Title

stata.com

icd9 — ICD-9-CM diagnostic and procedure codes

Syntax

Verify that variable contains defined codes

{icd9|icd9p} check *varname* [, any list generate(*newvar*)]

Verify and clean variable

{icd9|icd9p} clean *varname* [, dots pad]

Generate new variable from existing variable

{icd9|icd9p} generate *newvar* = *varname*, main

{icd9|icd9p} generate *newvar* = *varname*, description [long end]

{icd9|icd9p} generate *newvar* = *varname*, range(*icd9rangelist*)

Display code descriptions

{icd9|icd9p} lookup *icd9rangelist*

Search for codes from descriptions

{icd9|icd9p} search ["]*text*["] [["]*text*["] ...] [, or]

Display ICD-9 code source

{icd9|icd9p} query

where *icd9rangelist* is

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

or any combination of the above, such as 001* 018/019 E* 018.02. *icd9codes* must be typed with leading zeros: 1 is an error; type 001 (diagnostic code) or 01 (procedure code).

icd9 is for use with ICD-9 *diagnostic* codes, and icd9p is for use with *procedure* codes. The two commands' syntaxes parallel each other.

Menu

{icd9 | icd9p} check

Data > Other utilities > ICD9 utilities > Verify variable is valid

{icd9 | icd9p} clean

Data > Other utilities > ICD9 utilities > Clean and verify variable

{icd9 | icd9p} generate

Data > Other utilities > ICD9 utilities > Generate new variable from existing

{icd9 | icd9p} lookup

Data > Other utilities > ICD9 utilities > Display code descriptions

{icd9 | icd9p} search

Data > Other utilities > ICD9 utilities > Search for codes from descriptions

{icd9 | icd9p} query

Data > Other utilities > ICD9 utilities > Display ICD-9 code source

Description

`icd9` and `icd9p` help when working with ICD-9-CM codes.

ICD-9 codes come in two forms: diagnostic codes and procedure codes. In this system, 001 (cholera) and 941.45 (deep 3rd deg burn nose) are examples of diagnostic codes, although some people write (and datasets record) 94145 rather than 941.45. Also, 01 (incise-excis brain/skull) and 55.01 (nephrotomy) are examples of procedure codes, although some people write 5501 rather than 55.01. `icd9` and `icd9p` understand both ways of recording codes.

Important note: What constitutes a valid ICD-9 code changes over time. For the rest of this entry, a *defined code* is any code that is either currently valid, was valid at some point since version V16 (effective October 1, 1998), or has meaning as a grouping of codes. Some examples would help. The diagnosis code 001, though not valid on its own, is useful because it denotes cholera. It is kept as a defined code whose description ends with an asterisk (*). The diagnosis code 645.01 was deleted between versions V16 and V18. It remains as a defined code, and its description ends with a hash mark (#).

`icd9` and `icd9p` parallel each other; `icd9` is for use with diagnostic codes, and `icd9p` is for use with procedure codes.

`icd9[p] check` verifies that existing variable *varname* contains defined ICD-9 codes. If not, `icd9[p] check` provides a full report on the problems. `icd9[p] check` is useful for tracking down problems when any of the other `icd9[p]` commands tell you that the "variable does not contain ICD-9 codes". `icd9[p] check` verifies that each recorded code actually exists in the defined code list.

`icd9[p] clean` also verifies that existing variable *varname* contains valid ICD-9 codes, and, if it does, `icd9[p] clean` modifies the variable to contain the codes in either of two standard formats. All `icd9[p]` commands work equally well with cleaned or uncleaned codes. There are many ways of writing the same ICD-9 code, and `icd9[p] clean` is designed to ensure consistency and to make subsequent output look better.

`icd9[p] generate` produces new variables based on existing variables containing (cleaned or uncleaned) ICD-9 codes. `icd9[p] generate, main` produces *newvar* containing the main code. `icd9[p] generate, description` produces *newvar* containing a textual description of the ICD-9 code. `icd9[p] generate, range()` produces numeric *newvar* containing 1 if *varname* records an ICD-9 code in the range listed and 0 otherwise.

`icd9[p] lookup` and `icd9[p] search` are utility routines that are useful interactively. `icd9[p] lookup` simply displays descriptions of the codes specified on the command line, so to find out what diagnostic E913.1 means, you can type `icd9 lookup e913.1`. The data that you have in memory are irrelevant—and remain unchanged—when you use `icd9[p] lookup`. `icd9[p] search` is similar to `icd9[p] lookup`, except that it turns the problem around; `icd9[p] search` looks for relevant ICD-9 codes from the description given on the command line. For instance, you could type `icd9 search liver` or `icd9p search liver` to obtain a list of codes containing the word "liver".

`icd9[p] query` displays the identity of the source from which the ICD-9 codes were obtained and the textual description that `icd9[p]` uses.

ICD-9 codes are commonly written in two ways: with and without periods. For instance, with diagnostic codes, you can write 001, 86221, E8008, and V822, or you can write 001., 862.21, E800.8, and V82.2. With procedure codes, you can write 01, 50, 502, and 5021, or 01., 50., 50.2, and 50.21. The `icd9[p]` command does not care which syntax you use or even whether you are consistent. Case also is irrelevant: v822, v82.2, V822, and V82.2 are all equivalent. Codes may be recorded with or without leading and trailing blanks.

`icd9[p]` works with V32, V31, V30, V29, V28, V27, V26, V25, V24, V22, V21, V19, V18, and V16 codes.

Options

Options are presented under the following headings:

Options for icd9[p] check

`any` tells `icd9[p] check` to verify that the codes fit the format of ICD-9 codes but not to check whether the codes are actually defined. This makes `icd9[p] check` run faster. For instance, diagnostic code 230.52 (or 23052, if you prefer) looks valid, but there is no such ICD-9 code. Without the `any` option, 230.52 would be flagged as an error. With `any`, 230.52 is not an error.

`list` reports any invalid codes that were found in the data by `icd9[p] check`. For example, 1, 1.1.1, and perhaps 230.52, if `any` is not specified, are to be individually listed.

`generate(`*newvar*`)` specifies that `icd9[p] check` create new variable *newvar* containing, for each observation, 0 if the code is defined and a number from 1 to 10 otherwise. The positive numbers indicate the kind of problem and correspond to the listing produced by `icd9[p] check`. For instance, 10 means that the code could be valid, but it turns out not to be on the list of defined codes.

Options for icd9[p] clean

`dots` specifies whether periods are to be included in the final format. Do you want the diagnostic codes recorded, for instance, as 86221 or 862.21? Without the `dots` option, the 86221 format would be used. With the `dots` option, the 862.21 format would be used.

`pad` specifies that the codes are to be padded with spaces, front and back, to make the codes line up vertically in listings. Specifying `pad` makes the resulting codes look better when used with most other Stata commands.

Options for icd9[p] generate

`main`, `description`, and `range(`*icd9rangelist*`)` specify what `icd9[p] generate` is to calculate. *varname* always specifies a variable containing ICD-9 codes.

`main` specifies that the main code be extracted from the ICD-9 code. For procedure codes, the main code is the first two characters. For diagnostic codes, the main code is usually the first three or four characters (the characters before the dot if the code has dots). In any case, `icd9[p] generate` does not care whether the code is padded with blanks in front or how strangely it might be written; `icd9[p] generate` will find the main code and extract it. The resulting variable is itself an ICD-9 code and may be used with the other `icd9[p]` subcommands. This includes `icd9[p] generate, main`.

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

`long` is for use with `description`. It specifies that the new variable, in addition to containing the text describing the code, contain the code, too. Without `long`, *newvar* in an observation might contain "bronchus injury-closed". With `long`, it would contain "862.21 bronchus injury-closed".

`end` modifies `long` (specifying `end` implies `long`) and places the code at the end of the string: "bronchus injury-closed 862.21".

`range(`*icd9rangelist*`)` allows you to create indicator variables equal to 1 when the ICD-9 code is in the inclusive range specified.

Option for icd9[p] search

`or` specifies that ICD-9 codes be searched for entries that contain any word specified after `icd9[p] search`. The default is to list only entries that contain all the words specified.

Remarks and examples

stata.com

Let's begin with the diagnostic codes that `icd9` processes. The format of an ICD-9 diagnostic code is

$$[\,\texttt{blanks}\,]\{\texttt{0–9,V,v}\}\{\texttt{0–9}\}\{\texttt{0–9}\}[\,\texttt{.}\,][\,\texttt{0--9}[\,\texttt{0--9}\,]\,][\,\texttt{blanks}\,]$$

or

$$[\,\texttt{blanks}\,]\{\texttt{E,e}\}\{\texttt{0–9}\}\{\texttt{0–9}\}\{\texttt{0–9}\}[\,\texttt{.}\,][\,\texttt{0--9}\,][\,\texttt{blanks}\,]$$

`icd9` can deal with ICD-9 diagnostic codes written in any of the ways that this format allows. Items in square brackets are optional. The code might start with some number of blanks. Braces, { }, indicate required items. The code then has a digit from `0` to `9`, the letter `V` (uppercase or lowercase, first line), or the letter `E` (uppercase or lowercase, second line). After that, it has two or more digits, perhaps followed by a period, and then it may have up to two more digits (perhaps followed by more blanks).

All the following codes meet the above definition:

```
001
001.
   001
001.9
         0019
86222
862.22
E800.2
e8002
V82
v82.2
V822
```

Meeting the above definition does not make the code valid. There are 133,100 possible codes meeting the above definition, of which fewer than 20,000 are currently defined.

Examples of currently defined diagnostic codes include

Code	Description
001	cholera*
001.0	cholera d/t vib cholerae
001.1	cholera d/t vib el tor
001.9	cholera nos
...	
999	complic medical care nec*
...	
V01	communicable dis contact*
V01.0	cholera contact
V01.1	tuberculosis contact
V01.2	poliomyelitis contact
V01.3	smallpox contact
V01.4	rubella contact
V01.5	rabies contact
V01.6	venereal dis contact
V01.7	viral dis contact nec#
V01.71	varicella contact/exp
V01.79	viral dis contact nec
V01.8	communic dis contact nec#
V01.81	contact/exposure-anthrax
V01.82	exposure to sars
V01.83	e. coli contact/exp
V01.84	meningococcus contact
V01.89	communic dis contact nec
V01.9	communic dis contact nos
...	
E800	rr collision nos*
E800.0	rr collision nos-employ
E800.1	rr coll nos-passenger
E800.2	rr coll nos-pedestrian
E800.3	rr coll nos-ped cyclist
E800.8	rr coll nos-person nec
E800.9	rr coll nos-person nos
...	

The *main code* refers to the part of the code to the left of the period. 001, 002, . . . , 999; V01, . . . , V82; and E800, . . . , E999 are main codes.

The main code corresponding to a detailed code can be obtained by taking the part of the code to the left of the period, except for codes beginning with 176, 764, 765, V29, and V69. Those main codes are not defined, yet there are more detailed codes under them:

Code	Description
176	CODE DOES NOT EXIST:
176.0	skin - kaposi's sarcoma
176.1	sft tisue - kpsi's srcma
...	
764	CODE DOES NOT EXIST:
764.0	lt-for-dates w/o fet mal*
764.00	light-for-dates wtnos
...	
765	CODE DOES NOT EXIST:
765.0	extreme immaturity*
765.00	extreme immatur wtnos
...	
V29	CODE DOES NOT EXIST:
V29.0	nb obsrv suspct infect
V29.1	nb obsrv suspct neurlgcl
...	
V69	CODE DOES NOT EXIST:
V69.0	lack of physical exercise
V69.1	inapprt diet eat habits
...	

Our solution is to define five new codes:

Code	Description
176	kaposi's sarcoma (Stata)*
764	light-for-dates (Stata)*
765	immat & preterm (Stata)*
V29	nb suspct cnd (Stata)*
V69	lifestyle (Stata)*

Things are less confusing with respect to the procedure codes processed by `icd9p`. The format of ICD-9 procedure codes is

$$[\texttt{blanks}]\{\texttt{0–9}\}\{\texttt{0–9}\}[\texttt{.}][\texttt{0--9}[\texttt{0--9}]][\texttt{blanks}]$$

Thus there are 10,000 possible procedure codes, of which fewer than 5,000 are currently valid. The first two digits represent the main code, of which 100 are feasible and 98 are currently used (00 and 17 are not used).

Descriptions

The description given for each of the codes is as found in the original source. The procedure codes contain the addition of five new codes created by Stata. An asterisk on the end of a description indicates that the corresponding ICD-9 diagnostic code has subcategories. A hash mark (#) at the end of a description denotes a code that is not valid in the most current version but that was valid at some time between version V16 and the present version.

`icd9[p] query` reports the original source of the information on the codes:

```
. icd9 query

_dta:
  1.  ICD9 Diagnostic Code Mapping Data for use with Stata, History
  2.  ------ V16 ----------------------------------------------------------
  3.  Dataset obtained 24aug1999 from http://www.hcfa.gov/stats/pufiles.htm,
      file http://www.hcfa.gov/stats/icd9v16.exe
  4.  Codes 176, 764, 765, V29, and V69 defined by StataCorp: 176 [kaposi's
      sarcoma (Stata)*], 765 [immat & preterm (Stata)*], 764 [light-for-dates
      (Stata)*], V29 [nb suspct cnd (Stata)*], V69 [lifestyle (Stata)*]
  5.  ------ V18 ----------------------------------------------------------
 (output omitted)
 12.  ------ V19 ----------------------------------------------------------
 13.  Dataset obtained 3jan2002 from http://www.hcfa.gov/stats/pufiles.htm,
      file http://www.hcfa.gov/stats/icd9v19.zip, file 9v19diag.txt
 14.  27feb2002: V19 put into Stata distribution
 (output omitted)

. icd9p query

_dta:
  1.  ICD9 Procedure Code Mapping Data for use with Stata, History
  2.  ------ V16 ----------------------------------------------------------
  3.  Dataset obtained 24aug1999 from http://www.hcfa.gov/stats/pufiles.htm,
      file http://www.hcfa.gov/stats/icd9v16.exe
  4.  ------ V18 ----------------------------------------------------------
  5.  Dataset obtained 10may2001 from http://www.hcfa.gov/stats/pufiles.htm,
      file http://www.hcfa.gov/stats/icd9v18.zip, file V18SURG.TXT
  6.  11jun2001: V18 data put into Stata distribution
  7.  BETWEEN V16 and V18: 9 codes added: 3971 3979 4107 4108 4109 4697 6096
      6097 9975
 (output omitted)
```

▷ Example 1

We have a dataset containing up to three diagnostic codes and up to two procedures on a sample of 1,000 patients:

```
. use http://www.stata-press.com/data/r13/patients
. list in 1/10
```

	patid	diag1	diag2	diag3	proc1	proc2
1.	1	65450			9383	
2.	2	23v.6	37456		8383	17
3.	3	V10.02				
4.	4	102.6			629	
5.	5	861.01				
6.	6	38601	2969		9337	
7.	7	705			7309	8385
8.	8	v53.32			7878	951
9.	9	20200	7548	E8247	0479	
10.	10	464.11	20197		4641	

Do not try to make sense of these data because, in constructing this example, the diagnostic and procedure codes were randomly selected.

First, variable diag1 is recorded sloppily—sometimes the dot notation is used and sometimes not, and sometimes there are leading blanks. That does not matter. We decide to begin by using icd9 clean to clean up this variable:

```
. icd9 clean diag1
diag1 contains invalid ICD-9 codes
r(459);
```

icd9 clean refused because there are invalid codes among the 1,000 observations. We can use icd9 check to find and flag the problem observations (or observation, as here):

```
. icd9 check diag1, gen(prob)
diag1 contains invalid codes:
    1.  Invalid placement of period                    0
    2.  Too many periods                               0
    3.  Code too short                                 0
    4.  Code too long                                  0
    5.  Invalid 1st char (not 0-9, E, or V)            0
    6.  Invalid 2nd char (not 0-9)                     0
    7.  Invalid 3rd char (not 0-9)                     1
    8.  Invalid 4th char (not 0-9)                     0
    9.  Invalid 5th char (not 0-9)                     0
   10.  Code not defined                               0
                                                  ------
        Total                                          1
. list patid diag1 prob if prob
```

	patid	diag1	prob
2.	2	23v.6	7

Let's assume that we go back to the patient records and determine that this should have been coded 230.6:

```
. replace diag1 = "230.6" if patid==2
(1 real change made)
. drop prob
```

We now try again to clean up the formatting of the variable:

```
. icd9 clean diag1
(643 changes made)
. list in 1/10
```

	patid	diag1	diag2	diag3	proc1	proc2
1.	1	65450			9383	
2.	2	2306	37456		8383	17
3.	3	V1002				
4.	4	1026			629	
5.	5	86101				
6.	6	38601	2969		9337	
7.	7	705			7309	8385
8.	8	V5332			7878	951
9.	9	20200	7548	E8247	0479	
10.	10	46411	20197		4641	

Perhaps we prefer the dot notation. `icd9 clean` can be used again on `diag1`, and now we will clean up `diag2` and `diag3`:

```
. icd9 clean diag1, dots
(936 changes made)
. icd9 clean diag2, dots
(551 changes made)
. icd9 clean diag3, dots
(100 changes made)
. list in 1/10
```

	patid	diag1	diag2	diag3	proc1	proc2
1.	1	654.50			9383	
2.	2	230.6	374.56		8383	17
3.	3	V10.02				
4.	4	102.6			629	
5.	5	861.01				
6.	6	386.01	296.9		9337	
7.	7	705			7309	8385
8.	8	V53.32			7878	951
9.	9	202.00	754.8	E824.7	0479	
10.	10	464.11	201.97		4641	

We now turn to cleaning the procedure codes. We use `icd9p` (emphasis on the *p*) to clean these codes:

```
. icd9p clean proc1, dots
(816 changes made)
. icd9p clean proc2, dots
(140 changes made)
. list in 1/10
```

	patid	diag1	diag2	diag3	proc1	proc2
1.	1	654.50			93.83	
2.	2	230.6	374.56		83.83	17
3.	3	V10.02				
4.	4	102.6			62.9	
5.	5	861.01				
6.	6	386.01	296.9		93.37	
7.	7	705			73.09	83.85
8.	8	V53.32			78.78	95.1
9.	9	202.00	754.8	E824.7	04.79	
10.	10	464.11	201.97		46.41	

Both `icd9 clean` and `icd9p clean` verify only that the variable being cleaned follows the construction rules for the code; it does not check that the code is itself valid. `icd9[p] check` does that:

```
. icd9p check proc1
(proc1 contains valid ICD-9 procedure codes; 168 missing values)
. icd9p check proc2

proc2 contains invalid codes:
    1.  Invalid placement of period                    0
    2.  Too many periods                               0
    3.  Code too short                                 0
    4.  Code too long                                  0
    5.  Invalid 1st char (not 0-9)                     0
    6.  Invalid 2nd char (not 0-9)                     0
    7.  Invalid 3rd char (not 0-9)                     0
    8.  Invalid 4th char (not 0-9)                     0
   10.  Code not defined                               1
                                                  ------
        Total                                          1
```

`proc2` has an invalid code. We could find it by using `icd9p check, generate()`, just as we did above with `icd9 check, generate()`.

`icd9[p]` can create new variables containing textual descriptions of our diagnostic and procedure codes:

```
. icd9 generate td1 = diag1, description
. sort patid
. list patid diag1 td1 in 1/10
```

	patid	diag1	td1
1.	1	654.50	cerv incompet preg-unsp
2.	2	230.6	ca in situ anus nos
3.	3	V10.02	hx-oral/pharynx malg nec
4.	4	102.6	yaws of bone & joint
5.	5	861.01	heart contusion-closed
6.	6	386.01	actv meniere,cochlvestib
7.	7	705	disorders of sweat gland*
8.	8	V53.32	ftng autmtc dfibrillator
9.	9	202.00	ndlr lym unsp xtrndl org
10.	10	464.11	ac tracheitis w obstruct

`icd9[p] generate, description` does not preserve the sort order of the data (and neither does `icd9[p] check`, unless you specify the `any` option).

Procedure code `proc2` had an invalid code. Even so, `icd9p generate, description` is willing to create a textual description variable:

```
. icd9p gen tp2 = proc2, description
(1 nonmissing value invalid and so could not be labeled)
. sort patid
. list patid proc2 tp2 in 1/10
```

	patid	proc2	tp2
1.	1		
2.	2	17	
3.	3		
4.	4		
5.	5		
6.	6		
7.	7	83.85	musc/tend lng change nec
8.	8	95.1	form & structur eye exam*
9.	9		
10.	10		

`tp2` contains nothing when `proc2` is 17 because 17 is not a valid procedure code.

`icd9`[`p`] `generate` can also create variables containing main codes:

```
. icd9 generate main1 = diag1, main
. list patid diag1 main1 in 1/10
```

	patid	diag1	main1
1.	1	654.50	654
2.	2	230.6	230
3.	3	V10.02	V10
4.	4	102.6	102
5.	5	861.01	861
6.	6	386.01	386
7.	7	705	705
8.	8	V53.32	V53
9.	9	202.00	202
10.	10	464.11	464

`icd9p generate, main` can similarly generate main procedure codes.

Sometimes we might merely be examining an observation:

```
. list diag* if patid==563
```

	diag1	diag2	diag3
563.	526.4		

If we wondered what 526.4 was, we could type

```
. icd9 lookup 526.4
1 match found:
    526.4    inflammation of jaw
```

`icd9[p] lookup` can list ranges of codes:

```
. icd9 lookup 526/526.99
15 matches found:
    526      jaw diseases*
    526.0    devel odontogenic cysts
    526.1    fissural cysts of jaw
    526.2    cysts of jaws nec
    526.3    cent giant cell granulom
    526.4    inflammation of jaw
    526.5    alveolitis of jaw
    526.61   perfor root canal space
    526.62   endodontic overfill
    526.63   endodontic underfill
    526.69   periradicular path nec
    526.8    other jaw diseases*
    526.81   exostosis of jaw
    526.89   jaw disease nec
    526.9    jaw disease nos
```

The same result could be found by typing

```
. icd9 lookup 526*
```

`icd9[p] search` can find a code from the description:

```
. icd9 search jaw disease
4 matches found:
    526      jaw diseases*
    526.8    other jaw diseases*
    526.89   jaw disease nec
    526.9    jaw disease nos
```

◁

Stored results

`icd9 check` and `icd9p check` store the following in `r()`:

Scalars

`r(e#)`	number of errors of type #
`r(esum)`	total number of errors

`icd9 clean` and `icd9p clean` store the following in `r()`:

Scalars

`r(N)`	number of changes

Reference

Gould, W. W. 2000. dm76: ICD-9 diagnostic and procedure codes. *Stata Technical Bulletin* 54: 8–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 77–87. College Station, TX: Stata Press.