

# Package: healthdb (via r-universe)

September 29, 2026

**Type** Package

**Title** Working with Healthcare Databases

**Version** 0.5.1.9000

**Description** A system for identifying diseases or events from healthcare databases and preparing data for epidemiological studies. It includes capabilities not supported by 'SQL', such as matching strings by 'stringr' style regular expressions, and can compute comorbidity scores (Quan et al. (2005) <[doi:10.1097/01.mlr.0000182534.19832.83](https://doi.org/10.1097/01.mlr.0000182534.19832.83)>) directly on a database server. The implementation is based on 'dbplyr' with full 'tidyverse' compatibility.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** clock, data.table (>= 1.18.4), dbplyr (>= 2.5.1), dplyr (>= 1.1.0), glue, lubridate, magrittr, purrr, rlang, stringr (>= 1.5.0), tidyr

**URL** <https://github.com/KevinHzq/healthdb>,  
<https://kevinhzq.github.io/healthdb/>

**BugReports** <https://github.com/KevinHzq/healthdb/issues>

**Suggests** DBI, knitr, odbc, rmarkdown, RPostgres, testthat (>= 3.0.0), withr, RSQLite

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

**Depends** R (>= 3.5)

**Config/pak/sysreqs** libicu-dev

**Repository** <https://kevinhzq.r-universe.dev>

**Date/Publication** 2026-07-31 06:18:35 UTC

**RemoteUrl** <https://github.com/kevinhzq/healthdb>

**RemoteRef** HEAD  
**RemoteSha** 03f31af47eb2204292f2ab8d66f34a86cbcdfaef

**Contents**

|                                |           |
|--------------------------------|-----------|
| bind_source . . . . .          | 2         |
| build_def . . . . .            | 3         |
| collapse_episode . . . . .     | 5         |
| compute_comorbidity . . . . .  | 6         |
| compute_duration . . . . .     | 10        |
| cut_period . . . . .           | 11        |
| define_case . . . . .          | 12        |
| define_case_with_age . . . . . | 15        |
| elix_codes . . . . .           | 19        |
| exclude . . . . .              | 21        |
| execute_def . . . . .          | 22        |
| fetch_var . . . . .            | 23        |
| identify_rows . . . . .        | 25        |
| if_date . . . . .              | 28        |
| lookup . . . . .               | 29        |
| make_test_dat . . . . .        | 30        |
| pool_case . . . . .            | 31        |
| report_n . . . . .             | 34        |
| restrict_date . . . . .        | 35        |
| restrict_n . . . . .           | 37        |
| <b>Index</b>                   | <b>39</b> |

---

|             |                                                                                             |
|-------------|---------------------------------------------------------------------------------------------|
| bind_source | <i>Row-bind a list of data.frames or remote tables with variable selection and renaming</i> |
|-------------|---------------------------------------------------------------------------------------------|

---

**Description**

Row bind a list of data.frames or remote tables with variable selection and renaming.

**Usage**

```
bind_source(data, ..., force_proceed = getOption("healthdb.force_proceed"))
```

**Arguments**

data                    A list of data.frame or remote tables, e.g., output from `execute_def()`.

... Named arguments for each variable included in the output. The argument name should be the new name in the output, and the right hand side of the argument is a character vector of the original names. The name vector and the list elements in data will be matched by position. if an output variable only came from some of the sources, fill the name vector to a length equal to the number of sources with NA. For example, if var only comes from the second out of three sources, then var = c(NA, 'nm\_in\_src2', NA).

force\_proceed A logical for whether to ask for user input in order to proceed when remote tables are needed to be collected for binding. The default is FALSE to let user be aware of that the downloading process may be slow. Use options(healthdb.force\_proceed = TRUE) to suppress the prompt once and for all. In non-interactive sessions (e.g., scripts run via Rscript, knitr), the confirmation prompt cannot be displayed, and the function stops with an error unless force\_proceed = TRUE.

### Value

A data.frame or remote table containing combined rows of the input list with variables specified by ...

### Examples

```
df1 <- subset(iris, Species == "setosa")
df2 <- subset(iris, Species == "versicolor")
df3 <- subset(iris, Species == "virginica")

bind_source(list(df1, df2, df3),
  s_l = "Sepal.Length",
  s_w = "Sepal.Width",
  p_l_setosa = c("Petal.Length", NA, NA),
  p_l_virginica = c(NA, NA, "Petal.Length")
) %>%
  head()
```

---

build\_def

*Build case definition function calls*

---

### Description

This function assembles function calls from the supplied functions and their required arguments, leaving the data argument empty for easy re-use of the definition calls with different data and batch execution (see [execute\\_def\(\)](#) for detail). It is useful for defining multiple diseases/events across multiple sources.

### Usage

```
build_def(def_lab, src_labs, def_fn = define_case, fn_args)
```

## Arguments

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| def_lab  | A single character label for the definition, e.g., some disease.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| src_labs | A character vector of place-holder names for the data sources that will be used to execute the definition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| def_fn   | A list of functions (default: <code>define_case()</code> ) that will filter the source data sets and keep clients that met the case definition. The length of the list should be either 1 or equal to the length of <code>src_labs</code> . If length = 1, the same function will be applied to all sources; otherwise, <code>def_fn</code> should match <code>src_lab</code> by position. User can supply custom functions but must put input data as the first argument and name it data. Note: Mixing define functions (e.g., <code>define_case()</code> and <code>define_case_with_age()</code> ) that accept . . . additional arguments may cause error. It is recommended to use the same define function for all sources. |
| fn_args  | A named list of arguments passing to the <code>def_fn</code> . Each element in the list should have the same name as an argument in the source-specific <code>def_fn</code> , and the element length should also be either 1 or equal to the number of sources. If you have <code>def_fn</code> functions taking different sets of arguments, include the union in one list.                                                                                                                                                                                                                                                                                                                                                     |

## Value

A tibble with a number of rows equal to the length of `src_labs`, containing the input arguments and the synthetic function call in the `fn_call` column.

## Examples

```
sud_def <- build_def("SUD", # usually a disease name
  src_lab = c("src1", "src2"), # identify from multiple sources, e.g., hospitalization, ED visits.
  # functions that filter the data with some criteria,
  # including mean here for src2 as a trivial example
  # to show only valid arguments will be in the call
  def_fn = list(define_case, mean),
  fn_args = list(
    vars = list(starts_with("diagx"), "diagx_2"),
    match = "start", # "start" will be applied to all sources as length = 1
    vals = list(c("304"), c("305")),
    clnt_id = "clnt_id",
    # c() can be used in place of list
    # if this argument only takes one value for each source
    n_per_clnt = c(2, 3),
    x = list(1:10) # src2 with mean as def_fn will only accept this argument
  )
)

# the result is a tibble
sud_def

# the fn_call column stores the code that can be run with execute_def
sud_def # fn_call
```

---

|                  |                                                            |
|------------------|------------------------------------------------------------|
| collapse_episode | <i>Group records no more than n days apart as episodes</i> |
|------------------|------------------------------------------------------------|

---

## Description

This function is useful for collapsing, e.g., medication dispensation or hospitalization, records into episodes if the records' dates are no more than *n* days apart. The length of the gap can be relaxed by another grouping variable. Records with a missing (NA) start or end date are removed with a warning before episodes are derived, because the gap to such records is undefined.

## Usage

```
collapse_episode(
  data,
  cInt_id,
  start_dt,
  end_dt = NULL,
  gap,
  overwrite = NULL,
  gap_overwrite = 99999,
  .dt_trans = data.table::as.IDate,
  ...
)
```

## Arguments

|                            |                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>          | A data.frame or remote table that contains the id and date variables.                                                                                                                                                                                                                                                                                                                                  |
| <code>cInt_id</code>       | Column name of subject/person ID.                                                                                                                                                                                                                                                                                                                                                                      |
| <code>start_dt</code>      | Column name of the starting date of records.                                                                                                                                                                                                                                                                                                                                                           |
| <code>end_dt</code>        | Column name of the end date of records. The default is NULL, assuming the record lasts one day and only the start date will be used to calculate the gaps between records.                                                                                                                                                                                                                             |
| <code>gap</code>           | A number in days that will be used to separate episodes. For example, <code>gap = 7</code> means collapsing records no more than 7 days apart. Note that the number of days apart will be calculated as numeric difference between two days, so that 2020-01-07 and 2020-01-01 is considered as 6 days apart.                                                                                          |
| <code>overwrite</code>     | Column name of a grouping variable determining whether the consecutive records are related and should have a different gap value. For example, dispensing records may have the same original prescription number, and a different gap value can be assigned for such situation, e.g., the days between two records is <code>&gt; gap</code> , but these records still belong to the same prescription. |
| <code>gap_overwrite</code> | A different gap value used for related records. The default is 99999, which practically means all records with the same overwrite variable will be collapsed.                                                                                                                                                                                                                                          |
| <code>.dt_trans</code>     | Function to transform <code>start_dt/end_dt</code> . For data.frame input only. Default is <code>data.table::as.IDate()</code> .                                                                                                                                                                                                                                                                       |

... Additional arguments passing to the .dt\_trans function. For data.frame input only.

### Value

The original data.frame or remote table with new columns indicating episode grouping. The new variables include:

- epi\_id: unique identifier of episodes across the whole data set
- epi\_no: identifier of episodes within a client/group
- epi\_seq: identifier of records within an episode
- epi\_start/stop\_dt: start and end dates corresponding to epi\_id

### Examples

```
# toy dispensing records: each row has a supply start and end date
rx <- data.frame(
  clnt_id = c(1, 1, 1, 2, 2),
  start = as.Date(c(
    "2020-01-01", "2020-03-01", "2020-06-01",
    "2020-01-01", "2020-01-10"
  )),
  end = as.Date(c(
    "2020-01-15", "2020-03-15", "2020-06-15",
    "2020-01-20", "2020-01-25"
  )),
  rx_id = c("a", "a", "b", "c", "c")
)

# collapse records no more than 14 days apart into episodes
collapse_episode(rx, clnt_id, start_dt = start, end_dt = end, gap = 14)

# end_dt may be omitted, then each record is assumed to last a single day
collapse_episode(rx, clnt_id, start_dt = start, gap = 14)

# overwrite keeps related records (e.g. refills of the same prescription)
# in one episode regardless of the gap: consecutive records with the same
# rx_id are collapsed using gap_overwrite, while a change in rx_id falls
# back to the regular gap. Here client 1's two "a" records merge even though
# they are more than 14 days apart.
collapse_episode(rx, clnt_id,
  start_dt = start, end_dt = end, gap = 14, overwrite = rx_id
)
```

## Description

This function computes unweighted Elixhauser Comorbidity Index for both data.frame and remote table input. The ICD codes used to identify the 31 disease categories is from Quan et al. (2005).

## Usage

```
compute_comorbidity(
  data,
  vars,
  icd_ver = c("ICD-10", "ICD-9-CM-3digits", "ICD-9-CM-5digits"),
  clnt_id,
  uid = NULL,
  sum_by = c("row", "clnt"),
  excl = NULL
)
```

## Arguments

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data    | Data.frames or remote tables (e.g., from <code>dbplyr::tbl_sql()</code> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| vars    | An expression passing to <code>dplyr::select()</code> . It can be quoted/unquoted column names, or helper functions, such as <code>dplyr::starts_with()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| icd_ver | One of <code>c("ICD-10", "ICD-9-CM-3digits", "ICD-9-CM-5digits")</code> . Specify the ICD code version used in data. The ICD-10 and ICD-9-CM 5 digits version are from Quan et al. (2005). The ICD-9-CM 3 digits version is adopted from Manitoba Centre for Health Policy. It uses BOTH 3-digit and 5-digit codes in search. See their web page for cautions and limitations of the 3 digit version if your data only has 3-digit codes ( <a href="http://mchp-appserv.cpe.umanitoba.ca/viewConcept.php?printer=Y&amp;conceptID=1436#CAUTIONS">http://mchp-appserv.cpe.umanitoba.ca/viewConcept.php?printer=Y&amp;conceptID=1436#CAUTIONS</a> ). For all versions, the codes in the lists are at the category/subcategory level and cover all their subdivisions (the ".x" notation in Quan et al. (2005)). Therefore, matching is done by prefix: a diagnosis value in data matches a listed code if its leading characters equal the code. For example, "E1152" in data would be captured by ICD-10 "E115" (Diabetes Complicated), and "4280" by ICD-9 "428" (Congestive Heart Failure). This reproduces the reference SAS implementation, which compares every code with the SAS IN: (starts-with) operator (see <a href="http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/_ElixhauserICD9CM.sas.txt">http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/_ElixhauserICD9CM.sas.txt</a> and <a href="http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/_ElixhauserICD10.sas.txt">http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/_ElixhauserICD10.sas.txt</a> ). See the exported dataset <a href="#">elix_codes</a> for the code lists and matching lengths. Codes in data must not contain dots (e.g., use "E1152" not "E11.52", and "4280" not "428.0"); otherwise, codes may not be matched correctly. Codes should also be upper case (e.g., "E1152" not "e1152"): the comparison is case-sensitive for data.frame input and on most database backends (including SQLite and PostgreSQL), but follows the database collation on SQL Server, which is commonly case-insensitive. Note that some codes belong to multiple categories in Quan et al. (2005), e.g., I11.0 indicates both Congestive Heart Failure (I099, I110, ...) and Hypertension Complicated (I11.x); records with such codes are counted in all the categories they match. |
| clnt_id | Grouping variable (quoted/unquoted).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

|        |                                                                                                                                                                                                                                                                                                              |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uid    | Variable name for a unique row identifier. It is necessary for SQL to produce consistent result based on sorting.                                                                                                                                                                                            |
| sum_by | One of "row" or "clnt". The "row" option computes total score for each row (default), and the "clnt" option summarizes total score by clnt_id. Each disease categories will be counted only once in the calculation regardless of multiple records in a category.                                            |
| excl   | A character vector of disease categories labels that should be excluded in the total score calculation. This is useful when some of the categories are the exposure/outcome of interest, and the goal is to measure comorbidity excluding these disease. See detail for a list of the categories and labels. |

## Details

List of disease categories - labels (in quote):

1. Congestive Heart Failure - "chf"
2. Cardiac Arrhythmia - "arrhy"
3. Valvular Disease - "vd"
4. Pulmonary Circulation Disorders - "pcd"
5. Peripheral Vascular Disorders - "pvd"
6. Hypertension Uncomplicated - "hptn\_nc"
7. Hypertension Complicated - "hptn\_c"
8. Paralysis - "para"
9. Other Neurological Disorders - "othnd"
10. Chronic Pulmonary Disease - "copd"
11. Diabetes Uncomplicated - "diab\_nc"
12. Diabetes Complicated - "diab\_c"
13. Hypothyroidism - "hptothy"
14. Renal Failure - "rf"
15. Liver Disease - "ld"
16. Peptic Ulcer Disease excluding bleeding - "pud\_nb"
17. AIDS/HIV - "hiv"
18. Lymphoma - "lymp"
19. Metastatic Cancer - "mets"
20. Solid Tumor without Metastasis - "tumor"
21. Rheumatoid Arthritis/collagen - "rheum\_a"
22. Coagulopathy - "coag"
23. Obesity - "obesity"
24. Weight Loss - "wl"
25. Fluid and Electrolyte Disorders - "fluid"
26. Blood Loss Anemia - "bla"

- 27. Deficiency Anemia - "da"
- 28. Alcohol Abuse - "alcohol"
- 29. Drug Abuse - "drug"
- 30. Psychoses - "psycho"
- 31. Depression - "dep"

The full ICD code lists defining these categories, including the matching rule for each code (prefix vs. exact), are available in the exported dataset [elix\\_codes](#).

### Value

A data.frame or remote table with binary indicators for each categories as columns.

### References

Quan H, Sundararajan V, Halfon P, Fong A, Burnand B, Luthi JC, Saunders LD, Beck CA, Feasby TE, Ghali WA. Coding algorithms for defining comorbidities in ICD-9-CM and ICD-10 administrative data. Med Care 2005;43(11):1130-1139.

### See Also

[elix\\_codes](#)

### Examples

```
# make ICD-9 toy data
df <- data.frame(
  uid = 1:10, clnt_id = sample(1:3, 10, replace = TRUE),
  diagx_1 = c("193", "2780", "396", "4254", "4150", "401", "401", "0932", "5329", "2536"),
  diagx_2 = c(NA, NA, "72930", "V6542", "493", "405", "5880", "2409", "714", NA)
)

# compute Elixhauser Comorbidity Index by row
# uid is needed for by row calculation
# 3 categories were excluded in total_eci
compute_comorbidity(df,
  vars = starts_with("diagx"),
  icd_ver = "ICD-9-CM-5digits",
  clnt_id = clnt_id, uid = uid,
  excl = c("drug", "psycho", "dep")
)

# compute ECI by person
compute_comorbidity(df,
  vars = starts_with("diagx"),
  icd_ver = "ICD-9-CM-5digits",
  clnt_id = clnt_id,
  sum_by = "clnt"
)
```

---

|                  |                                           |
|------------------|-------------------------------------------|
| compute_duration | <i>Compute duration between two dates</i> |
|------------------|-------------------------------------------|

---

## Description

This function is meant to be for data frame input only and used with `dplyr::mutate()` to compute age or duration between two character or Date columns. If a vector of breaks is given, the output will be converted to factor with labels generated automatically.

## Usage

```
compute_duration(
  from,
  to,
  lower_brks = NULL,
  unit = c("year", "day", "week", "month"),
  trans = FALSE,
  .transfn = lubridate::ymd,
  verbose = getOption("healthdb.verbose"),
  ...
)
```

## Arguments

|            |                                                                                                                                                                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| from       | A character or Date vector for start dates.                                                                                                                                                                                                                                                                                                                                 |
| to         | A character or Date vector for end dates.                                                                                                                                                                                                                                                                                                                                   |
| lower_brks | A numeric vector for lower breaks passing to the base <code>base::cut()</code> function to convert the numeric result to a factor. The level labels will be auto generated. For example, the level labels are <code>c("&lt;19", "19-24", "25-34", "35-44", "45-54", "55+")</code> for <code>lower_brks = c(0, 19, 25, 35, 45, 55)</code> . Default is NULL (no conversion). |
| unit       | A character string specifying the unit of the output. One of "year" (default), "day", "week", or "month".                                                                                                                                                                                                                                                                   |
| trans      | A logical for whether to transform both from and to with the <code>.transfn</code> function                                                                                                                                                                                                                                                                                 |
| .transfn   | A function for transforming the inputs. Default is <code>lubridate::ymd()</code> .                                                                                                                                                                                                                                                                                          |
| verbose    | A logical for whether to print a summary of the output and warn about missing values. Default is fetching from options. Use <code>options(healthdb.verbose = FALSE)</code> to suppress once and for all.                                                                                                                                                                    |
| ...        | Additional arguments passing to <code>base::cut()</code> .                                                                                                                                                                                                                                                                                                                  |

## Value

A numeric or factor vector of the duration.

**Examples**

```
# toy data
n <- 5
df <- data.frame(
  id = 1:n,
  start_dt = sample(seq(as.Date("1970-01-01"), as.Date("2000-12-31"), by = 1), size = n),
  end_dt = sample(seq(as.Date("2001-01-01"), as.Date("2023-12-31"), by = 1), size = n)
)

# get age group at a cut-off
df %>% dplyr::mutate(
  age_grp = compute_duration(start_dt, "2023-01-01", lower_brks = c(0, 19, 25, 35, 45, 55))
)

# compute gaps between two dates in weeks
df %>% dplyr::mutate(
  gap_wks = compute_duration(start_dt, end_dt, unit = "week")
)
```

cut\_period

*Cut the time period in one row into multiple rows by interval***Description**

This function is for cutting time periods into segments, which could be useful for subsequent overlap joins. Each original period (per row) will be expanded to multiple rows by weeks, months, etc. Only data.frame input is accepted as the output size is greater than the input. Thus, remote tables should be collected before running this function for optimal performance.

**Usage**

```
cut_period(
  data,
  start,
  end,
  len,
  unit = c("day", "week", "month", "quarter", "year"),
  .dt_trans = NULL
)
```

**Arguments**

|       |                                                                           |
|-------|---------------------------------------------------------------------------|
| data  | Input data.frame that each row has start and end dates                    |
| start | Record start date column (unquoted)                                       |
| end   | Record end date column (unquoted)                                         |
| len   | An integer, the interval that would be used to divide the record duration |

|           |                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------|
| unit      | One of "day" (default), "week", "month", "quarter", or "year" used in combination of len to specify the time length of the interval. |
| .dt_trans | Function to transform start/end, such as <code>lubridate::ymd()</code> . Default is NULL.                                            |

### Value

Data frame that each row is now a segment of the period defined by `c(start, end)` in the original row. Original variables are retained and repeated for each segment plus new variables defining the segment interval.

### Examples

```
# toy data
df <- data.frame(sample_id = 1, period_id = 1, start_date = "2015-01-01", end_date = "2019-12-31")

# divide period into segments (multiple rows per period)
df_seg <- cut_period(
  data = df, start = start_date, end = end_date,
  len = 1,
  unit = "year",
  .dt_trans = lubridate::ymd
)

# categorize segment_id as factor
df_seg$segment <- cut(df_seg$segment_id,
  breaks = c(0, 1, 2, Inf),
  labels = c("< 1 year", "1 - 2 years", "Remainder")
)

head(df_seg)
```

---

define\_case

*Identify diseases/events from administrative records*

---

### Description

This function is a composite of `identify_row()`, `exclude()`, `restrict_n()`, and `restrict_date()`. It aims to implement common case definitions in epidemiological studies using administrative database as a one-shot big query. The intended use case is for definitions in the form of, e.g., two or more physician visits with some diagnostic code at least 30 days apart within two years. The component functions mentioned above are chained in the following order if all arguments were supplied: `identify_row(vals) %>% exclude(identify_row(excl_vals), by = cInt_id) %>% restrict_n() %>% restrict_date()`. Only the necessary steps in the chain will be run if some arguments are missing, see the verbose output for what was done. Note that if `date_var` is supplied, `n_per_cInt` will be counted by distinct dates instead of number of records.

If your case definition also restricts records to an age range (e.g., only count visits made when the client was aged 18-65), use `define_case_with_age()`, which extends this function with `birth_date/age` and `age_range` arguments.

**Usage**

```
define_case(
  data,
  vars,
  match = "in",
  vals,
  clnt_id,
  n_per_clnt = 1,
  date_var = NULL,
  apart = NULL,
  within = NULL,
  uid = NULL,
  excl_vals = NULL,
  excl_args = NULL,
  keep = c("all", "first", "last"),
  if_all = FALSE,
  ignore_case = TRUE,
  mode = c("flag", "filter"),
  force_collect = FALSE,
  verbose = getOption("healthdb.verbose"),
  ...
)
```

**Arguments**

|            |                                                                                                                                                                           |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data       | Data.frames or remote tables (e.g., from <code>dbplyr::tbl_sql()</code> )                                                                                                 |
| vars       | An expression passing to <code>dplyr::select()</code> . It can be quoted/unquoted column names, or helper functions, such as <code>dplyr::starts_with()</code> .          |
| match      | One of "in", "start", "regex", "like", "between", and "glue_sql". It determines how values would be matched. See <code>identify_row()</code> for detail.                  |
| vals       | Depending on match, it takes different input. See <code>identify_row()</code> .                                                                                           |
| clnt_id    | Grouping variable (quoted/unquoted).                                                                                                                                      |
| n_per_clnt | A single number specifying the minimum group size.                                                                                                                        |
| date_var   | Variable name (quoted/unquoted) for the dates to be interpreted.                                                                                                          |
| apart      | An integer specifying the minimum gap (in days) between adjacent dates in a draw.                                                                                         |
| within     | An integer specifying the maximum time span (in days) of a draw.                                                                                                          |
| uid        | Variable name for a unique row identifier. It is necessary for SQL to produce consistent result based on sorting.                                                         |
| excl_vals  | Same as vals but clients/groups with these values are going to be removed from the result. This is intended for exclusion criteria of a case definition.                  |
| excl_args  | A named list of arguments passing to the second <code>identify_row()</code> call for excl_vals. If not supplied, var, match and if_all of the first call will be re-used. |
| keep       | One of:                                                                                                                                                                   |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"> <li>• "first" (keeping each client's earliest record),</li> <li>• "last" (keeping the latest),</li> <li>• and "all" (keeping all relevant records, default).</li> <li>• Note that "first"/"last" should not be used with "flag" mode.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                  |
| if_all        | A logical for whether combining the predicates (if multiple columns were selected by vars) with AND instead of OR. Default is FALSE, e.g., var1 in vals OR var2 in vals.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| ignore_case   | A logical for whether match = "like" and "start" should ignore case. Default is TRUE, because codes in administrative data are often inconsistently cased, and a case-sensitive match would silently miss records. Both the values and the patterns are lower-cased before comparison, so the result does not depend on the backend: without it, matching would be case-sensitive for data.frames, case-sensitive or not on remote tables depending on the database, the database's collation, and the 'dbplyr' version. Set to FALSE for a case-sensitive match, which on a large table can also be much faster, as wrapping the column in LOWER() prevents the database from using an index on it. |
| mode          | <p>Either:</p> <ul style="list-style-type: none"> <li>• "flag" - add new columns starting with "flag_" indicating if the client met the condition,</li> <li>• or "filter" - remove clients that did not meet the condition from the data.</li> <li>• This will be passed to both <a href="#">restrict_n()</a> AND <a href="#">restrict_date()</a>. Default is "flag".</li> </ul>                                                                                                                                                                                                                                                                                                                     |
| force_collect | A logical for whether force downloading the result table if it is not a local data.frame. Downloading data could be slow, so the user has to opt in; default is FALSE.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| verbose       | A logical for whether to print an explanation of the operation. Default is fetching from options. Use options(healthdb.verbose = FALSE) to suppress once and for all.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| ...           | Additional arguments, e.g., flag_at, passing to <a href="#">restrict_date()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Value**

A subset of the input data that satisfied the specified case definition.

**See Also**

[define\\_case\\_with\\_age\(\)](#) for the same workflow with an added age restriction.

**Examples**

```
sample_size <- 30
df <- data.frame(
  clnt_id = rep(1:3, each = 10),
  service_dt = sample(seq(as.Date("2020-01-01"), as.Date("2020-01-31"), by = 1),
    size = sample_size, replace = TRUE
),
  diagx = sample(letters, size = sample_size, replace = TRUE),
```

```

    diagx_1 = sample(c(NA, letters), size = sample_size, replace = TRUE),
    diagx_2 = sample(c(NA, letters), size = sample_size, replace = TRUE)
  )

  # define from one source
  define_case(df,
    vars = starts_with("diagx"), "in", vals = letters[1:4],
    clnt_id = clnt_id, date_var = service_dt,
    excl_args = list(if_all = TRUE),
    # remove non-case
    mode = "filter",
    # keeping the first record
    keep = "first"
  )

  # multiple sources with purrr::pmap
  # arguments with length = 1 will be recycle to match the number of sources
  # wrap expressions/unquoted variables with bquote(),
  # or rlang::exprs() to prevent immediate evaluation,
  # or just use quoted variable names
  purrr::pmap(
    list(
      data = list(df, df),
      vars = rlang::exprs(starts_with("diagx")),
      match = c("in", "start"),
      vals = list(letters[1:4], letters[5:10]),
      clnt_id = list(bquote(clnt_id)), n_per_clnt = c(2, 3),
      date_var = "service_dt",
      excl_vals = list(letters[11:13], letters[14:16]),
      excl_args = list(list(if_all = TRUE), list(if_all = FALSE))
    ),
    define_case
  )

```

---

|                      |                                                                                  |
|----------------------|----------------------------------------------------------------------------------|
| define_case_with_age | <i>Identify diseases/events from administrative records with age restriction</i> |
|----------------------|----------------------------------------------------------------------------------|

---

## Description

This function extends the standard case definition function `define_case()` by allowing age-based filtering. See `define_case()` for more general description of what this function does.

Note that when using this function with an existing age variable, the age should be determined at the time of the record. Records that are not in the eligible age range are removed before counting records/dates per client and before interpreting the temporal relationship between records. In other words, the age restriction is applied before both `restrict_n()` and `restrict_date()`, so that `n_per_clnt`, `apart`, and `within` are evaluated only on age-eligible records (e.g., "two or more visits while aged 18-65").

The age restriction is an eligibility filter on the population, not a per-record case flag, so age-ineligible records are always removed regardless of mode. This differs from the `n_per_clnt/apart/within` criteria, which only add `flag_` columns (and keep all rows) when `mode = "flag"`. As a result, the output of `mode = "flag"` may still have fewer rows than the input.

For other age restrictions based on a fixed time point (e.g., age at the baseline of follow-up), it can be done by filtering the input data or output of `define_case()` instead of using this function.

## Usage

```
define_case_with_age(
  data,
  vars,
  match = "in",
  vals,
  clnt_id,
  n_per_clnt = 1,
  date_var = NULL,
  apart = NULL,
  within = NULL,
  uid = NULL,
  excl_vals = NULL,
  excl_args = NULL,
  keep = c("all", "first", "last"),
  if_all = FALSE,
  ignore_case = TRUE,
  mode = c("flag", "filter"),
  birth_date = NULL,
  age = NULL,
  age_range = NULL,
  force_collect = FALSE,
  verbose = getOption("healthdb.verbose"),
  ...
)
```

## Arguments

|                         |                                                                                                                                                                  |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>       | Data.frames or remote tables (e.g., from <code>dbplyr::tbl_sql()</code> )                                                                                        |
| <code>vars</code>       | An expression passing to <code>dplyr::select()</code> . It can be quoted/unquoted column names, or helper functions, such as <code>dplyr::starts_with()</code> . |
| <code>match</code>      | One of "in", "start", "regex", "like", "between", and "glue_sql". It determines how values would be matched. See <code>identify_row()</code> for detail.         |
| <code>vals</code>       | Depending on <code>match</code> , it takes different input. See <code>identify_row()</code> .                                                                    |
| <code>clnt_id</code>    | Grouping variable (quoted/unquoted).                                                                                                                             |
| <code>n_per_clnt</code> | A single number specifying the minimum group size.                                                                                                               |
| <code>date_var</code>   | Variable name (quoted/unquoted) for the dates to be interpreted.                                                                                                 |
| <code>apart</code>      | An integer specifying the minimum gap (in days) between adjacent dates in a draw.                                                                                |

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| within      | An integer specifying the maximum time span (in days) of a draw.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| uid         | Variable name for a unique row identifier. It is necessary for SQL to produce consistent result based on sorting.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| excl_vals   | Same as vals but clients/groups with these values are going to be removed from the result. This is intended for exclusion criteria of a case definition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| excl_args   | A named list of arguments passing to the second <code>identify_row()</code> call for <code>excl_vals</code> . If not supplied, <code>var</code> , <code>match</code> and <code>if_all</code> of the first call will be re-used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| keep        | One of: <ul style="list-style-type: none"> <li>• "first" (keeping each client's earliest record),</li> <li>• "last" (keeping the latest),</li> <li>• and "all" (keeping all relevant records, default).</li> <li>• Note that "first"/"last" should not be used with "flag" mode.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| if_all      | A logical for whether combining the predicates (if multiple columns were selected by vars) with AND instead of OR. Default is FALSE, e.g., <code>var1</code> in vals OR <code>var2</code> in vals.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ignore_case | A logical for whether <code>match = "like"</code> and <code>"start"</code> should ignore case. Default is TRUE, because codes in administrative data are often inconsistently cased, and a case-sensitive match would silently miss records. Both the values and the patterns are lower-cased before comparison, so the result does not depend on the backend: without it, matching would be case-sensitive for <code>data.frames</code> , case-sensitive or not on remote tables depending on the database, the database's collation, and the 'dbplyr' version. Set to FALSE for a case-sensitive match, which on a large table can also be much faster, as wrapping the column in <code>LOWER()</code> prevents the database from using an index on it. |
| mode        | Either: <ul style="list-style-type: none"> <li>• "flag" - add new columns starting with "flag_" indicating if the client met the condition,</li> <li>• or "filter" - remove clients that did not meet the condition from the data.</li> <li>• This will be passed to both <code>restrict_n()</code> AND <code>restrict_date()</code>. Default is "flag".</li> <li>• Note that the age restriction (<code>age_range</code>) always removes age-ineligible records regardless of mode; see Description.</li> </ul>                                                                                                                                                                                                                                          |
| birth_date  | Optional. The name of the column containing birth dates. Used to calculate age when <code>age_range</code> is specified. Requires <code>date_var</code> to be supplied. Age will be calculated as $(\text{date\_var} - \text{birth\_date})/365.25$ . The age is signed, so a record dated before its birth date (e.g., a data error) yields a negative age and is treated as outside <code>age_range</code> and removed.                                                                                                                                                                                                                                                                                                                                  |
| age         | Optional. The name of the column containing age values. Used directly for age filtering when <code>age_range</code> is specified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| age_range   | Optional. A length 2 numeric vector <code>c(min, max)</code> specifying the age range in years (inclusive on both ends). Use NA for one-sided bounds (e.g., <code>c(10, NA)</code> for age $\geq 10$ , or <code>c(NA, 65)</code> for age $\leq 65$ ). At least one non-NA value must be provided. Records outside this range are always removed regardless of mode.                                                                                                                                                                                                                                                                                                                                                                                       |

|               |                                                                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| force_collect | A logical for whether force downloading the result table if it is not a local data.frame. Downloading data could be slow, so the user has to opt in; default is FALSE. |
| verbose       | A logical for whether to print an explanation of the operation. Default is fetching from options. Use options(healthdb.verbose = FALSE) to suppress once and for all.  |
| ...           | Additional arguments, e.g., mode, passing to <a href="#">restrict_date()</a> .                                                                                         |

### Value

A subset of the input data that satisfied the specified case definition.

### Examples

```
sample_size <- 30
df <- data.frame(
  clnt_id = rep(1:3, each = 10),
  service_dt = sample(seq(as.Date("2020-01-01"), as.Date("2020-01-31"), by = 1),
    size = sample_size, replace = TRUE
  ),
  diagx = sample(letters, size = sample_size, replace = TRUE),
  diagx_1 = sample(c(NA, letters), size = sample_size, replace = TRUE),
  diagx_2 = sample(c(NA, letters), size = sample_size, replace = TRUE)
)

# define from one source
define_case_with_age(df,
  vars = starts_with("diagx"), "in", vals = letters[1:4],
  clnt_id = clnt_id, date_var = service_dt,
  excl_args = list(if_all = TRUE),
  # remove non-case
  mode = "filter",
  # keeping the first record
  keep = "first"
)

# age restriction using birth_date
# give each client a different birth date so age is calculated
# at the time of each record (date_var - birth_date)
df_with_birth <- df
birth_dts <- as.Date(c("2008-05-01", "1985-08-20", "1950-03-15"))
df_with_birth$birth_dt <- birth_dts[df_with_birth$clnt_id]

# keep only records made when the client was aged 18-65;
# the age restriction drops client 1 (~12) and client 3 (~70)
define_case_with_age(df_with_birth,
  vars = starts_with("diagx"), "in", vals = letters[1:4],
  clnt_id = clnt_id, date_var = service_dt,
  birth_date = birth_dt, age_range = c(18, 65),
  mode = "filter"
)
```

```

# one-sided bound: keep records made at age >= 18 only;
# the age restriction drops only client 1 (~12)
define_case_with_age(df_with_birth,
  vars = starts_with("diagx"), "in", vals = letters[1:4],
  clnt_id = clnt_id, date_var = service_dt,
  birth_date = birth_dt, age_range = c(18, NA),
  mode = "filter"
)

# age restriction using a pre-computed age column
# (age recorded at the time of the service, no birth date needed)
df_with_age <- df
df_with_age$age <- c(12, 35, 70)[df_with_age$clnt_id]
define_case_with_age(df_with_age,
  vars = starts_with("diagx"), "in", vals = letters[1:4],
  clnt_id = clnt_id, date_var = service_dt,
  age = age, age_range = c(18, 65),
  mode = "filter"
)

# multiple sources with purrr::pmap
# arguments with length = 1 will be recycle to match the number of sources
# wrap expressions/unquoted variables with bquote(),
# or rlang::exprs() to prevent immediate evaluation,
# or just use quoted variable names
purrr::pmap(
  list(
    data = list(df, df),
    vars = rlang::exprs(starts_with("diagx")),
    match = c("in", "start"),
    vals = list(letters[1:4], letters[5:10]),
    clnt_id = list(bquote(clnt_id)), n_per_clnt = c(2, 3),
    date_var = "service_dt",
    excl_vals = list(letters[11:13], letters[14:16]),
    excl_args = list(list(if_all = TRUE), list(if_all = FALSE))
  ),
  define_case_with_age
)

```

---

elix\_codes

---

*ICD codes defining the Elixhauser comorbidity categories*


---

## Description

The International Classification of Diseases (ICD) diagnostic codes that define the 31 Elixhauser comorbidity categories, as used by `compute_comorbidity()`. The "ICD-10" and "ICD-9-CM-5digits" versions are from Quan et al. (2005). The "ICD-9-CM-3digits" version is adopted from the Manitoba Centre for Health Policy, which searches BOTH 3-digit and 5-digit codes (see <http://mchp-appserv.cpe.umanitoba.ca/viewConcept.php?printer=Y&conceptID=1436#CAUTIONS> for cautions and limitations).

**Usage**

```
elix_codes
```

**Format**

A data frame with 893 rows and 5 columns:

**icd\_ver** ICD code version; one of "ICD-10", "ICD-9-CM-5digits", and "ICD-9-CM-3digits", corresponding to the `icd_ver` argument of `compute_comorbidity()`.

**category** Short label of the comorbidity category, e.g., "chf". These labels are the column names in `compute_comorbidity()` output and the valid values of its `excl` argument.

**description** Full name of the comorbidity category, e.g., "Congestive Heart Failure".

**code** The ICD code, e.g., "I50". Codes contain no dots.

**match\_len** The number of leading characters of a diagnosis value that is compared with the code (prefix match), equal to the number of characters in the code. Codes in the sources are listed at the category/subcategory level and cover all their subdivisions (the ".x" notation in Quan et al.), hence the prefix match, e.g., ICD-10 "I50" covers "I500", and ICD-9 "428" covers "4280" and "42800". This reproduces the behavior of the reference SAS implementation, which compares every code with the SAS IN: (starts-with) operator.

**Source**

The code lists were verified code-for-code against the SAS macros published by the Manitoba Centre for Health Policy (based on Quan's "Enhanced Elixhauser Diagnosis-Type SAS code"):

- ICD-9-CM: [http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/\\_ElixhauserICD9CM.sas.txt](http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/_ElixhauserICD9CM.sas.txt)
- ICD-10: [http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/\\_ElixhauserICD10.sas.txt](http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/_ElixhauserICD10.sas.txt)

**References**

Quan H, Sundararajan V, Halfon P, Fong A, Burnand B, Luthi JC, Saunders LD, Beck CA, Feasby TE, Ghali WA. Coding algorithms for defining comorbidities in ICD-9-CM and ICD-10 administrative data. *Med Care* 2005;43(11):1130-1139.

**Examples**

```
head(elix_codes)

# codes defining Congestive Heart Failure in ICD-10
subset(elix_codes, icd_ver == "ICD-10" & category == "chf")
```

---

|         |                                                            |
|---------|------------------------------------------------------------|
| exclude | <i>Remove rows based on conditions or another data set</i> |
|---------|------------------------------------------------------------|

---

### Description

This function combines `dplyr::anti_join()`, and negation of `dplyr::filter()`. When a second data set is supplied through the `excl` argument, anti join would be performed; otherwise, data would be filtered with the expression given via the `condition` argument, and the filter result would in turn be removed using `dplyr::setdiff()`.

### Usage

```
exclude(
  data,
  excl = NULL,
  by = NULL,
  condition = NULL,
  verbose = getOption("healthdb.verbose"),
  report_on = NULL,
  ...
)
```

### Arguments

|                        |                                                                                                                                                                                                                                                                                |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>      | Data.frames or remote tables (e.g., from <code>dbplyr::tbl_sql()</code> ). A subset will be removed from this data.                                                                                                                                                            |
| <code>excl</code>      | Data frames or remote tables (e.g., from 'dbplyr'). Rows/values present in it will be removed from data if there is a match. This will be passed to <code>dplyr::anti_join()</code> as the second argument.                                                                    |
| <code>by</code>        | Column names that should be matched by <code>dplyr::anti_join()</code> , or an expression made with <code>dplyr::join_by()</code> . See <code>dplyr::anti_join()</code> 's <code>by</code> argument for detail. Default NULL is the same as <code>setdiff(data, excl)</code> . |
| <code>condition</code> | An expression that will be passed to <code>dplyr::filter()</code> . The rows that satisfy condition are those to be removed from data.                                                                                                                                         |
| <code>verbose</code>   | A logical for whether to print an explanation of the operation. Default is fetching from options. Use <code>options(healthdb.verbose = FALSE)</code> to suppress once and for all.                                                                                             |
| <code>report_on</code> | A quoted/unquoted column name for counting how many of its distinct values were removed from data, e.g., counting how many client IDs were removed. Default is NULL.                                                                                                           |
| <code>...</code>       | Additional arguments passing to <code>dplyr::filter()/dplyr::anti_join()</code> for finer control of matching, e.g., <code>na action</code> , <code>by-group</code> filtering, etc.                                                                                            |

### Value

A data frame or remote table that is a subset of data.

## Examples

```
# exclude with condition
cyl_not_4 <- exclude(mtcars, condition = cyl == 4, report_on = cyl)

# exclude with another data
exclude(mtcars, cyl_not_4, dplyr::join_by(cyl), report_on = cyl)
```

---

|             |                                               |
|-------------|-----------------------------------------------|
| execute_def | <i>Execute parameterized case definitions</i> |
|-------------|-----------------------------------------------|

---

## Description

This function executes the function calls stored in the output tibble from [build\_def()] with data objects supplied through a named list and returns the results as a list. It is intended to facilitate re-use of pre-defined calls with different data.

## Usage

```
execute_def(
  def,
  with_data,
  bind = FALSE,
  force_proceed = getOption("healthdb.force_proceed")
)
```

## Arguments

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| def           | A tibble created by [build_def()].                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| with_data     | A named list which the elements are in the form of src_lab = data, where 'src_lab' corresponds to the src_labs argument from [build_def()] and 'data' is the data object that will be passed to calls stored in def. The names (and length) of 'with_data' must match the unique values of src_labs in 'def'.                                                                                                                                                                            |
| bind          | A logical for whether row-binding records from multiple sources into one table. Note that the binding may fail in ways that are difficult to anticipate in advance, such as data type conflict (e.g., Date vs. character) between variables in the same name from different sources. The default is FALSE. If TRUE, the behavior is to try and return the unbinded result when failed.                                                                                                   |
| force_proceed | A logical for whether to ask for user input in order to proceed when remote tables are needed to be collected for binding. The default is FALSE to let user be aware of that the downloading process may be slow. Use options(healthdb.force_proceed = TRUE) to suppress the prompt once and for all. In non-interactive sessions (e.g., scripts run via Rscript, knitr), the confirmation prompt cannot be displayed, and the function stops with an error unless force_proceed = TRUE. |

## Value

A single (if bind = TRUE) or a list of data.frames or remote tables.

**See Also**

[bind\_source()] for binding the output with convenient renaming features.

**Examples**

```
# toy data
sample_size <- 30
df <- data.frame(
  clnt_id = rep(1:3, each = 10),
  service_dt = sample(seq(as.Date("2020-01-01"), as.Date("2020-01-31"), by = 1),
    size = sample_size, replace = TRUE
),
  diagx = sample(letters, size = sample_size, replace = TRUE),
  diagx_1 = sample(c(NA, letters), size = sample_size, replace = TRUE),
  diagx_2 = sample(c(NA, letters), size = sample_size, replace = TRUE)
)

# make df a database table
con <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
db <- dplyr::copy_to(con, df, "df")

# use build_def to make a toy definition
sud_def <- build_def("SUD", # usually a disease name
  src_lab = c("src1", "src2"), # identify from multiple sources, e.g., hospitalization, ED visits.
  # functions that filter the data with some criteria
  def_fn = define_case,
  fn_args = list(
    vars = starts_with("diagx"),
    match = "start", # "start" will be applied to all sources as length = 1
    vals = list(c("304"), c("305")),
    clnt_id = "clnt_id", # list()/c() could be omitted for single element
    # c() can be used in place of list
    # if this argument only takes one value for each source
    n_per_clnt = c(2, 3)
  )
)

# save the definition for re-use
# saveRDS(sud_def, file = some_path)

sud_def %>% execute_def(with_data = list(src1 = df, src2 = db), force_proceed = TRUE)
```

---

 fetch\_var

---

*Get variables from multiple tables with common ID columns*


---

**Description**

This function fetches variables from different tables that are linked by common IDs. It calls `dplyr::left_join()` multiple times with various source tables (y argument of the join) to gather

variables. It is not meant to replace `left_join()` but simplify syntax for the situation that you started off with a table of your study sample and wanted to gather covariates from different sources linked by common client IDs, which is often the case when working with healthcare databases. **Caution:** this function is intended for one-to-one joins only because it could be problematic when we do not know which source caused a one-to-many join and changed the number of rows. For `data.frame` input, an error will be given when one-to-many joins were detected. However, such checking could be an expensive operation on remote source. Therefore, for database input, the result will not be checked.

The names of the fetched variables must not clash with each other, with columns already in `data`, or with any key not used in their linkage (see the `|` syntax below); an error will be given in these cases, as the clashing columns would otherwise be silently renamed (e.g., `var.x/var.y`) with results that no longer mean what their names suggest. Also note that `|` is reserved for separating variables and keys in the linkage formulas and thus cannot be used within the 'tidyselect' expressions, e.g., use `c(starts_with("a"), ends_with("b"))` instead of `starts_with("a") | ends_with("b")`.

### Usage

```
fetch_var(data, keys, linkage, ...)
```

### Arguments

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>    | A <code>data.frame</code> or remote table ( <code>tbl_sql</code> ) which must be an object and not from a pipe. It would be used as the <code>x</code> argument in <code>left_join()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>keys</code>    | A vector of quoted/unquoted variable names, or 'tidyselect' expression (see <a href="#">dplyr::select()</a> ). These variables must be present in <code>data</code> and would be used as the <code>by</code> argument in <code>left_join()</code> . The <code>y</code> tables must have a subset of these if not all.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>linkage</code> | <p>A list of formulas in the form of "from_tab ~ get_vars by_keys":</p> <ul style="list-style-type: none"> <li>• source table on the left-hand-side</li> <li>• variables on the right-hand-side</li> <li>• If a source table does not have all the variables in <code>keys</code>, use <code>"I"</code> on RHS to specify the subset of keys to be used.</li> </ul> <p>For example, given <code>keys</code> has 3 variables, <code>list(y1 ~ tidyselect_expr1, y2 ~ tidyselect_expr2 key1 + key2)</code> meaning:</p> <ol style="list-style-type: none"> <li>1. from table <code>y1</code> get variables picked by the tidyselect expression matching on all 3 keys;</li> <li>2. from table <code>y2</code> get variables matching on only <code>key1</code> and <code>key2</code>.</li> </ol> |
| <code>...</code>     | Additional arguments, e.g., <code>copy = TRUE</code> , passing to <code>left_join()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

### Value

A `data.frame` or remote table containing all original columns of `x` and new variables matched from other tables based on the specified linkage.

## Examples

```
# make toy data
size <- 30
n <- 10
df1 <- data.frame(
  id = sample(1:n, size = size, replace = TRUE),
  service_dt = sample(seq(as.Date("2020-01-01"), as.Date("2022-01-31"), by = 1),
    size = size
  )
) %>%
  dplyr::mutate(year = lubridate::year(service_dt))
df2 <- data.frame(
  id = rep(1:n, size / n), year = rep(2020:2022, each = n),
  status_1 = sample(0:1, size = size, replace = TRUE),
  status_2 = sample(0:1, size = size, replace = TRUE)
)
df3 <- data.frame(id = 1:n, sex = sample(c("F", "M"), size = n, replace = TRUE))

# simple joins
# note that for left_join(df1, df2), boths keys have to be used,
# otherwise, error as the relation would not be one-to-one
fetch_var(df1,
  keys = c(id, year),
  linkage = list(
    df2 ~ starts_with("s"), # match both keys without '|'
    df3 ~ sex | id
  ) # match by id only; otherwise failed because df3 has no year
)

# example if some y is remote
# make df2 as database table
con <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
db2 <- dplyr::copy_to(con, df2, "df2")

fetch_var(df1,
  keys = c(id, year),
  linkage = list(
    db2 ~ starts_with("s"),
    df3 ~ sex | id
  ),
  copy = TRUE # pass to left_join for forced collection of remote table
)
```

---

identify\_rows

*Identify rows with a match*


---

## Description

Filter rows whose values satisfy the specified conditions. The functionality is identical to `dplyr::filter()` combined with `dplyr::if_any()` or `dplyr::if_all()`, but it uses the 'data.table' package (`vignette("datatable-intro")`).

package = "data.table")) for the `data.frame` method, and it supports regular expressions for remote database tables. The motivation is to take away some pain when working with databases, which often do not support regular expressions or the 'LIKE' operator with multiple string patterns.

## Usage

```
identify_rows(
  data,
  vars,
  match = c("in", "start", "regex", "like", "between", "glue_sql"),
  vals,
  if_all = FALSE,
  ignore_case = TRUE,
  verbose = getOption("healthdb.verbose"),
  query_only = TRUE,
  ...
)

identify_row(
  data,
  vars,
  match = c("in", "start", "regex", "like", "between", "glue_sql"),
  vals,
  if_all = FALSE,
  ignore_case = TRUE,
  verbose = getOption("healthdb.verbose"),
  query_only = TRUE,
  ...
)
```

## Arguments

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>  | Data.frames or remote tables (e.g., from <code>dbplyr::tbl_sql()</code> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>vars</code>  | An expression passing to <code>dplyr::select()</code> . It can be quoted/unquoted column names, or helper functions, such as <code>dplyr::starts_with()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>match</code> | One of "in", "start", "regex", "like", "between", and "glue_sql". It determines how values would be matched. The operations under each type: <ul style="list-style-type: none"> <li>• "in": <code>var %in% vals</code> (This is default)</li> <li>• "regex": <code>stringr::str_detect(var, vals)</code>. For remote tables, unique values in vars are collected locally before matching (may be slow).</li> <li>• "like": <code>stringr::str_like(var, vals)</code>. For remote tables, <code>WHERE var LIKE val</code>.</li> <li>• "start": same as regex or LIKE with modified vals, e.g., <code>"^val1 ^val2"</code> or <code>"val1% val2%"</code></li> <li>• "between": <code>dplyr::between(var, val1, val2)</code></li> <li>• "glue_sql": For remote table only, this gives full control of the WHERE clause using <code>dplyr::filter(dbplyr::sql(glue::glue_sql(...)))</code></li> </ul> |

Matching by "like"/"start" ignores case by default; see `ignore_case`. The other match types are unaffected: "in" and "between" compare values as they are, and "regex" follows the pattern you supply (use "(?i)" to make it case-insensitive).

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| vals        | <p>Depending on match, it takes different input:</p> <ul style="list-style-type: none"> <li>• "in": a vector of values (numeric/character/Date)</li> <li>• "start": a vector of numeric/character that would be modified into a regex or LIKE pattern string by adding "^" in front or "%" at the end</li> <li>• "regex"/"like": a string of the expression</li> <li>• "between": a vector of numeric or date with exactly two elements, e.g., <code>c(lower, upper)</code></li> <li>• "glue_sql": a string of a SQL WHERE clause, which will be passed to <code>glue::glue_sql()</code>. See examples for detail.</li> </ul>                                                                                                                             |
| if_all      | A logical for whether combining the predicates (if multiple columns were selected by vars) with AND instead of OR. Default is FALSE, e.g., <code>var1</code> in vals OR <code>var2</code> in vals.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ignore_case | A logical for whether <code>match = "like"</code> and <code>"start"</code> should ignore case. Default is TRUE, because codes in administrative data are often inconsistently cased, and a case-sensitive match would silently miss records. Both the values and the patterns are lower-cased before comparison, so the result does not depend on the backend: without it, matching would be case-sensitive for <code>data.frames</code> , case-sensitive or not on remote tables depending on the database, the database's collation, and the 'dbplyr' version. Set to FALSE for a case-sensitive match, which on a large table can also be much faster, as wrapping the column in <code>LOWER()</code> prevents the database from using an index on it. |
| verbose     | A logical for whether to print an explanation of the query and an overview of the result. Default is fetching from options. Use <code>options(healthdb.verbose = FALSE)</code> to suppress once and for all. Result overview is not for remote tables as the query is not executed immediately, thus no result is available for summary without adding an extra run (may be slow) of the query.                                                                                                                                                                                                                                                                                                                                                           |
| query_only  | A logical for whether keeping the output as remote table (Default TRUE) or downloading the query result as a tibble (FALSE). The argument is ignored when the input data is a <code>data.frame/tibble</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| ...         | For remote table method only. Additional arguments passing to <code>glue::glue_sql()</code> for parameterized queries.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

### Value

A `data.frame` or `tbl_sql` object depending on the input.

### Examples

```
# applying to data.frame; both sepal length and width in range 3-5
identify_rows(iris, starts_with("Sepal"), "between", c(3, 5), if_all = TRUE)

# applying to remote table; species starts with se or ends with ca
iris_db <- dbplyr::memdb_frame(iris)
identify_rows(iris_db, Species, "like", c("se%", "%ca"))
```

```

# using glue_sql to write the WHERE clause
# use `{vars}` to refer to the variables selected by vars
# supply additional values required in the query through '...'
# note that if you use LIKE here, you cannot supply multiple patterns in what
identify_rows(iris_db, Species, "glue_sql",
  "{vars} LIKE {what}",
  what = "se%"
)

# add * after a vector
identify_rows(iris_db, Species, "glue_sql",
  "{vars} IN ({what*})",
  what = c("setosa", "virginica")
)

```

---

if\_date

*Interpret if any n elements drawn from a date vector could be some days apart within some time span*

---

## Description

Given a vector of dates x, interpret if there could be at least one set of n elements taken from x satisfy that adjacent elements in the set are at least certain days apart AND the dates in the set are within the specified time span. When identifying events/diseases from administrative data, definitions often require, e.g., n diagnoses that are at least some days apart within some years. This function is intended for such use and optimized to avoid looping through all n-size combinations in x. See vignette("if\_date\_logic") for an explanation of the algorithm. This function does not work with remote table input.

## Usage

```

if_date(
  x,
  n,
  apart = NULL,
  within = NULL,
  detail = FALSE,
  align = c("left", "right"),
  dup.rm = TRUE,
  ...
)

```

## Arguments

|   |                                   |
|---|-----------------------------------|
| x | A character or Date vector        |
| n | An integer for the size of a draw |

|        |                                                                                                                                                                                                                                                                                                                                         |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| apart  | An integer specifying the minimum gap (in days) between adjacent dates in a draw.                                                                                                                                                                                                                                                       |
| within | An integer specifying the maximum time span (in days) of a draw.                                                                                                                                                                                                                                                                        |
| detail | Logical for whether return result per element of x. The default is FALSE, which returns one logical summarized by any(). Detail is not available if apart was supplied without within because sets that satisfied the condition could overlap, and records within a set may be far apart; thus, no unambiguous way to label by element. |
| align  | Character, define if the time span for each record should start ("left") or end ("right") at its current date. Defaults to "left". See 'flag_at' argument in <a href="#">restrict_date()</a> for detail.                                                                                                                                |
| dup.rm | Logical for whether multiple records on the same date should be counted as one in the calculation. Only applicable when within is supplied without apart; duplicated dates have no impact when apart is present as the n dates must be distinct if they were apart. Default is TRUE.                                                    |
| ...    | Additional argument passing to <a href="#">data.table::as.IDate()</a> for date conversion.                                                                                                                                                                                                                                              |

**Value**

Single or a vector of logical for whether there is any draw from x satisfied the conditions

**See Also**

[restrict\\_date\(\)](#)

**Examples**

```

dates_of_records <- sample(seq(as.Date("2015-01-01"), as.Date("2021-12-31"), 7), 10)

# whether there is any 3 records at least 30 days apart within 2 years
if_date(dates_of_records, n = 3, apart = 30, within = 365 * 2)

# specified either apart or within or both
if_date(dates_of_records, n = 2, within = 365)

```

---

lookup

---

*Find value corresponding to the input vector using a look-up table*


---

**Description**

Find value corresponding to the input vector using a look-up table

**Usage**

```
lookup(x, link, lu, verbose = getOption("healthdb.verbose"))
```

**Arguments**

|         |                                                                                                                                        |
|---------|----------------------------------------------------------------------------------------------------------------------------------------|
| x       | A variable name in a data.frame; this function should be called inside dplyr::mutate().                                                |
| link    | A formula in the form: name_of_x_in_lu ~ name_of_target_value. The left-hand-side can be omitted if x's name is also x in the look-up. |
| lu      | Look-up table in data.frame class.                                                                                                     |
| verbose | A logical for whether warn for missing values in the output.                                                                           |

**Value**

A vector of matched values.

**Examples**

```
df <- data.frame(drug_code = 1:10)
lu <- data.frame(drug_id = 1:20, drug_code = as.character(1:10), drug_name = sample(letters, 20))

df %>% dplyr::mutate(
  drug_nm = lookup(drug_code, drug_id ~ drug_name, lu),
  # this will work as lu also has drug_code column
  drug_nm = lookup(drug_code, ~drug_name, lu)
)
```

---

make\_test\_dat

---

*Make test data*


---

**Description**

Make a toy data set for testing and demo. This is for internal use purpose and not intended to be called by users.

**Usage**

```
make_test_dat(
  vals_kept = c("304", "305", 3040:3049, 3050:3059),
  noise_val = "999",
  IDs = 1:50,
  date_range = seq(as.Date("2015-01-01"), as.Date("2020-12-31"), by = 1),
  nrows = 100,
  n_any = 50,
  n_all = 10,
  seed = NULL,
  answer_id = NULL,
  type = c("data.frame", "database")
)
```

**Arguments**

|            |                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------|
| vals_kept  | A vector of values that suppose to be identified.                                                     |
| noise_val  | A vector of values that are not meant to be identified.                                               |
| IDs        | A vector of client IDs.                                                                               |
| date_range | A vector of all possible dates in the data.                                                           |
| nrows      | Number of rows of the output.                                                                         |
| n_any      | Number of rows to be identified if the criteria is that if any target column contains certain values. |
| n_all      | Number of rows to be identified if the criteria is that if all target columns contain certain values. |
| seed       | Seed for random number generation.                                                                    |
| answer_id  | Column name for the indicator of how the row should be identified: any, all, and noise.               |
| type       | Output type, "data.frame" or "database".                                                              |

**Value**

A data.frame or remote table from 'dbplyr'.

**Examples**

```
make_test_dat() %>% head()
```

---

pool\_case

*Pool qualified clients from results of multiple definitions*

---

**Description**

This function filters and pools, i.e., row-binds, qualified clients/groups from different sources with an option to summarize by client. Unlike [bind\\_source\(\)](#), no need to supply variable names; the function will guess what should be included and their names from the supplied definition from [build\\_def\(\)](#). Whether a client is qualified relies on the flag variables set by [define\\_case\(\)](#). Therefore, this function is intended to be used only with the built-in [define\\_case\(\)](#) as `def_fn` in [build\\_def\(\)](#).

**Usage**

```
pool_case(
  data,
  def,
  output_lvl = c("raw", "clnt"),
  include_src = c("all", "has_valid", "n_per_clnt"),
  ...
)
```

**Arguments**

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>        | A list of data.frame or remote table which should be output from <code>execute_def()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>def</code>         | A tibble of case definition generated by <code>build_def()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>output_lvl</code>  | Either: <ul style="list-style-type: none"> <li>• "raw" - output all records (default),</li> <li>• or "clnt" - output one record per client with summaries including the date and source of the earliest record ('first_entry_date/src'), the first valid record ('first_valid_date/src'), and the latest record ('last_entry_date/src'). Source-specific record counts are also provided (see the return section).</li> </ul>                                                                                                                                                                                                                                                                                                                              |
| <code>include_src</code> | Character. It determines records from which sources should be included. This matters when clients were identified only from, not all, but some of the sources. This choice will not impact the number of client that would be identified but has impact on the number of records and the first/latest entry date. The options are one of: <ul style="list-style-type: none"> <li>• "all" - records from all sources are included;</li> <li>• "has_valid" - for each client, records from sources that contain at least one valid record are included;</li> <li>• "n_per_clnt" - for each client, if they had fewer than <code>n_per_clnt</code> records in a source (see <code>restrict_n()</code>), then records from that source are removed.</li> </ul> |
| <code>...</code>         | Additional arguments passing to <code>bind_source()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

**Value**

A data.frame or remote table with clients that satisfied the predefined case definition. The columns depend on `output_lvl`.

With `output_lvl = "raw"`, the output is the row-bound records of the qualified clients. Note that these are the records `define_case()` returned for each source, i.e., only those matching the codes in `vals`, further subset by `include_src`; they are referred to as the client's "included records" below. The variables that `build_def()` named in the definition are kept and renamed to a common set of names:

- `def` - the definition label (`def_lab` in `build_def()`) the record was identified by.
- `src` - the source label (`src_labs` in `build_def()`) the record came from.
- `clnt_id`, `uid`, `date_var` - the client id, unique record id, and date variable, taken from the `clnt_id`, `uid`, and `date_var` arguments of `define_case()`. `uid` and `date_var` are present only if they were supplied in the definition.
- `flag_restrict_n`, `flag_restrict_date` - the flags produced by `restrict_n()` and `restrict_date()`, present only if the definition used `n_per_clnt` greater than 1, or `apart/within`, respectively.
- `flag_valid_record` - 1 if the record satisfied all the criteria of the definition, and 0 otherwise. It is taken from `flag_restrict_date` if present, otherwise from `flag_restrict_n`, and is filled with 1 for the sources that had no such flag (i.e., sources whose criteria were applied by filtering instead of flagging).
- `src_No` - the position of the source in the input list. This is added by `bind_source()` when binding local data.frames only, and is absent when all inputs are remote tables.

With `output_lvl = "clnt"`, the output has one row per def and `clnt_id` summarizing that client's included records. The date columns are present only if `date_var` was supplied for every source in the definition; otherwise they are omitted with a warning.

- `first_entry_date`, `first_entry_src` - the date of the client's earliest included record, and the source it came from, regardless of whether that record was valid.
- `first_valid_date`, `first_valid_src` - the date of the client's earliest record with `flag_valid_record` equal to 1, and the source it came from. Note that `first_valid_date` can be later than `first_entry_date`: the earliest record of a client is often not one of the records that made them a case.
- `last_entry_date`, `last_entry_src` - the date of the client's latest included record, and the source it came from, regardless of whether that record was valid.
- `raw_in[src]` - one column per source, counting the client's included records in that source.
- `valid_in[src]` - one column per source, counting the client's valid records (i.e., the number of flags) in that source.

The source of the first/last record is determined by comparing dates, and ties are broken by the alphabetical order of the source labels, e.g., if a client's earliest date is present in both "dad" and "msp", `first_entry_src` is "dad". Records with a missing date never contribute to the first/last date or source.

## Examples

```
# toy data
df1 <- make_test_dat()
df2 <- make_test_dat()

# use build_def to make a toy definition
sud_def <- build_def("SUD", # usually a disease name
  src_lab = c("src1", "src2"), # identify from multiple sources, e.g., hospitalization, ED visits.
  # functions that filter the data with some criteria
  def_fn = define_case,
  fn_args = list(
    vars = starts_with("diagx"),
    match = "start", # "start" will be applied to all sources as length = 1
    vals = list(c("304"), c("305")),
    clnt_id = "clnt_id", # list()/c() could be omitted for single element
    # c() can be used in place of list
    # if this argument only takes one value for each source
    n_per_clnt = c(2, 3)
  )
)

# save the definition for re-use
# saveRDS(sud_def, file = some_path)

# execute definition
sud_by_src <- sud_def %>% execute_def(with_data = list(src1 = df1, src2 = df2))

# pool results from src1 and src2 together at client level
pool_case(sud_by_src, sud_def, output_lvl = "clnt")
```

---

|          |                                                                            |
|----------|----------------------------------------------------------------------------|
| report_n | <i>Report the number of distinct values in a column across data frames</i> |
|----------|----------------------------------------------------------------------------|

---

## Description

This function is intended to mimic `dplyr::n_distinct()` for multiple inputs. It is useful to report the number of clients throughout a series of inclusion or exclusion steps. A use case could be getting the Ns for the sample definition flowchart in an epidemiological study. It is also useful for inline reporting of Ns in an R Markdown document.

## Usage

```
report_n(..., on, force_proceed = getOption("healthdb.force_proceed"))
```

## Arguments

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>...</code>           | Data frames or remote tables (e.g., from 'dbplyr')                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>on</code>            | The column to report on. It must be present in all data sources.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>force_proceed</code> | A logical for whether to ask for user input in order to proceed when the data is not local data.frames, and a query needs to be executed before reporting. The default is fetching from options (FALSE). Use <code>options(healthdb.force_proceed = TRUE)</code> to suppress the prompt once and for all. In non-interactive sessions (e.g., scripts run via Rscript, knitr), the confirmation prompt cannot be displayed, and the function stops with an error unless <code>force_proceed = TRUE</code> . |

## Value

A sequence of the number of distinct on values, one for each input

## Examples

```
# some exclusions
iris_1 <- subset(iris, Petal.Length > 1)
iris_2 <- subset(iris, Petal.Length > 2)

# get n at each operation
n <- report_n(iris, iris_1, iris_2, on = Species)
n

# get the difference at each step
diff(n)

# data in a list
iris_list <- list(iris_1, iris_2)
report_n(rlang::splice(iris_list), on = Species)
# if you loaded tidyverse, this will also work
# report_n(!!!iris_list, on = Species)
```

---

|               |                                                                            |
|---------------|----------------------------------------------------------------------------|
| restrict_date | <i>Remove or flag groups that failed to meet conditions based on dates</i> |
|---------------|----------------------------------------------------------------------------|

---

## Description

For each client or group, interpret if they have *n* records that are at least certain days apart AND within a specified time span. When identifying events/diseases from administrative data, definitions often require, e.g., *n* diagnoses that are at least some days apart within some years. This function is intended for such use and optimized to avoid looping through all *n*-size combinations of dates per client. See `vignette("if_date_logic")` for an explanation of the algorithm and how it is translated into SQL.

## Usage

```
restrict_date(
  data,
  clnt_id,
  date_var,
  n,
  apart = NULL,
  within = NULL,
  uid = NULL,
  mode = c("flag", "filter"),
  flag_at = c("left", "right"),
  dup.rm = TRUE,
  force_collect = FALSE,
  verbose = getOption("healthdb.verbose"),
  check_missing = FALSE,
  ...
)
```

## Arguments

|                       |                                                                                                                   |
|-----------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>data</code>     | Data frames or remote tables (e.g., from <code>dbplyr::tbl_sql()</code> )                                         |
| <code>clnt_id</code>  | Grouping variable (quoted/unquoted).                                                                              |
| <code>date_var</code> | Variable name (quoted/unquoted) for the dates to be interpreted.                                                  |
| <code>n</code>        | An integer for the size of a draw.                                                                                |
| <code>apart</code>    | An integer specifying the minimum gap (in days) between adjacent dates in a draw.                                 |
| <code>within</code>   | An integer specifying the maximum time span (in days) of a draw.                                                  |
| <code>uid</code>      | Variable name for a unique row identifier. It is necessary for SQL to produce consistent result based on sorting. |
| <code>mode</code>     | Either:                                                                                                           |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"> <li>• "flag" - add a new column 'flag_restrict_date' indicating if the condition was met (flag = 1 if the time period starting or ending at the current record satisfied the apart-within condition),</li> <li>• or "filter" - remove clients without any qualified record from the data. Default is "flag".</li> </ul>                                                                                                                                                                                        |
| flag_at       | Character, define if the flag should be placed at the start ("left") or end ("right") of a time period that contains n qualified records. Defaults to "left". Note that this would impact the first and last qualified/diagnosed dates of a client, e.g., using "right" will have the first flag not at the earliest but the date which the client became qualified. For example, if the condition was 2 records within a year, for c("2023-01-01", "2023-04-01", "2024-05-01"), flag will be c(0, 1, 0) for "right" while c(1, 0, 0) for "left". |
| dup.rm        | Logical for whether multiple records on the same date should be counted as one in the calculation. Only applicable when within is supplied without apart; duplicated dates have no impact when apart is present as the n dates must be distinct if they were apart. Default is TRUE.                                                                                                                                                                                                                                                              |
| force_collect | A logical for whether to download the data and use the local method when both apart and within are supplied. For remote tables only. The default (FALSE) runs everything on the database, which involves writing a temporary table and an overlap (non-equi) join; if the database does not permit these, set TRUE to download the data and compute locally instead (may be slow). The argument has no effect when only one of apart/within is supplied, as those cases always run on the database.                                               |
| verbose       | A logical for whether to explain the query and report how many groups were removed. Default is fetching from options. Use options(healthdb.verbose = FALSE) to suppress once and for all. Reporting is not for remote tables as the query is not executed immediately, thus no result is available for summary without adding an extra run (may be slow) of the query.                                                                                                                                                                            |
| check_missing | A logical for whether to check and remove missing entries in date_var before applying the conditions. Default is FALSE for faster run time.                                                                                                                                                                                                                                                                                                                                                                                                       |
| ...           | Additional argument passing to <code>data.table::as.IDate()</code> for date conversion.                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

**Value**

A subset of the input data that satisfied the date requirements, or the input data with a new flag column.

**See Also**

`if_date()`

**Examples**

```
sample_size <- 30
df <- data.frame(
  clnt_id = sample(1:sample_size, sample_size, replace = TRUE),
  service_dt = sample(seq(as.Date("2020-01-01"), as.Date("2020-01-31"), by = 1),
    size = sample_size, replace = TRUE
```

```

),
diagx = sample(letters, size = sample_size, replace = TRUE),
diagx_1 = sample(c(NA, letters), size = sample_size, replace = TRUE),
diagx_2 = sample(c(NA, letters), size = sample_size, replace = TRUE)
)

# Keep clients with 2 records that were 1 week apart within 1 month
restrict_date(df, clnt_id, service_dt, n = 2, apart = 7, within = 30)

```

---

|            |                                                          |
|------------|----------------------------------------------------------|
| restrict_n | <i>Remove or flag groups smaller than a minimum size</i> |
|------------|----------------------------------------------------------|

---

## Description

Remove or flag groups or clients that have fewer than some number of rows or fewer than some number of distinct values in a variable. For example, it can be used to remove clients that had fewer than `n` visits to some service on different dates from some administrative records. It offers filtering with `dplyr::n_distinct()` functionality for database input.

## Usage

```

restrict_n(
  data,
  clnt_id,
  n_per_clnt,
  count_by = NULL,
  mode = c("flag", "filter"),
  verbose = getOption("healthdb.verbose")
)

```

## Arguments

|                         |                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>       | Data.frames or remote tables (e.g., from <code>dbplyr::tbl_sql()</code> )                                                                                                                                                                                                                                                                                                           |
| <code>clnt_id</code>    | Grouping variable (quoted/unquoted).                                                                                                                                                                                                                                                                                                                                                |
| <code>n_per_clnt</code> | A single number specifying the minimum group size.                                                                                                                                                                                                                                                                                                                                  |
| <code>count_by</code>   | Another variable dictating the counting unit of <code>n_per_clnt</code> . The default is <code>NULL</code> meaning the inclusion criteria is the number of row, i.e., <code>dplyr::n() &gt;= n_per_clnt</code> . If it is not <code>NULL</code> , the criteria becomes equivalent to <code>dplyr::n_distinct(count_by) &gt;= n_per_clnt</code> .                                    |
| <code>mode</code>       | Either "flag" - add a new column 'flag_restrict_n' indicating if the client met the condition (all rows from a qualified client would have <code>flag = 1</code> ), or "filter" - remove clients that did not meet the condition from the data. Default is "flag".                                                                                                                  |
| <code>verbose</code>    | A logical for whether to explain the query and report how many groups were removed. Default is fetching from options. Use <code>options(healthdb.verbose = FALSE)</code> to suppress once and for all. Reporting is not for remote tables as the query is not executed immediately, thus no result is available for summary without adding an extra run (may be slow) of the query. |

**Value**

A subset of the input data that satisfied the group size requirement, or the input data with a new flag column.

**See Also**

[dplyr::n\(\)](#), [dplyr::n\\_distinct\(\)](#)

**Examples**

```
# flag cyl groups with less than 8 cars
restrict_n(mtcars, clnt_id = cyl, n_per_clnt = 8, mode = "flag") %>%
  head()

# remove cyl groups with less than 2 types of gear boxes
restrict_n(mtcars, clnt_id = cyl, n_per_clnt = 3, count_by = gear, mode = "filter")
```

# Index

- \* **datasets**
  - elix\_codes, 19
- base::cut(), 10
- bind\_source, 2
- bind\_source(), 31, 32
- build\_def, 3
- build\_def(), 31, 32
- collapse\_episode, 5
- compute\_comorbidity, 6
- compute\_comorbidity(), 19, 20
- compute\_duration, 10
- cut\_period, 11
- data.table::as.IDate(), 5, 29, 36
- dbplyr::tbl\_sql(), 7, 13, 16, 21, 26, 35, 37
- define\_case, 12
- define\_case(), 4, 15, 16, 31, 32
- define\_case\_with\_age, 15
- define\_case\_with\_age(), 4, 12, 14
- dplyr::anti\_join(), 21
- dplyr::filter(), 21, 25
- dplyr::if\_all(), 25
- dplyr::if\_any(), 25
- dplyr::join\_by(), 21
- dplyr::left\_join(), 23
- dplyr::mutate(), 10
- dplyr::n(), 38
- dplyr::n\_distinct(), 34, 37, 38
- dplyr::select(), 7, 13, 16, 24, 26
- dplyr::setdiff(), 21
- dplyr::starts\_with(), 7, 13, 16, 26
- elix\_codes, 7, 9, 19
- exclude, 21
- exclude(), 12
- execute\_def, 22
- execute\_def(), 2, 3, 32
- fetch\_var, 23
- glue::glue\_sql(), 27
- identify\_row(identify\_rows), 25
- identify\_row(), 12, 13, 16, 17
- identify\_rows, 25
- if\_date, 28
- if\_date(), 36
- lookup, 29
- lubridate::ymd(), 10, 12
- make\_test\_dat, 30
- pool\_case, 31
- report\_n, 34
- restrict\_date, 35
- restrict\_date(), 12, 14, 15, 17, 18, 29, 32
- restrict\_n, 37
- restrict\_n(), 12, 14, 15, 17, 32