Package ‘osdc’

December 10, 2025

Type Package

Title Open Source Diabetes Classifier for Danish Registers

Version 0.9.17

Description The algorithm first identifies a population of individuals from Danish register data with any type of diabetes as individuals with two or more inclusion events. Then, it splits this population into individuals with either type 1 diabetes or type 2 diabetes by identifying individuals with type 1 diabetes and classifying the remainder of the diabetes population as having type 2 diabetes.

License MIT + file LICENSE

URL `https://github.com/steno-aarhus/osdc`, `https://steno-aarhus.github.io/osdc/`

BugReports `https://github.com/steno-aarhus/osdc/issues`

Depends R (>= 4.2.0)

Imports checkmate, cli, codeCollection, dbplyr, dplyr, duckplyr, fabricatr, lifecycle, lubridate, purrr, rlang, rvest, stats, tidyselect, utils

Suggests glue, knitr, quarto, rmarkdown, spelling, stringr, testthat (>= 3.0.0), tidyr, tibble

VignetteBuilder quarto

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

NeedsCompilation no

Author Signe Kirk Brødbæk [aut] (ORCID: <`https://orcid.org/0009-0000-2208-7088`>), Anders Aasted Isaksen [aut] (ORCID: <`https://orcid.org/0000-0001-8457-5466`>), Luke William Johnston [aut, cre] (ORCID:

<https://orcid.org/0000-0003-4169-2616>),
Steno Diabetes Center Aarhus [cph],
Aarhus University [cph]

Maintainer Luke William Johnston <lwjohnst@gmail.com>

Repository CRAN

Date/Publication 2025-12-10 21:20:02 UTC

Contents

algorithm 2
classify_diabetes 3
edge_cases 4
non_cases 5
non_cases_metadata 6
registers 6
simulate_registers 7

Index **8**

`algorithm` *A list of the algorithmic logic underlying osdc.*

Description

This nested list contains the logic details of the algorithm.

Usage

```
algorithm()
```

Format

Is a list with nested lists that have these named elements:

register Optional. The register used for this logic

title The title to use when displaying the logic in tables.

logic The logic itself.

comments Some additional comments on the logic.

Value

A nested list with the algorithmic logic. Contains fields `register`, `title`, `logic`, and `comments`.

See Also

See the `vignette("algorithm")` for the logic used to filter these patients.

Examples

```
algorithm()$is_hba1c_over_threshold
algorithm()$is_gld_code$logic
```

`classify_diabetes` *Classify diabetes status using Danish registers.*

Description

This function requires that each source of register data is represented as a single DuckDB object in R (e.g. a connection to Parquet files). Each DuckDB object must contain a single table covering all years of that data source, or at least the years you have and are interested in.

Usage

```
classify_diabetes(
  kontakter,
  diagnoser,
  lpr_diag,
  lpr_adm,
  sysi,
  sssy,
  lab_forsker,
  bef,
  lmdb,
  stable_inclusion_start_date = "1998-01-01"
)
```

Arguments

`kontakter`	The contacts information table from the LPR3 patient register
`diagnoser`	The diagnoses information table from the LPR3 patient register
`lpr_diag`	The diagnoses information table from the LPR2 patient register
`lpr_adm`	The administrative information table from the LPR2 patient register
`sysi`	The SYSI table from the health service register
`sssy`	The SSSY table from the health service register
`lab_forsker`	The register for laboratory results for research
`bef`	The BEF table from the civil register
`lmdb`	The LMDB table from the prescription register
`stable_inclusion_start_date`	Cutoff date after which inclusion events are considered true incident diabetes cases. Defaults to "1998-01-01", which is one year after the data on pregnancy events from the Patient Register are considered valid for dropping gestational diabetes-related purchases of glucose-lowering drugs. This default assumes that the user is using LPR and LMDB data from at least Jan 1 1997 onward. If the user only has access to LPR and LMDB data from a later date, this parameter should be set to one year after the beginning of the user's data coverage.

Value

The same object type as the input data, which would be a `duckplyr::duckdb_tibble()` type object.

See Also

See the osdc vignette for a detailed description of the internal implementation of this classification function.

Examples

```
# Can't run this multiple times, will cause an error as the table
# has already been created in the DuckDB connection.
register_data <- registers() |>
  names() |>
  simulate_registers() |>
  purrr::map(duckplyr::as_duckdb_tibble) |>
  purrr::map(duckplyr::as_tbl)

classify_diabetes(
  kontakter = register_data$kontakter,
  diagnoser = register_data$diagnoser,
  lpr_diag = register_data$lpr_diag,
  lpr_adm = register_data$lpr_adm,
  sysi = register_data$sysi,
  sssy = register_data$sssy,
  lab_forsker = register_data$lab_forsker,
  bef = register_data$bef,
  lmdb = register_data$lmdb
)
```

`edge_cases` *Create a synthetic dataset of edge case inputs*

Description

This function generates a list of tibbles representing the Danish health registers and the data necessary to run the algorithm. The dataset contains 23 individual cases (`pnrs`), each designed to test a specific logical branch of the diabetes classification algorithm, including inclusion, exclusion, censoring, and type classification rules.

The generated data is used in `testthat` tests to ensure the algorithm behaves as expected under a wide range of conditions, but it is also intended to be explored by users to better understand how the algorithm logic works.

Usage

```
edge_cases()
```

Value

A named list of 9 `tibble::tibble()` objects, each representing a different health register: `bef`, `lmdb`, `lpr_adm`, `lpr_diag`, `kontakter`, `diagnoser`, `sysi`, `sssy`, and `lab_forsker`.

Examples

```
edge_cases()
```

`non_cases` *List of non-cases to test the diabetes classification algorithm*

Description

This function generates a list of tibbles representing the Danish health registers and the data necessary to run the algorithm. The dataset contains individuals who should *not* be included in the final classified cohort.

Usage

```
non_cases()
```

Details

The generated data is used in `testthat` tests to ensure the algorithm behaves as expected under a wide range of conditions, but it is also intended to be explored by users to better understand how the algorithm logic works and to be shown in the documentation.

Value

A named list of 9 `tibble::tibble()` objects, each representing a different health register: `bef`, `lmdb`, `lpr_adm`, `lpr_diag`, `kontakter`, `diagnoser`, `sysi`, `sssy`, and `lab_forsker`.

Examples

```
non_cases()
```

`non_cases_metadata` *Description of the different non-cases included in* `non_cases()`

Description

All cases, aside from what would exclude them from being classified as described in the metadata here, would otherwise be classified as having diabetes.

Usage

```
non_cases_metadata()
```

Value

A named list of character strings, where each name corresponds to a non-case PNR in the dataset generated by `non_cases()`.

Examples

```
non_cases_metadata()
```

`registers` *Register variables (with descriptions) required for the osdc algorithm.*

Description

Register variables (with descriptions) required for the osdc algorithm.

Usage

```
registers()
```

Value

Outputs a list of registers and variables required by osdc. Each list item contains the official Danish name of the register, the start year, the end year, and the variables with their descriptions. The variables item is a data frame with 4 columns:

name The official name of the variable found in the register.

danish_description The official Danish description of the variable.

english_description The translated English description of the variable.

data_type The data type, e.g. "character" of the variable. Could have multiple options (e.g. "Date" or "character").

Source

Many of the details within the `registers()` metadata come from the full official list of registers from Statistics Denmark (DST): `https://www.dst.dk/extranet/forskningvariabellister/Oversigt%20over%20registre.html`

Examples

```
registers()
```

`simulate_registers` *Simulate a fake data frame of one or more Danish registers*

Description

Simulate a fake data frame of one or more Danish registers

Usage

```
simulate_registers(registers, n = 1000)
```

Arguments

`registers`	The name of the register you want to simulate.
`n`	The number of rows to simulate for the resulting register.

Value

A list with simulated register data, as a `tibble::tibble()`.

Examples

```
simulate_registers(c("bef", "sysi"))
simulate_registers("bef")
simulate_registers("diagnoser")
```

Index

algorithm, 2

classify_diabetes, 3

duckplyr::duckdb_tibble(), *4*

edge_cases, 4

non_cases, 5
non_cases_metadata, 6

osdc, *4*

registers, 6
registers(), *7*

simulate_registers, 7

tibble::tibble(), *5*, *7*