

Description

`icd9` is a suite of commands for working with ICD-9-CM diagnosis codes from the 16th version (effective October 1998) to the 32nd version. To see the current version of the ICD-9-CM diagnosis codes and any changes that have been applied, type `icd9 query`.

`icd9 check`, `icd9 clean`, and `icd9 generate` are data management commands. `icd9 check` verifies that a variable contains defined ICD-9-CM diagnosis codes and provides a summary of any problems encountered. `icd9 clean` standardizes the format of the codes. `icd9 generate` can create a binary indicator variable for whether the code is in a specified set of codes, a variable containing a corresponding higher-level code, or a variable containing the description of the code.

`icd9 lookup` and `icd9 search` are interactive utilities. `icd9 lookup` displays descriptions of the codes specified on the command line. `icd9 search` looks for relevant ICD-9-CM diagnosis codes from keywords given on the command line.

Quick start

Determine whether ICD-9-CM diagnosis codes in `diag1` are invalid, and store reasons in `invalid`

```
icd9 check diag1, generate(invalid)
```

Standardize display of codes in `diag2` to remove all periods, and align codes by padding with spaces

```
icd9 clean diag2, pad
```

Create `descr3` as the diagnosis code prepended to short description of diagnosis code in `diag3`

```
icd9 generate descr3 = diag3, description long
```

Create `diabetes` as an indicator for a diabetes diagnosis in `diag4` using ICD-9-CM codes 250.xx

```
icd9 generate diabetes = diag4, range(25000/25093)
```

Look up descriptions for ICD-9-CM diagnosis codes E827.0 to E828.9

```
icd9 lookup E8270/E8289
```

Menu

Data > ICD codes > ICD-9

Syntax

Verify that variable contains defined codes

```
icd9 check varname [ if ] [ in ] [ , any list generate(newvar) ]
```

Clean variable and verify format of codes

```
icd9 clean varname [ if ] [ in ] [ , dots pad ]
```

Generate new variable from existing variable

```
icd9 generate newvar = varname [ if ] [ in ] , category
```

```
icd9 generate newvar = varname [ if ] [ in ] , description [ long end ]
```

```
icd9 generate newvar = varname [ if ] [ in ] , range(codelist)
```

Display code descriptions

```
icd9 lookup codelist
```

Search for codes from descriptions

```
icd9 search [ " ]text[ " ] [ [ " ]text[ " ] ... ] [ , or ]
```

Display ICD-9 code source

```
icd9 query
```

codelist is

<i>icd9code</i>	(the particular code)
<i>icd9code*</i>	(all codes starting with)
<i>icd9code/icd9code</i>	(the code range)

or any combination of the above, such as 001* 018/019 E* 018.02. *icd9codes* must be typed with leading 0s. For example, type 001; typing 1 will result in an error.

collect is allowed with icd9 check, icd9 clean, and icd9 lookup; see [U] 11.1.10 Prefix commands.

The icd9 suite of commands does not allow alias variables; see [D] [frunalias](#) for advice on how to get around this restriction.

Options

Options are presented under the following headings:

Options for icd9 check
Options for icd9 clean
Options for icd9 generate
Option for icd9 search

Options for icd9 check

`any` tells `icd9 check` to verify that the codes fit the format of ICD-9-CM diagnosis codes but not to check whether the codes are defined.

`list` specifies that `icd9 check` list the observation number, the invalid or undefined ICD-9-CM diagnosis code, and the reason the code is invalid or whether it is an undefined code.

`generate`(*newvar*) specifies that `icd9 check` create a new variable containing, for each observation, 0 if the observation contains a defined code or is missing. Otherwise, it contains a number from 1 to 10. The positive numbers indicate the kind of problem and correspond to the listing produced by `icd9 check`.

Options for icd9 clean

`dots` specifies that the period be included in the final format. If `dots` is not specified, then all periods are removed.

`pad` specifies that `icd9 clean` pad the codes with spaces, front and back, to make the (implied) dots align vertically in listings. Specifying `pad` makes the resulting codes look better when used with most other Stata commands.

Options for icd9 generate

`category`, `description`, and `range`(*codelist*) specify the contents of the new variable that `icd9 generate` is to create. You do not need to `icd9 clean varname` before using `icd9 generate`; it will accept any supported format or combination of formats.

`category` creates a new variable that contains ICD-9-CM diagnosis category codes. The resulting variable may be used with the other `icd9` subcommands. For diagnosis codes, the category code is the first three characters, except for E-codes, when it is the first four characters.

`description` creates *newvar* containing descriptions of the ICD-9-CM diagnosis codes.

`long` is for use with `description`. It specifies that the code be prepended to the text describing the code.

`end` modifies `long` (specifying `end` implies `long`) and places the code at the end of the string.

`range`(*codelist*) creates a new indicator variable equal to 1 when the ICD-9-CM diagnosis code is in the range specified, equal to 0 when the ICD-9-CM diagnosis code is not in the range, and equal to missing when *varname* is missing.

Option for icd9 search

`or` specifies that ICD-9-CM diagnosis codes be searched for descriptions that contain any word specified with `icd9 search`. The default is to list only descriptions that contain all the words specified.

Remarks and examples

Remarks are presented under the following headings:

Using icd9 and icd9p
Verifying and cleaning variables
Interactive utilities
Creating new variables

If you have not yet read *Introduction to ICD coding* in [D] **icd**, please do so before using the `icd9` commands.

Using icd9 and icd9p

The ICD-9-CM coding system includes diagnosis and procedure codes. Some examples of diagnosis codes are 552.3 (Diaphragmatic hernia with obstruction) and E871.0 (Foreign object left in body during surgical operation). Some example of procedure codes are 01.2 (Craniotomy and craniectomy) and 55.23 (Closed renal biopsy).

Many datasets record (and some people write) codes without the period; for example, diagnosis code 550.1 may appear as 5501. The `icd9` commands understand both ways of recording codes. The commands are also insensitive to codes recorded with or without leading and trailing blanks. For E-codes and V-codes, the `icd9` commands are case insensitive. All the following codes are acceptable formats.

diagnosis	procedure
001	27.62
001.	72
00581	32.6
552.3	97.11
E800.2	872
e8002	5523
v82.2	08.51

Important note: What constitutes a valid code changes between versions. For the rest of this entry, a defined code is any code that is currently valid, was valid at some point since version 16 (V16, effective 1 October 1998), or has meaning as a grouping of codes. The list of valid codes and their associated descriptions is from the US Centers for Medicare and Medicaid Services (CMS). These codes are jointly maintained and distributed by the US Centers for Disease Control and Prevention's National Center for Health Statistics and by CMS ([Centers for Disease Control and Prevention 2013](#)).

In `icd9`, descriptions that end with an asterisk (*) are used to denote codes that are invalid for medical coding purposes but are defined as a category code or a subcategory code that has been further subdivided. For example, diagnosis code 001 (Cholera) is invalid without a fourth digit but is defined as a category code, so its description appears as `cholera*`. CMS does not distribute short descriptions of category and subcategory codes that are defined but not valid for coding. To ensure that Stata reports that these codes are defined, we added them to the dataset `icd9` uses with a description of `*`.

Codes that were valid in the past, but no longer are, have descriptions that end with a hash mark (#). For example, the diagnosis code 645.01 was deleted between V16 and V18. It remains a defined code, and its description appears as `prolonged preg-delivered#`.

To view the current version of ICD-9-CM diagnosis codes in Stata, its source, and a log of changes that have been made to the list of ICD-9-CM codes since the `icd9` commands were implemented, type

```
. icd9 query
ICD9 Diagnostic Code Mapping Data for use with Stata, History
(output omitted)
V32
Dataset obtained 26aug2014 from
<http://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/
> codes.html>, by selecting the 'Version 32...' file. Can be gotten
directly via
<http://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/
> Downloads/ICD-9-CM-v32-master-descriptions.zip>. After unzipping, the
useful file name is "CMS32_DESC_SHORT_DX.txt (there are other files we
did not use)."
09oct2014: V32 put into Stata distribution
BETWEEN V31 and V32: There were no additional codes.
BETWEEN V31 and V32: 0 codes were deleted.
BETWEEN V31 and V32: There were no description changes.
(output omitted)
```

Throughout the remainder of this entry, we use `nhds2010.dta`, an extract of adult same-day discharges from the 2010 National Hospital Discharge Survey (NHDS). Below we [describe](#) the data and [list](#) the first five observations for the diagnosis and procedure code variables.

```
. use https://www.stata-press.com/data/r19/nhds2010
(Adult same-day discharges, 2010)
. describe
Contains data from https://www.stata-press.com/data/r19/nhds2010.dta
Observations:      2,210      Adult same-day discharges, 2010
Variables:         15        30 Jan 2024 15:03
                        (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
ageu	byte	%8.0g	ageu	Units for age
age	byte	%8.0g		Age
sex	byte	%8.0g	sex	Sex
race	byte	%8.0g	race	Race
month	byte	%8.0g		Discharge month
status	byte	%8.0g	status	Discharge status
region	byte	%8.0g	region	Region
atype	byte	%8.0g	atype	Type of admission
dx1	str5	%9s		Diagnosis 1
dx2	str5	%9s		Diagnosis 2
dx3	str5	%9s		Diagnosis 3 (imported incorrectly)
dx3corr	str5	%9s		Diagnosis 3 (corrected)
pr1	str4	%9s		Procedure 1
wgt	int	%12.0g		Frequency weight
recid	float	%9.0g		Order of record (raw data)

Sorted by: recid

```
. list recid dx1 dx2 dx3 pr1 in 1/5
```

	recid	dx1	dx2	dx3	pr1
1.	84	4414	99811	14275	3834
2.	105	25013	3572	-2506	
3.	255	51909	1489	-V146	
4.	651	9678	E8528	8	
5.	696	V271	64421	16564	7359

Verifying and cleaning variables

`icd9 check` verifies that *varname* contains defined ICD-9-CM codes and, if not, provides a full report on the problems. It is a good idea to begin with this command and fix any potential problems before proceeding to other `icd9` commands. However, the check subcommand is also useful for tracking down problems when any of the other `icd9` commands tell you that the “variable does not contain ICD-9 codes”.

`icd9 clean` modifies the variable to ensure consistency and to make subsequent output look better. This is not strictly necessary because all `icd9` commands work equally well with cleaned or uncleaned codes. `icd9 clean` also can be used to verify that the codes in a variable conform with the ICD-9-CM diagnosis format, without checking to see whether the codes are defined.

► Example 1: Checking the validity of a variable

We noticed when we listed our data that `dx3` appears to be padded with dashes instead of spaces. As a preemptive step, we replace the dashes with spaces by using the `subinstr()` function because the `icd9` commands ignore spaces.

```
. replace dx3=subinstr(dx3,"-", " ",.)
(1,009 real changes made)
. list recid dx1 dx2 dx3 pr1 in 1/5
```

	recid	dx1	dx2	dx3	pr1
1.	84	4414	99811	14275	3834
2.	105	25013	3572	2506	
3.	255	51909	1489	V146	
4.	651	9678	E8528	8	
5.	696	V271	64421	16564	7359

Now that we have replaced the characters we know will be a problem, we can `icd9 check` the diagnosis variables. We add the `generate()` option so that we can identify any observations with invalid codes.

```
. icd9 check dx1, generate(prob1)
(dx1 contains defined ICD-9-CM codes; no missing values)
. icd9 check dx2, generate(prob2)
(dx2 contains defined ICD-9-CM codes; 179 missing values)
```

```
. icd9 check dx3, generate(prob3)
(dx3 contains 277 missing values)
dx3 contains invalid codes:
```

1. Invalid placement of period	0
2. Too many periods	0
3. Code too short	79
4. Code too long	0
5. Invalid 1st char (not 0-9, E, or V)	0
6. Invalid 2nd char (not 0-9)	128
7. Invalid 3rd char (not 0-9)	0
8. Invalid 4th char (not 0-9)	0
9. Invalid 5th char (not 0-9)	0
10. Code not defined	793
	<hr/>
Total	1,000

We see that all codes in dx1 are valid and all discharges have a primary diagnosis recorded. Likewise, all codes in dx2 are defined, and we see that 179 observations did not have a second diagnosis.

However, icd9 check reports that 1,000 of the 2,210 observations on dx3 have some sort of problem: 79 codes are too short, 128 have an invalid second character, and 793 are undefined. After some investigation, we discover that when we imported the data, we started reading from the wrong position in the file. Hereafter, we use the correctly imported variable, dx3corr.

```
. icd9 check dx3corr
(dx3corr contains defined ICD-9-CM codes; 356 missing values)
```

◀

Rather than typing the icd9 check command once for each variable, we could have checked all three simultaneously. See [Working with multiple codes](#) in [D] [icd](#).

► Example 2: Standardizing the format of codes

If we plan to do any reporting with these codes later, we may want to make them more readable. Suppose we want to report the primary diagnosis and procedure for each discharge. We can use icd9 cclean with the dots and pad options to add the period between the category code and any subsequent digits and to align the periods.

```
. icd9 cclean dx1, dots pad
(2210 changes made)
```

Using icd9 cclean with undefined codes will not result in an error message. So if you are using codes from a country other than the United States, the cclean subcommand can still be used to standardize the format of your codes and check for correct placement of the period.

▶

Interactive utilities

icd9 search looks for relevant ICD-9-CM diagnosis codes from the description given on the command line, and icd9 lookup lists the descriptions of codes given on the command line. The two commands complement each other.

► Example 3: Finding diagnosis codes

Suppose that we want to identify the observations for which the primary diagnosis is congestive heart failure (CHF). As part of a quick exploratory analysis, we can use `icd9 search` to find ICD-9-CM codes that we may want to use to define our study population. We use the terms “heart failure” and “chf”. We enclose “heart failure” in quotation marks and use the `or` option so that `icd9 search` looks for either term.

```
. icd9 search "heart failure" chf, or
5 matches found:
  398.91    rheumatic heart failure
  428       heart failure*
  428.0     chf nos
  428.1     left heart failure
  428.9     heart failure nos
```

Because the descriptions are abbreviated, we are concerned that some of the 428 codes may be left out. So we use `icd9 lookup` to list a range of codes.

```
. icd9 lookup 428*
19 matches found:
  428       heart failure*
  428.0     chf nos
  428.1     left heart failure
  428.2     *
  428.20    systolic hrt failure nos
  428.21    ac systolic hrt failure
  428.22    chr systolic hrt failure
  428.23    ac on chr syst hrt fail
  428.3     *
  428.30    diastolic hrt failure nos
  428.31    ac diastolic hrt failure
  428.32    chr diastolic hrt fail
  428.33    ac on chr diast hrt fail
  428.4     *
  428.40    syst/diast hrt fail nos
  428.41    ac syst/diastol hrt fail
  428.42    chr syst/diastl hrt fail
  428.43    ac/chr syst/dia hrt fail
  428.9     heart failure nos
```

The same result could be found by typing

```
. icd9 lookup 428/4289
```

if we knew that 428.9 was the last code in the 428 category.



Creating new variables

`icd9 generate` produces new variables based on existing variables containing (cleaned or uncleaned) ICD-9-CM diagnosis codes. `icd9 generate, category` creates *newvar* containing the category code that corresponds to the code in the existing variable. `icd9 generate, description` creates *newvar* containing the abbreviated textual description of the ICD-9-CM diagnosis code. `icd9 generate, range()` produces numeric *newvar* containing 1 if *varname* records an ICD-9-CM diagnosis code in the range listed and containing 0 otherwise.

► Example 4: Creating an indicator variable

We review the list of codes we found in [example 3](#) and decide that we will use 398.91 and all of the 428 codes in our definition of a CHF diagnosis. Now we can use `icd9 generate` with the `range()` option to create an indicator variable.

```
. icd9 generate chf = dx1, range(398.91 428*)
. tabulate chf [fweight=wgt]
```

chf	Freq.	Percent	Cum.
0	563,048	97.88	97.88
1	12,192	2.12	100.00
Total	575,240	100.00	

After [tabulating](#) the results, we see that about 2.1% of all same-day discharges were for CHF in 2010.



□ Technical note

The dataset that supports `icd9` includes all codes that were added or deleted between V16 and the last version (V32). However, the descriptions were updated with each new version. If you are using `icd9 generate` with option `description` for codes from a version other than 32, please review the `icd9` query log for any changes to descriptions between the version you are using and version 32.



► Example 5: Combining commands for reporting

The `icd9 generate` commands are useful in isolation, but their real power comes when they are combined. For example, suppose that we want to make a graph showing the number of discharges in each diagnosis category for ICD-9-CM chapter 4, “Diseases of Blood and Blood-Forming Organs”. We could use several `generate` commands and string functions, but `icd9 generate` greatly reduces our work.

First, we extract the category code from the detailed diagnosis code. Then, because the `icd9` commands work equally well with complete codes or category codes, we can use `icd9 generate` with the `range(280/289)` option to create an indicator variable for whether the discharge had a primary diagnosis in chapter 4.

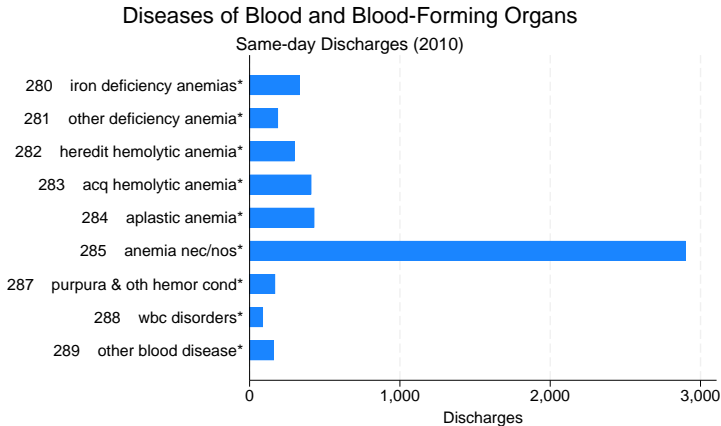
```
. icd9 generate dx1cat = dx1, category
. icd9 generate ch4 = dx1cat, range(280/289)
```

Next, we create a variable with the descriptions of the category codes in chapter 4.

```
. icd9 generate ch4des = dx1cat if ch4==1, description long
```

Finally, we use `graph hbar` to make a horizontal bar graph showing the frequencies of same-day discharges by diagnosis category.

```
. graph hbar (count) [fweight=wt], over(ch4des) ytitle(Discharges)
> title(Diseases of Blood and Blood-Forming Organs, span)
> subtitle(Same-day Discharges (2010), span)
```



See [G-2] **graph bar** for information about customizing the graph above. For more information about graphing results, see [G-2] **graph**.



Stored results

`icd9 check` stores the following in `r()`:

Scalars

<code>r(e#)</code>	number of errors of type #
<code>r(esum)</code>	total number of errors

`icd9 clean` stores the following in `r()`:

Scalars

<code>r(N)</code>	number of changes
-------------------	-------------------

`icd9 lookup` stores the following in `r()`:

Scalars

<code>r(N)</code>	number of codes found
-------------------	-----------------------

References

- Centers for Disease Control and Prevention. 2013. International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM). <https://www.cdc.gov/nchs/icd/icd9cm.htm>.
- National Center for Health Statistics. 2011. International Classification of Diseases, Ninth Revision, Clinical Modification. https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/ICD9-CM/2011/.
- . 2012. National Hospital Discharge Survey: 2010 Public Use Data File Documentation. https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/NHDS/NHDS_2010_Documentation.pdf.

Also see

- [D] [icd](#) — Introduction to ICD commands
- [D] [icd9p](#) — ICD-9-CM procedure codes
- [D] [icd10cm](#) — ICD-10-CM diagnosis codes
- [D] [frunalias](#) — Change storage type of alias variables

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow and NetCourseNow are trademarks of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.

For suggested citations, see the FAQ on [citing Stata documentation](#).

