept_ids = [],
...     add_descendants_to_exclude = False,
...     included_covariate_ids = []
... )
```

filter_by_cohort_definition_id(*covariate_data*, *cohort_id*)

Filter covariates by cohort definition ID

Wraps the R `FeatureExtraction::filterByCohortDefinitionId` function defined in `FeatureExtraction/R/HelperFunctions.R`.

Parameters

- **covariate_data** (*RS4 | CovariateData*) – An object of type `CovariateData`.
- **cohort_id** (*int*) – The cohort definition ID to keep.

Returns

An object of type `CovariateData`.

Return type

CovariateData

Examples

```
>>> covariate_data = filter_by_cohort_definition_id(
...     covariate_data = covariate_data,
...     cohort_id = 1
... )
```

filter_by_row_id(*covariate_data*, *row_ids*)

Filter covariates by row ID

Wraps the R `FeatureExtraction::filterByRowId` function defined in `FeatureExtraction/R/HelperFunctions.R`.

Parameters

- **covariate_data** (*RS4 | CovariateData*) – An object of type `CovariateData`.
- **row_ids** (*list[int]*) – A vector containing the row_ids to keep.

Returns
An object of type `CovariateData`.

Return type
CovariateData

Examples

```
>>> covariate_data <- filter_by_row_id(
...     covariate_data = covariate_data,
...     row_ids = [1,2]
... )
```

get_db_covariate_data(*cdm_database_schema*, *covariate_settings*, *connection_details=None*, *connection=None*, *oracle_temp_schema=None*, *cdm_version='5'*, *cohort_table='cohort'*, *cohort_database_schema=None*, *cohort_table_is_temp=False*, *cohort_id=-1*, *row_id_field='subject_id'*, *aggregated=False*)

Get covariate information from the database

Uses one or several covariate builder functions to construct covariates. This function uses the data in the CDM to construct a large set of covariates for the provided cohort. The cohort is assumed to be in an existing table with these fields: 'subject_id', 'cohort_definition_id', 'cohort_start_date'. Optionally, an extra field can be added containing the unique identifier that will be used as rowID in the output.

Wraps the R `FeatureExtraction::getDbCovariateData` function defined in `FeatureExtraction/R/GetCovariates.R`.

Parameters

- **connection_details** (`Optional[RS4]`) – An R object of type `connectionDetails` created using the function `createConnectionDetails` in the `DatabaseConnector` package. Either the `connection` or `connectionDetails` argument should be specified.
- **connection** (`Optional[RS4]`) – A connection to the server containing the schema as created using the `connect` function in the `DatabaseConnector` package. Either the `connection` or `connectionDetails` argument should be specified.
- **oracle_temp_schema** (`Optional[str]`) – A schema where temp tables can be created in Oracle.

- **cdm_database_schema** (str) – The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specify both the database and the schema, so for example 'cdm_instance.dbo'.
- **cdm_version** (str) – Define the OMOP CDM version used: currently supported is "5".
- **cohort_table** (str) – Name of the (temp) table holding the cohort for which we want to construct covariates
- **cohort_database_schema** (Optional[str]) – If the cohort table is not a temp table, specify the database schema where the cohort table can be found. On SQL Server, this should specify both the database and the schema, so for example 'cdm_instance.dbo'.
- **cohort_table_is_temp** (bool) – Is the cohort table a temp table?
- **cohort_id** (int) – For which cohort ID(s) should covariates be constructed? If set to -1, covariates will be constructed for all cohorts in the specified cohort table.
- **row_id_field** (str) – The name of the field in the cohort table that is to be used as the row_id field in the output table. This can be especially useful if there is more than one period per person.
- **covariate_settings** (ListVector) – Either an object of type covariateSettings as created using one of the createCovariate functions, or a list of such objects.
- **aggregated** (bool) – Should aggregate statistics be computed instead of covariates per cohort entry?

Returns
Returns an object of type covariateData, containing information on the covariates.

Return type
CovariateData

Examples

```
>>> cov_data = get_db_covariate_data(
...     connection_details = connection_details,
...     oracle_temp_schema = None,
...     cdm_database_schema = "main",
...     cdm_version = "5",
...     cohort_table = "cohort",
...     cohort_database_schema = "main",
...     cohort_table_is_temp = False,
...     cohort_id = -1,
...     row_id_field = "subject_id",
...     covariate_settings = cov_settings,
...     aggregated = False
... )
```

get_db_default_covariate_data(*cdm_database_schema*, *covariate_settings=None*, *target_database_schema=None*, *target_covariate_table=None*, *target_covariate_ref_table=None*, *target_analysis_ref_table=None*, *connection=None*, *oracle_temp_schema=None*, *cohort_table='#cohort_person'*, *cohort_id=-1*, *cdm_version='5'*, *row_id_field='subject_id'*, *aggregated=False*)

Get default covariate information from the database

Constructs a large default set of covariates for one or more cohorts using data in the CDM schema. Includes covariates for all drugs, drug classes, condition, condition classes, procedures, observations, etc.

Wraps the R `FeatureExtraction::getDbDefaultCovariateData` function defined in `FeatureExtraction/R/GetDefaultCovariates.R`.

Parameters

- **cdm_database_schema** (*str*) – The name of the database schema that contains the OMOP CDM instance.
- **covariate_settings** (*ListVector | ListVectorExtended*) – Either an object of type `covariateSettings` as created using one of the createCovariate functions, or a list of such objects.
- **(Optional)** (*aggregated*) – The name of the database schema where the resulting covariates should be stored.
- **(Optional)** – The name of the table where the resulting covariates will be stored. If not provided, results will be fetched to R. The table can be a permanent table in the `targetDatabaseSchema` or a temp table. If it is a temp table, do not specify `targetDatabaseSchema`.
- **(Optional)** – The name of the table where the covariate reference will be stored.
- **(Optional)** – The name of the table where the analysis reference will be stored.
- **connection** (*RS4 (Optional)*) – A connection to the OMOP CDM, as generated by `DatabaseConnector.connect`.
- **(Optional)** – The name of the schema where the temp tables should be created. This is only relevant for Oracle.
- **(Optional)** – The name of the (temp) table holding the cohort for which we want to construct covariates.
- **(Optional)** – The ID of the cohort for which we want to construct covariates. If set to -1, covariates will be constructed for all cohorts in the specified cohort table.
- **(Optional)** – The version of the CDM. Can be "4" or "5".
- **(Optional)** – The name of the field in the cohort table that is to be used as the row_id field in the output table.
- **(Optional)** – Should aggregate statistics be computed instead of covariates per cohort entry?

Returns

An object of class `covariateData`.

Return type

CovariateData

Examples

```
>>> results = get_db_default_covariate_data(
...     connection = connection,
...     cdm_database_schema = "main",
...     cohort_table = "cohort",
...     covariate_settings = create_default_covariate_settings(),
...     target_database_schema = "main",
...     target_covariate_table = "ut_cov",
...     target_covariate_ref_table = "ut_cov_ref",
...     target_analysis_ref_table = "ut_cov_analysis_ref"
... )
```

get_default_table1_specifications()

Get the default table 1 specifications

Loads the default specifications for a table 1, to be used with the `createTable1` function.

Wraps the R `FeatureExtraction::getDefaultTable1Specifications` function defined in `FeatureExtraction/R/Table1.R`.

Returns
Returns a `specifications` DataFrame.

Return type
DataFrame

Examples

```
>>> default_table1_specs = Table1.get_default_table1_specifications()
```

is_aggregated_covariate_data(*x*)

Check whether covariate data is aggregated

Wraps the R `FeatureExtraction::isAggregatedCovariateData` function defined in `FeatureExtraction/R/CovariateData.R`.

Parameters
x (*Any*) – The covariate data object to check.

Returns
True if `x` is an aggregated `CovariateData` object, False otherwise.

Return type
bool

Examples

```
>>> is_aggregated_cov_data = is_aggregated_covariate_data(covariate_data)
```

is_covariate_data(*x*)

Check whether an object is a CovariateData object

Wraps the R FeatureExtraction::isCovariateData function defined in FeatureExtraction/R/CovariateData.R.

Parameters
x (*Any*) – The object to check.

Returns
True if x is a CovariateData object, False otherwise.

Return type
bool

Examples

```
>>> is_cov_data = is_covariate_data(covariate_data)
```

is_temporal_covariate_data(*x*)

Check whether covariate data is temporal

Wraps the R FeatureExtraction::isTemporalCovariateData function defined in FeatureExtraction/R/CovariateData.R.

Parameters
x (*Any*) – The covariate data object to check.

Returns
True if x is a temporal CovariateData object, False otherwise.

Return type
bool

Examples

```
>>> is_temp_cov_data = is_temporal_covariate_data(covariate_data)
```

load_covariate_data(*file*, *read_only=False*)

Load the covariate data from a folder

This function loads an object of type covariateData from a folder in the file system.

Wraps the R FeatureExtraction::loadCovariateData function defined in FeatureExtraction/R/CovariateData.R.

Parameters

- **file** (*str*) – The name of the file containing the data.
- **read_only** (*bool | None*) – DEPRECATED: If True, the data is opened read only.

Returns
An object of class CovariateData.

Return type
CovariateData

Examples

```
>>> covariate_data = load_covariate_data(filename)
```

save_covariate_data(*covariate_data*, *file*)

Save the covariate data to folder

This function saves an object of type `covariateData`. The data will be written to a file specified by the user.

Wraps the R `FeatureExtraction::saveCovariateData` function defined in `FeatureExtraction/R/CovariateData.R`.

Parameters

- **covariate_data** (*RS4*) – An object of type `covariateData` as generated using `getDbCovariateData`.
- **file** (*str*) – The name of the file where the data will be written.
- **Effects** (*Side*) –
- **----------** –
- **to** (*A file containing an object of class covariateData will be written*) –
- **system.** (*the file*) –

Return type
None

Examples

```
>>> save_covariate_data(covariate_data, file = filename)
```

tidy_covariate_data(*covariate_data*, *min_fraction=0.001*, *normalize=True*, *remove_redundancy=True*)

Tidy covariate data

Normalize covariate values by dividing by the max and/or remove redundant covariates and/or remove infrequent covariates. For temporal covariates, redundancy is evaluated per time ID.

Wraps the R `FeatureExtraction::tidyCovariateData` function defined in `FeatureExtraction/R/Normalization.R`.

Parameters

- **covariate_data** (*RS4 | CovariateData*) – An object as generated using the `getDbCovariateData` function.
- **min_fraction** (*float*) – Minimum fraction of the population that should have a non-zero value for a covariate for that covariate to be kept. Set to 0 to don't filter on frequency.
- **normalize** (*bool*) – Normalize the covariates? (dividing by the max).
- **remove_redundancy** (*bool*) – Should redundant covariates be removed?

Returns
An object of class `covariateData`.

Return type
CovariateData

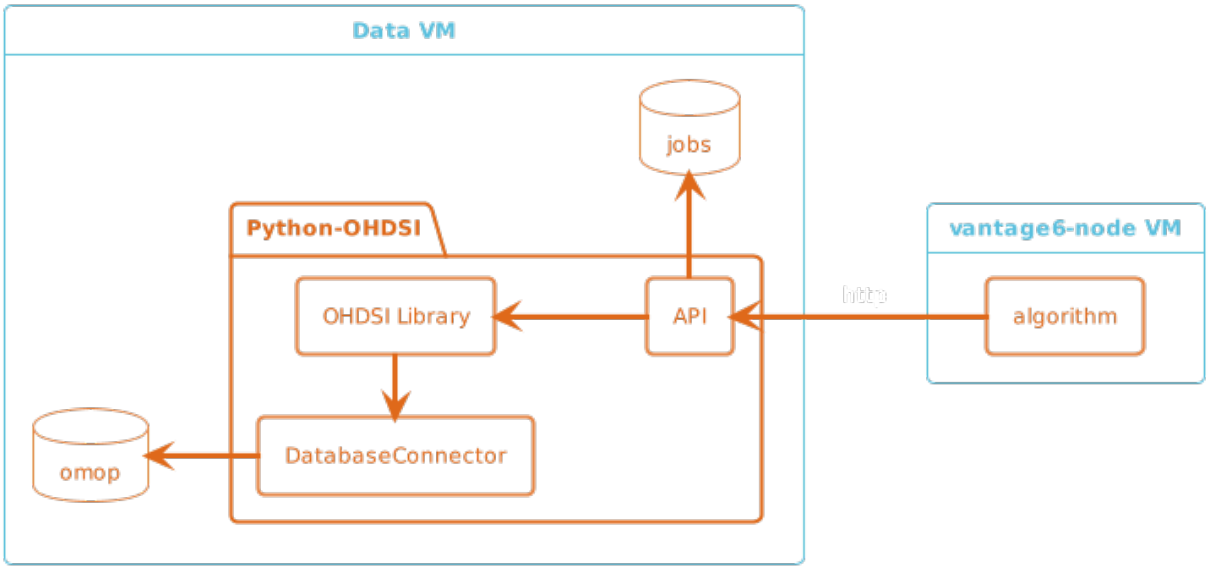
Examples

```
>>> covariate_data = tidy_covariate_data(
...     covariate_data = covariate_data,
...     min_fraction = 0.001,
...     normalize = True,
...     removeRedundancy = True
... )
```

4.7 API

The API allows applications to retrieve data from an OMOP source through a http interface. The API is implemented in Python and uses the Flask framework.

Because OMOP queries can be quite time consuming, therefore the API will make use of background tasks. The API will return a job id, which can be used to retrieve the results of the query later.



4.7.1 Start the API

```
docker compose up -d
```

Or to run it in development mode:

```
docker compose -f docker-compose.yml -f docker-compose.dev.yml up -d
```

4.7.2 Build the API

```
docker compose build
```

4.8 Contributing

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX