Package: coder (via r-universe)

August 29, 2024

Type Package

Title Deterministic Categorization of Items Based on External Code Data

Version 0.13.10

Description Fast categorization of items based on external code data identified by regular expressions. A typical use case considers patient with medically coded data, such as codes from the International Classification of Diseases ('ICD') or the Anatomic Therapeutic Chemical ('ATC') classification system. Functions of the package relies on a triad of objects: (1) case data with unit id:s and possible dates of interest; (2) external code data for corresponding units in (1) and with optional dates of interest and; (3) a classification scheme ('classcodes' object) with regular expressions to identify and categorize relevant codes from (2). It is easy to introduce new classification schemes ('classcodes' objects) or to use default schemes included in the package. Use cases includes patient categorization based on 'comorbidity indices' such as 'Charlson', 'Elixhauser', 'RxRisk V', or the 'comorbidity-polypharmacy' score (CPS), as well as adverse events after hip and knee replacement surgery.

License GPL-2

Depends R (>= 3.3)

Suggests covr, testthat, knitr, rmarkdown, writexl

Imports data.table, decoder, generics, methods, tibble

LazyData TRUE

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

URL https://docs.ropensci.org/coder/

BugReports https://github.com/ropensci/coder/issues

Encoding UTF-8

Language en-US

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/coder

RemoteRef master

RemoteSha 957d87186c11f55b6df0406684bc787b86b74d39

Contents

---

ae *Classcodes for adverse events after knee and hip arthroplasty*

Description

ICD-10 group names are prefixed by two letters as given by the references. Two groups (DB and DM) are split into two due to different conditions.

Usage

```
knee_ae

hip_ae
```

Format

Data frame with 3 columns:

group Different types of adverse events (see reference section)

icd10 regular expressions identifying ICD-10 codes for each group

icd10_fracture regular expressions for fracture patients. Essentially the same as `regex` but with some additional codes for group "DM1 other"

kva regular expressions identifying KVA codes

condition special conditions are used, see below.

An object of class `classcodes` (inherits from `tbl_df`, `tbl`, `data.frame`) with 7 rows and 5 columns.

Hip fractures

Adverse events (AE) codes for hip fractures are based on codes for elective cases but with some additional codes for DM 1 (N300, N308, N309 and N390).

Conditions

Special conditions apply to all categories. Those require non-standard modifications of the classcodes data prior to categorization.

hbdia1_hdia `TRUE` if the code was given as any type of diagnose during hospital visit for index operation, or as main diagnose for later visits, otherwise `FALSE`

late_hdia `TRUE` if the code was given as main diagnose at a later visit after the index operation, otherwise `FALSE`

post_op `TRUE` if the code was given at a later visit after the index operation, otherwise `FALSE`

References

Magneli M, Unbeck M, Rogmark C, Rolfson O, Hommel A, Samuelsson B, et al. Validation of adverse events after hip arthroplasty: a Swedish multi-centre cohort study. BMJ Open. 2019 Mar 7;9(3):e023773.

See Also

hip_ae_hailer

Other default classcodes: `charlson`, `cps`, `elixhauser`, `hip_ae_hailer`, `rxriskv`

all_classcodes *Summary data for all default classcodes object in the package*

Description

Tabulate object names and list all related versions of implemented regular expressions and index weights.

Usage

```
all_classcodes()
```

Value

tibble::tibble with columns describing all default classcodes objects from the package.

See Also

Other classcodes: `as.data.frame.classified()`, `classcodes`, `codebook()`, `print.classcodes()`, `print.classified()`, `set_classcodes()`, `summary.classcodes()`, `visualize.classcodes()`

Examples

```
all_classcodes()
```

as.data.frame.classified *Convert output from classify() to matrix/data.frame/data.table*

Description

Convert output from classify() to matrix/data.frame/data.table

Usage

```
## S3 method for class 'classified'
as.data.frame(x, ...)

## S3 method for class 'classified'
as.data.table(x, ...)

## S3 method for class 'classified'
as.matrix(x, ...)
```

Arguments

x	output from classify()
...	ignored

Value

data frame/data table with:

- first column named as "id" column specified as input to classify() and with data from row.names(x)
- all columns from classified
- no row names

or simply the input matrix without additional attributes

See Also

Other classcodes: all_classcodes(), classcodes, codebook(), print.classcodes(), print.classified(), set_classcodes(), summary.classcodes(), visualize.classcodes()

Examples

```
x <- classify(c("C80", "I20", "unvalid_code"), "elixhauser")

as.matrix(x)[, 1:3]
as.data.frame(x)[, 1:3]
data.table::as.data.table(x)[, 1:3]

# `as_tibble()` works automatically due to internal use of `as.data.frame()`.
tibble::as_tibble(x)
```

---

as.keyvalue.classcodes

Make keyvalue object from classcodes object

Description

S3-method for generic decoder::as.keyvalue()

Usage

```
## S3 method for class 'classcodes'
as.keyvalue(x, coding, cc_args = list(), ...)
```

Arguments

x	classcodes object
coding	either a vector with codes from the original classification, or a name (character vector of length one) of a keyvalue object from package "decoder" (for example "icd10cm" or "atc")
cc_args	List of named arguments passed to set_classcodes()
...	additional arguments passed to decoder::as.keyvalue()

Value

Object of class keyvalue where key is the subset of codes from object$keyidentified by the regular expression from x and where value is the corresponding x$group. Hence, note that the original object$value is not used in the output.

Examples

```
# List all codes with corresponding classes as recognized by the Elixhauser
# comorbidity classification according to the Swedish version of the
# international classification of diseases version 10 (ICD-10-SE)
head(decoder::as.keyvalue(elixhauser, "icd10se"))

# Similar but with the American ICD-10-CM instead
# Note that the `value` column is similar as above
# (with names from `x$group`) and not
# from `object$value`
head(decoder::as.keyvalue(elixhauser, "icd10cm"))

# Codes identified by regular expressions based on ICD-9-CM and found in
# the Swedish version of ICD-9 used within the national cancer register
# (thus, a subset of the whole classification).
head(
  decoder::as.keyvalue(
    elixhauser, "icd9",
    cc_args = list(regex = "icd9cm")
  )
)
```

---

categorize *Categorize cases based on external data and classification scheme*

Description

This is the main function of the package, which relies of a triad of objects: (1) data with unit id:s and possible dates of interest; (2) codedata for corresponding units and with optional dates of interest and; (3) a classification scheme (classcodes object; cc) with regular expressions to identify and categorize relevant codes. The function combines the three underlying steps performed by codify(), classify() and index(). Relevant arguments are passed to those functions by codify_args and cc_args.

Usage

```
categorize(x, ...)

## S3 method for class 'data.frame'
categorize(x, ...)

## S3 method for class 'tbl_df'
categorize(x, ...)

## S3 method for class 'data.table'
categorize(x, ..., codedata, id, code, codify_args = list())

## S3 method for class 'codified'
categorize(
  x,
  ...,
  cc,
  index = NULL,
  cc_args = list(),
  check.names = TRUE,
  .data_cols = NULL
)
```

Arguments

x	data set with mandatory character id column (identified by argument `id = "<col_name>"`), and optional `Date` of interest (identified by argument `date = "<col_name>"`). Alternatively, the output from `codify()`
...	arguments passed between methods
codedata	external code data with mandatory character id column (identified by `id = "<col_name>"`), code column (identified by argument `code = "<col_name>"`) and optional `Date` column (identified by `codify_args = list(code_date = "<col_name>")`).
id	name of unique character id column found in both xand codedata. (where it must not be unique).
code	name of code column in codedata.
codify_args	Lists of named arguments passed to `codify()`
cc	`classcodes` object (or name of a default object from `all_classcodes()`).
index	Argument passed to `index()`. A character vector of names of columns with index weights from the corresponding classcodes object (as supplied by the ccargument). See `attr(cc, "indices")` for available options. Set to `FALSE` if no index should be calculated. If `NULL`, the default, all available indices (from `attr(cc, "indices")`) are provided.
cc_args	List with named arguments passed to `set_classcodes()`
check.names	Column names are based on `cc$group`, which might include spaces. Those names are changed to syntactically correct names by `check.names = TRUE`. Syntactically invalid, but grammatically correct names might be preferred for presentation of the data as achieved by `check.names = FALSE`. Alternatively, if

categorize is called repeatedly, longer informative names might be created by cc_args = list(tech_names = TRUE).

.data_cols used internally

Value

Object of the same class as x with additional logical columns indicating membership of groups identified by the classcodes object (the cc argument). Numeric indices are also included if requested by the index argument.

See Also

Other verbs: classify(), codify(), index_fun

Examples

```
# For this example, 1 core would suffice:
old_threads <- data.table::getDTthreads()
data.table::setDTthreads(1)

# For some patient data (ex_people) and related hospital visit code data
# with ICD 10-codes (ex_icd10), add the Elixhauser comorbidity
# conditions based on all registered ICD10-codes
categorize(
   x            = ex_people,
   codedata     = ex_icd10,
   cc           = "elixhauser",
   id           = "name",
   code         = "icd10"
)


# Add Charlson categories and two versions of a calculated index
# ("quan_original" and "quan_updated").
categorize(
   x            = ex_people,
   codedata     = ex_icd10,
   cc           = "charlson",
   id           = "name",
   code         = "icd10",
   index        = c("quan_original", "quan_updated")
)


# Only include recent hospital visits within 30 days before surgery,
categorize(
   x            = ex_people,
   codedata     = ex_icd10,
   cc           = "charlson",
   id           = "name",
   code         = "icd10",
   index        = c("quan_original", "quan_updated"),
```

```
   codify_args  = list(
      date      = "surgery",
      days      = c(-30, -1),
      code_date = "admission"
   )
)



# Multiple versions ------------------------------------------------------

# We can compare categorization by according to Quan et al. (2005); "icd10",
# and Armitage et al. (2010); "icd10_rcs" (see `?charlson`)
# Note the use of `tech_names = TRUE` to distinguish the column names from the
# two versions.

# We first specify some common settings ...
ind <- c("quan_original", "quan_updated")
cd  <- list(date = "surgery", days = c(-30, -1), code_date = "admission")

# ... we then categorize once with "icd10" as the default regular expression ...
categorize(
   x            = ex_people,
   codedata     = ex_icd10,
   cc           = "charlson",
   id           = "name",
   code         = "icd10",
   index        = ind,
   codify_args  = cd,
   cc_args      = list(tech_names = TRUE)
) %>%

# .. and once more with `regex = "icd10_rcs"`
categorize(
   codedata     = ex_icd10,
   cc           = "charlson",
   id           = "name",
   code         = "icd10",
   index        = ind,
   codify_args  = cd,
   cc_args      = list(regex = "icd10_rcs", tech_names = TRUE)
)



# column names -----------------------------------------------------------

# Default column names are based on row names from corresponding classcodes
# object but are modified to be syntactically correct.
default <-
   categorize(ex_people, codedata = ex_icd10, cc = "elixhauser",
              id = "name", code = "icd10")
```

```
# Set `check.names = FALSE` to retain original names:
original <-
  categorize(
    ex_people, codedata = ex_icd10, cc = "elixhauser",
    id = "name", code = "icd10",
    check.names = FALSE
   )

# Or use `tech_names = TRUE` for informative but long names (use case above)
tech <-
  categorize(ex_people, codedata = ex_icd10, cc = "elixhauser",
    id = "name", code = "icd10",
    cc_args = list(tech_names = TRUE)
  )

# Compare
tibble::tibble(names(default), names(original), names(tech))

# Go back to original number of threads
data.table::setDTthreads(old_threads)
```

`charlson` *Classcodes for Charlson comorbidity based on ICD-codes*

Description

Classcodes for Charlson comorbidity based on ICD-codes

Usage

```
charlson
```

Format

A data frame with 17 rows and 8 variables:

- `group`: comorbidity groups
- `description`: Verbal description of codes as described by Deyo et al. (1992).
- `icd10`: regular expressions identifying ICD-10 codes of each group as decoded from Quan et al. 2005. Note that this classification was not originally used with all weights! To simply use this classification table with weights other than `quan_original` and `quan_updated` might therefore lead to different results than originally intended for each index.
- `icd9cm_deyo`:Codes from table 1 column "Deyo's ICD-9-CM" in Quan et al. (2005). Procedure code 38.48 for peripheral vascular disease ignored.
- `icd9cm_enhanced`: Codes from table 1 column "Enhanced ICD-9-CM" in Quan et al. (2005).
- `icd10_rcs`: Codification by Armitage (2010). Note that Peptic ulcer disease is not included. All liver diseases (including mild) are included in "moderate or severe liver disease". All diabetes is included in "diabetes complication"

- `icd8_brusselaers`: Back translated version from ICD-10 to ICD-8 by Brusselaers et al. (2017). "Moderate and severe liver disease" contains all liver disease and "diabetes complication" contains all diabetes.
- `icd9_brusselaers`: Back translated version from ICD-10 to ICD-9 by Brusselaers et al. (2017). "Moderate and severe liver disease" contains all liver disease and "diabetes complication" contains all diabetes.
- `charlson`: original weights as suggested by Charlson et al. (1987)*
- `deyo_ramano`: weights suggested by Deyo and Romano*
- `dhoore`: weights suggested by D'Hoore*
- `ghali`: weights suggested by Ghali*
- `quan_original`: weights suggested by Quan (2005)
- `quan_updated`: weights suggested by Quan (2011)

- Weights decoded from Yurkovich et al. (2015).

References

Armitage, J. N., & van der Meulen, J. H. (2010). Identifying co-morbidity in surgical patients using administrative data with the Royal College of Surgeons Charlson Score. British Journal of Surgery, 97(5), 772–781.

Brusselaers N, Lagergren J. (2017) The Charlson Comorbidity Index in Registry-based Research. Methods Inf Med 2017;56:401–6.

Deyo, R. A., Cherkin, D. C., & Ciol, M. A. (1992). Adapting a clinical comorbidity index for use with ICD-9-CM administrative databases. Journal of Clinical Epidemiology, 45(6), 613–619.

Quan Hude et al. (2005). Coding algorithms for defining comorbidities in ICD-9-CM and ICD-10 administrative data. Medical care, 1130-1139.

Yurkovich, M., Avina-Zubieta, J. A., Thomas, J., Gorenchtein, M., & Lacaille, D. (2015). A systematic review identifies valid comorbidity indices derived from administrative health data. Journal of clinical epidemiology, 68(1), 3-14.

See Also

Other default classcodes: `ae`, `cps`, `elixhauser`, `hip_ae_hailer`, `rxriskv`

`classcodes` *Classcodes methods*

Description

`classcodes` are classification schemes based on regular expression stored in data frames. These are essential to the package and constitute the third part of the triad of case data, code data and a classification scheme.

Usage

```
as.classcodes(x, ...)

## S3 method for class 'classcodes'
as.classcodes(
  x,
  ...,
  regex = attr(x, "regexpr"),
  indices = attr(x, "indices"),
  hierarchy = attr(x, "hierarchy")
)

## S3 method for class 'data.frame'
as.classcodes(
  x,
  ...,
  regex = NULL,
  indices = NULL,
  hierarchy = attr(x, "hierarchy"),
  .name = NULL
)

is.classcodes(x)
```

Arguments

x	data frame with columns described in the details section. Alternatively a classcodes object to be modified.
...	arguments passed between methods#'
regex, indices	character vector with names of columns in x containing regular expressions/indices.
hierarchy	named list of pairwise group names to appear as superior and subordinate for indices. To be used for indexing when the subordinate class is redundant (see the details section of elixhauser for an example).
.name	used internally for name dispatch

Details

A classcodes object is a data frame with mandatory columns:

- group: unique and non missing class names
- At least one column with regular expressions (regex without Perl-like versions) defining class membership. Those columns can have arbitrary names (as specified by the regex argument). Occurrences of non unique regular expressions will lead to the same class having multiple names. This is accepted but will raise a warning. Classes do not have to be disjunct.

The object can have additional optional columns:

- description: description of each category

- condition: a class might have conditions additional to what is expressed by the regular expressions. If so, these should be specified as quoted expressions that can be evaluated within the data frame used by classify()
- weights for each class used by index(). Could be more than one and could have arbitrary names (as specified by the indicesargument).

Value

Object of class classcodes (inheriting from data frame) with additional attributes:

- code: the coding used (for example "icd10", or "ATC"). NULL for unknown/arbitrary coding.
- regexprs: name of columns with regular expressions (as specified by the regexargument)
- indices: name of columns with (optional) index weights (as specified by the indicesargument)
- hierarchy: list as specified by the hierarchy argument.
- name: name as specified by the .name argument.

See Also

vignette("classcodes") vignette("Interpret_regular_expressions") The package have several default classcodes included, see all_classcodes().

Other classcodes: all_classcodes(), as.data.frame.classified(), codebook(), print.classcodes(), print.classified(), set_classcodes(), summary.classcodes(), visualize.classcodes()

Other classcodes: all_classcodes(), as.data.frame.classified(), codebook(), print.classcodes(), print.classified(), set_classcodes(), summary.classcodes(), visualize.classcodes()

Examples

```
# The Elixhauser comorbidity classification is already a classcodes object
is.classcodes(coder::elixhauser)

# Strip its class attributes to use in examples
df <- as.data.frame(coder::elixhauser)

# Specify which columns store regular expressions and indices
# (assume no hierarchy)
elix <-
  as.classcodes(
    df,
    regex     = c("icd10", "icd10_short", "icd9cm", "icd9cm_ahrqweb", "icd9cm_enhanced"),
    indices   = c("sum_all", "sum_all_ahrq", "walraven",
                  "sid29", "sid30", "ahrq_mort", "ahrq_readm"),
    hierarchy = NULL
  )
elix

# Specify hierarchy for patients with different types of cancer and diabetes
# See `?elixhauser` for details
as.classcodes(
  elix,
  hierarchy = list(
```

```
    cancer   = c("metastatic cancer", "solid tumor"),
    diabetes = c("diabetes complicated", "diabetes uncomplicated")
  )
)

# Several checks are performed to not allow any erroneous classcodes object
## Not run:
  as.classcodes(iris)
  as.classcodes(iris, regex = "Species")

## End(Not run)
```

`classify` *Classify codified data*

Description

This is the second step of `codify()` %>% `classify()` %>% `index()`. Hence, the function takes a codified data set and classify each case based on relevant codes as identified by the classification scheme provided by a `classcodes` object.

Usage

```
classify(codified, cc, ..., cc_args = list())

## Default S3 method:
classify(codified, cc, ..., cc_args = list())

## S3 method for class 'codified'
classify(codified, ...)

## S3 method for class 'data.frame'
classify(codified, ...)

## S3 method for class 'data.table'
classify(codified, cc, ..., id, code, cc_args = list())
```

Arguments

`codified`	output from `codify()`
`cc`	`classcodes` object (or name of a default object from `all_classcodes()`).
`...`	arguments passed between methods
`cc_args`	List with named arguments passed to `set_classcodes()`
`code, id`	name of code/id columns (in `codified`).

Value

Object of class "classified". Inheriting from a Boolean matrix with one row for each element/row of `codified` and columns for each class with corresponding class names (according to the `classcodes` object). Note, however, that `print.classified()` preview this output as a tibble.

See Also

`as.data.frame.classified()`, `as.data.table.classified()` and `as.matrix.classified()`, `print.classified()`

Other verbs: `categorize()`, `codify()`, `index_fun`

Examples

```
# classify.default() ------------------------------------------------------

# Classify individual ICD10-codes by Elixhauser
classify(c("C80", "I20", "unvalid_code"), "elixhauser")



# classify.codified() -----------------------------------------------------

# Prepare some codified data with ICD-10 codes during 1 year (365 days)
# before surgery
x <-
  codify(
    ex_people,
    ex_icd10,
    id        = "name",
    code      = "icd10",
    date      = "surgery",
    days      = c(-365, 0),
    code_date = "admission"
  )

# Classify those patients by the Charlson and Elixhasuer comorbidity indices
classify(x, "charlson")        # classcodes object by name ...
classify(x, coder::elixhauser) # ... or by the object itself


# -- start/stop --
# Assume that a prefix "ICD-10 = " is used for all codes and that some
# additional numbers are added to the end
x$icd10 <- paste0("ICD-10 = ", x$icd10)

# Set start = FALSE to identify codes which are not necessarily found in the
# beginning of the string
classify(x, "charlson", cc_args = list(start = FALSE))
```

```
# -- regex --
# Use a different version of Charlson (as formulated by regular expressions
# according to the Royal College of Surgeons (RCS) by passing arguments to
# `set_classcodes()` using the `cc_args` argument
y <-
  classify(
    x,
    "charlson",
    cc_args = list(regex = "icd10_rcs")
  )


# -- tech_names --
# Assume that we want to compare the results using the default ICD-10
# formulations (from Quan et al. 2005) and the RCS version and that the result
# should be put into the same data frame. We can use `tech_names = TRUE`
# to distinguish variables with otherwise similar names
cc <- list(tech_names = TRUE) # Prepare sommon settings
compare <-
  merge(
  classify(x, "charlson", cc_args = cc),
  classify(x, "charlson", cc_args = c(cc, regex = "icd10_rcs"))
)
names(compare) # long but informative and distinguishable column names



# classify.data.frame() / classify.data.table() ------------------------

# Assume that `x` is a data.frame/data.table without additional attributes
# from `codify()` ...
xdf <- as.data.frame(x)
xdt <- data.table::as.data.table(x)

# ... then the `id` and `code` columns must be specified explicitly
classify(xdf, "charlson", id = "name", code = "icd10")
classify(xdt, "charlson", id = "name", code = "icd10")
```

---

codebook *codebook(s) for classcodes object*

Description

summary.classcodes() and visualize.classcodes() are used to summarize/visualize classcodes in R. A codebook, on the other hand, is an exported summary saved in an Excel spreadsheet to use in collaboration with non R-users. Several codebooks might be combined into a single Excel document with several sheets (one for each codebook).

Usage

```
codebook(object, coding, ..., file = NULL)

## S3 method for class 'codebook'
print(x, ...)

codebooks(..., file = NULL)
```

Arguments

object	classcodes object
coding	either a vector with codes from the original classification, or a name (character vector of length one) of a keyvalue object from package "decoder" (for example "icd10cm" or "atc")
...	Additional arguments for each function: • codebook(): arguments passed to summary.classcodes() • codebooks(): multiple named outputs from codebook() • print.codebook(): arguments passed to tibble::print.tbl()
file	name/path to Excel file for data export
x	output from codebook()

Value

Functions are primarily called for their side effects (exporting data to Excel or printing to screen). In addition:

- codebook()returns list of data frames describing relationship between groups and individual codes
- codebooks() returns a concatenated list with output from codebook(). Only one 'README' object is kept however and renamed as such.
- print.codebook()returns x (invisible)

See Also

Other classcodes: all_classcodes(), as.data.frame.classified(), classcodes, print.classcodes(), print.classified(), set_classcodes(), summary.classcodes(), visualize.classcodes()

Examples

```
# codebook() ------------------------------------------------------------
## Not run:
# Export codebook (to temporary file) with all codes identified by the
# Elixhauser comorbidity classification based on ICD-10-CM
codebook(elixhauser, "icd10cm", file = tempfile("codebook", fileext = ".xlsx"))

# All codes from ICD-9-CM Disease part used by Elixhauser enhanced version
codebook(elixhauser, "icd9cmd",
```

```
  cc_args = list(regex = "icd9cm_enhanced",
  file = tempfile("codebook", fileext = ".xlsx"))
)

# The codebook returns a list with three objects.
# Access a dictionary table with translates of each code to text:
codebook(charlson, "icd10cm")$all_codes


# print.codebook() ------------------------------------------------------

# If argument `file` is unspecified, a preview of each sheet of the codebook is
# printed to the screen
(cb <- codebook(charlson, "icd10cm"))

# The preview can be modified by arguments to the print-method
print(cb, n = 20)


# codebooks() -----------------------------------------------------------

# Combine codebooks based on different versions of the regular expressions
# and export to a single (temporary) Excel file
c1 <- codebook(elixhauser, "icd10cm")
c2 <- codebook(elixhauser, "icd9cmd",
  cc_args = list(regex = "icd9cm_enhanced")
  )

codebooks(
  elix_icd10 = c1, elix_icd9cm = c2,
  file = tempfile("codebooks", fileext = ".xlsx")
)

## End(Not run)
```

`codify` *Codify case data with external code data (within specified time frames)*

Description

This is the first step of `codify()` %>% `classify()` %>% `index()`. The function combines case data from one data set with related code data from a second source, possibly limited to codes valid at certain time points relative to case dates.

Usage

```
codify(x, codedata, ..., id, code, date = NULL, code_date = NULL, days = NULL)

## S3 method for class 'data.frame'
codify(x, ..., id, date = NULL, days = NULL)
```

```
## S3 method for class 'data.table'
codify(
  x,
  codedata,
  ...,
  id,
  code,
  date = NULL,
  code_date = NULL,
  days = NULL,
  alnum = FALSE,
  .copy = NA
)

## S3 method for class 'codified'
print(x, ..., n = 10)
```

Arguments

x	data set with mandatory character id column (identified by argument id = "<col_name>"), and optional Date of interest (identified by argument date = "<col_name>"). Alternatively, the output from codify()
codedata	additional data with columns including case id (character), code and an optional date (Date) for each code. An optional column condition might distinguish codes/dates with certain characteristics (see example).
...	arguments passed between methods
id, code, date, code_date	column names with case id (character from x and codedata), code (from x) and optional date (Date from x) and code_date (Date from codedata).
days	numeric vector of length two with lower and upper bound for range of relevant days relative to date. See "Relevant period".
alnum	Should codes be cleaned from all non alphanumeric characters?
.copy	Should the object be copied internally by data.table::copy()? NA (by default) means that objects smaller than 1 GB are copied. If the size is larger, the argument must be set explicitly. Set TRUE to make copies regardless of object size. This is recommended if enough RAM is available. If set to FALSE, calculations might be carried out but the object will be changed by reference. IMPORTANT! This might lead to undesired consequences and should only be used if absolutely necessary!
n	number of rows to preview as tibble. The output is technically a data.table::data.table, which might be an unusual format to look at. Use n = NULL to print the object as is.

Value

Object of class codified (inheriting from data.table::data.table). Essentially x with additional columns: code, code_date: left joined from codedata or NA if no match within period. in_period:

Boolean indicator if the case had at least one code within the specified period.

The output has one row for each combination of "id" from `x` and "code" from `codedata`. Rows from `x` might be repeated accordingly.

Relevant period

Some examples for argument `days`:

- `c(-365, -1)`: window of one year prior to the `date` column of `x`. Useful for patient comorbidity.
- `c(1, 30)`: window of 30 days after `date`. Useful for adverse events after a surgical procedure.
- `c(-Inf, Inf)`: no limitation on non-missing dates.
- `NULL`: no time limitation at all.

See Also

Other verbs: `categorize()`, `classify()`, `index_fun`

Examples

```
# Codify all patients from `ex_people` with their ICD-10 codes from `ex_icd10`
x <- codify(ex_people, ex_icd10, id = "name", code = "icd10")
x

# Only consider codes if recorded at hospital admissions within one year prior
# to surgery
codify(
  ex_people,
  ex_icd10,
  id        = "name",
  code      = "icd10",
  date      = "surgery",
  code_date = "admission",
  days      = c(-365, 0)   # admission during one year before surgery
)

# Only consider codes if recorded after surgery
codify(
  ex_people,
  ex_icd10,
  id        = "name",
  code      = "icd10",
  date      = "surgery",
  code_date = "admission",
  days      = c(1, Inf)     # admission any time after surgery
)


# Dirty code data ------------------------------------------------------------

# Assume that codes contain unwanted "dirty" characters
```

```
# Those could for example be a dot used by ICD-10 (i.e. X12.3 instead of X123)
dirt <- c(strsplit(c("!#%&/()=?`,.-_"), split = ""), recursive = TRUE)
rdirt <- function(x) sample(x, nrow(ex_icd10), replace = TRUE)
sub <- function(i) substr(ex_icd10$icd10, i, i)
ex_icd10$icd10 <-
  paste0(
    rdirt(dirt), sub(1),
    rdirt(dirt), sub(2),
    rdirt(dirt), sub(3),
    rdirt(dirt), sub(4),
    rdirt(dirt), sub(5)
  )
head(ex_icd10)

# Use `alnum = TRUE` to ignore non alphanumeric characters
codify(ex_people, ex_icd10, id = "name", code = "icd10", alnum = TRUE)



# Big data ----------------------------------------------------------------

# If `data` or `codedata` are large compared to available
# Random Access Memory (RAM) it might not be possible to make internal copies
# of those objects. Setting `.copy = FALSE` might help to overcome such problems

# If no copies are made internally, however, the input objects (if data tables)
# would change in the global environment
x2 <- data.table::as.data.table(ex_icd10)
head(x2) # Look at the "icd10" column (with dirty data)

# Use `alnum = TRUE` combined with `.copy = FALSE`
codify(ex_people, x2, id = "name", code = "icd10", alnum = TRUE, .copy = FALSE)

# Even though no explicit assignment was specified
# (neither for the output of codify(), nor to explicitly alter `x2`,
# the `x2` object has changed (look at the "icd10" column!):
head(x2)

# Hence, the `.copy` argument should only be used if necessary
# and if so, with caution!


# print.codify() ----------------------------------------------------------

x # Preview first 10 rows as a tibble
print(x, n = 20) # Preview first 20 rows as a tibble
print(x, n = NULL) # Print as data.table (ignoring the 'classified' class)
```

---

cps *Classcodes for the comorbidity-polypharmacy score (CPS) based on ICD-10 codes*

Description

Classcodes for the comorbidity-polypharmacy score (CPS) based on ICD-10 codes

Usage

```
cps
```

Format

A data frame with 2 rows and 2 variables:

group comorbidity groups, either "ordinary" for most ICD-10-codes or "special" for codes beginning with "UA", "UB" and "UP"

icd10 regular expressions identifying ICD-10 codes of each group

only_ordinary index weights, 1 for ordinary and 0 for special

References

Stawicki, Stanislaw P., et al. "Comorbidity polypharmacy score and its clinical utility: A pragmatic practitioner's perspective." Journal of emergencies, trauma, and shock 8.4 (2015): 224.

See Also

Other default classcodes: ae, charlson, elixhauser, hip_ae_hailer, rxriskv

---

`elixhauser` *Classcodes for Elixhauser based on ICD-codes*

Description

Solid tumors are subordinate to metastatic cancer. A patient with both conditions will still be classified as such but a possible (weighted) index value will only account for metastatic cancer. The same is true for "diabetes uncomplicated", which is subordinate of "diabetes complicated". See Elixhauser et al. (1998).

Usage

```
elixhauser
```

Format

A data frame with 31 rows and 8 variables:

group comorbidity groups

icd10 regular expressions identifying ICD-10 codes of each group. Corresponds to column 'ICD-10' in table 2 of Quan et al. (2005).

icd10_short regular expressions identifying only the first three characters of ICD-10 codes of each group. This alternative version was added only to use in emergency when only the first three digits are available. It is not an official version and we do not recommend to use it!

icd9cm Corresponds to column 'Elixhauser's Original ICD-9-CM' in table 2 of Quan et al. (2005).

icd9cm_ahrqweb Corresponds to column 'Elixhauser AHRQ-Web ICD-9-CM' in table 2 of Quan et al. (2005).

icd9cm_enhanced Corresponds to column 'Enhanced ICD-9-CM' in table 2 of Quan et al. (2005).

sum_all all weights = 1 (thus no weights)

sum_all_ahrq as `sum_all` excluding "cardiac arrhythmia. Compare to `icd9cm_ahrqweb` which does not consider this condition.

walraven weights suggested by Walraven et al. (2009)

sid29 weights suggested by Thompson et al. (2015) based on all conditions except cardiac arrhythmia

sid30 weights suggested by Thompson et al. (2015) based on all conditions

ahrq_mort weights for in-hospital mortality suggested by Moore et al. (2017)

ahrq_readm weights for readmissions suggested by Moore et al. (2017)

Details

Note that "DRG screen" as proposed in table 1 of Elixhauser et al. (1998) is not handled by the coder package. This should instead be considered as an additional pre- or post-processing step!

References

Elixhauser A, Steiner C, Harris DR, Coffey RM (1998). Comorbidity Measures for Use with Administrative Data. Med Care. 1998;36(1):8–27.

Moore, B. J., White, S., Washington, R., Coenen, N., & Elixhauser, A. (2017). Identifying Increased Risk of Readmission and In-hospital Mortality Using Hospital Administrative Data. Medical Care, 55(7), 698–705.

Quan Hude et al. (2005). Coding algorithms for defining comorbidities in ICD-9-CM and ICD-10 administrative data. Medical care, 1130-1139.

Thompson, N. R., Fan, Y., Dalton, J. E., Jehi, L., Rosenbaum, B. P., Vadera, S., & Griffith, S. D. (2015). A new Elixhauser-based comorbidity summary measure to predict in-hospital mortality. Med Care, 53(4), 374–379.

Walraven, C. Van, Austin, P. C., Jennings, A., Quan, H., Alan, J., Walraven, C. Van, ... Jennings, A. (2009). A Modification of the Elixhauser Comorbidity Measures Into a Point System for Hospital Death Using Administrative Data. Medical Care, 47(6), 626–633.

See Also

Other default classcodes: `ae`, `charlson`, `cps`, `hip_ae_hailer`, `rxriskv`

ex_atc *Example data for random ATC codes*

Description

Example data for fictive people to use for testing and in examples.

Usage

```
ex_atc
```

Format

Data frames with 100 rows and 2 variables:

name random person names

atc Random codes from the Anatomic Therapeutic Chemical classification (ATC) system.

prescription random dates of prescription of medications with corresponding ATC codes

See Also

Other example data: ex_icd10, ex_people

ex_icd10 *Example data for random codes assigned to random people*

Description

Example data for fictive ICD-10-diagnoses to use for testing and in examples.

Usage

```
ex_icd10
```

Format

Data frames with 1,000 rows and 4 variables:

id Random names corresponding to column name in dataset ex_people

date random dates corresponding to registered (comorbidity) codes

code ICD-10 codes from the uranium_pathology dataset in the icd.data package by Jack Wasey originating from the United States Transuranium and Uranium Registries, published in the public domain.

hdia boolean marker if corresponding code is the main diagnose of the hospital visit (randomly assigned to 10 percent of the codes)

Source

https://github.com/jackwasey/icd.data https://ustur.wsu.edu/about-us/

See Also

Other example data: ex_atc, ex_people

`ex_people` *Example data for random people*

Description

Example data for fictive people to use for testing and in examples.

Usage

```
ex_people
```

Format

Data frames with 100 rows and 2 variables:

name random person names

surgery random dates for a relevant event

See Also

Other example data: ex_atc, ex_icd10

`hip_ae_hailer` *Classcodes for infection and dislocation after hip arthroplasty*

Description

Classcodes for infection and dislocation after hip arthroplasty

Usage

```
hip_ae_hailer
```

Format

Data frame with 3 columns:

group Infection or dislocation

icd10 regular expressions based on ICD-10

kva regular expressions based on NOMESCO/KVA codes

See Also

ae

Other default classcodes: ae, charlson, cps, elixhauser, rxriskv

---

`index_fun` *Calculate index based on classification scheme*

Description

This is the third step of `codify() %>% classify() %>% index()`. The function takes classified case data and calculates (weighted) index sums as specified by weights from a `classcodes` object.

Usage

```
index(classified, ...)

## S3 method for class 'data.frame'
index(classified, ...)

## S3 method for class 'matrix'
index(classified, index = NULL, cc = NULL, ...)
```

Arguments

`classified`	output from `classify()`
`...`	used internally
`index`	name of column with 'weights' from corresponding `classcodes` object. Can be `NULL` if the index is just a unweighted count of all identified groups.
`cc`	`classcodes` object. Can be `NULL` if a `classcodes` object is already available as an attribute of `classified` (which is often the case) and/or if `index = NULL`.

Details

Index weights for subordinate hierarchical classes (as identified by `attr(cc, "hierarchy")`) are excluded in presence of superior classes if index specified with argument `index`.

Value

Named numeric index vector with names corresponding to `rownames(classified)`

See Also

Other verbs: `categorize()`, `classify()`, `codify()`

Examples

```
# Prepare some codified data with ICD-10 codes during 1 year (365 days)
# before surgery
x <-
  codify(
    ex_people,
    ex_icd10,
    id        = "name",
    code      = "icd10",
    date      = "surgery",
    days      = c(-365, 0),
    code_date = "admission"
  )

# Classify those patients by the Charlson comorbidity indices
cl <- classify(x, "charlson")

# Calculate (weighted) index values
head(index(cl))                  # Un-weighted sum/no of conditions for each patient
head(index(cl, "quan_original")) # Weighted index (Quan et al. 2005; see `?charlson`)
head(index(cl, "quan_updated"))  # Weighted index (Quan et al. 2011; see `?charlson`)

# Tabulate index for all patients.
# As expected, most patients are healthy and have index = 0/NA,
# where NA indicates no recorded hospital visits
# found in `ex_icd10` during codification.
# In practice, those patients might be assumed to have 0 comorbidity as well.
table(index(cl, "quan_original"), useNA = "always")

# If `cl` is a matrix without additional attributes (as imposed by `codify()`)
# an explicit classcodes object must be specified by the `cc` argument
cl2 <- as.matrix(cl)
head(index(cl2, cc = "charlson"))
```

`print.classcodes` *Print classcodes object*

Description

Print classcodes object

Usage

```
## S3 method for class 'classcodes'
print(x, n = NULL, ...)
```

Arguments

`x`	object of type classcodes

n	number of rows to preview (n = 0 is allowed)
...	arguments passed to print method for tibble

See Also

Other classcodes: all_classcodes(), as.data.frame.classified(), classcodes, codebook(), print.classified(), set_classcodes(), summary.classcodes(), visualize.classcodes()

Examples

```
# Default printing
elixhauser

# Print attributes data but no data preview
print(elixhauser, n = 0)

# Print all rows
print(elixhauser, n = 31)
```

---

print.classified *Printing classified data*

Description

Preview first n rows as tibble

Usage

```
## S3 method for class 'classified'
print(x, ...)
```

Arguments

x	output from classify()
...	additional arguments passed to printing method for a tibble. n is the number of rows to preview. Set n = NULL to disable the tibble preview and print the object as is (a matrix).

See Also

Other classcodes: all_classcodes(), as.data.frame.classified(), classcodes, codebook(), print.classcodes(), set_classcodes(), summary.classcodes(), visualize.classcodes()

Examples

```
# Preview all output
classify(c("C80", "I20", "unvalid_code"), "elixhauser")

# Preview only the first row
print(classify(c("C80", "I20", "unvalid_code"), "elixhauser"), n = 1)

# Print object as is (matrix)
print(classify(c("C80", "I20", "unvalid_code"), "elixhauser"), n = NULL)
```

`rxriskv` *Classcodes for RxRisk V based on ATC codes*

Description

Note that desired implementation might differ over time and by country.

Usage

```
rxriskv
```

Format

Data frames with 46 rows and 6 variables:

group medical condition

pratt ATC codes from table 1 in Pratt et al. 2018 (ignoring PBS item codes and extra conditions).

garland Modified version by Anne Garland to resemble medical use in Sweden 2016 (Unpublished).

caughey From appendix 1 in Caughey et al. 2010

pratt Mortality weights from table 1 in Pratt et al. 2018

sum_all Unweighted count of all conditions.

References

Caughey GE, Roughead EE, Vitry AI, McDermott RA, Shakib S, Gilbert AL. Comorbidity in the elderly with diabetes: Identification of areas of potential treatment conflicts. Diabetes Res Clin Pract 2010;87:385–93.

Pratt NL, Kerr M, Barratt JD, Kemp-Casey A, Kalisch Ellett LM, Ramsay E, et al. The validity of the Rx-Risk Comorbidity Index using medicines mapped to the Anatomical Therapeutic Chemical (ATC) Classification System. BMJ Open 2018;8.

See Also

Other default classcodes: `ae`, `charlson`, `cps`, `elixhauser`, `hip_ae_hailer`

set_classcodes *Set classcodes object*

Description

Prepare a classcodesobject by specifying the regular expressions to use for classification.

Usage

```
set_classcodes(
  cc,
  classified = NULL,
  regex = NULL,
  start = TRUE,
  stop = FALSE,
  tech_names = NULL
)
```

Arguments

cc	classcodes object (or name of a default object from all_classcodes()).
classified	object that classcodes could be inherited from
regex	name of column with regular expressions to use for classification. NULL (default) uses attr(obj, "regexpr")[1].
start, stop	should codes start/end with the specified regular expressions? If TRUE, column "regex" is prefixed/suffixed by ^/$.
tech_names	should technical column names be used? If FALSE, colnames are taken directly from group names of cc, if TRUE, these are changed to more technical names avoiding special characters and are prefixed by the name of the classification scheme. NULL (by default) preserves previous names if cc is inherited from classified (fall backs to FALSE if not already set).

Value

classcodes object.

See Also

Other classcodes: all_classcodes(), as.data.frame.classified(), classcodes, codebook(), print.classcodes(), print.classified(), summary.classcodes(), visualize.classcodes()

Examples

```
# Prepare a classcodes object for the Charlson comorbidity classification
# based on the default regular expressions
set_classcodes(charlson)   # by object
set_classcodes("charlson") # by name
```

```
# Same as above but based on regular expressions for ICD-8 (see `?charlson`)
set_classcodes(charlson, regex = "icd8_brusselaers")

# Only recognize codes if no other characters are found after the relevant codes
# Hence if the code vector stops with the code
set_classcodes(charlson, stop = TRUE)

# Accept code vectors with strings which do not necessarily start with the code.
# This is useful if the code might appear in the middle of a longer character
# string or if a common prefix is used for all codes.
set_classcodes(charlson, start = FALSE)

# Use technical names to clearly describe the origin of each group.
# Note that the `cc` argument must be specified by a character string
# since this name is used as part of the column names
x <- set_classcodes("charlson", tech_names = TRUE)
x$group
```

`summary.classcodes` *Summarizing a classcodes object*

Description

Classification schemes are formalized by regular expressions within the classcodes objects. These are computationally effective but sometimes hard to interpret. Use this function to list all codes identified for each group.

Usage

```
## S3 method for class 'classcodes'
summary(object, coding, ..., cc_args = list())

## S3 method for class 'summary.classcodes'
print(x, ...)
```

Arguments

`object`	classcodes object
`coding`	either a vector with codes from the original classification, or a name (character vector of length one) of a keyvalue object from package "decoder" (for example "icd10cm" or "atc")
`...`	• `summary.classcodes()`: ignored • `print.summary.classcodes()`: arguments passed to `tibble:::print.tbl()`
`cc_args`	List of named arguments passed to `set_classcodes()`
`x`	output from `summary.classcodes()`

Value

Methods primarily called for their side effects (printing to the screen) but with additional invisible objects returned:

- `summary.classcodes()`: list with input arguments `object` and `coding` unchanged, as well as a data frame (`summary`) with columns for groups identified (`group`); the number of codes to be recognized for each group (`n`) and individual codes within each group (`codes`).
- `print.summary.classcodes()`: argument `x` unchanged

See Also

Other classcodes: `all_classcodes()`, `as.data.frame.classified()`, `classcodes`, `codebook()`, `print.classcodes()`, `print.classified()`, `set_classcodes()`, `visualize.classcodes()`

Examples

```
# summary.classcodes() ------------------------------------------------

# Summarize all ICD-10-CM codes identified by the Elixhauser
# comorbidity classification
# See `?decoder::icd10cm` for details
summary(elixhauser, coding = "icd10cm")

# Is there a difference if instead considering the Swedish ICD-10-SE?
# See `?decoder::icd10se` for details
summary(elixhauser, coding = "icd10se")

# Which ICD-9-CM diagnostics codes are recognized by Charlson according to
# Brusselears et al. 2017 (see `?charlson`)
summary(
  charlson, coding = "icd9cmd",
  cc_args = list(regex = "icd9_brusselaers")
)


# print.summary.classcodes() ------------------------------------------

# Print all 31 lines of the summarized Elixhauser classcodes object
print(
  summary(elixhauser, coding = "icd10cm"),
  n = 31
)
```

`visualize.classcodes` *Visualize classification scheme in web browser*

Description

Groups from a classcodes object are visualized by their regular expressions in the default web browser. The visualization does not give any details on group names, conditions or weights but might be useful both for understanding of a classification scheme in use, and during the creation and debugging of such.

Usage

```
## S3 method for class 'classcodes'
visualize(x, group = NULL, show = TRUE, ...)
```

Arguments

x	classcodes object or name of such object included in the package (see all_classcodes()).
group	names (as character vector) of groups to visualize (subset of rownames(x)). (All groups if NULL.)
show	should a visualization be shown in the default web browser. Set to FALSE to just retrieve a URL for later use.
...	Arguments passed on to set_classcodes regex name of column with regular expressions to use for classification. NULL (default) uses attr(obj, "regexpr")[1].

Value

URL to website with visualization (invisible)

See Also

Other classcodes: all_classcodes(), as.data.frame.classified(), classcodes, codebook(), print.classcodes(), print.classified(), set_classcodes(), summary.classcodes()

Examples

```
# The default behavior is to open a visualization in the default web browser
## Not run:

 # How is depression classified according to Elixhauser?
 visualize("elixhauser", "depression")

 # Compare the two diabetes groups according to Charlson
 visualize("charlson",
   c("diabetes without complication", "diabetes complication"))

 # Is this different from the "Royal College of Surgeons classification?
 # Yes, there is only one group for diabetes
 visualize("charlson",
   c("diabetes without complication", "diabetes complication"),
   regex = "rcs"
 )
```

```
 # Show all groups from Charlson
 visualize("charlson")

 # It is also possible to visualize an arbitrary regular expression
 # from a character string
 visualize("I2([12]|52)")

## End(Not run)

 # The URL is always returned (invisable) but the visual display can
 # also be omitted
url <- visualize("hip_ae", show = FALSE)
url
```

Index