

Package: osdc (via r-universe)

October 20, 2026

Type Package

Title Open Source Diabetes Classifier for Danish Registers

Version 0.11.3

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, dplyr (>= 1.2.0), dbplyr (>= 2.5.1), duckplyr (>= 1.2.1), fabricatr, lifecycle, lubridate (>= 1.9.5), purrr (>= 1.2.1), rlang (>= 1.1.7), stats, tidyselect (>= 1.2.1), utils

Suggests glue, knitr, quarto, rmarkdown, spelling, stringr, testthat (>= 3.0.0), tidyr, tibble, arrow (>= 22.0.0.1), duckdb (>= 1.5.4)

VignetteBuilder quarto

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev xz-utils

Repository <https://steno-aarhus.r-universe.dev>
Date/Publication 2026-10-20 09:25:37 UTC
RemoteUrl <https://github.com/steno-aarhus/osdc>
RemoteRef HEAD
RemoteSha cf348a600c357983281c62c50483921a9c619742

Contents

algorithm	2
classify_diabetes	3
edge_cases	4
join_registers	5
non_cases	5
non_cases_metadata	6
prepare_lpr2	7
prepare_lpr3a	8
prepare_lpr3f	9
registers	10
simulate_registers	10
Index	12

algorithm	<i>A list of the algorithmic logic underlying osdc.</i>
-----------	---

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	<i>Classify diabetes status using Danish registers.</i>
-------------------	---

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(
  lpr,
  hsr,
  lab_forsker,
  bef,
  lmdb,
  stable_inclusion_start_date = "1998-01-01"
)
```

Arguments

lpr	The unified LPR register, see join_registers()
hsr	The unified health services registers (SYSI and SSSY), see join_registers()
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 `vignette("osdc", package = "osdc")` vignette for a more details and on how to use this function.

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`, `lpr3a_kontakt`, `lpr3a_diagnose`, `lpr3f_kontakter`, `lpr3f_diagnoser`, `sysi`, `sssy`, and `lab_forsker`.

Examples

```
edge_cases()
```

join_registers	<i>Join prepared registers</i>
----------------	--------------------------------

Description

Join prepared registers

Usage

```
join_registers(register_list)
```

Arguments

register_list A list of the prepared registers, from e.g. [prepare_lpr2\(\)](#).

Value

A single object with all rows from each register in register_list.

Examples

```
register_data <- simulate_registers(c(
  "lpr_adm",
  "lpr_diag",
  "lpr3f_kontakter",
  "lpr3f_diagnoser",
  "sssy",
  "sysi"
))
join_registers(list(
  prepare_lpr2(register_data$lpr_adm, register_data$lpr_diag),
  prepare_lpr3f(
    register_data$lpr3f_kontakter,
    register_data$lpr3f_diagnoser
  )
))
join_registers(list(register_data$sysi, register_data$sssy))
```

non_cases	<i>List of non-cases to test the diabetes classification algorithm</i>
-----------	--

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`, `lpr3a_kontakt`, `lpr3a_diagnose`, `lpr3f_kontakter`, `lpr3f_diagnoser`, `sysi`, `sssy`, and `lab_forsker`.

Examples

```
non_cases()
```

non_cases_metadata	<i>Description of the different non-cases included in non_cases()</i>
--------------------	---

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()
```

prepare_lpr2	<i>Prepare and join the two LPR2 registers to extract diabetes and pregnancy diagnoses.</i>
--------------	---

Description

Prepare and join the two LPR2 registers to extract diabetes and pregnancy diagnoses.

Usage

```
prepare_lpr2(lpr_adm, lpr_diag)
```

Arguments

lpr_adm	The LPR2 register containing hospital admissions.
lpr_diag	The LPR2 register containing diabetes diagnoses.

Value

The same type as the input data, as a `duckplyr::duckdb_tibble()`, with the following columns:

- `pnr`: The personal identification variable.
- `date`: The date of all the recorded diagnosis (renamed from `d_inddto` or `dato_start`).
- `is_primary_diagnosis`: Whether the diagnosis was a primary diagnosis.
- `is_diabetes_code`: Whether the diagnosis was any type of diabetes.
- `is_t1d_code`: Whether the diagnosis was T1D-specific.
- `is_t2d_code`: Whether the diagnosis was T2D-specific.
- `is_pregnancy_code`: Whether the person has an event related to pregnancy like giving birth or having a miscarriage at the given date.
- `is_endocrinology_dept`: Whether the diagnosis was made by an endocrinology medical department.
- `is_medical_dept`: Whether the diagnosis was made by a non-endocrinology medical department.

See Also

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

prepare_lpr3a	<i>Prepare and join the two LPR3A registers to extract diabetes and pregnancy diagnoses.</i>
---------------	--

Description

Prepare and join the two LPR3A registers to extract diabetes and pregnancy diagnoses.

Usage

```
prepare_lpr3a(lpr3a_kontakt, lpr3a_diagnose)
```

Arguments

lpr3a_kontakt The LPR3A register containing hospital contacts/admissions.

lpr3a_diagnose The LPR3A register containing diabetes diagnoses.

Value

The same type as the input data, as a `duckplyr::duckdb_tibble()`, with the following columns:

- pnr: The personal identification variable.
- date: The date of all the recorded diagnosis (renamed from d_inddto or dato_start).
- is_primary_diagnosis: Whether the diagnosis was a primary diagnosis.
- is_diabetes_code: Whether the diagnosis was any type of diabetes.
- is_t1d_code: Whether the diagnosis was T1D-specific.
- is_t2d_code: Whether the diagnosis was T2D-specific.
- is_pregnancy_code: Whether the person has an event related to pregnancy like giving birth or having a miscarriage at the given date.
- is_endocrinology_dept: Whether the diagnosis was made by an endocrinology medical department.
- is_medical_dept: Whether the diagnosis was made by a non-endocrinology medical department.

See Also

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

prepare_lpr3f	<i>Prepare and join the two LPR3F registers to extract diabetes and pregnancy diagnoses.</i>
---------------	--

Description

Prepare and join the two LPR3F registers to extract diabetes and pregnancy diagnoses.

Usage

```
prepare_lpr3f(lpr3f_kontakter, lpr3f_diagnoser)
```

Arguments

lpr3f_kontakter	The LPR3F register containing hospital contacts/admissions.
lpr3f_diagnoser	The LPR3F register containing diabetes diagnoses.

Value

The same type as the input data, as a `duckplyr::duckdb_tibble()`, with the following columns:

- pnr: The personal identification variable.
- date: The date of all the recorded diagnosis (renamed from d_inddto or dato_start).
- is_primary_diagnosis: Whether the diagnosis was a primary diagnosis.
- is_diabetes_code: Whether the diagnosis was any type of diabetes.
- is_t1d_code: Whether the diagnosis was T1D-specific.
- is_t2d_code: Whether the diagnosis was T2D-specific.
- is_pregnancy_code: Whether the person has an event related to pregnancy like giving birth or having a miscarriage at the given date.
- is_endocrinology_dept: Whether the diagnosis was made by an endocrinology medical department.
- is_medical_dept: Whether the diagnosis was made by a non-endocrinology medical department.

See Also

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

registers	<i>Register variables (with descriptions) required for the osdc algorithm.</i>
-----------	--

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. Each register item is a list with 4 items:

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.

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	<i>Simulate a fake data frame of one or more Danish registers</i>
--------------------	---

Description

Simulate a fake data frame of one or more Danish registers

Usage

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

Arguments

<code>registers</code>	The name of the register you want to simulate.
<code>n</code>	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")
```

Index

algorithm, [2](#)

classify_diabetes, [3](#)

duckplyr::duckdb_tibble(), [4](#), [7–9](#)

edge_cases, [4](#)

join_registers, [5](#)
join_registers(), [3](#)

non_cases, [5](#)
non_cases_metadata, [6](#)

prepare_lpr2, [7](#)
prepare_lpr2(), [5](#)
prepare_lpr3a, [8](#)
prepare_lpr3f, [9](#)

registers, [10](#)
registers(), [10](#)

simulate_registers, [10](#)

tibble::tibble(), [4](#), [6](#), [11](#)