

# Package ‘icd’

November 18, 2018

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

**Version** 3.3

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

**Description** Calculate comorbidities, Charlson and van Walraven scores, perform fast and accurate validation, conversion, manipulation, filtering and comparison of ICD-9 and ICD-10 codes. This package enables a work flow from raw lists 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.

**License** GPL-3

**URL** <https://jackwasey.github.io/icd/>

**BugReports** <https://github.com/jackwasey/icd/issues>

**Depends** R (>= 3.4), icd.data

**Imports** checkmate (>= 1.7.0), magrittr, methods, stats, Rcpp (>= 0.12.3), utils

**Suggests** knitr, roxygen2 (>= 5.0.0), rmarkdown, rticles, testthat (>= 0.11.1), tinytex, xml2

**LinkingTo** Rcpp (>= 0.12.3), RcppEigen, testthat

**VignetteBuilder** knitr

**ByteCompile** true

**SystemRequirements** pandoc >= 2.2 to build the JSS vignette

**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** 6.1.1

**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] (Hierarchical Condition Codes),  
 Vitaly Druker [ctb] (AHRQ CCS),  
 Ed Lee [ctb] (explain codes in table format)

**Repository** CRAN

**Date/Publication** 2018-11-18 17:40:07 UTC

## R topics documented:

icd-package . . . . .	3
as.decimal_diag . . . . .	5
as.icd_long_data . . . . .	6
categorize_simple . . . . .	7
charlson . . . . .	8
charlson_from_comorbid . . . . .	10
children . . . . .	11
combine . . . . .	13
comorbid . . . . .	14
comorbid_df_to_mat . . . . .	19
comorbid_hcc . . . . .	21
comorbid_mat_to_df . . . . .	22
comorbid_pccc_dx . . . . .	23
convert . . . . .	25
count_codes . . . . .	25
count_codes_wide . . . . .	27
count_comorbid . . . . .	28
diff_comorbid . . . . .	28
filter_poa . . . . .	30
filter_valid . . . . .	32
get_defined . . . . .	33
icd10cm2016 . . . . .	34
icd10_chapters . . . . .	34
icd10_map_ahrq_pcs . . . . .	35
icd10_pcs . . . . .	35
icd10_sub_chapters . . . . .	36
icd9cm_billable . . . . .	36
icd9cm_hierarchy . . . . .	37
icd9_chapters . . . . .	37
icd9_map_ahrq . . . . .	38
icd9_map_elix . . . . .	39
icd9_map_hcc . . . . .	39
icd9_map_pccc . . . . .	40

icd9_map_quan_deyo . . . . .	41
icd9_map_quan_elix . . . . .	41
icd9_map_single_ccs . . . . .	42
is.icd9 . . . . .	43
long_to_wide . . . . .	44
names_elix . . . . .	45
poa_choices . . . . .	46
print.comorbidity_map . . . . .	46
subset_icd . . . . .	47
unzip_single . . . . .	48
uranium_pathology . . . . .	48
van_walraven . . . . .	49
vermont_dx . . . . .	51
wide_to_long . . . . .	51
[[.comorbidity_map . . . . .	53

**Index** **54**

icd-package *icd: Comorbidity Calculations and Tools for ICD-9 and ICD-10 Codes*

**Description**

Calculate comorbidities, Charlson and van Walraven scores, perform fast and accurate validation, conversion, manipulation, filtering and comparison of ICD-9 and ICD-10 codes. This package enables a work flow from raw lists 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.

**Details**

**Comorbidities** `comorbid` determines comorbidities for a set of patients with one or more ICD codes each. All the comorbidity functions attempt to guess which are your identifier and ICD code fields, and these can also be specified exactly.

- The AHRQ comorbidity mappings from ICD-9 and ICD-10 are provided as `icd9_map_ahrq` and `icd10_map_ahrq`. The easiest way to use them is by calling the function `comorbid_ahrq` directly with your `icd_long_data` format patient data.
- Quan revised both Deyo/Charlson and Elixhauser ICD-9 and ICD-10 to comorbidity mappings. These are presented as: `icd9_map_quan_deyo`, `icd10_map_quan_deyo`, `icd9_map_quan_elix`, and `icd10_map_quan_elix`. Like the AHRQ mappings, these are all carefully extracted from the original publications or source code. These mappings can be used on patient data by calling `comorbid_quan_deyo` and `comorbid_quan_elix`
- There is no canonical Charlson ICD-9 or ICD-10 mapping, so `comorbid_charlson` uses the thoroughly researched and widely cited `comorbid_quan_deyo` method by default.
- The original Elixhauser mappings are provided by the lists `icd9_map_elix` and `icd10_map_elix`, and Elixhauser comorbidities can be calculated directly from patient data using `comorbid_elix`.

- AHRQ publishes Hierarchical Condition Codes (HCC) which are essentially comorbidity maps with very many comorbidities, complicated by a single- or multi-level system. These categories can be computed using [comorbid\\_hcc](#).
- AHRQ also publishes Clinical Classification Software (CCS) which provides another set of disease groups, and this SAS code is implemented in 'icd' by [comorbid\\_ccs](#)

**Risk scoring systems** [charlson](#) calculates Charlson scores (Charlson Comorbidity Indices) directly from your patient data. If you already calculated the Charlson comorbidities, it is more efficient to use [charlson\\_from\\_comorbid](#). Similarly, [van\\_walraven](#) calculates Van Walraven scores (based on the Elixhauser comorbidities, instead of Charlson), and [van\\_walraven\\_from\\_comorbid](#) if you already calculated Elixhauser comorbidities

**Validation** [is\\_valid](#) checks whether ICD-9 codes are syntactically valid (although not necessarily genuine ICD-9 diagnoses). In contrast, [is\\_defined](#) checks whether ICD-9 codes correspond to diagnoses in the current ICD-9-CM definition from CMS.

**Conversion** There are many functions to convert ICD-9 codes or their components between different formats and structures. The most commonly used are: [decimal\\_to\\_short](#), [short\\_to\\_decimal](#) to convert, e.g., 002.3 to 0023 and back again. See [convert](#) for other options. Conversion between ICD-9, ICD-10, and ICD-11 codes is not yet supported.

**Manipulation** You can find children of a higher-level ICD-9 code with [children](#) and find a common parent to a set of children (or arbitrary list of ICD-9 codes) with [condense](#). [sort\\_icd](#) sorts in hierarchical, then numerical order, so '100.0' comes before '100.00', for example.

[wide\\_to\\_long](#) and [long\\_to\\_wide](#) convert the two most common data structures containing patient disease data. This is more sophisticated and tailored to the problem than base reshaping or packages like [dplyr](#), although these could no doubt be used.

**Explanation and decoding** Use [explain\\_code](#) to convert a list of codes into human-readable descriptions. This function can optionally reduce the codes to a their top-level groups if all the child members of a group are present. [diff\\_comorbid](#) allows summary of the differences between comorbidity mappings, e.g. to find what has changed from year-to-year or between revisions by different authors. [icd9cm\\_hierarchy](#) is a `data.frame` containing the full ICD-9 classification for each diagnosis. [icd9\\_chapters](#) contains definitions of chapters, sub-chapters and three-digit groups.

#### Author(s)

Jack O. Wasey <[jack@jackwasey.com](mailto:jack@jackwasey.com)>

#### See Also

Useful links:

- <https://jackwasey.github.io/icd/>
- Report bugs at <https://github.com/jackwasey/icd/issues>

---

as.decimal_diag	<i>Get or set whether ICD codes have have an attribute indicating 'short' or 'decimal' format</i>
-----------------	---

---

### 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, must_work = FALSE)
is.icd_decimal_diag(x, must_work = FALSE)
is.short_diag(x, must_work = FALSE)
is.icd_short_diag(x, must_work = FALSE)
```

### 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.
must_work	single logical value, if FALSE (the default) this may return NULL if the attribute is not present. If TRUE, then either TRUE or FALSE is returned.

### Getting the attribute

is.short\_diag tests for presence of an attribute, not whether the code is a valid ICD code. If must\_work is TRUE then NULL (i.e. no attribute set) returns FALSE, otherwise NULL is returned. To test validity, 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.

## Examples

```
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
```

---

as.icd_long_data	<i>Convert between and identify 'long' and 'wide' patient data formats</i>
------------------	--

---

## Description

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.

icd\_long\_data and icd\_wide\_data create data.frames using all the arguments, and sets the class, whereas as.icd\_long\_data and as.icd\_wide\_data just set the class of an existing data.frame.

## Usage

```
as.icd_long_data(x)
as.icd_wide_data(x)
icd_long_data(...)
icd_wide_data(...)
is.icd_long_data(x)
is.icd_wide_data(x)
```

## Arguments

x	data.frame or matrix to set class, or convert.
...	arguments passed on to create a data.frame

## Details

Long or wide format ICD data is expected to be in a data frame. It does not carry any ICD classes at the top level, even if it only contains one type of code, but its constituent columns may have a class specified, e.g. 'icd9'.

## 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`: Set class on a matrix or data.frame to `icd_wide_data`. To convert long to wide data, use [long\\_to\\_wide](#).
- `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.
- `is.icd_long_data`: Return TRUE if x has the `icd_long_data` class.
- `is.icd_wide_data`: Return TRUE if x has the `icd_wide_data` class.

## See Also

Other ICD code conversion: [convert](#), [long\\_to\\_wide](#), [wide\\_to\\_long](#)

Other ICD code conversion: [convert](#), [long\\_to\\_wide](#), [wide\\_to\\_long](#)

---

categorize_simple	<i>Categorize codes according to a mapping</i>
-------------------	--

---

## Description

This is the function which prepares the input data for the categorization, and forms the core of the package, along with the C++ matrix code. This is pure data manipulation and generalizable beyond medical data.

## Usage

```
categorize_simple(x, map, id_name, code_name, return_df = FALSE,
  return_binary = FALSE, restore_id_order = TRUE, unique_ids = FALSE,
  preserve_id_type = FALSE, comorbid_fun = comorbidMatMulSimple)
```

## Arguments

x	Data frame containing a column for an 'id' and a column for a code, e.g., an ICD-10 code.
map	named list containing vectors of ICD-9 codes. E.g. the AHRQ ICD-9 comorbidities, contains <code>list(OBESE = c("2780", "27800", "27801", "27803", "V8554", "79391", "649", "3004", "30112", "3090", "3091", "311"))</code> amongst other longer groups.
id_name	The name of the data.frame field which is the unique identifier.

code_name	String with name of column containing the codes.
return_df	single logical value, if TRUE, return the result as a data frame with the first column being the <code>visit_id</code> , and the second being the count. If <code>visit_id</code> was a factor or named differently in the input, this is preserved.
return_binary	Logical value, if TRUE, the output will be in 0s and 1s instead of TRUE and FALSE.
restore_id_order	Logical value, if TRUE, the default, the order of the visit IDs will match the order of visit IDs first encountered in the input data. This takes a third of the time in calculations on data with tens of millions of rows, so, if the visit IDs will be discarded when summarizing data, this can be set to FALSE for a big speed-up.
unique_ids	Single logical value, if TRUE then the visit IDs in column given by <code>id_name</code> are assumed to be unique. Otherwise, the default action is to ensure they are unique.
preserve_id_type	Single logical value, if TRUE, the visit ID column will be converted back to its original type. The default of FALSE means only factors and character types are restored in the returned data frame. For matrices, the row names are necessarily stored as character vectors.
comorbid_fun	function i.e. the function symbol (not character string) to be called to do the comorbidity calculation

### Examples

```
## Not run:
u <- uranium_pathology
m <- icd10_map_ahrq
u$icd10 <- decimal_to_short(u$icd10)
j <- categorize_simple(u, m, id_name = "case", code_name = "icd10")

## End(Not run)
```

---

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"),
         ...)

icd_charlson(...)

icd_charlson.data.frame(...)
```

### Arguments

<code>x</code>	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 <code>visit_name</code> is not specified, the first column is used.
<code>visit_name</code>	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 <code>NULL</code> , 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 <code>visit_id</code> was not specified, then the first column of the data frame is used.
<code>scoring_system</code>	One of <code>original</code> , <code>charlson</code> , or <code>quan</code> . The first two will give the original Charlson weights for each comorbidity, whereas <code>quan</code> uses the updated weights from Quan 2011.
<code>return_df</code>	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. <code>visit_name</code> ), containing all the visits, and the second column containing the Charlson Comorbidity Index.
<code>stringsAsFactors</code>	single logical, passed on when constructing <code>data.frame</code> if <code>return_df</code> is <code>TRUE</code> . If the input data frame <code>x</code> has a factor for the <code>visit_name</code> , this is not changed, but a non-factor <code>visit_name</code> may be converted or not converted according to your system default or this setting.
<code>...</code>	further arguments to pass on to <code>icd9_comorbid_quan_deyo</code> , e.g. <code>name</code>

### 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 ([dx.doi.org/10.1093/aje/kwq433](https://doi.org/10.1093/aje/kwq433)). 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`.

**Deprecated function names**

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

**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"))

icd_charlson_from_comorbid(...)
```

**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 and 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.
...	arguments passed on to other functions

### Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

---

children	<i>Get children of ICD codes</i>
----------	----------------------------------

---

### 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, ...)

icd_children.character(...)

icd_children.icd10(...)

icd_children.icd10cm(...)

icd_children.icd9(...)

icd_children(...)

icd_children_defined(...)

icd_children_defined.icd10cm(...)

```

### 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, 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

### Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

### See Also

Other ICD-9 ranges: [condense](#), [expand\\_minor](#), [expand\\_range](#)

### Examples

```
library(magrittr, warn.conflicts = FALSE, quietly = TRUE) # optional

# 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)
children("003", short_code = TRUE, defined = TRUE) %>%
  explain_code(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 combination 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

**Examples**

```
# Care with the following:
c(as.icd9("E998"), as.icd10("A10"))
# which results in both codes sharing the 'icd9' class.
(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 = TRUE, visitId = NULL,
  icd9Field = NULL, categorize_fun = categorize_simple,
  comorbid_fun = comorbidMatMulSimple, ...)

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(...)
```

```
icd_comorbid(...)
```

```
icd_comorbid_ahrq(...)
```

```
icd_comorbid_elix(...)
```

```
icd_comorbid_hcc(...)
```

```
icd_comorbid_quan_deyo(...)
```

```
icd_comorbid_quan_elix(...)
```

## Arguments

<code>x</code>	data.frame containing a column of patient-visit identifiers and a column of ICD codes. The data.frame should be in ‘long’ format, like the example <code>vermont_dx</code> data. If it is in ‘wide’ format, it must be converted to ‘long’ using <code>wide_to_long</code> before calling any comorbidity functions.
<code>map</code>	list of the comorbidities with each list item containing a vector of decimal ICD-9 codes. This is 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. <code>comorbid_ahrq</code> , since these also name the fields correctly, apply any hierarchical rules (see hierarchy below)
<code>visit_name</code>	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 <code>visit_id</code> was not specified, then the first column of the data frame is used.
<code>icd_name</code>	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, <code>icd9</code> 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.
<code>short_code</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 the result as 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 <code>icd:::logical_to_binary</code> and <code>icd:::binary_to_logical</code> , or using other tools, e.g. <code>apply(x, 2, as.integer)</code>
categorize_fun	Internal. Function used for the categorization problem.
...	arguments passed on to other functions
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 <code>icd9_add_leading_zeroes</code> ), then <code>preclean</code> can be set to FALSE to save time.
visitId	Deprecated. Use <code>visit_name</code> instead.
icd9Field	Deprecated. Use <code>icd_name</code> instead.
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. <code>ahrqComorbidNamesAbbrev</code> 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.

## Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

## 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](#), [comorbid\\_pccc\\_dx](#)

Other comorbidities: [comorbid\\_hcc](#), [comorbid\\_pccc\\_dx](#), [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**

```

vermont_dx[1:5, 1:10]
vd <- wide_to_long(vermont_dx)
# get first few rows and columns of Charlson comorbidities using Quan's mapping
comorbid_quan_deyo(vd)[1:5, 1:14]

# get summary AHRQ (based on Elixhauser) comorbidities for ICD-10 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)

```

---

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"))

icd_comorbid_df_to_mat(...)

icd_comorbid_mat_to_df(...)
```

**Arguments**

<code>x</code>	data frame, with a <code>visit_name</code> column (not necessarily first), and other columns with flags for comorbidities, as such column names are required.
<code>visit_name</code>	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 <code>visit_id</code> was not specified, then the first column of the data frame is used.
<code>stringsAsFactors</code>	Single logical value, describing whether the resulting data frame should have strings, e.g. <code>visit_id</code> converted to factor. Default is to follow the current session option. This is identical to the argument used in, among other base functions <code>as.data.frame</code> .
<code>...</code>	arguments passed on to other functions

**Deprecated function names**

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

**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**

<code>x</code>	data frame with columns for patient/visit ID, ICD code and date
<code>date_name</code>	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'.
<code>visit_name</code>	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 <code>visit_id</code> was not specified, then the first column of the data frame is used.
<code>icd_name</code>	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, <code>icd9</code> 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\\_pccc\\_dx](#), [comorbid](#)

Other comorbidities: [comorbid\\_pccc\\_dx](#), [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.

**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"]])
```

---

comorbid_pccc_dx	<i>Calculate pediatric complex chronic conditions (PCCC) comorbidities</i>
------------------	--

---

### Description

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.

### Usage

```
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, ...)

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, ...)

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

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

### Arguments

x	data.frame containing a column of patient-visit identifiers and a column of ICD codes. The data.frame should be in 'long' format, like the example <code>vermont_dx</code> data. If it is in 'wide' format, it must be converted to 'long' using <a href="#">wide_to_long</a> before calling any comorbidity functions.
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 <code>visit_id</code> 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.
return_df	single logical value, if TRUE, return the result as 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)
...	arguments passed on to other functions

## 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

Other comorbidity computations: [comorbid\\_hcc](#), [comorbid](#)

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\\_map\\_single\\_ccs](#)

## Examples

```
# not pediatric data, but let's look for this example
head(icd9_comorbid_pccc_dx(wide_to_long(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")
```



---

convert	<i>Convert ICD9 codes between formats and structures.</i>
---------	---

---

### Description

ICD-9 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 convert ICD-9 to ICD-10 codes yet.

### See Also

Other ICD code conversion: [as.icd\\_long\\_data](#), [long\\_to\\_wide](#), [wide\\_to\\_long](#)

---

count_codes	<i>Count ICD codes or comorbidities for each patient</i>
-------------	--

---

### 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)
```

```
icd_count_codes(...)
```

```
icd_count_comorbid(...)
```

### 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.

... arguments passed on to other functions

### 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)

### Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

### 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:
library(magrittr)
wide %>% wide_to_long %>% count_codes
```

---

count_codes_wide	<i>Count ICD codes given in wide format</i>
------------------	---

---

### 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)
```

```
icd_count_codes_wide(...)
```

### 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 the result as 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.
...	arguments passed on to other functions

### Deprecated function names

Future versions of **icd** will drop the icd\_ prefix. For example, charlson should be used in favor of icd\_charlson. To distinguish **icd** function calls, consider using the prefix icd:: instead, e.g., icd::charlson. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. icd9\_comorbid\_ahrq. **icd** specific classes also retain the prefix, e.g., icd\_wide\_data.

---

count_comorbid	<i>Count number of comorbidities per patient</i>
----------------	--

---

### 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 <code>visit_name</code> 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 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 <code>visit_id</code> was not specified, then the first column of the data frame is used.
return_df	single logical value, if TRUE, return the result as a data frame with the first column being the <code>visit_id</code> , and the second being the count. If <code>visit_id</code> was a factor or named differently in the input, this is preserved.

---

diff_comorbid	<i>show the difference between two comorbidity mappings</i>
---------------	---

---

### 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)

icd_diff_comorbid.list(...)

icd_diff_comorbid(...)
```

**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.
...	arguments passed on to other functions

**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

**Deprecated function names**

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

**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)

```

---

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")

icd_filter_poa(...)

icd_filter_poa_no(...)

icd_filter_poa_not_no(...)

icd_filter_poa_not_yes(...)

icd_filter_poa_yes(...)

```

### 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."
...	arguments passed on to other functions

## 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.

## Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

## Examples

```
## Not run:
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)
# can call the core comorbid() function with an arbitrary mapping
myData %>%
  filter_poa_yes %>%
  comorbid_elix(icd_name = "diag", visit_name = "visit_id",
  short_mapping = 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)
```

```
icd_filter_invalid(...)
```

```
icd_filter_valid(...)
```

### 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

## Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

---

<code>get_defined</code>	<i>Select only defined ICD codes</i>
--------------------------	--------------------------------------

---

## 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)
```

```
icd_get_defined.icd9(...)
```

```
icd_get_defined(...)
```

## Arguments

<code>x</code>	input vector or factor, possibly with an ICD class
<code>short_code</code>	logical value, whether short-form ICD code
<code>billable</code>	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.
<code>...</code>	arguments passed on to other functions

## Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

---

`icd10cm2016`*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**

There are annual revisions to this data. Currently, the 2016 edition is included.

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

**See Also**

[ls\\_icd\\_data](#)

---

`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.

**See Also**

[ls\\_icd\\_data](#)

---

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](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_pccc_dx`, `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\_pcs      *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.

### Details

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

`ls_icd_data`

---

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.

### See Also

[ls\\_icd\\_data](#)

---

icd9cm\_billable      *list of annual versions of billable leaf nodes of ICD-9-CM*

---

### Description

These are derived from the CMS published updates, with versions 23 to 32 currently available going back to 2004/5. The source files back to version 27 have short and long descriptions. The short descriptions are in ASCII with no special characters, whereas the long descriptions contain accented characters which seem to be interpreted 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.

### Details

Format: list of data frames. Each list item is named by the version as a string, e.g. "32". The constituent data frames have columns icd9, shortDesc, and longDesc.

### See Also

[ls\\_icd\\_data](#)

---

icd9cm_hierarchy	<i>Latest ICD-9-CM diagnosis codes, in flat data.frame format</i>
------------------	---

---

**Description**

Short-form ICD-9 codes with short and long descriptions, and description of each hierarchy level containing each code.

**Details**

Format: data frame

**See Also**

[ls\\_icd\\_data](#)

---

icd9_chapters	<i>ICD-9 chapters</i>
---------------	-----------------------

---

**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.

### See Also

[ls\\_icd\\_data](#)

---

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 arrhythmia.

### 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](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\\_pccc\\_dx](#), [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\\_pccc\\_dx](#), [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](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\\_pccc\\_dx](#), [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**

[comorbid\\_pccc\\_dx](#) [icd9\\_comorbid\\_pccc\\_dx](#) [icd10\\_comorbid\\_pccc\\_dx](#) [comorbid\\_pccc\\_pcs](#) [icd9\\_comorbid\\_pccc\\_pcs](#) [icd10\\_comorbid\\_pccc\\_pcs](#) [https://feudtnerlab.research.chop.edu/ccv\\_version\\_2.php](https://feudtnerlab.research.chop.edu/ccv_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\\_pccc\\_dx](#), [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\\_pccc\\_dx](#), [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\\_pccc\\_dx](#), [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/toolsoftware/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/toolsoftware/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\\_pccc\\_dx](#), [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](#)

---

is.icd9	<i>test ICD-related classes</i>
---------	---------------------------------

---

**Description**

currently no checks on correctness of the classes for these functions

**Usage**

`is.icd9(x)`

`is.icd10(x)`

`is.icd9cm(x)`

`is.icd10cm(x)`

`is.comorbidity_map(x)`

`is.icd_comorbidity_map(...)`

**Arguments**

`x` Any object which may have ICD-related classes set

`...` arguments passed on to other functions

**Deprecated function names**

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. [icd9\\_comorbid\\_ahrq](#). **icd** specific classes also retain the prefix, e.g., [icd\\_wide\\_data](#).

---

long\_to\_wide                      *Convert ICD data from long to wide format*

---

### Description

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

### Usage

```
long_to_wide(x, visit_name = get_visit_name(x),
            icd_name = get_icd_name(x), prefix = "icd_", min_width = 0,
            aggr = TRUE, return_df = FALSE)
```

```
icd_long_to_wide(...)
```

### 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.
aggr	single logical value, if TRUE (the default) will take more time to find out-of-order visit_names, and combine all the codes for each unique visit_name. If FALSE, then out-of-order visit_names will result in a row in the output data per contiguous block of identical visit_names.
return_df	single logical value, if TRUE, return a data frame with a field for the visit_name. This may be more convenient, but the default of FALSE gives the more natural return data of a matrix with row names being the visit IDs from visit_names.
...	arguments passed on to other functions

### Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

### See Also

Other ICD code conversion: [as.icd\\_long\\_data](#), [convert](#), [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 arrhythmia 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.

---

poa_choices	<i>Present-on-admission flags</i>
-------------	-----------------------------------

---

**Description**

See [filter\\_poa](#) for more details.

**Usage**

```
poa_choices
icd_poa_choices
```

**Format**

An object of class character of length 4.

**Deprecated function names**

Future versions of **icd** will drop the `icd_` prefix. For example, `char1son` should be used in favor of `icd_char1son`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::char1son`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. [icd9\\_comorbid\\_ahrq](#). **icd** specific classes also retain the prefix, e.g., [icd\\_wide\\_data](#).

**Examples**

```
poa_choices
```

---

```
print.comorbidity_map Print a comorbidity map
```

---

**Description**

The default is to summarize by printing the first seven comorbidities, and the first seven codes for each. To print the whole thing, just convert it to a list.

**Usage**

```
## S3 method for class 'comorbidity_map'
print(x, ..., n_comorbidities = 7,
      n_codes = 7)
```

**Arguments**

x                    a list optionally with class comorbidity\_map  
 ...                  further arguments are passed to print  
 n\_comorbidities    single integer, number of comorbidities to print  
 n\_codes             single integer, number of codes per comorbidity to print

**Examples**

```
icd9_map_ahrq
## Not run:
print(icd9_map_ahrq)
print(icd9_map_ahrq, n_comorbidities = 3, n_codes = 3)
print.list(icd9_map_ahrq)
print(list(icd9_map_ahrq))

## End(Not run)
```

---

subset\_icd

*extract subset 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

**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"))
```

---

unzip_single	<i>unzip a single file from URL</i>
--------------	-------------------------------------

---

**Description**

take a single file from zip located at a given URL, unzip into temporary directory, and copy to the given save\_path

**Usage**

```
unzip_single(url, file_name, save_path)
```

**Arguments**

url	URL of a zip file
file_name	file name of the resource within the zip file
save_path	file path to save the first file from the zip

---

uranium_pathology	<i>United States Transuranium &amp; Uranium Registries</i>
-------------------	--

---

**Description**

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

**See Also**

[ls\\_icd\\_data](#)



---

 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)

icd_van_walraven.data.frame(...)

icd_van_walraven(...)

icd_van_walraven_from_comorbid(...)
```

### 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. <code>visit_id</code> converted to factor. Default is to follow the current session option. This is identical to the argument used in, among other base functions <code>as.data.frame</code> .
...	arguments passed on to other functions
hierarchy	single logical value that defaults to <code>TRUE</code> , 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

### Deprecated function names

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

### 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.

### See Also

[ls\\_icd\\_data](#)

---

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

---

### Description

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.

### 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"))
```

```
icd_wide_to_long(...)
```

**Arguments**

<code>x</code>	data.frame in wide format, i.e. one row per patient, and multiple columns containing ICD codes, empty strings or NA.
<code>visit_name</code>	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 and leave 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 <code>visit_id</code> was not specified, then the first column of the data frame is used.
<code>icd_labels</code>	vector of column names in which codes are found. If NULL, all columns matching the regular expression <code>icd_regex</code> will be included.
<code>icd_name</code>	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, <code>icd9</code> 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.
<code>icd_regex</code>	vector of character strings containing a regular expression to identify ICD-9 diagnosis columns to try (case-insensitive) in order. Default is <code>c("icd", "diag", "dx_", "dx")</code>
<code>...</code>	arguments passed on to other functions

**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.

**Deprecated function names**

Future versions of **icd** will drop the `icd_` prefix. For example, `charlson` should be used in favor of `icd_charlson`. To distinguish **icd** function calls, consider using the prefix `icd::` instead, e.g., `icd::charlson`. Functions which specifically operate on either ICD-9 or ICD-10 codes or their sub-types will retain the prefix. E.g. `icd9_comorbid_ahrq`. **icd** specific classes also retain the prefix, e.g., `icd_wide_data`.

**See Also**

Other ICD code conversion: [as.icd\\_long\\_data](#), [convert](#), [long\\_to\\_wide](#)

**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)
```

---

[[.comorbidity\_map     *Extract vector of codes from an ICD comorbidity map*

---

**Description**

Equivalent to a list, but preserves class of extracted vector.

**Usage**

```
## S3 method for class 'comorbidity_map'  
x[[index, ...]]
```

**Arguments**

x	comorbidity map, which is a named list
index	integer
...	arguments passed on to other functions

**Examples**

```
# show that attributes are preserved when subsetting  
stopifnot(is.short_diag(icd10_map_ahrq[[1]]))
```

# Index

- \*Topic **character**
  - poa\_choices, 46
- \*Topic **datasets**
  - icd10\_chapters, 34
  - icd10\_map\_ahrq\_pcs, 35
  - icd10\_pcs, 35
  - icd10\_sub\_chapters, 36
  - icd10cm2016, 34
  - icd9\_chapters, 37
  - icd9\_map\_ahrq, 38
  - icd9\_map\_elix, 39
  - icd9\_map\_hcc, 39
  - icd9\_map\_pccc, 40
  - icd9\_map\_quan\_deyo, 41
  - icd9\_map\_quan\_elix, 41
  - icd9\_map\_single\_ccs, 42
  - icd9cm\_billable, 36
  - icd9cm\_hierarchy, 37
  - names\_elix, 45
  - poa\_choices, 46
  - uranium\_pathology, 48
  - vermont\_dx, 51
- \*Topic **manip**
  - children, 11
  - filter\_poa, 30
  - filter\_valid, 32
- \*Topic **misc**
  - icd-package, 3
- \*Topic **utilities**
  - icd-package, 3
  - [.icd10 (subset\_icd), 47
  - [.icd9 (subset\_icd), 47
  - [[.comorbidity\_map, 53
  - [[.icd10 (subset\_icd), 47
  - [[.icd9 (subset\_icd), 47
  - \_PACKAGE (icd-package), 3
  
  - ahrq (icd9\_map\_ahrq), 38
  - as.decimal\_diag, 5
  - as.icd\_decimal\_diag (as.decimal\_diag), 5
  
  - as.icd\_long\_data, 6, 25, 45, 52
  - as.icd\_short\_diag (as.decimal\_diag), 5
  - as.icd\_wide\_data (as.icd\_long\_data), 6
  - as.short\_diag (as.decimal\_diag), 5
  
  - c.icd10 (combine), 13
  - c.icd9 (combine), 13
  - categorize\_simple, 7
  - charlson, 4, 8
  - charlson\_from\_comorbid, 4, 10
  - children, 4, 11
  - combine, 13
  - comorbid, 3, 14, 22, 24, 35, 38–43
  - comorbid\_ahrq, 3, 18, 38
  - comorbid\_ahrq (comorbid), 14
  - comorbid\_ccs, 4, 14, 43
  - comorbid\_ccs (comorbid), 14
  - comorbid\_charlson, 3
  - comorbid\_charlson (comorbid), 14
  - comorbid\_df\_to\_mat, 19
  - comorbid\_elix, 3, 18, 39
  - comorbid\_elix (comorbid), 14
  - comorbid\_hcc, 4, 14, 18, 21, 24, 35, 38–43
  - comorbid\_mat\_to\_df, 22
  - comorbid\_pccc\_dx, 18, 22, 23, 35, 38–43
  - comorbid\_pccc\_pcs, 40
  - comorbid\_pccc\_pcs (comorbid\_pccc\_dx), 23
  - comorbid\_quan\_deyo, 3, 41
  - comorbid\_quan\_deyo (comorbid), 14
  - comorbid\_quan\_elix, 3, 42
  - comorbid\_quan\_elix (comorbid), 14
  - condense, 4, 13
  - convert, 4, 7, 25, 45, 52
  - count\_codes, 25
  - count\_codes\_wide, 27
  - count\_comorbid, 28
  
  - decimal\_to\_short, 4, 5, 25
  - diff\_comorbid, 4, 28

- expand\_minor, 13
- expand\_range, 13
- explain\_code, 4
  
- filter\_invalid(filter\_valid), 32
- filter\_poa, 30, 46
- filter\_poa\_no(filter\_poa), 30
- filter\_poa\_not\_no(filter\_poa), 30
- filter\_poa\_not\_yes(filter\_poa), 30
- filter\_poa\_yes(filter\_poa), 30
- filter\_valid, 32
  
- get\_defined, 33
  
- icd(icd-package), 3
- icd-package, 3
- icd10-package(icd-package), 3
- icd10\_chapters, 34
- icd10\_comorbid, 35
- icd10\_comorbid(comorbid), 14
- icd10\_comorbid\_ahrq, 38
- icd10\_comorbid\_ahrq(comorbid), 14
- icd10\_comorbid\_ccs, 43
- icd10\_comorbid\_ccs(comorbid), 14
- icd10\_comorbid\_charlson(comorbid), 14
- icd10\_comorbid\_elix, 39
- icd10\_comorbid\_elix(comorbid), 14
- icd10\_comorbid\_hcc, 40
- icd10\_comorbid\_hcc(comorbid\_hcc), 21
- icd10\_comorbid\_pccc\_dx, 40
- icd10\_comorbid\_pccc\_dx
  - (comorbid\_pccc\_dx), 23
- icd10\_comorbid\_pccc\_pcs, 40
- icd10\_comorbid\_pccc\_pcs
  - (comorbid\_pccc\_dx), 23
- icd10\_comorbid\_quan\_deyo, 41
- icd10\_comorbid\_quan\_deyo(comorbid), 14
- icd10\_comorbid\_quan\_elix, 42
- icd10\_comorbid\_quan\_elix(comorbid), 14
- icd10\_filter\_invalid(filter\_valid), 32
- icd10\_filter\_valid(filter\_valid), 32
- icd10\_map\_ahrq, 3
- icd10\_map\_ahrq(icd9\_map\_ahrq), 38
- icd10\_map\_ahrq\_pcs, 18, 22, 24, 35, 38–43
- icd10\_map\_cc(icd9\_map\_hcc), 39
- icd10\_map\_ccs(icd9\_map\_single\_ccs), 42
- icd10\_map\_charlson
  - (icd9\_map\_quan\_deyo), 41
- icd10\_map\_elix, 3
- icd10\_map\_elix(icd9\_map\_elix), 39
- icd10\_map\_pccc\_dx(icd9\_map\_pccc), 40
- icd10\_map\_pccc\_pcs(icd9\_map\_pccc), 40
- icd10\_map\_quan\_deyo, 3
- icd10\_map\_quan\_deyo
  - (icd9\_map\_quan\_deyo), 41
- icd10\_map\_quan\_elix, 3
- icd10\_map\_quan\_elix
  - (icd9\_map\_quan\_elix), 41
- icd10\_pcs, 35
- icd10\_sub\_chapters, 36
- icd10cm2016, 34
- icd9-package(icd-package), 3
- icd9\_chapters, 4, 37
- icd9\_comorbid, 35
- icd9\_comorbid(comorbid), 14
- icd9\_comorbid\_ahrq, 10, 11, 13, 18, 20, 26, 27, 29, 31, 33, 38, 43, 45, 46, 50, 52
- icd9\_comorbid\_ahrq(comorbid), 14
- icd9\_comorbid\_ccs, 43
- icd9\_comorbid\_ccs(comorbid), 14
- icd9\_comorbid\_charlson(comorbid), 14
- icd9\_comorbid\_elix, 39
- icd9\_comorbid\_elix(comorbid), 14
- icd9\_comorbid\_hcc, 40
- icd9\_comorbid\_hcc(comorbid\_hcc), 21
- icd9\_comorbid\_pccc\_dx, 40
- icd9\_comorbid\_pccc\_dx
  - (comorbid\_pccc\_dx), 23
- icd9\_comorbid\_pccc\_pcs, 40
- icd9\_comorbid\_pccc\_pcs
  - (comorbid\_pccc\_dx), 23
- icd9\_comorbid\_quan\_deyo, 41
- icd9\_comorbid\_quan\_deyo(comorbid), 14
- icd9\_comorbid\_quan\_elix, 42
- icd9\_comorbid\_quan\_elix(comorbid), 14
- icd9\_filter\_invalid(filter\_valid), 32
- icd9\_filter\_valid(filter\_valid), 32
- icd9\_majors(icd9\_chapters), 37
- icd9\_map\_ahrq, 3, 18, 22, 24, 35, 38, 39–43
- icd9\_map\_cc(icd9\_map\_hcc), 39
- icd9\_map\_charlson(icd9\_map\_quan\_deyo), 41
- icd9\_map\_elix, 3, 18, 22, 24, 35, 38, 39, 40–43
- icd9\_map\_hcc, 18, 22, 24, 35, 38, 39, 39, 40–43
- icd9\_map\_multi\_ccs

- (icd9\_map\_single\_ccs), 42
- icd9\_map\_pccc, 18, 22, 24, 35, 38–40, 40, 41–43
- icd9\_map\_pccc\_dx (icd9\_map\_pccc), 40
- icd9\_map\_pccc\_pcs (icd9\_map\_pccc), 40
- icd9\_map\_quan\_deyo, 3, 18, 22, 24, 35, 38–40, 41, 42, 43
- icd9\_map\_quan\_elix, 3, 18, 22, 24, 35, 38–41, 41, 43
- icd9\_map\_single\_ccs, 18, 22, 24, 35, 38–42, 42
- icd9\_sub\_chapters (icd9\_chapters), 37
- icd9Billable (icd9cm\_billable), 36
- icd9cm\_billable, 36
- icd9cm\_hierarchy, 4, 37
- icd\_charlson (charlson), 8
- icd\_charlson\_from\_comorbid (charlson\_from\_comorbid), 10
- icd\_children (children), 11
- icd\_children\_defined (children), 11
- icd\_comorbid (comorbid), 14
- icd\_comorbid\_ahrq (comorbid), 14
- icd\_comorbid\_df\_to\_mat (comorbid\_df\_to\_mat), 19
- icd\_comorbid\_elix (comorbid), 14
- icd\_comorbid\_hcc (comorbid), 14
- icd\_comorbid\_mat\_to\_df (comorbid\_df\_to\_mat), 19
- icd\_comorbid\_quan\_deyo (comorbid), 14
- icd\_comorbid\_quan\_elix (comorbid), 14
- icd\_count\_codes (count\_codes), 25
- icd\_count\_codes\_wide (count\_codes\_wide), 27
- icd\_count\_comorbid (count\_codes), 25
- icd\_diff\_comorbid (diff\_comorbid), 28
- icd\_filter\_invalid (filter\_valid), 32
- icd\_filter\_poa (filter\_poa), 30
- icd\_filter\_poa\_no (filter\_poa), 30
- icd\_filter\_poa\_not\_no (filter\_poa), 30
- icd\_filter\_poa\_not\_yes (filter\_poa), 30
- icd\_filter\_poa\_yes (filter\_poa), 30
- icd\_filter\_valid (filter\_valid), 32
- icd\_get\_defined (get\_defined), 33
- icd\_long\_data, 3
- icd\_long\_data (as.icd\_long\_data), 6
- icd\_long\_to\_wide (long\_to\_wide), 44
- icd\_map\_cc\_hcc (icd9\_map\_hcc), 39
- icd\_names\_ahrq (names\_elix), 45
- icd\_names\_ahrq\_abbrev (names\_elix), 45
- icd\_names\_ahrq\_htn (names\_elix), 45
- icd\_names\_ahrq\_htn\_abbrev (names\_elix), 45
- icd\_names\_cc (names\_elix), 45
- icd\_names\_charlson (names\_elix), 45
- icd\_names\_charlson\_abbrev (names\_elix), 45
- icd\_names\_elix (names\_elix), 45
- icd\_names\_elix\_abbrev (names\_elix), 45
- icd\_names\_elix\_htn (names\_elix), 45
- icd\_names\_elix\_htn\_abbrev (names\_elix), 45
- icd\_names\_quan\_elix (names\_elix), 45
- icd\_names\_quan\_elix\_abbrev (names\_elix), 45
- icd\_names\_quan\_elix\_htn (names\_elix), 45
- icd\_names\_quan\_elix\_htn\_abbrev (names\_elix), 45
- icd\_poa\_choices (poa\_choices), 46
- icd\_van\_walraven (van\_walraven), 49
- icd\_van\_walraven\_from\_comorbid (van\_walraven), 49
- icd\_wide\_data, 10, 11, 13, 18, 20, 26, 27, 29, 31, 33, 43, 45, 46, 50, 52
- icd\_wide\_data (as.icd\_long\_data), 6
- icd\_wide\_to\_long (wide\_to\_long), 51
- is.comorbidity\_map (is.icd9), 43
- is.decimal\_diag (as.decimal\_diag), 5
- is.icd10 (is.icd9), 43
- is.icd10cm (is.icd9), 43
- is.icd9, 43
- is.icd9cm (is.icd9), 43
- is.icd\_comorbidity\_map (is.icd9), 43
- is.icd\_decimal\_diag (as.decimal\_diag), 5
- is.icd\_long\_data (as.icd\_long\_data), 6
- is.icd\_short\_diag (as.decimal\_diag), 5
- is.icd\_wide\_data (as.icd\_long\_data), 6
- is.short\_diag (as.decimal\_diag), 5
- is\_defined, 4
- is\_valid, 4, 5
- long\_to\_wide, 4, 7, 25, 44, 52
- ls\_icd\_data, 34–38, 48, 51
- names\_ahrq (names\_elix), 45
- names\_ahrq\_abbrev (names\_elix), 45
- names\_ahrq\_htn (names\_elix), 45
- names\_ahrq\_htn\_abbrev (names\_elix), 45



names\_cc (names\_elix), 45  
names\_charlson (names\_elix), 45  
names\_charlson\_abbrev (names\_elix), 45  
names\_elix, 45  
names\_elix\_abbrev (names\_elix), 45  
names\_elix\_htn (names\_elix), 45  
names\_elix\_htn\_abbrev (names\_elix), 45  
names\_quan\_elix (names\_elix), 45  
names\_quan\_elix\_abbrev (names\_elix), 45  
names\_quan\_elix\_htn (names\_elix), 45  
names\_quan\_elix\_htn\_abbrev  
    (names\_elix), 45

package-icd (icd-package), 3  
package-icd10 (icd-package), 3  
package-icd9 (icd-package), 3  
poa\_choices, 46  
print.comorbidity\_map, 46

reshape, 51

short\_to\_decimal, 4, 5, 25  
sort\_icd, 4  
subset\_icd, 47

unzip\_single, 48  
uranium\_pathology, 48

van\_walraven, 4, 49  
van\_walraven\_from\_comorbid, 4  
van\_walraven\_from\_comorbid  
    (van\_walraven), 49  
vermont\_dx, 51

wide\_to\_long, 4, 7, 16, 23, 25, 45, 51