Package ‘icd’

May 31, 2020

Title Comorbidity Calculations and Tools for ICD-9 and ICD-10 Codes

Version 4.0.9

Maintainer Jack O. Wasey <jack@jackwasey.com>

Description Calculate comorbidities, medical risk scores, and work very quickly and precisely with ICD-9 and ICD-10 codes. This package enables a work flow from raw tables of ICD codes in hospital databases to comorbidities. ICD-9 and ICD-10 comorbidity mappings from Quan (Deyo and Elixhauser versions), Elixhauser and AHRQ included. Common ambiguities and code formats are handled. Comorbidity computation includes Hierarchical Condition Codes, and an implementation of AHRQ Clinical Classifications. Risk scores include those of Charlson and van Walraven. US Clinical Modification, Word Health Organization, Belgian and French ICD-10 codes are supported, most of which are downloaded on demand.

License GPL-3

URL https://jackwasey.github.io/icd/

BugReports https://github.com/jackwasey/icd/issues

Depends R (>= 3.4)

Imports methods, Rcpp (>= 0.12.3)

Suggests graphics, httr, jsonlite, knitr, magrittr, nhds, readxl, rmarkdown (>= 1.11), RODBC, roxygen2 (>= 5.0.0), stats, testthat (>= 0.11.1), utils, xml2

LinkingTo Rcpp (>= 0.12.3), RcppEigen

VignetteBuilder knitr

Biarch true

Classification/ACM-2012 Social and professional topics~Medical records, Applied computing~Health care information systems, Applied computing~Health informatics, Applied computing~Bioinformatics

Copyright See file (inst/)COPYRIGHTS

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

RoxygenNote 7.1.0

NeedsCompilation yes

Author Jack O. Wasey [aut, cre, cph] (<https://orcid.org/0000-0003-3738-4637>),
William Murphy [ctb] (Van Walraven scores),
Anobel Odisho [ctb] (CMS Hierarchical Condition Codes),
Vitaly Druker [ctb] (AHRQ CCS),
Ed Lee [ctb] (explain codes in table format),
Patrick McCormick [ctb] (Multiple great pull requests),
Alessandro Gasparini [ctb],
Michel Lang [aut] (backport of R_user-dir for R version >4. ORCID =
0000-0001-9754-0393),
R Core Team [aut, cph] (utils::askYesNo backport and m4 script for
OpenMP detection.)

Repository CRAN

Date/Publication 2020-05-31 02:00:07 UTC

R topics documented:

`as.comorbidity_map` *Set the class of a named list to show it is a comorbidity map.*

Description

Set the class of a named list to show it is a comorbidity map.

Usage

```
as.comorbidity_map(x)
```

Arguments

x	A list of depth one, with unique names, and elements that are character vectors.

See Also

Other ICD data types: `set_icd_class`, `wide_vs_long`

`as.decimal_diag` *Get or set whether ICD codes have have an attribute indicating 'short' or 'decimal' format*

Description

Get or set whether ICD codes have have an attribute indicating 'short' or 'decimal' format

Usage

```
as.decimal_diag(x, value = TRUE)

as.icd_decimal_diag(x, value = TRUE)

as.short_diag(x, value = TRUE)

as.icd_short_diag(x, value = TRUE)

is.decimal_diag(x)
is.icd_decimal_diag(x)

is.short_diag(x)

is.icd_short_diag(x)
```

Arguments

x	ICD data
value	TRUE or FALSE, default is TRUE which sets the attribute to whatever is indicated in the function name. See examples.

Getting the attribute

is.short_diag tests for presence of an attribute, not whether the code is a valid ICD code. To test validity of the codes themselves, see is_valid.

Setting the attribute

Similarly, as.icd_short_diag and as.icd_decimal_diag set the attribute, but do not convert the codes themselves. For conversion between 'short' and 'decimal' forms, use decimal_to_short and short_to_decimal.

The attribute icd_short_code should be either TRUE or FALSE. There is no attribute named icd_decimal_code. These functions set and get the attribute safely. If the attribute is not present, both is.icd_short_diag and is.icd_decimal_diag (or their synonyms is.short_diag and is.decimal_diag) will return FALSE.

Examples

```
library(icd)
as.icd_short_diag("6670")
as.icd_short_diag("667.0") # no warning or error!
is.icd_short_diag(decimal_to_short("667.0"))
decimal_type_code <- as.icd_short_diag("667.0", FALSE)
stopifnot(is.icd_decimal_diag(decimal_type_code))
codes <- as.icd9(c("100.1", "441.3"))
codes <- as.decimal_diag(codes)
codes
```

---

charlson *Calculate Charlson Comorbidity Index (Charlson Score)*

Description

Charlson score is calculated in the basis of the Quan revision of Deyo's ICD-9 mapping. (Peptic ulcer disease no longer warrants a point.) Quan published an updated set of scores, but it seems most people use the original scores for easier comparison between studies, even though Quan's were more predictive.

Usage

```
charlson(
  x,
  visit_name = NULL,
  scoring_system = c("original", "charlson", "quan"),
  return_df = FALSE,
  stringsAsFactors = getOption("stringsAsFactors"),
  ...
)

## S3 method for class 'data.frame'
charlson(
  x,
  visit_name = NULL,
  scoring_system = c("original", "charlson", "quan"),
  return_df = FALSE,
  stringsAsFactors = getOption("stringsAsFactors"),
  ...
)
```

Arguments

`x`	data frame containing a column of visit or patient identifiers, and a column of ICD-9 codes. It may have other columns which will be ignored. By default, the first column is the patient identifier and is not counted. If `visit_name` is not specified, the first column is used.
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`scoring_system`	One of `original`, `charlson`, or `quan`. The first two will give the original Charlson weights for each comorbidity, whereas `quan` uses the updated weights from Quan 2011.
`return_df`	single logical value, if `TRUE`, a two column data frame will be returned, with the first column named as in input data frame (i.e., `visit_name`), containing all the visits, and the second column containing the Charlson Comorbidity Index.
`stringsAsFactors`	single logical, passed on when constructing data.frame if `return_df` is `TRUE`. If the input data frame `x` has a factor for the `visit_name`, this is not changed, but a non-factor `visit_name` may be converted or not converted according to your system default or this setting.
`...`	further arguments to pass on to `icd9_comorbid_quan_deyo`, e.g. `name`

Details

When used, hierarchy is applied per Quan, “The following comorbid conditions were mutually exclusive: diabetes with chronic complications and diabetes without chronic complications; mild liver disease and moderate or severe liver disease; and any malignancy and metastatic solid tumor.” The Quan scoring weights come from the 2011 paper. The comorbidity weights were recalculated using updated discharge data, and some changes, such as Myocardial Infarction decreasing from 1 to 0, may reflect improved outcomes due to advances in treatment since the original weights were determined in 1984.

Methods (by class)

- `data.frame`: Charlson scores from data frame of visits and ICD-9 codes. ICD-10 Charlson can be calculated simply by getting the Charlson (e.g. Quan Deyo) comorbidities, then calling `charlson_from_comorbid`.

Examples

```
mydf <- data.frame(
  visit_name = c("a", "b", "c"),
  icd9 = c("441", "412.93", "042")
)
charlson(mydf)
cmb <- icd9_comorbid_quan_deyo(mydf)
cmb
# can specify short_code directly instead of guessing
charlson(mydf, short_code = FALSE, return_df = TRUE)
charlson_from_comorbid(cmb)
```

`charlson_from_comorbid` *Calculate Charlson scores from precomputed Charlson comorbidities*

Description

Calculate Charlson scores from precomputed Charlson comorbidities, instead of directly from the ICD codes. This is useful if the comorbidity calculation is time consuming. Commonly, both the Charlson comorbidities and the Charlson scores will be calculated, and this function provides just that second step.

Usage

```
charlson_from_comorbid(
  x,
  visit_name = NULL,
  hierarchy = FALSE,
  scoring_system = c("original", "charlson", "quan")
)
```

Arguments

`x`	data.frame or matrix, typically the output of a comorbidity calculation which uses the Charlson categories, e.g. `comorbid_quan_deyo`
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`hierarchy`	single logical value, default is `FALSE`. If `TRUE`, will drop `DM` if `DMcx` is present, etc.
`scoring_system`	One of `original`, `charlson`, or `quan`. The first two will give the original Charlson weights for each comorbidity, whereas `quan` uses the updated weights from Quan 2011.

`children` *Get children of ICD codes*

Description

Expand ICD codes to all possible sub-codes, optionally limiting to those codes which are *defined* or *billable* (leaf nodes).

Usage

```
children(x, ...)

## S3 method for class 'character'
children(x, ...)

## S3 method for class 'icd9cm'
children(x, short_code = guess_short(x), defined = TRUE, billable = FALSE, ...)

## S3 method for class 'icd9'
children(x, short_code = guess_short(x), defined = TRUE, billable = FALSE, ...)

## S3 method for class 'icd10'
children(x, short_code = guess_short(x), defined, billable = FALSE, ...)

## S3 method for class 'icd10cm'
children(x, short_code = guess_short(x), defined, billable = FALSE, ...)
```

```
## S3 method for class 'icd10who'
children(
  x,
  short_code = guess_short(x),
  defined,
  billable = NULL,
  leaf = NULL,
  ...
)
```

Arguments

`x`	data, e.g. character vector of ICD codes.
`...`	arguments passed on to other functions
`short_code`	single logical value which determines whether the ICD-9 code provided is in short (`TRUE`) or decimal (`FALSE`) form. Where reasonable, this is guessed from the input data.
`defined`	single logical value, whether returned codes should only include those which have definitions. Definition is based on the ICD version being used, e.g. ICD-9-CM, the WHO version of ICD-10, or other.
`billable`	single logical value, identical to 'leaf'. Leaf is preferred as most adaptations of WHO ICD codes are not oriented around money.
`leaf`	single logical value, whether to limit return codes also by whether they are billable, i.e. leaf nodes. This is really only designed for use with ICD-9-CM, ICD-10-CM etc, since the WHO versions are not designed for billing, but for public health and death reporting.

Value

Returns a vector of ICD codes, with class of `character` and the class of the identified or specified ICD code, e.g. `icd9`

Methods (by class)

- `character`: Get child codes, guessing ICD version and short versus decimal format
- `icd9cm`: Get children of ICD-9-CM codes
- `icd9`: Get children of ICD-9 codes, based on the super-set ICD-9-CM at present
- `icd10`: Get children of ICD-10 codes (warns because this only applies to ICD-10-CM for now).
- `icd10cm`: Get children of ICD-10-CM codes
- `icd10who`: Get children of ICD-10-CM codes

See Also

Other ICD-9 ranges: `condense()`, `expand_range()`

Examples

```
# N.b. magrittr not required by icd
library(magrittr, warn.conflicts = FALSE, quietly = TRUE)
# no children other than self
children("10201", short_code = TRUE, defined = FALSE)

# guess it was ICD-9 and a short, not decimal code
children("0032")

# empty because 102.01 is not meaningful
children("10201", short_code = TRUE, defined = TRUE)
x <- children("003", short_code = TRUE, defined = TRUE)
explain_code(x, condense = FALSE, short_code = TRUE)

children(short_code = FALSE, "100.0")
children(short_code = FALSE, "100.00")
children(short_code = FALSE, "2.34")
```

---

combine *Combine ICD codes*

Description

These function implement ICD specific methods for c, i.e., combinations of lists or vectors of codes, while preserving ICD classes. Base R c just drops all user defined classes and casts down to lowest common denominator, e.g. if mixing numbers and characters. No attempt here to catch all possible combinations of feeding in mixed ICD types and other types. Let R do what it normally does, but just try to keep classes of the first item in the list.

Usage

```
## S3 method for class 'icd9'
c(..., warn = FALSE)

## S3 method for class 'icd10'
c(..., warn = FALSE)
```

Arguments

...	elements to combine
warn	single logical value, if TRUE, will give warnings when incompatible types are combined using c

See Also

ICD data types

Examples

```
# Care with the following:
c(as.icd9("E998"), as.icd10("A10"))
# which results in both codes sharing the 'icd9' class.
# ICD-10 codes
(a <- as.icd10("A100SSX"))
(b <- as.icd10("Z999A"))
c(a, b)
c(as.icd_short_diag(a), as.icd_short_diag(b))
(d <- as.icd10("A10.0SSX"))
(e <- as.icd10("Z99.9A"))
c(d, e)
c(as.icd_decimal_diag(d), as.icd_decimal_diag(e))
# warn when mixing attribute types
suppressWarnings(
  c(as.icd_short_diag(a), as.icd_decimal_diag(e))
)
```

---

comorbid *Find comorbidities from ICD-9 codes.*

Description

This is the main function which extracts comorbidities from a set of ICD-9 codes. Some comorbidity schemes have rules, for example, what to do when both 'hypertension' and 'hypertension with complications' are present. These rules are applied by default; if the exact fields from the original mappings are needed, use `hierarchy = FALSE`. For comorbidity counting, Charlson or Van Walraven scores the default should be used to apply the rules. For more about computing Hierarchical Condition Codes (HCC), see `comorbid_hcc` For more about comorbidities following the Clinical Classification Software (CCS) rules from AHRQ, see `comorbid_ccs`.

Usage

```
comorbid(
  x,
  map,
  visit_name = NULL,
  icd_name = NULL,
  short_code = guess_short(x, icd_name = icd_name),
  short_map = guess_short(map),
  return_df = FALSE,
  return_binary = FALSE,
  categorize_fun = categorize_simple,
  ...
)

icd10_comorbid(
  x,
  map,
  visit_name = NULL,
  icd_name = NULL,
  short_code = NULL,
  short_map = guess_short(map),
  return_df = FALSE,
  return_binary = FALSE,
  icd10_comorbid_fun = icd10_comorbid_reduce,
  ...
)

icd9_comorbid(
  x,
  map,
  visit_name = NULL,
  icd_name = NULL,
  short_code = guess_short(x, icd_name = icd_name),
  short_map = guess_short(map),
  return_df = FALSE,
  return_binary = FALSE,
  preclean = FALSE,
  categorize_fun = categorize_simple,
  comorbid_fun = comorbid_mat_mul_wide,
  ...
)

icd9_comorbid_ahrq(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd10_comorbid_ahrq(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd9_comorbid_elix(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd10_comorbid_elix(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd9_comorbid_quan_elix(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd10_comorbid_quan_elix(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd9_comorbid_quan_deyo(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd10_comorbid_quan_deyo(x, ..., abbrev_names = TRUE, hierarchy = TRUE)

icd9_comorbid_charlson(...)

icd10_comorbid_charlson(...)

comorbid_ccs(x, icd_name = get_icd_name(x), ...)
```

```
icd9_comorbid_ccs(
  x,
  ...,
  single = TRUE,
  lvl = NULL,
  map = icd::icd9_map_single_ccs,
  short_map = TRUE
)

icd10_comorbid_ccs(x, ..., single = TRUE, lvl = NULL)

comorbid_ahrq(x, ...)

comorbid_elix(x, ...)

comorbid_quan_elix(x, ...)

comorbid_quan_deyo(x, ...)

comorbid_charlson(...)

comorbid_pccc_dx(
  x,
  visit_name = get_visit_name(x),
  icd_name = get_icd_name(x),
  short_code = guess_short(x, icd_name = icd_name),
  return_df = FALSE,
  return_binary = FALSE,
  ...
)

comorbid_pccc_pcs(
  x,
  visit_name = get_visit_name(x),
  icd_name,
  return_df = FALSE,
  return_binary = FALSE,
  ...
)

icd9_comorbid_pccc_dx(
  x,
  visit_name = NULL,
  icd_name = NULL,
  short_code = guess_short(x, icd_name = icd_name),
  return_df = FALSE,
  return_binary = FALSE,
  ...,
  abbrev_names = TRUE
)

icd10_comorbid_pccc_dx(
  x,
  visit_name = NULL,
  icd_name = NULL,
  short_code = guess_short(x, icd_name = icd_name),
  return_df = FALSE,
  return_binary = FALSE,
  ...,
  abbrev_names = TRUE
)

icd9_comorbid_pccc_pcs(
  x,
  visit_name = get_visit_name(x),
  icd_name = get_icd_pc_name(x),
  return_df = FALSE,
  return_binary = FALSE,
  ...,
  abbrev_names = TRUE
)

icd10_comorbid_pccc_pcs(
  x,
  visit_name = get_visit_name(x),
  icd_name,
  return_df = FALSE,
  return_binary = FALSE,
  ...,
  abbrev_names = TRUE
)
```

Arguments

x	`data.frame` containing a column of patient-visit identifiers and a column of ICD codes. The `data.frame` may be in 'long' or 'wide' format, like the example `vermont_dx` and `uranium_pathology` data.
map	A named list of the comorbidities with each list item containing a vector of decimal ICD-9 codes. **icd** includes a number of these, e.g., `icd9_map_elix`. Alternatively, this can be omitted if the convenience functions, such as `icd10_comorbid_charlson` are used directly. `map` should be in the form of a list, with the names of the items corresponding to the comorbidities (e.g. 'HTN', or 'diabetes') and the contents of each list item being a character vector of short-form (no decimal place, zero left-padded) ICD codes. There is no default: the user should use the family of functions, e.g. `comorbid_ahrq`, since these also name the fields correctly, and these functions also apply any hierarchical rules (see `hierarchy` below)

`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`icd_name`	The name of the column in the `data.frame` which contains the ICD codes. This is a character vector of length one. If it is `NULL`, `icd9` will attempt to guess the column name, looking for progressively less likely possibilities until it matches a single column. Failing this, it will take the first column in the data frame. Specifying the column using this argument avoids the guesswork.
`short_code`	single logical value which determines whether the ICD-9 code provided is in short (`TRUE`) or decimal (`FALSE`) form. Where reasonable, this is guessed from the input data.
`short_map`	Same as short, but applied to `map` instead of the data frame of ICD codes, `x`. All the codes in a mapping should be of the same type, i.e. short or decimal.
`return_df`	single logical value, if `TRUE`, return 'tidy' data, i.e., the result is a data frame with the first column being the `visit_id`, and the second being the count. If `visit_id` was a factor or named differently in the input, this is preserved.
`return_binary`	Single logical value, if `TRUE`, the returned `matrix` or `data.frame` will be composed of `1` and `0`, instead of `TRUE` and `FALSE`, respectively. This conversion can also be done by the internal functions `icd:::logical_to_binary` and `icd:::binary_to_logical`, or using other tools, e.g. `apply(x,2,as.integer)`
`categorize_fun`	Internal. Function used for the categorization problem.
`...`	Arguments passed through to `comorbid`, and ultimately `categorize_simple`, e.g. `restore_id_order`.
`icd10_comorbid_fun`	Internal function Default will be fast and accurate. A function which calculates comorbidities for ICD-10 codes, in which the comorbidity map only specifies parent codes, not every possible child.
`preclean`	single logical value, which, if `TRUE` causes ICD-9 'short' code input to be padded to correct three (or four for E code) length before applying the comorbidity map. For very large data sets, e.g. ten million rows, this is much slower than the comorbidity calculation. If you know that the source ICD-9 codes are already well formed (or have already run `icd9_add_leading_zeroes`), then `preclean` can be set to `FALSE` to save time.
`comorbid_fun`	Internal. Function used inside categorization.
`abbrev_names`	single logical value that defaults to `TRUE`, in which case the shorter human-readable names stored in e.g. `ahrqComorbidNamesAbbrev` are applied to the data frame column names.
`hierarchy`	single logical value that defaults to `TRUE`, in which case the hierarchy defined for the mapping is applied. E.g. in Elixhauser, you can't have uncomplicated and complicated diabetes both flagged.

`single`	a logical value, if `TRUE` then use single level CCS, otherwise use multi level
`lvl`	If multiple level CCS, then level must be selected as a number between one and four.

Details

The order of visits may change depending on the original sequence, and the underlying algorithm used. Usually this would be the order of the first occurrence of each visit/patient identifier, but this is not guaranteed unless `restore_id_order` is set to `TRUE`.

`data.frames` of patient data may have columns within them which are of class `icd9`, `icd10` etc., but do not themselves have a class: therefore, the S3 mechanism for dispatch is not suitable. I may add a wrapper function which looks inside a `data.frame` of comorbidities, and dispatches to the appropriate function, but right now the user must call the `icd9_` or `icd10_` prefixed function directly.

Functions

- `icd10_comorbid`: ICD-10 comorbidities
- `icd9_comorbid`: Get comorbidities from `data.frame` of ICD-9 codes
- `icd9_comorbid_ahrq`: AHRQ comorbidities for ICD-9 codes
- `icd10_comorbid_ahrq`: AHRQ comorbidities for ICD-10 codes
- `icd9_comorbid_elix`: Elixhauser comorbidities for ICD-9 codes
- `icd10_comorbid_elix`: Elixhauser comorbidities for ICD-10 codes
- `icd9_comorbid_quan_elix`: Quan's Elixhauser comorbidities for ICD-9 codes
- `icd10_comorbid_quan_elix`: Quan's Elixhauser comorbidities for ICD-10 codes
- `icd9_comorbid_quan_deyo`: Quan's Deyo (Charlson) comorbidities for ICD-9 codes
- `icd10_comorbid_quan_deyo`: Quan's Deyo (Charlson) comorbidities for ICD-10 codes
- `icd9_comorbid_charlson`: Currently synonym for `icd9_comorbid_quan_deyo`
- `icd10_comorbid_charlson`: Currently synonym for `icd10_comorbid_quan_deyo`
- `comorbid_ccs`: Use AHRQ CCS for comorbidity classification
- `icd9_comorbid_ccs`: Compute AHRQ Clinical Classifications Software (CCS) scores from ICD-9 codes
- `icd10_comorbid_ccs`: Compute AHRQ Clinical Classifications Software (CCS) scores from ICD-10 codes
- `comorbid_ahrq`: AHRQ comorbidities, infers whether to use ICD-9 or ICD-10 codes
- `comorbid_elix`: Elixhauser comorbidities, infers whether to use ICD-9 or ICD-10 codes
- `comorbid_quan_elix`: Quan's Elixhauser comorbidities, infers whether to use ICD-9 or ICD-10 codes
- `comorbid_quan_deyo`: Quan's Deyo (Charlson) comorbidities, infers whether to use ICD-9 or ICD-10 codes
- `comorbid_charlson`: Calculate comorbidities using Charlson categories according to Quan/Deyo ICD categories. Synonymous with `link{comorbid_quan_deyo}` in this release.

- comorbid_pccc_dx: Calculate pediatric complex chronic conditions (PCCC) comorbidities
  Unlike with ICD-9 and ICD-10 diagnostic codes, 'icd' doesn't currently have a method for guessing which fields are procedure codes, so icd_name must be specified for the _pcs functions.
- comorbid_pccc_pcs: Calculate the PCCC comorbidities based on procedure codes,
- icd9_comorbid_pccc_dx: Calculate PCCC comorbidities from ICD-9 diagnosis codes
- icd10_comorbid_pccc_dx: Calculate PCCC comorbidities from ICD-10 diagnosis codes
- icd9_comorbid_pccc_pcs: Calculate PCCC comorbidities from ICD-9 procedure codes
- icd10_comorbid_pccc_pcs: Calculate PCCC comorbidities from ICD-10 procedure codes

See Also

icd9_map_single_ccs

Consider using comorbid_ahrq instead of comorbid_elix for more recently updated mappings based on the Elixhauser scheme.

Other comorbidity computations: comorbid_hcc()

Other comorbidities: comorbid_hcc(), icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

Other comorbidity computations: comorbid_hcc()

Other comorbidities: comorbid_hcc(), icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

Examples

```
library(icd)
vermont_dx[1:5, 1:10]
# get first few rows and columns of Charlson comorbidities using Quan/Deyo
# mapping of ICD-9 or ICD-10 codes Charlson categories
comorbid_quan_deyo(vermont_dx)[1:5, 1:14]

# Note that the comorbidity calculations automatically finds the ICD code
# columns, and uses 'wide' or 'long' format data.

stopifnot(
  identical(
    comorbid_quan_deyo(vermont_dx),
    comorbid_quan_deyo(wide_to_long(vermont_dx))
  )
)

# get summary AHRQ (based on Elixhauser) comorbidities for the Uranium data:
summary(comorbid_ahrq(uranium_pathology))

pts <- icd_long_data(
  visit_name = c("2", "1", "2", "3", "3"),
  icd9 = c("39891", "40110", "09322", "41514", "39891")
)
comorbid(pts, icd9_map_ahrq, short_code = TRUE) # visit_name is now sorted
```

```
pts <- icd_long_data(
  visit_name = c("1", "2", "3", "4", "4"),
  icd_name = c("20084", "1742", "30410", "41514", "95893"),
  date = as.Date(c(
    "2011-01-01", "2011-01-02", "2011-01-03",
    "2011-01-04", "2011-01-04"
  ))
)

pt_hccs <- comorbid_hcc(pts, date_name = "date")
head(pt_hccs)

pts10 <- icd_long_data(
  visit_name = c("a", "b", "c", "d", "e"),
  icd_name = c("I058", NA, "T82817A", "", "I69369"),
  date = as.Date(
    c("2011-01-01", "2011-01-02", "2011-01-03", "2011-01-03", "2011-01-03")
  )
)

icd10_comorbid(pts10, map = icd10_map_ahrq)
# or if library(icd) hasn't been called first:
icd::icd10_comorbid(pts10, map = icd::icd10_map_ahrq)
# or most simply:
icd::icd10_comorbid_ahrq(pts10)

# specify a simple custom comorbidity map:
my_map <- list(
  "malady" = c("100", "2000"),
  "ailment" = c("003", "040")
)
two_pts <- data.frame(
  visit_id = c("v01", "v01", "v02", "v02"),
  icd9 = as.icd9(c("040", "000", "100", "000")),
  stringsAsFactors = FALSE
)
comorbid(two_pts, map = my_map)
# not pediatric data, but let's look for this example
head(icd9_comorbid_pccc_dx(vermont_dx))
# Six random codes from each PCCC procedure code map. 'icd' will use
# an heuristic to guess whether ICD-9 or ICD-10:
pts <- data.frame(
  encounters = c(10, 11, 12),
  icd9_pcs = c("0152", "304", "0050"),
  icd10_pcs = c("0B110Z4", "02YA0Z2", "031209D")
)
comorbid_pccc_pcs(pts, icd_name = "icd9_pcs", return_binary = TRUE)
comorbid_pccc_pcs(pts, icd_name = "icd10_pcs", return_binary = TRUE)

# All ICD-9 procedure codes are numeric, some ICD-10 procedure codes
# are numeric, so best to call functions directly:
pts <- data.frame(encounters = c(100), icd10_pcs = c("0016070"))
icd10_comorbid_pccc_pcs(pts, icd_name = "icd10_pcs")
```

comorbid_df_to_mat *convert comorbidity matrix to data frame*

Description

convert matrix of comorbidities into data frame, preserving visit_name information

Usage

```
comorbid_df_to_mat(
  x,
  visit_name = get_visit_name(x),
  stringsAsFactors = getOption("stringsAsFactors")
)
```

Arguments

`x`	data frame, with a `visit_name` column (not necessarily first), and other columns with flags for comorbidities, as such column names are required.
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`stringsAsFactors`	Single logical value, describing whether the resulting data frame should have strings, e.g. `visit_id` converted to factor. Default is to follow the current session option. This is identical to the argument used in, among other base functions `as.data.frame`.

See Also

`comorbid_mat_to_df`

Other ICD data conversion: `comorbid_mat_to_df()`, `convert`, `decimal_to_short()`, `long_to_wide()`, `short_to_decimal()`, `wide_to_long()`

Examples

```
longdf <- icd_long_data(
  visit = c("a", "b", "b", "c"),
  icd9 = c("441", "4240", "443", "441")
)
cmbdf <- icd9_comorbid_elix(longdf, return_df = TRUE)
```

```
class(cmbdf)
rownames(cmbdf)
mat.out <- comorbid_df_to_mat(cmbdf)
stopifnot(is.matrix(mat.out))
mat.out[, 1:4]
```

---

comorbid_hcc *Get Hierarchical Condition Codes (HCC)*

Description

Applying CMS Hierarchical Condition Categories `comorbid_hcc` works differently from the rest of the comorbidity assignment functions. This is because CMS publishes a detailed ICD to Condition Category mapping including all child ICD codes. While these mappings were the same for 2007-2012, after 2013 there are annual versions, so date must be taken into consideration. Also, there is a many:many linkage between ICD and Condition Categories (CC). Once CCs are assigned, a series of hierarchy rules (which can also change annually) are applied to create the HCCs.

Usage

```
comorbid_hcc(
  x,
  date_name = "date",
  visit_name = get_visit_name(x),
  icd_name = get_icd_name(x)
)

icd9_comorbid_hcc(x, date_name = "date", visit_name = NULL, icd_name = NULL)

icd10_comorbid_hcc(x, date_name = "date", visit_name = NULL, icd_name = NULL)
```

Arguments

`x`	data frame with columns for patient/visit ID, ICD code and date
`date_name`	the name of the column representing the date of each record. Needed because each year there is a different ICD9/10 to CC mapping). Default value is 'date'.
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.

`icd_name`	The name of the column in the `data.frame` which contains the ICD codes. This is a character vector of length one. If it is `NULL`, `icd9` will attempt to guess the column name, looking for progressively less likely possibilities until it matches a single column. Failing this, it will take the first column in the data frame. Specifying the column using this argument avoids the guesswork.

Functions

- `icd9_comorbid_hcc`: Get HCCs from a data frame of ICD-9 codes
- `icd10_comorbid_hcc`: Get HCCs from a data frame of ICD-10 codes

See Also

Other comorbidity computations: `comorbid()`

Other comorbidities: `comorbid()`, `icd10_map_ahrq_pcs`, `icd9_map_ahrq`, `icd9_map_elix`, `icd9_map_hcc`, `icd9_map_pccc`, `icd9_map_quan_deyo`, `icd9_map_quan_elix`, `icd9_map_single_ccs`

`comorbid_mat_to_df` *convert comorbidity data frame from matrix*

Description

convert matrix of comorbidities into data frame, preserving visit_name information

Usage

```
comorbid_mat_to_df(
  x,
  visit_name = "visit_id",
  stringsAsFactors = getOption("stringsAsFactors")
)
```

Arguments

`x`	Matrix of comorbidities, with row and columns names defined
`visit_name`	Single character string with name for new column in output data frame. Everywhere else, `visit_name` describes the input data, but here it is for output data.
`stringsAsFactors`	Single logical value, describing whether the resulting data frame should have strings, e.g. `visit_id` converted to factor. Default is to follow the current session option. This is identical to the argument used in, among other base functions `as.data.frame`.

See Also

comorbid_df_to_mat

Other ICD data conversion: comorbid_df_to_mat(), convert, decimal_to_short(), long_to_wide(), short_to_decimal(), wide_to_long()

Examples

```
longdf <- icd_long_data(
  visit_id = c("a", "b", "b", "c"),
  icd9 = as.icd9(c("441", "4240", "443", "441"))
)
mat <- icd9_comorbid_elix(longdf)
class(mat)
typeof(mat)
rownames(mat)
df.out <- comorbid_mat_to_df(mat)
stopifnot(is.data.frame(df.out))
# output data frame has a factor for the visit_name column
stopifnot(identical(rownames(mat), as.character(df.out[["visit_id"]])))
df.out[, 1:4]
# when creating a data frame like this, stringsAsFactors uses
# the system-wide option you may have set e.g. with
# options("stringsAsFactors" = FALSE).
is.factor(df.out[["visit_id"]])
```

`convert` *Convert ICD data between formats and structures.*

Description

ICD codes are represented in *short* and *decimal* forms. The short form has up to 5 digits, or V or E followed by up to four digits. The decimal form has a decimal point to delimit the top-level (henceforth *major*) category, and the *minor* part containing the subsidiary classifications.

Details

For conversions of ICD-9 or ICD-10 codes between the *short* and *decimal* forms, use short_to_decimal and decimal_to_short.

icd does not covert ICD-9 to ICD-10 codes yet.

See Also

Other ICD data conversion: comorbid_df_to_mat(), comorbid_mat_to_df(), decimal_to_short(), long_to_wide(), short_to_decimal(), wide_to_long()

`count_codes` *Count ICD codes or comorbidities for each patient*

Description

`count_codes` takes a data frame with a column for `visit_name` and another for ICD-9 code, and returns the number of distinct codes for each patient.

Usage

```
count_codes(x, visit_name = get_visit_name(x), return_df = FALSE)
```

Arguments

`x`	data frame with one row per patient, and a true/false or 1/0 flag for each column. By default, the first column is the patient identifier and is not counted. If `visit_name` is not specified, the first column is used.
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`return_df`	single logical, if `TRUE`, return the result as a data frame with the first column being the `visit_name`, and the second being the count. If `visit_name` was a factor or named differently in the input, this is preserved.

Details

The `visit_name` field is typically the first column. If there is no column called `visit_name` and `visit_name` is not specified, the first column is used.

Value

vector of the count of comorbidities for each patient. This is sometimes used as a metric of comorbidity load, instead of, or in addition to metrics like the Charlson Comorbidity Index (aka Charlson Score)

Examples

```
mydf <- data.frame(
  visit_name = c("r", "r", "s"),
  icd9 = c("441", "412.93", "042")
)
```

```
count_codes(mydf, return_df = TRUE)
count_codes(mydf)

cmb <- icd9_comorbid_quan_deyo(mydf, short_code = FALSE, return_df = TRUE)
count_comorbid(cmb)

wide <- data.frame(
  visit_name = c("r", "s", "t"),
  icd9_1 = c("0011", "441", "456"),
  icd9_2 = c(NA, "442", NA),
  icd9_3 = c(NA, NA, "510")
)
count_codes_wide(wide)
# or:
## Not run:
library(magrittr, warn.conflicts = FALSE)
wide %>%
  wide_to_long() %>%
  count_codes()

## End(Not run)
```

---

count_codes_wide — *Count ICD codes given in wide format*

Description

For count_codes, it is assumed that all the columns apart from visit_name represent actual or possible ICD-9 codes. Duplicate visit_names are repeated as given and aggregated.

Usage

```
count_codes_wide(
  x,
  visit_name = get_visit_name(x),
  return_df = FALSE,
  aggr = FALSE
)
```

Arguments

x	data.frame with one row per patient, hospital visit, encounter, etc., and multiple columns containing any ICD codes attributed to that encounter or patient. i.e. data frame with ICD codes in wide format.
visit_name	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or NULL, then an attempt is made to guess which field has the ID for

	the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`return_df`	single logical value, if `TRUE`, return 'tidy' data, i.e., the result is a data frame with the first column being the `visit_id`, and the second being the count. If `visit_id` was a factor or named differently in the input, this is preserved.
`aggr`	single logical, default is `FALSE`. If `TRUE`, the length (or rows) of the output will no longer match the input, but duplicate `visit_names` will be counted together.

`count_comorbid` *Count number of comorbidities per patient*

Description

`count_comorbid` differs from the other counting functions in that it counts *comorbidities*, not individual diagnoses. It accepts any `data.frame` with either logical or binary contents, with a single column for visit_name. No checks are made to see whether visit_name is duplicated.

Usage

```
count_comorbid(x, visit_name = get_visit_name(x), return_df = FALSE)
```

Arguments

`x`	data frame with one row per patient, and a true/false or 1/0 flag for each column. By default, the first column is the patient identifier and is not counted. If `visit_name` is not specified, the first column is used.
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`return_df`	single logical value, if `TRUE`, return 'tidy' data, i.e., the result is a data frame with the first column being the `visit_id`, and the second being the count. If `visit_id` was a factor or named differently in the input, this is preserved.

diff_comorbid — *show the difference between two comorbidity mappings*

Description

Compares two comorbidity to ICD code mappings. The results are returned invisibly as a list. Only those comorbidities with (case sensitive) overlapping names are compared.

Usage

```
diff_comorbid(
  x,
  y,
  all_names = NULL,
  x_names = NULL,
  y_names = NULL,
  show = TRUE,
  explain = TRUE
)

## S3 method for class 'list'
diff_comorbid(
  x,
  y,
  all_names = NULL,
  x_names = NULL,
  y_names = NULL,
  show = TRUE,
  explain = TRUE
)
```

Arguments

`x`	list of character vectors
`y`	list of character vectors
`all_names`	character vector of the comorbidity names
`x_names`	character vector of the comorbidity names from `x` to compare
`y_names`	character vector of the comorbidity names from `y` to compare
`show`	single logical value. The default is `TRUE` which causes a report to be printed.
`explain`	single logical value. The default is `TRUE` which means the differing codes are attempted to be reduced to their parent codes, in order to give a more succinct summary.

Value

A list, each item of which is another list containing the intersections and both asymmetric differences.

Methods (by class)

- `list`: Show difference between comorbidity maps with ICD-9 codes

Examples

```
# compare CHF for ICD-10 mappings from Elixhauser and AHRQ
diff_comorbid(icd10_map_elix, icd10_map_ahrq, show = FALSE)[["CHF"]]
## Not run:
# default is to show the results in a human readable manner:
diff_result <- diff_comorbid(icd9_map_elix, icd9_map_ahrq)[["CHF"]]
# show differences for
# give full report on all comorbidities for these mappings
diff_result <- diff_comorbid(icd9_map_elix, icd9_map_ahrq, show = FALSE)

# the following outputs a summary to the console:
diff_comorbid(icd9_map_elix, icd9_map_ahrq)

## End(Not run)
```

`download_all_icd_data` *Download all the additional data at once*

Description

It will download and parse WHO ICD-10, French, and Belgian codes and descriptions. It will also get years 2014, 2015, 2017, and 2018 for ICD-10-CM (diagnostic codes), and 2014–2019 procedure codes. 2016 and 2019 diagnostic codes are included in the package data. The total amount of data is about 340Mb. It is not necessary to do call `download_all_icd_data` for normal use: you may simply call the functions like `get_icd10cm2014`, which will download data when needed.

Usage

```
download_all_icd_data()
```

See Also

`set_icd_data_dir`

Examples

```
## Not run:
# set_icd_data_dir()
# set_icd_data_dir("/tmp/icd")

# The following would download, and make all the known ICD data available
# download_all_icd_data()

## End(Not run)
```

`filter_poa` *Filters data frame based on present-on-arrival flag*

Description

Present On Arrival (POA) is not a simple flag, since many codes are exempt, unspecified, or unknown. Therefore, two options are given: get all the comorbidities where the POA flag was definitely negative, coded as 'N' or definitely positive and coded as 'Y'. Negating one set won't give the other set unless all codes were either Y or N.

Usage

```
filter_poa(x, poa_name = "poa", poa = poa_choices)

filter_poa_yes(x, poa_name = "poa")

filter_poa_no(x, poa_name = "poa")

filter_poa_not_no(x, poa_name = "poa")

filter_poa_not_yes(x, poa_name = "poa")
```

Arguments

`x`	input vector of ICD codes
`poa_name`	The name of column in the data frame which contains the Present On Arrival (POA) flag. The flag itself is a single character, typically one of 'Y', 'N', 'E', 'X', 'U', or empty.
`poa`	single character value, being one of `Yes`, `No`, `NotYes`, and `NotNo`, indicating whether to account for comorbidities flagged as present-on-arrival. This is not a simple flag, because many codes are exempt, unspecified, or unknown. The intermediate codes, such as 'exempt', 'unknown' and `NA` mean that 'yes' is not the same as 'not no'.

Functions

- `filter_poa_yes`: Select rows where Present-on-Arrival flag is explicitly 'Yes.'
- `filter_poa_no`: Select rows where Present-on-Arrival flag is explicitly 'No.'
- `filter_poa_not_no`: Select rows where Present-on-Arrival flag is anything but 'No.' This includes unknown, exempt, other codes, and of course all those marked 'Yes.'
- `filter_poa_not_yes`: Select rows where Present-on-Arrival flag is anything but 'Yes.' This would group exempt, unknown and other codes under 'Not POA' which is unlikely to be a good choice, since exempt codes, of which there are a quite large number, tend to describe chronic or out-of-hospital characteristics.

Examples

```
## Not run:
library(icd)
# magrittr not required for icd to work, just for this example
library(magrittr, warn.conflicts = FALSE, quietly = TRUE)
myData <- data.frame(
  visit_id = c("v1", "v2", "v3", "v4"),
  diag = c("39891", "39790", "41791", "4401"),
  poa = c("Y", "N", NA, "Y"),
  stringsAsFactors = FALSE
)
myData %>%
  filter_poa_not_no() %>%
  comorbid_ahrq()
# can fill out named fields also:
myData %>%
  filter_poa_yes(poa_name = "poa") %>%
  comorbid_ahrq(
    icd_name = "diag", visit_name = "visit_id",
    short_code = TRUE
  )
my_map <- head(icd9_map_elix)
# can call the core comorbid() function with an arbitrary mapping
myData %>%
  filter_poa_yes() %>%
  comorbid(
    icd_name = "diag",
    visit_name = "visit_id",
    map = my_map,
    short_map = TRUE
  )

## End(Not run)
```

`filter_valid` *Filter ICD codes by validity.*

Description

Filters a data.frame of patients for valid or invalid ICD codes

Usage

```
filter_valid(
  x,
  icd_name = get_icd_name(x),
  short_code = guess_short(.subset2(x, icd_name)),
  invert = FALSE,
  ...
)

filter_invalid(
  x,
  icd_name = get_icd_name(x),
  short_code = guess_short(x[[icd_name]]),
  invert = FALSE
)

icd9_filter_valid(
  x,
  icd_name = get_icd_name(x),
  short_code = guess_short(x[[icd_name]]),
  invert = FALSE
)

icd10_filter_valid(
  x,
  icd_name = get_icd_name(x),
  short_code = guess_short(x[[icd_name]]),
  invert = FALSE
)

icd9_filter_invalid(
  x,
  icd_name = get_icd_name(x),
  short_code = guess_short(x[[icd_name]]),
  invert = FALSE
)

icd10_filter_invalid(
  x,
  icd_name = get_icd_name(x),
  short_code = guess_short(x[[icd_name]]),
  invert = FALSE
)
```

Arguments

`x`	a data.frame containing a column of ICD codes
`icd_name`	The name of the column in the `data.frame` which contains the ICD codes. This is a character vector of length one. If it is `NULL`, `icd9` will attempt to guess the column name, looking for progressively less likely possibilities until it matches a single column. Failing this, it will take the first column in the data frame. Specifying the column using this argument avoids the guesswork.
`short_code`	single logical value which determines whether the ICD-9 code provided is in short (`TRUE`) or decimal (`FALSE`) form. Where reasonable, this is guessed from the input data.

`invert`	Single logical value. Returns the inverse of the result. E.g. if seeking valid ICD-9 codes, the invalid ones are returned.
`...`	arguments passed to the class-specific functions

Functions

- `filter_invalid`: Filter invalid rows from data frame of patients with ICD codes. This can also be achieved with `filter_valid` and `invert = TRUE`
- `icd9_filter_valid`: Filter data frame for valid ICD codes

`get_billable` *Get the subset of codes that are billable according to ICD-9-CM or ICD-10-CM*

Description

Using the equivalent `get_leaf()` is preferred.

Usage

```
get_billable(...)

## S3 method for class 'icd9'
get_billable(...)

## S3 method for class 'icd9cm'
get_billable(x, short_code = guess_short(x), invert = FALSE, ...)

## S3 method for class 'icd10'
get_billable(x, short_code = guess_short(x), invert = FALSE, ...)

## S3 method for class 'icd10cm'
get_billable(x, short_code = guess_short(x), invert = FALSE, ...)

## Default S3 method:
get_billable(x, short_code = guess_short(x), ...)
```

Arguments

`...`	arguments passed on to other functions
`x`	input vector of ICD codes
`short_code`	single logical value which determines whether the ICD-9 code provided is in short (`TRUE`) or decimal (`FALSE`) form. Where reasonable, this is guessed from the input data.
`invert`	Single logical value. Returns the inverse of the result. E.g. if seeking valid ICD-9 codes, the invalid ones are returned.

Methods (by class)

- `icd9`: Prefer 'leaf' to 'billable' for generality.
- `icd9cm`: Prefer 'leaf' to 'billable' for generality.
- `icd10`: Prefer 'leaf' to 'billable' for generality.
- `icd10cm`: Prefer 'leaf' to 'billable' for generality.
- `default`: Prefer 'leaf' to 'billable' for generality.

See Also

`is_leaf()`

`get_cim10fr2019` *Localised synonym for `get_icd10fr2019`, with French column names*

Description

Localised synonym for `get_icd10fr2019`, with French column names

Usage

```
get_cim10fr2019()
```

See Also

`get_icd10fr2019`

`get_defined` *Select only defined ICD codes*

Description

Return only those codes which are heading or leaf (billable), specifying whether codes are all short-form or all decimal-form

Usage

```
get_defined(x, short_code = guess_short(x), billable = FALSE, leaf = billable)
```

Arguments

`x`	input vector or factor, possibly with an ICD class
`short_code`	logical value, whether short-form ICD code
`billable`	single logical value, identical to 'leaf'. Leaf is preferred as most adaptations of WHO ICD codes are not oriented around money.
`leaf`	single logical value, whether to limit return codes also by whether they are billable, i.e. leaf nodes. This is really only designed for use with ICD-9-CM, ICD-10-CM etc, since the WHO versions are not designed for billing, but for public health and death reporting.

`get_icd10be2014` *Belgian ICD-10-BE*

Description

This is based heavily on ICD-10-CM. 2014 is identical, with translations for most of the codes into Dutch and French. 2017 has about a hundred additional code definitions over ICD-10-CM 2017. The 2014 data also has the interesting fields for gender specificity of a given code, and whether it is permissible as Present-on-Arrival (POA).

Source

`https://www.health.belgium.be/en/node/30433` `https://www.health.belgium.be/sites/default/files/uploads/fields/fpshealth_theme_file/fy2017_reflist_icd-10-be.xlsx_last_updatet_28-07-2017_1.xlsx` `https://www.health.belgium.be/fr/sante/organisation-des-soins-de-san`
`hopitaux/systemes-denregistrement/icd-10-be` `https://www.health.belgium.be/fr/fy2014reflisticd-10-be`

See Also

`get_icd10be2017` `get_icd10be2014_pc` `get_icd10be2017_pc`

`get_icd10be2014_pc` *ICD-10-BE 2014 procedure codes*

Description

ICD-10-BE 2014 procedure codes

See Also

`get_icd10be2014` `get_icd10be2017` `get_icd10be2017_pc`

get_icd10be2017 *ICD-10-BE 2017*

Description

ICD-10-BE 2017

See Also

`get_icd10be2014` `get_icd10be2014_pc` `get_icd10be2017_pc`

get_icd10be2017_pc *ICD-10-BE 2017 procedure codes*

Description

ICD-10-BE 2017 procedure codes

See Also

`get_icd10be2014` `get_icd10be2014_pc` `get_icd10be2017`

get_icd10cm2014 *ICD-10-CM 2014*

Description

ICD-10-CM 2014

See Also

`icd10cm2019`

get_icd10cm2014_pc *ICD-10-CM Procedure codes for 2014*

Description

ICD-10-CM Procedure codes for 2014

See Also

`get_icd10cm2015_pc` `get_icd10cm2016_pc` `get_icd10cm2017_pc` `get_icd10cm2018_pc` `get_icd10cm2019_pc`

`get_icd10cm2015` *ICD-10-CM 2015*

Description

ICD-10-CM 2015

See Also

`icd10cm2019`

`get_icd10cm2015_pc` *ICD-10-CM Procedure codes for 2015*

Description

ICD-10-CM Procedure codes for 2015

See Also

`get_icd10cm2014_pc` `get_icd10cm2016_pc` `get_icd10cm2017_pc` `get_icd10cm2018_pc` `get_icd10cm2019_pc`

`get_icd10cm2016` *ICD-10-CM 2016*

Description

ICD-10-CM 2016

See Also

`icd10cm2019`

`get_icd10cm2016_pc` *ICD-10-CM Procedure codes for 2016*

Description

ICD-10-CM Procedure codes for 2016

See Also

`get_icd10cm2014_pc` `get_icd10cm2015_pc` `get_icd10cm2017_pc` `get_icd10cm2018_pc` `get_icd10cm2019_pc`

`get_icd10cm2017` *ICD-10-CM 2017*

Description

ICD-10-CM 2017

See Also

`icd10cm2019`

`get_icd10cm2017_pc` *ICD-10-CM Procedure codes for 2017*

Description

ICD-10-CM Procedure codes for 2017

See Also

`get_icd10cm2014_pc` `get_icd10cm2015_pc` `get_icd10cm2016_pc` `get_icd10cm2018_pc` `get_icd10cm2019_pc`

`get_icd10cm2018` *ICD-10-CM 2018*

Description

ICD-10-CM 2018

See Also

`icd10cm2019`

`get_icd10cm2018_pc` *ICD-10-CM Procedure codes for 2018*

Description

ICD-10-CM Procedure codes for 2018

See Also

`get_icd10cm2014_pc` `get_icd10cm2015_pc` `get_icd10cm2016_pc` `get_icd10cm2017_pc` `get_icd10cm2019_pc`

`get_icd10cm2019` *ICD-10-CM 2019*

Description

ICD-10-CM 2019

See Also

`icd10cm2019`

`get_icd10cm2019_pc` *ICD-10-CM Procedure Codes*

Description

ICD-10-PCS is the annually-updated set of procedure codes designed by 3M for the US CMS. There is no directory of WHO ICD procedure codes.

Format

A named list of data frames. The elements of the list are named by the year, e.g., `"2018"`. Each data frame contains two character columns, the first, named code is the procedure code; the second, named `desc`, has the description.

See Also

`get_icd10cm2014_pc` `get_icd10cm2015_pc` `get_icd10cm2016_pc` `get_icd10cm2017_pc` `get_icd10cm2018_pc` `https://www.cms.gov/Medicare/Coding/ICD10/downloads/pcs_refman.pdf`

`get_icd10cm_available` *Get the ICD-10-CM versions available in this package*

Description

Get the ICD-10-CM versions available in this package

Usage

```
get_icd10cm_available(dx = TRUE, return_year = FALSE)
```

Arguments

`dx`	Single logical value, if `TRUE` the default, diagnostic codes will be retrieved and processed. If `FALSE`, procedure codes will be used. Note that most ICD-10 schemes around the world do not add procedure codes. The US uses them extensively, and these form the basis of the Belgian version of ICD-10.
`return_year`	Logical, which, if `TRUE`, will result in only a character vector of year (or year-like version) being returned.

Value

By default, the names of all the data available, for diagnostic ICD-10-CM codes, e.g. `icd10cm2019`.

Examples

```
# Diagnostic codes:
get_icd10cm_available()
# Just get the years avaiable for ICD-10-CM procedure codes
get_icd10cm_available(dx = FALSE, return_year = TRUE)
# How to use the data name - most are not package data, due to severe CRAN
# package size limitations, so they are retrieved and cached as needed.
# The latest ICD-10-CM is included.
tail(get_icd10cm_available(), n = 1)
```

`get_icd10cm_latest` *The latest available version of ICD-10-CM in this package*

Description

The latest available version of ICD-10-CM in this package

The latest available ICD-10-CM data in this package

Usage

```
get_icd10cm_latest()
```

```
get_icd10cm_latest()
```

Details

This is an active binding, so is exported explicitly

This is an active binding, so is exported explicitly

Examples

```
a <- get_icd10cm_latest()
identical(a, icd10cm2019)
```

`get_icd10cm_version` *Get the data for a given version (four-digit year) of ICD-10-CM*

Description

When called without an argument, it returns the currently active version as set by `set_icd10cm_active_year()`

Usage

```
get_icd10cm_version(ver)

get_icd10cm_active()
```

Arguments

`ver`	Character vector of length one: the version of ICD-10-CM to use, corresponding to the suffix of the `data.frame` name, e.g., for 2019 ICD-10-CM, use `"icd10cm2019"` for Dutch 2014 ICD-CM translations, use `"icd10cm2014_nl"`

Functions

- `get_icd10cm_active`: Get the currently active version of ICD-10-CM.

Examples

```
## Not run:
get_icd10cm_version("2018")

## End(Not run)
```

`get_icd10fr2019` *French ICD-10-FR modification of WHO ICD-10 used in France*

Description

La Classification internationale statistique des maladies (CIM), version 10, edition française (The International Classification of Diseases (ICD), version 10, French edition.) Comme la version américaine, l'édition française a beaucoup de changéements par rapport à l'édition de l'OMS. Juste l'année 2018 est présentée pour le moment. The short descriptions are capitalized, and, as is correct in French, do not require accents. These were not converted to lower or sentence case to avoid introducing spelling errors. Définitions CIM-10-FR de l'OMS (WHO ICD-10 definitions)

Source

https://www.atih.sante.fr/cim-10-fr-2018-usage-pmsi

References

ATIH CIM-10-FR

`get_icd10who2008fr` *2008 WHO ICD-10 data in French*

Description

This data must be downloaded on a per-user basis. A prompt is given when the data is first attempted to be accessed.

Source

`http://www.who.int`

`get_icd10who2016` *2016 WHO ICD-10 data*

Description

This data must be downloaded on a per-user basis. A prompt is given when the data is first attempted to be accessed.

Source

`http://www.who.int`

`get_icd9cm2014_leaf` *ICD-9-CM, just billable/leaf codes*

Description

ICD-9-CM, just billable/leaf codes

Format

data frames with columns `code`, `short_desc`, and `long_desc`.

Details

These are derived from the final CMS published version 32 for 2014, which was unchanged since 2011. The short descriptions are in ASCII with no special characters, whereas the long descriptions contain accented characters which are stored as Unicode, `latin-1` or `cp1252`.

This all done during package creation, but can be repeated by package users, including pulling the data from the web pages directly. Despite my best efforts, current locale can give different results, but this packaged data is correct, with some `UTF-8` encoded strings. `icd9cm_billable` has been removed, and is replaced by `icd9cm2014_leaf` now includes only the latest version (32).

Source

http://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes.html

---

`icd10cm2019` *United States and Belgium ICD-10-CM*

Description

The public domain modified ICD-10 classification as published in the public domain by the US CDC. Currently this has a slightly different structure to `icd9cm_hierarchy` because the published data helpfully has a *leaf* flag indicating whether a code is a *billable* leaf node, or a code higher in the hierarchy which nevertheless will have a description.

Details

Format: data frame, with columns for code, leaf status (0 or 1), short and long descriptions.

Editions

There are annual revisions to this data in the US.

Source

http://www.cdc.gov/nchs/icd/icd10cm.htm

References

https://www.cms.gov/Medicare/Coding/ICD10/ https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2018-ICD-10-PCS-Tables-And-Index.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2018-ICD-10-PCS-Order-File.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2017-PCS-Code-Tables.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2017-PCS-Long-Abbrev-Titles.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2016-Code-Descriptions-in-Tabular-Order.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2015-code-descriptions.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2015-tables-index.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2015-Code_Tables-and-Index.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2015-PCS-long-and-abbreviated-titles.zip https://www.cms.gov/Medicare/

Coding/ICD10/Downloads/2014-ICD10-Code-Descriptions.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2014-ICD10-Code-Tables-and-Index.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2014-Code-Tables-and-Index.zip https://www.cms.gov/Medicare/Coding/ICD10/Downloads/2014-PCS-long-and-abbreviated-titles.zip

See Also

`get_icd10cm2014` `get_icd10cm2015` `get_icd10cm2017` `get_icd10cm2018`

`icd10_chapters` *ICD-10 chapters*

Description

The WHO ICD-10 scheme chapters. The chapter level is the highest in the hierarchy, each chapter containing sets of codes which span multiple three-digit 'major' codes, and in some cases also span codes across two alphabetic initial characters. E.g. Chapter I spans A00 to B99.

Details

2017 ICD-10-CM does not have any U codes (codes for special purposes). U00-U49 - Provisional assignment of new diseases of uncertain etiology or emergency use U82-U85 - Resistance to antimicrobial and anti-neoplastic drugs

Format: list with chapter names stored in list names, each with two element named character vector with start and end codes.

Source

http://apps.who.int/classifications/icd10/browse/2016/en

See Also

`icd10_sub_chapters`

`icd10_map_ahrq_pcs` *AHRQ ICD-10-PCS categories*

Description

The AHRQ has categorized each of the ICD-10-PCS (Procedure Codes) into one of four groups: minor diagnostic, minor therapeutic, major diagnostic or major therapeutic. This mapping can be used to get the type(s) of procedure(s) performed on a patient from a `data.frame` of patients and associated procedure codes in 'long' format. See the ICD-10 vignette for an example.

Details

Currently there is no specific comorbidity function to use this data, so the generic comorbid icd9_comorbid icd10_comorbid should be used, and this data specified as the map.

See Also

comorbid icd9_comorbid icd10_comorbid https://www.hcup-us.ahrq.gov/toolssoftware/procedureicd10/procedure_icd10.jsp

Other comorbidity maps: icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

Other comorbidities: comorbid_hcc(), comorbid(), icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

Examples

```
icd10_map_ahrq_pcs[["Major Diagnostic"]][1:5]
icd10_map_ahrq_pcs[["Minor Therapeutic"]][1:5]
```

---

icd10_sub_chapters *ICD-10 sub-chapters*

Description

The WHO ICD-10 scheme sub-chapters. N.b. there may be WHO vs CM differences: please file bug if noted. In the XML definition of ICD-10-CM there are some intermediate hierarchical levels, e.g. for neoplasms. Sub-chapter here is defined as the lowest-level grouping of three-digit codes, e.g. C00-C14 "Malignant neoplasms of lip, oral cavity and pharynx", not C00-C96 "Malignant neoplasms" which itself is a subset of the chapter C00-D49 "Neoplasms"

Details

Format: list with sub-chapter or major names stored in list names, each with two element named character vector with start and end codes.

Source

http://apps.who.int/classifications/icd10/browse/2016/en

See Also

icd10_chapters

icd9cm2014_leaf *The final ICD-9-CM list of leaf ('billable') codes*

Description

Other years are available from get_icd9cm2013_leaf, etc..

icd9cm_hierarchy *ICD-9-CM diagnosis codes including leaf nodes and branch names up to the three-digit codes.*

Description

Unlike get_icd9cm2014_leaf and friends, these data frames contain the full structure of the ICD-9-CM scheme up to the three-digit codes. Unlike the equivalent ICD-10-CM data frames (e.g., icd10cm2019), they do not have columns indicating billable/leaf status, or chapter designations.

Details

icd9cm_hierarchy is the deprecated name, which currently points to the final ICD-9-CM 2014 release. 2011 – 2014 are in fact identical. These are generated dynamically by parsing an awkward RTF file, and only the 2011 – 2014 data has been tested and used thoroughly.

Format: data frame

Source

http://wonder.cdc.gov/wonder/sci_data/codes/icd9/type_txt/icd9cm.asp

Rich text descriptions here: http://www.cdc.gov/nchs/icd/icd9cm.htm http://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes.html This page has versions 23 to 32 (2005 to 2014). At present, only the 2014 data is included in this package.

http://wonder.cdc.gov/wonder/sci_data/codes/icd9/type_txt/icd9abb.asp

http://wonder.cdc.gov/wonder/sci_data/codes/icd9/type_txt/icd9cm.asp

http://wonder.cdc.gov/wonder/sci_data/codes/icd9/type_txt/icdcm.asp

http://wonder.cdc.gov/wonder/sci_data/codes/icd9/type_txt/icd9abb.asp `icd9_chapters` *ICD-9 chapters*

Description

`icd9_chapters`, `icd9_chapters_sub` and `icd9_majors` contain mappings from the higher level descriptions of ICD-9 codes to the ranges of ICD-9 codes they describe. Helpful in summarizing codes or grouping for human-readable output. These can easily be converted to a co-morbidity mapping, as shown in the vignette.

Details

- 001-139 Infectious And Parasitic Diseases
- 140-239 Neoplasms
- 240-279 Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders
- 280-289 Diseases Of The Blood And Blood-Forming Organs
- 290-319 Mental Disorders
- 320-389 Diseases Of The Nervous System And Sense Organs
- 390-459 Diseases Of The Circulatory System
- 460-519 Diseases Of The Respiratory System
- 520-579 Diseases Of The Digestive System
- 580-629 Diseases Of The Genitourinary System
- 630-679 Complications Of Pregnancy, Childbirth, And The Puerperium
- 680-709 Diseases Of The Skin And Subcutaneous Tissue
- 710-739 Diseases Of The Musculoskeletal System And Connective Tissue
- 740-759 Congenital Anomalies
- 760-779 Certain Conditions Originating In The Perinatal Period
- 780-799 Symptoms, Signs, And Ill-Defined Conditions
- 800-999 Injury And Poisoning
- V01-V91 Supplementary Classification Of Factors Influencing Health Status And Contact With Health Services
- E000-E999 Supplementary Classification Of External Causes Of Injury And Poisoning

Format: list with chapter/sub-chapter or major names stored in list names, each with two element named character vector with start and end codes.

Source

`http://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes.html`

icd9_map_ahrq *AHRQ comorbidities*

Description

These mappings of comorbidities to ICD-9 and ICD-10 codes are derived directly from SAS code provided by AHRQ then translated into this R data structure. This is a revision of the Elixhauser system, notably excluding cardiac arrythmia.

Format

list of character vectors

Source

http://www.hcup-us.ahrq.gov/toolssoftware/comorbidity/comorbidity.jsp http://www.hcup-us.ahrq.gov/toolssoftware/comorbidityicd10/comorbidity_icd10.jsp

See Also

comorbid_ahrq icd9_comorbid_ahrq icd10_comorbid_ahrq

Other comorbidity maps: icd10_map_ahrq_pcs, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

Other comorbidities: comorbid_hcc(), comorbid(), icd10_map_ahrq_pcs, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

icd9_map_elix *Elixhauser comorbidities*

Description

This data comprises the original mapping of Elixhauser's ICD-9-CM to 30 comorbidities. According to Sharabiani, this mapping provides the best long-term mortality prediction. The weaknesses of this mapping are that it is based on slightly out-dated ICD-9 codes. I have not yet verified what changes to the ICD-9-CM specification between 1998 and now would impact this mapping.

Format

list of character vectors, each named by co-morbidity

References

Sharabiani, Mansour T. A., Paul Aylin, and Alex Bottle. "Systematic Review of Comorbidity Indices for Administrative Data." Medical Care December 2012 50, no. 12 (2012): 1109-18. doi:10.1097/MLR.0b013e31825f64d0. http://www.ncbi.nlm.nih.gov/pubmed/22929993

Elixhauser, Anne, Claudia Steiner, D. Robert Harris, and Rosanna M. Coffey. "Comorbidity Measures for Use with Administrative Data." Medical Care January 1998 36, no. 1 (1998): 8-27.

See Also

comorbid_elix icd9_comorbid_elix icd10_comorbid_elix

Other comorbidity maps: icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

Other comorbidities: comorbid_hcc(), comorbid(), icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

---

`icd9_map_hcc` *Medicare Hierarchical Condition Categories*

Description

Medicare HCC model was developed to use current year diagnoses and demographics predict current year healthcare expenditure. This classification has been used for additional risk adjustment models. ICD codes are first assigned to numeric Condition Categories ('CCs'). A hierarchy rule is then applied so that each patient is coded for only the most severe of the Condition Categories in a group. For example, if a patient has metastatic lung cancer, they will only be assigned the 'CC' for "Metastatic Cancer and Acute Leukemia", and will not be assigned the 'CC' for "Lung and other Severe Cancers". Once the hierarchy rules are applied, the codes are referred to as HCCs. This mapping can change over time. It remained the same from 2007-10

Format

`dataframe` with 3 columns (`icd_code`, `cc`, and `year`)

References

Pope, Gregory C., et al. "Diagnostic cost group hierarchical condition category models for Medicare risk adjustment." Health Economics Research, Inc. Waltham, MA (2000). https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Reports/Downloads/Pope_2000_2.pdf

Risk Adjustment, Centers for Medicare and Medicaid Services https://www.cms.gov/Medicare/Health-Plans/MedicareAdvtgSpecRateStats/Risk-Adjustors.html

See Also

comorbid_hcc icd9_comorbid_hcc icd10_comorbid_hcc

Other comorbidity maps: icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

Other comorbidities: comorbid_hcc(), comorbid(), icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_quan_elix, icd9_map_single_ccs

`icd9_map_pccc` *Pediatric Complex Chronic Conditions*

Description

There are seven comorbidity maps which represent the combinations of ICD-9, ICD-10, diagnosis and procedure codes, and three also with a set of 'fixed' codes. (See reference).

References

Feudtner C, Feinstein JA, Zhong W, Hall M, Dai D. Pediatric complex chronic conditions classification system version 2: updated for ICD-10 and complex medical technology dependence and transplantation. BMC Pediatr. 2014 Aug8;14:199. doi: 10.1186/1471-2431-14-199. `https://www.ncbi.nlm.nih.gov/pubmed/25102958`

See Also

`https://feudtnerlab.research.chop.edu/ccc_version_2.php`

Other comorbidity maps: `icd10_map_ahrq_pcs`, `icd9_map_ahrq`, `icd9_map_elix`, `icd9_map_hcc`, `icd9_map_quan_deyo`, `icd9_map_quan_elix`, `icd9_map_single_ccs`

Other comorbidities: `comorbid_hcc()`, `comorbid()`, `icd10_map_ahrq_pcs`, `icd9_map_ahrq`, `icd9_map_elix`, `icd9_map_hcc`, `icd9_map_quan_deyo`, `icd9_map_quan_elix`, `icd9_map_single_ccs`

`icd9_map_quan_deyo` *Quan adaptation of Deyo/Charlson comorbidities*

Description

Derived automatically from the SAS code used in the original publication. According to the referenced study, this provides the best predictor of in-patient to <30d mortality. Of note, Deyo drops the distinction between leukemia, lymphoma and non-metastatic cancer. As far as I have looked into this, in the rare cases where someone had two or three of leukemia, lymphoma and non-metastatic cancer, the Quan adaptation would give a lower Charlson score than the original scheme. The original Deyo Charlson to ICD-9-CM groups does include distinct categories for these things.

Format

list of character vectors, each named by co-morbidity

References

Quan, Hude, Vijaya Sundararajan, Patricia Halfon, Andrew Fong, Bernard Burnand, Jean-Christophe Luthi, L. Duncan Saunders, Cynthia A. Beck, Thomas E. Feasby, and William A. Ghali. "Coding Algorithms for Defining Comorbidities in ICD-9-CM and ICD-10 Administrative Data." Medical Care 43, no. 11 (November 1, 2005): 1130-39. `http://www.ncbi.nlm.nih.gov/pubmed/16224307` `http://web.archive.org/web/20110225042437/http://www.chaps.ucalgary.ca/sas`

See Also

comorbid_quan_deyo icd9_comorbid_quan_deyo icd10_comorbid_quan_deyo

Other comorbidity maps: icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_elix, icd9_map_single_ccs

Other comorbidities: comorbid_hcc(), comorbid(), icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_elix, icd9_map_single_ccs

icd9_map_quan_elix *Quan adaptation of Elixhauser comorbidities*

Description

These were transcribed directly from the Quan paper referenced.

Format

list of character vectors, each named by co-morbidity

References

Quan, Hude, Vijaya Sundararajan, Patricia Halfon, Andrew Fong, Bernard Burnand, Jean-Christophe Luthi, L. Duncan Saunders, Cynthia A. Beck, Thomas E. Feasby, and William A. Ghali. "Coding Algorithms for Defining Comorbidities in ICD-9-CM and ICD-10 Administrative Data." Medical Care 43, no. 11 (November 1, 2005): 1130-39. http://www.ncbi.nlm.nih.gov/pubmed/16224307 http://web.archive.org/web/20110225042437/http://www.chaps.ucalgary.ca/sas

See Also

comorbid_quan_elix icd9_comorbid_quan_elix icd10_comorbid_quan_elix

Other comorbidity maps: icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_single_ccs

Other comorbidities: comorbid_hcc(), comorbid(), icd10_map_ahrq_pcs, icd9_map_ahrq, icd9_map_elix, icd9_map_hcc, icd9_map_pccc, icd9_map_quan_deyo, icd9_map_single_ccs

icd9_map_single_ccs *Clinical Classifications Software (CCS) for ICD9/10-CM*

Description

The Clinical Classifications Software (CCS) for ICD-9/10-CM is one in a family of databases and software tools developed as part of the Healthcare Cost and Utilization Project (HCUP),a Federal-State-Industry partnership sponsored by the Agency for Healthcare Research and Quality. HCUP databases, tools, and software inform decision making at the national, State, and community levels. The software contains two different mappings. One is a single level mapping and one is a multi level classification. This data set contains the numeric representations of all of the codes.

Format

list of character vectors, each numbered by co-morbidity

Details

This file contains the updated ICD9 version that includes categories for Mental Health and Substance Abuse. More information on the ICD-9-CM data set can be found `https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp`.

The file `icd10_map_ccs` contains the 2018.1 ICD10 Version of the mapping. More information on the ICD10 code set can be found at `https://www.hcup-us.ahrq.gov/toolssoftware/ccs10/ccs10.jsp`

See Also

`comorbid_ccs` `icd9_comorbid_ccs` `icd10_comorbid_ccs`

Other comorbidity maps: `icd10_map_ahrq_pcs`, `icd9_map_ahrq`, `icd9_map_elix`, `icd9_map_hcc`, `icd9_map_pccc`, `icd9_map_quan_deyo`, `icd9_map_quan_elix`

Other comorbidities: `comorbid_hcc()`, `comorbid()`, `icd10_map_ahrq_pcs`, `icd9_map_ahrq`, `icd9_map_elix`, `icd9_map_hcc`, `icd9_map_pccc`, `icd9_map_quan_deyo`, `icd9_map_quan_elix`

icd9_sub_chapters *ICD-9 sub-chapters*

Description

ICD-9 sub-chapters

See Also

`icd9_chapters`

is.icd9 *Test presence of ICD classes*

Description

This merely checks whether the given object is a certain type of ICD code, it does no validation of any kind. For validation, see is_valid.

Usage

```
is.icd9(x)

is.icd9cm(x)

is.icd9cm_pc(x)

is.icd9who(x)

is.icd10(x)

is.icd10cm(x)

is.icd10cm_pc(x)

is.icd10who(x)

is.icd10fr(x)

is.icd10be(x)

is.comorbidity_map(x)
```

Arguments

x	Any object which may have ICD-related classes set

Examples

```
# A character string is not itself an ICD code
is.icd9("100.1")
is_valid("100.1")
is.icd9(as.icd9cm("100.1"))
```

is.icd_long_data *Test for class describing patient data*

Description

This function does not examine the data itself; it just checks whether one of the classes `icd_long_data` or `icd_wide_data` class is set.

Usage

```
is.icd_long_data(x)

is.icd_wide_data(x)
```

Arguments

x	Typically a `data.frame`

See Also

`icd_long_data`

is_billable *Check whether a code is billable according to ICD-9-CM or ICD-10-CM*

Description

Using the equivalent `is_leaf()` is preferred.

Usage

```
is_billable(x, short_code = guess_short(x), ...)

## S3 method for class 'icd9'
is_billable(x, short_code = guess_short(x), ...)

## S3 method for class 'icd9cm'
is_billable(x, short_code = guess_short(x), ...)

## S3 method for class 'icd10'
is_billable(x, short_code = guess_short(x), ...)

## S3 method for class 'icd10cm'
is_billable(x, short_code = guess_short(x), ...)

## Default S3 method:
is_billable(x, short_code = guess_short(x), ...)
```

Arguments

`x`	input vector to test
`short_code`	single logical value which determines whether the ICD-9 code provided is in short (`TRUE`) or decimal (`FALSE`) form. Where reasonable, this is guessed from the input data.
`...`	arguments passed on to other functions

Methods (by class)

- `icd9`: Prefer 'leaf' to 'billable' for generality.
- `icd9cm`: Prefer 'leaf' to 'billable' for generality.
- `icd10`: Prefer 'leaf' to 'billable' for generality.
- `icd10cm`: Prefer 'leaf' to 'billable' for generality.
- `default`: Prefer 'leaf' to 'billable' for generality.

See Also

`get_leaf()`

`long_to_wide` *Convert ICD data from long to wide format*

Description

Convert ICD data from long to wide format

Note the distinction between labelling existing data with any classes which `icd` provides, and actually converting the structure of the data.

Usage

```
long_to_wide(
  x,
  visit_name = get_visit_name(x),
  icd_name = get_icd_name(x),
  prefix = "icd_",
  min_width = 1L
)
```

Arguments

`x`	data.frame of long-form data, one column for visit_name and one for ICD code
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`icd_name`	The name of the column in the `data.frame` which contains the ICD codes. This is a character vector of length one. If it is `NULL`, `icd9` will attempt to guess the column name, looking for progressively less likely possibilities until it matches a single column. Failing this, it will take the first column in the data frame. Specifying the column using this argument avoids the guesswork.
`prefix`	character, default `icd_` to prefix new columns
`min_width,`	single integer, if specified, writes out this many columns even if no patients have that many codes. Must be greater than or equal to the maximum number of codes per patient.

Details

This is more complicated than expected using `base::reshape` or `reshape2::dcast` allows. This is a reasonably simple solution using built-in functions.

Long and Wide Formats

As is common with many data sets, key variables can be concentrated in one column or spread over several. Tools format of clinical and administrative hospital data, we can perform the conversion efficiently and accurately, while keeping some metadata about the codes intact, e.g. whether they are ICD-9 or ICD-10.

Data structure

Long or wide format ICD data are all expected to be in a data frame. The `data.frame` itself does not carry any ICD classes at the top level, even if it only contains one type of code; whereas its constituent columns may have a class specified, e.g. `icd9` or `icd10who`.

See Also

Other ICD data conversion: `comorbid_df_to_mat()`, `comorbid_mat_to_df()`, `convert`, `decimal_to_short()`, `short_to_decimal()`, `wide_to_long()`

Examples

```
longdf <- data.frame(
  visit_name = c("a", "b", "b", "c"),
  icd9 = c("441", "4424", "443", "441")
)
long_to_wide(longdf)
long_to_wide(longdf, prefix = "ICD10_")
```

---

names_elix *Comorbidity names*

Description

These lists provide correctly sorted names of the comorbidities and their particular permutations in both full and abbreviated forms.

Format

list, with character/numeric code. 'Hypertension, uncomplicated' and 'Hypertension, complicated' are labelled '6a' and '6b'. Diabetes, cancer, and metastasis are counted independently, as in the original paper, giving the original 30 groups. "01" to "30"

Details

In the Elixhauser derived mappings, uncomplicated and complicated hypertension are listed separately, but are always combined in the final analyses. Uncomplicated and complicated hypertension are list separately and as "Hypertension, combined." _abbrev suffix indicates a very short space-free description. Quan's version of Elixhauser is identical. AHRQ updates drops the arrythmia field. The naming convention is a root, e.g. `icd9_map_elix` or `icd10_map_elix`, with neither/either/both suffixes `_htn` and `_abbrev`. The Charlson derived mappings do not include hypertension. _abbreviated comorbidity names are helpful for interactive work, whereas the full names might be preferred for plotting.

---

plot_comorbid *Basic ordered bar plot showing counts of each comorbidity*

Description

Basic ordered bar plot showing counts of each comorbidity

Usage

```
plot_comorbid(x, comorbid_fun = icd::comorbid_ahrq, ...)

plot_comorbid_results(
  x,
  sort = TRUE,
  fix_margin = FALSE,
  las = 2,
  cex.names = 0.75,
  ...
)
```

Arguments

`x`	input patient data
`comorbid_fun`	Character name of function or function itself, default being `comorbid_ahrq`
`...`	Passed to `barplot`
`sort`	Logical, default `TRUE` which sorts the frequencies from high to low.
`fix_margin`	Logical, default `TRUE`, which causes a `par` margin to be set so the x axis labels are less likely to be truncated.
`las`	Integer, default is 2 which rotates the x axis labels appropriately
`cex.names`	Numeric, default is 0.75 which scales the text size for labels appropriately

Functions

- `plot_comorbid_results`: Plot the results of a call to one of the comorbidity settings.

Examples

```
## Not run:
library(icd)
plot_comorbid(vermont_dx)
plot_comorbid(uranium_pathology)
# Or calculate the comorbidities, then plot the results
cmb <- comorbid_ahrq(vermont_dx)
# plot with full, not abbreviated names
plot_comorbid_results(cmb, names.arg = names_ahrq)
# or return with full names, and plot those:
comorbid_ahrq(vermont_dx, abbrev_names = FALSE) %>%
  plot_comorbid_results()

## End(Not run)
```

`poa_choices` *Present-on-admission flags*

Description

See filter_poa for more details.

Usage

```
poa_choices
```

Format

An object of class `character` of length 4.

Examples

```
poa_choices
```

print.icd9 *Print ICD codes and comorbidity maps cleanly*

Description

Print ICD codes and comorbidity maps cleanly

Usage

```
## S3 method for class 'icd9'
print(x, verbose = FALSE, ...)

## S3 method for class 'icd10'
print(x, verbose = FALSE, ...)
```

Arguments

x	ICD codes to be printed
verbose	Annotate based on code attributes, e.g., decimal versus short codes.
...	arguments passed on to other functions

Examples

```
x <- structure(
  c("40201", "2258", "7208", "25001", "34400", "4011", "4011", NA),
  class = c("icd9cm", "icd9", "character"),
  icd_short_diag = TRUE
)
## Not run:
print(x)
print(x, verbose = TRUE)
# as.factor drops any 'icd' classes
print(as.factor(x), verbose = TRUE)

## End(Not run)
## Not run:
u <- uranium_pathology[1:10, "icd10"]
print(u)
print(u, verbose = TRUE)
# as.character will unclass the 'icd' classes
print(as.character(u), verbose = TRUE)
a <- structure(c("R21", "Z21"),
  class = c("icd10cm", "icd10", "character")
)
print(a, verbose = TRUE)

## End(Not run)
```

set_icd10cm_active_year — *Get or set the annual version of ICD-10-CM to use*

Description

Get or set the annual version of ICD-10-CM to use

Usage

```
set_icd10cm_active_year(ver, check_exists = TRUE)

get_icd10cm_active_year()
```

Arguments

`ver`	Character vector of length one: the version of ICD-10-CM to use, corresponding to the suffix of the `data.frame` name, e.g., for 2019 ICD-10-CM, use `"icd10cm2019"` for Dutch 2014 ICD-CM translations, use `"icd10cm2014_nl"`
`check_exists`	`TRUE` by default, which forces a check that the requested version is actually available in this R session.

set_icd_class — *Construct ICD-9 and ICD-10 data types*

Description

Takes an R structure and sets class to an ICD type. In the case of ICD-9 and ICD-10 codes, if a particular sub-type is set, e.g. ICD-9-CM (`icd9cm`), then an ICD-9 class (`icd9`) is also set.

Usage

```
as.icd9(x)

as.icd9cm(x)

as.icd9cm_pc(x)

as.icd10(x)

as.icd10cm(x, short_code = NULL)

as.icd10cm_pc(x)

as.icd10who(x, short_code = NULL)
as.icd10fr(x, short_code = NULL)

as.icd10be(x, short_code = NULL)
```

Arguments

`x`	object to set class `icd9`
`short_code`	single logical value which determines whether the ICD-9 code provided is in short (`TRUE`) or decimal (`FALSE`) form. Where reasonable, this is guessed from the input data.

Details

The `as.*` functions e.g. `as.icd9`, do checking and try to put multiple classes in a nice order. Calling the internal bare constructor, e.g. `icd:::icd9` just prepends the new class and returns without any checks, which is much faster.

Some features make more sense as attributes. E.g. setting code type to `short` or `decimal`.

Functions

- `as.icd9`: Use generic ICD-9 class for this data. Ideally, use the more specific `icd9cm` or other sub-classes (when available).
- `as.icd9cm`: Use ICD-9-CM
- `as.icd9cm_pc`: Indicate the data are ICD-9-CM procedure codes.
- `as.icd10`: Use generic ICD-10 class for this data. If possible, use the more specific `icd10who` or `icd10cm`.
- `as.icd10cm`: Use ICD-10-CM (USA) class for the given data
- `as.icd10cm_pc`: Indicate the data are ICD-10-CM procedure codes.
- `as.icd10who`: Use WHO ICD-10 class for the given data
- `as.icd10fr`: Use ICD-10-FR (France) class for the given data
- `as.icd10be`: Use ICD-10-BE (Belgium) class for the given data

See Also

Other ICD data types: `as.comorbidity_map()`, `wide_vs_long`

Examples

```
x <- as.icd10("A1009")
attr(x, "icd_short_diag") <- TRUE
x
attributes(x) <- list(icd_short_diag = NULL)
x

y <- as.decimal_diag(as.icd10("A10.09"))
y
```

```
is.short_diag(y)

j <- as.short_diag(as.icd10(c("A11", "B2222")))
j[2] <- "C33"
stopifnot(is.short_diag(j))
stopifnot(is.icd10(j), is.icd10(j[1]), is.icd10(j[[1]]))
j[[1]] <- "D44001"
stopifnot(is.short_diag(j))
stopifnot(is.icd10(j), is.icd10(j[2]), is.icd10(j[[2]]))
```

`set_icd_data_dir` *Set up the data download cache, give permission to download data*

Description

This must be called by the user, as prompted on package attach with `library(icd)`.

Usage

```
set_icd_data_dir(path = NULL)

get_icd_data_dir(must_work = TRUE)
```

Arguments

`path`	Path to a directory where cached online raw and parsed data will be cached. It will be created if it doesn't exist.
`must_work`	Logical, the default of `TRUE` will cause this to stop with an error if a usable icd data directory cannot be found or set.

Value

The path to the cache directory, or `NULL` if it could not be found.

Invisibly returns the data path which was set, or NULL if not done.

Functions

- `get_icd_data_dir`: Get the currently active data directory, and check it exists and is writable.

See Also

`download_all_icd_data`

Examples

```
## Not run:
set_icd_data_dir()
# or choose another directory:
# set_icd_data_dir("/var/cache/icd.data")
# If you choose a custom directory, you may wish to add this command to your .Rprofile .
# then you may use:
# download_all_icd_data()
# or let 'icd' download data when needed.

## End(Not run)
```

---

sort_icd *Sort or order ICD-9 or ICD-10 codes according to published sequence*

Description

The default method will guess whether ICD-9 or ICD-10 then sort based on that type. For ICD-10 codes, note that setting `short` is unnecessary and ignored. All codes should consistently use the decimal divider.

Usage

```
sort_icd(x, decreasing = FALSE, short_code = guess_short(x), ...)

## S3 method for class 'icd10'
sort(x, decreasing = FALSE, ...)

## S3 method for class 'icd10cm'
sort(x, decreasing = FALSE, ...)

## S3 method for class 'icd10be'
sort(x, decreasing = FALSE, ...)

## S3 method for class 'icd9'
sort(x, decreasing = FALSE, short_code = guess_short(x), ...)

order.icd9(x, na.last = TRUE)

order.icd10cm(x)

order.icd10be(x)
```

Arguments

`x`	vector of ICD codes to sort or order
`decreasing`	Logical See `sort`.

`short_code`	single logical value which determines whether the ICD-9 code provided is in short (`TRUE`) or decimal (`FALSE`) form. Where reasonable, this is guessed from the input data.
`...`	arguments passed on to other functions
`na.last`	Logical, analogous to `order`, so `NA` drops NA. `FALSE` is not currently supported.

Details

Note that `sort` is an S3 generic, whereas `order` is not. Thus we export `order.icd10cm`, but not `sort.icd10cm`, etc..

Value

For sort, a sorted vector of ICD-9 codes. Numeric, then E codes, then V codes. For order, an integer vector is returned with the order of each code.

ICD-9

Sorts lists of numeric, V or E codes. Note that a simple numeric sort does not work for ICD-9 codes, since `162` > `1620`, and also 'V' codes precede 'E' codes. Numeric codes are first, then 'V', then 'E'. A factor is returned if a factor is given.

ICD-10-CM and ICD-10-BE

There are some codes which are sequenced out of lexicographic order, e.g., `C7A` and `C7B` are between `C80` and `C81`; `D3A` is between `D48` and `D49`.

Examples

```
# order ICD-10-CM is not lexicographic:
codes <- as.icd10cm(c("C7A", "C79", "C80", "C81", "C7B"))
# as the class is set, use S3 dispatch to get the right answer
sort(codes)
# or call directly, but recall S3 dispatch will only work once 'icd' is
# attached using:
library(icd)
icd:::sort.icd10cm(c("C7A", "C79", "C80", "C81", "C7B"))
stopifnot(!identical(
  order.icd10cm(as.character(codes)),
  order(codes)
))
icd::order.icd9(c("V20", NA, "100", NA, "E998", "101"))
codes[order.icd10cm(codes)]
# Note that base::order does NOT do S3 dispatch, so the following does not work:
codes[order(codes)]
```

subset *extract subset(s) from ICD data*

Description

exactly the same as using `x[n]` or `x[[n]]` but preserves the ICD classes in result

Usage

```
## S3 method for class 'icd9'
x[...]

## S3 method for class 'icd9'
x[[...]]

## S3 method for class 'icd10'
x[...]

## S3 method for class 'icd10'
x[[...]]
```

Arguments

`x`	input data with list, vector, factor, and class set to an ICD type.
`...`	arguments passed on to other functions

Methods (by class)

- `icd9`: Extract ICD-9 codes
- `icd9`: Extract ICD-9 codes
- `icd10`: Extract ICD-10 codes
- `icd10`: Extract ICD-10 codes

Examples

```
x <- list(my_codes = as.icd9(c("V10.1", "441.1")))
x[1]
x[[1]]
x[[1]][2]
# subsetting a list should give the underlying data structure type,
# preserving the ICD class
stopifnot(!inherits(x[[1]], "list"))
stopifnot(!inherits(x[[1]][2], "list"))

y <- as.icd10(c("A01", "B0234"))
y[2]
y[[2]]
```

```
stopifnot(inherits(y[2], "icd10"))
stopifnot(inherits(y[[2]], "icd10"))
```

---

uranium_pathology *United States Transuranium & Uranium Registries*

Description

This is an ICD-10 data set (not ICD-10-CM) with mortality from the United States Transuranium & Uranium Registries, published in the public domain.

Source

https://ustur.wsu.edu/about-us/

---

van_walraven *Calculate van Walraven Elixhauser Score*

Description

van Walraven Elixhauser score is calculated from the Quan revision of Elixhauser's ICD-9 mapping. This function allows for the hierarchical exclusion of less severe versions of comorbidities when their more severe version is also present via the `hierarchy` argument. For the Elixhauser comorbidities, this is diabetes v. complex diabetes and solid tumor v. metastatic tumor

Usage

```
van_walraven(
  x,
  visit_name = NULL,
  return_df = FALSE,
  stringsAsFactors = getOption("stringsAsFactors"),
  ...
)

## S3 method for class 'data.frame'
van_walraven(
  x,
  visit_name = NULL,
  return_df = FALSE,
  stringsAsFactors = getOption("stringsAsFactors"),
  ...
)

van_walraven_from_comorbid(x, visit_name = NULL, hierarchy = FALSE)
```

Arguments

`x`	data frame containing a column of visit or patient identifiers, and a column of ICD-9 codes. It may have other columns which will be ignored. By default, the first column is the patient identifier and is not counted. If `visit_name` is not specified, the first column is used.
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`return_df`	single logical value, if `TRUE`, a two column data frame will be returned, with the first column named as in input data frame (i.e., `visit_name`), containing all the visits, and the second column containing the Charlson Comorbidity Index.
`stringsAsFactors`	Single logical value, describing whether the resulting data frame should have strings, e.g. `visit_id` converted to factor. Default is to follow the current session option. This is identical to the argument used in, among other base functions `as.data.frame`.
`...`	arguments passed on to other functions
`hierarchy`	single logical value that defaults to `TRUE`, in which case the hierarchy defined for the mapping is applied. E.g. in Elixhauser, you can't have uncomplicated and complicated diabetes both flagged.

Methods (by class)

- `data.frame`: van Walraven scores from data frame of visits and ICD-9 codes

Author(s)

wmurphyrd

References

van Walraven C, Austin PC, Jennings A, Quan H, Forster AJ. A Modification to the Elixhauser Comorbidity Measures Into a Point System for Hospital Death Using Administrative Data. Med Care. 2009; 47(6):626-633. `http://www.ncbi.nlm.nih.gov/pubmed/19433995`

Examples

```
mydf <- data.frame(
  visit_name = c("a", "b", "c"),
  icd9 = c("412.93", "441", "042")
)
van_walraven(mydf)
```

```
# or calculate comorbidities first:
cmb <- icd9_comorbid_quan_elix(mydf, short_code = FALSE, hierarchy = TRUE)
vwr <- van_walraven_from_comorbid(cmb)
stopifnot(identical(van_walraven(mydf), vwr))

# alternatively return as data frame in 'tidy' format
van_walraven(mydf, return_df = TRUE)
```

`vermont_dx` *Hospital discharge data from Vermont*

Description

Anonymous data from public Vermont source for 2013

Details

Conditions of Release Release of public use data is subject to the following conditions, which the requestor agrees to upon accepting copies of the data:

1. The data may not be used in any manner that attempts to or does identify, directly or indirectly, any individual patient or physician.

2. The requestor agrees to incorporate the following, or a substantially similar, disclaimer in all reports or publications that include public use data: “Hospital discharge data for use in this study were supplied by the Vermont Association of Hospitals and Health Systems-Network Services Organization (VAHHS-NSO) and the Vermont Department of Banking, Insurance, Securities and Health Care Administration (BISHCA). All analyses, interpretations or conclusions based on these data are solely that of [the requestor]. VAHHS-NSO and BISHCA disclaim responsibility for any such analyses, interpretations or conclusions. In addition, as the data have been edited and processed by VAHHS-NSO, BISHCA assumes no responsibility for errors in the data due to coding or processing”

Format: CSV original, minimally processed into R data frame.

Author(s)

Vermont Division of Health Care Administration

Source

http://www.healthvermont.gov/health-statistics-vital-records/health-care-systems-reporting/hospital-discharge-data

wide_to_long — *Convert ICD data from wide to long format*

Description

Convert ICD data from wide to long format

Note the distinction between labelling existing data with any classes which `icd` provides, and actually converting the structure of the data.

Usage

```
wide_to_long(
  x,
  visit_name = get_visit_name(x),
  icd_labels = NULL,
  icd_name = "icd_code",
  icd_regex = c("icd", "diag", "dx_", "dx")
)
```

Arguments

`x`	`data.frame` in wide format, i.e. one row per patient, and multiple columns containing ICD codes, empty strings or NA.
`visit_name`	The name of the column in the data frame which contains the patient or visit identifier. Typically this is the visit identifier, since patients come leave and enter hospital with different ICD-9 codes. It is a character vector of length one. If left empty, or `NULL`, then an attempt is made to guess which field has the ID for the patient encounter (not a patient ID, although this can of course be specified directly). The guesses proceed until a single match is made. Data frames may be wide with many matching fields, so to avoid false positives, anything but a single match is rejected. If there are no successful guesses, and `visit_id` was not specified, then the first column of the data frame is used.
`icd_labels`	vector of column names in which codes are found. If NULL, all columns matching the regular expression `icd_regex` will be included.
`icd_name`	The name of the column in the `data.frame` which contains the ICD codes. This is a character vector of length one. If it is `NULL`, `icd9` will attempt to guess the column name, looking for progressively less likely possibilities until it matches a single column. Failing this, it will take the first column in the data frame. Specifying the column using this argument avoids the guesswork.
`icd_regex`	vector of character strings containing a regular expression to identify ICD-9 diagnosis columns to try (case-insensitive) in order. Default is `c("icd","diag","dx_","dx")`

Details

Reshaping data is a common task, and is made easier here by knowing more about the underlying structure of the data. This function wraps the `reshape` function with specific behavior and checks related to ICD codes. Empty strings and NA values will be dropped, and everything else kept. No validation of the ICD codes is done.

Value

`data.frame` with visit_name column named the same as input, and a column named by `icd.name` containing all the non-NA and non-empty codes found in the wide input data.

Long and Wide Formats

As is common with many data sets, key variables can be concentrated in one column or spread over several. Tools format of clinical and administrative hospital data, we can perform the conversion efficiently and accurately, while keeping some metadata about the codes intact, e.g. whether they are ICD-9 or ICD-10.

Data structure

Long or wide format ICD data are all expected to be in a data frame. The `data.frame` itself does not carry any ICD classes at the top level, even if it only contains one type of code; whereas its constituent columns may have a class specified, e.g. `icd9` or `icd10who`.

See Also

Other ICD data conversion: `comorbid_df_to_mat()`, `comorbid_mat_to_df()`, `convert`, `decimal_to_short()`, `long_to_wide()`, `short_to_decimal()`

Examples

```
widedf <- data.frame(
  visit_name = c("a", "b", "c"),
  icd9_01 = c("441", "4424", "441"),
  icd9_02 = c(NA, "443", NA)
)
wide_to_long(widedf)
```

`wide_vs_long` *Set the ICD data structure class of a* `matrix` *or* `data.frame`.

Description

These functions take your patient data, and allow you to describe whether it is wide or long. `icd` never requires you do this, but it may help avoid errors, especially if you have atypical data that might confuse `icd`'s heuristics.

Usage

```
as.icd_long_data(x, warn = TRUE)

as.icd_wide_data(x, warn = TRUE)

icd_long_data(..., warn = TRUE)

icd_wide_data(x, ..., warn = TRUE)
```

Arguments

`x`	Input data is a `matrix`, `data.frame`, or a class that inherits one of these base structures, such as a `tibble`.
`warn`	Single logical, if `TRUE`, the default, a warning will be shown if changing class between long and wide types.
`...`	Data used to construct data frame before setting the appropriate class.

Functions

- `as.icd_long_data`: Set class on a matrix or data.frame to `icd_long_data`. To convert wide to long data, use `wide_to_long`.
- `as.icd_wide_data`: Construct a `data.frame`, adding the `icd_long_data` class.
- `icd_long_data`: Construct a `data.frame`, adding the `icd_long_data` class.
- `icd_wide_data`: Construct a `data.frame`, adding the `icd_wide_data` class.

Long and Wide Formats

As is common with many data sets, key variables can be concentrated in one column or spread over several. Tools format of clinical and administrative hospital data, we can perform the conversion efficiently and accurately, while keeping some metadata about the codes intact, e.g. whether they are ICD-9 or ICD-10.

Data structure

Long or wide format ICD data are all expected to be in a data frame. The `data.frame` itself does not carry any ICD classes at the top level, even if it only contains one type of code; whereas its constituent columns may have a class specified, e.g. `icd9` or `icd10who`.

Conversion

To convert between long and wide data, use `long_to_wide` or `wide_to_long`. Conversion functions in other packages, such as `ddplyr` will work, too, but will need some work to account for the typical structure of patient data and diagnostic codes. This is not done with `ddplyr`, `data.table` etc because it would add a big dependency burden. This package aims to be agnostic and use base R as much as possible.

See Also

long_to_wide and wide_to_long

Other ICD data types: as.comorbidity_map(), set_icd_class

Other ICD data types: as.comorbidity_map(), set_icd_class

Other ICD data types: as.comorbidity_map(), set_icd_class

Examples

```
(w <- icd_wide_data(
  id = c(1, 2, 3),
  dx01 = c("100", "441", "V20"),
  dx02 = c("E9981", "V10", "44004")
))
wide_to_long(w)
class(uranium_pathology)
class(vermont_dx)
```

---

with_icd10cm_version *Evaluate code with a particular version of ICD-10-CM*

Description

Temporarily sets and restores the option `icd.icd10cm_active_year`, analogous to functions in **withr**.

Usage

```
with_icd10cm_version(ver, code)
```

Arguments

ver	Character vector of length one: the version of ICD-10-CM to use, corresponding to the suffix of the `data.frame` name, e.g., for 2019 ICD-10-CM, use `"icd10cm2019"` for Dutch 2014 ICD-CM translations, use `"icd10cm2014_nl"`
code	Code block to execute, may be in braces, or a single statement without braces.

Examples

```
icd:::.show_options()
with_icd10cm_version("2014", icd:::.show_options())
```

Index