Introducing icd9: working with ICD-9 codes and comorbidities in R

Jack O. Wasey

January 9, 2015

1 Introduction

This package is designed to be used with a variety of input data, including multiple possible formats of ICD-9 codes, but some assumptions are made. There are many ways of misinterpreting ICD-9 codes, especially when dealing with ranges. The code in this package carefully considers a wide range of possibilities. **ICD-9 codes are not numeric**. Using numeric values for either decimal or non-decimal form will cause serious problems, hence the predominantly string-based processing here, and a robust set of unit tests.

When calcuating which patients have which comorbidities, the input data is typically structured as follows:

```
patientData

##   visitId   icd9 poa
## 1    1000 402010   Y
## 2    1000  27801   Y
## 3    1000   7208   N
## 4    1000  25001   Y
## 5    1001  34400   N
## 6    1001   4011   Y
## 7    1002   4011   N
```

Only the visitId column is propogated to the results. If 'present-on-arrival' is needed, it must be separated out first. The implicit default, therefore, is to ignore it, and give ICD-9 code regardless of POA status.

The comorbidities can be determined as follows:

```
icd9ComorbidAhrq(icd9df = patientData, isShort = TRUE)[, 1:10]

##   visitId   CHF Valvular  PHTN   PVD  HTN Paralysis NeuroOther Pulmonary    DM
## 1    1000  TRUE    FALSE FALSE FALSE TRUE     FALSE      FALSE     FALSE  TRUE
## 2    1001 FALSE    FALSE FALSE FALSE TRUE      TRUE      FALSE     FALSE FALSE
## 3    1002 FALSE    FALSE FALSE FALSE TRUE     FALSE      FALSE     FALSE FALSE
```

or

```
icd9ComorbidQuanDeyo(patientData, isShort = TRUE)[, 1:10]
```

```
##   visitId    MI   CHF   PVD Stroke Dementia Pulmonary Rheumatic   PUD LiverMild
## 1    1000 FALSE  TRUE FALSE  FALSE    FALSE     FALSE     FALSE FALSE     FALSE
## 2    1001 FALSE FALSE FALSE  FALSE    FALSE     FALSE     FALSE FALSE     FALSE
## 3    1002 FALSE FALSE FALSE  FALSE    FALSE     FALSE     FALSE FALSE     FALSE
```

or using magrittr to chain functions together:

```
patientData %>% icd9FilterPoaYes() %>%
  icd9ComorbidAhrq(isShort = TRUE) %>% extract(1:5)

## Error in icd9Comorbid(icd9df = icd9df, visitId = visitId, icd9Field = icd9Field, :  could not
find function "asCharacterNoWarn"
```

The following code shows gets the same result with default options written out:

```
icd9Comorbid(icd9df = icdFilterPoaYes(patientData),
             visitId = "visitId",
             icd9Field = "icd9",
             icd9Mapping = ahrqComorbid,
             validateMapping = FALSE,
             shortMapping = TRUE)[1:5]
```

2 Converting ICD-9 codes between types

These functions were designed with the problem of incorrectly or bizarrely formatted ICD-9 codes in mind. These functions make the reasonable assumption that short codes of three or fewer characters must be describing only the 'major' part. keepLoneDecimal allows retention of the decimal point even if there are no subsequent characters.

```
icd9DecimalToShort(c("10.20", "100", "123.45"))

## [1] "01020" "100"   "12345"

icd9ShortToDecimal(icd9DecimalToShort(c("10.20", "100", "123.45")))

## [1] "010.20" "100"    "123.45"
```

Only a subset of short codes can suffer dropping of leading zeroes.

```
icd9DecimalToShort(c("1", "22", "22.44", "100"))

## [1] "001"   "022"   "02244" "100"

icd9ShortToDecimal(icd9DecimalToShort(c("1", "1.2", "123.45")))

## [1] "001"    "001.2"  "123.45"

icd9ShortToDecimal(icd9DecimalToShort(c("1", "2.2", "100")))

## [1] "001"   "002.2" "100"
```

3 Validation of ICD-9 codes

```
icd9ValidDecimal("V10.2")

## [1] TRUE

icd9ValidShort(c("099.17", "-1"))

## [1] FALSE FALSE

icd9ValidShort(c("1", "001", "100", "10023"))

## [1] TRUE TRUE TRUE TRUE
```

Validation forces the package user to provide character format ICD-9 codes. If great care is taken, passing some integers could be valid, but given the high chance of mistakes, and the simplicity of dealing entirely with character input, character is enforced:

```
icd9ValidShort(100) # throws an error
```

4 Ranges of ICD-9 codes

These functions generate syntactically valid ICD-9 codes, without including parent codes when the range limit would subset the parent. E.g. "100.99" %i9d% "101.01" does not include "100" or "100.0", both of which imply large subsets than requested by the range command.

```
"10099" %i9s% "10101"

## [1] "10099" "10100" "10101"

"V10" %i9d% "V10.02"

## [1] "V10"    "V10.0"  "V10.00" "V10.01" "V10.02"

# "E987" %i9d% "E988.9"
```

Another way of specifying ranges are to use function calls. These are exactly equivalent to the %i9s% and %i9d% range operators. This example shows the result when the user specifies a range which would include parents but not all their children:

```
icd9ExpandRangeShort("V100", "V1002")

## [1] "V100"  "V1000" "V1001" "V1002"
```

Although V100 would include ten children, the range only returns 4 values. In all other cases, parents are omitted to avoid the range returning broader classifications than intended. A planned feature is to optionally enable returning these parent codes, which would then follow a more numerical pattern (although still distinguishing trailing zeroes).

We can easily find the children of a given higher-level ICD-9 code:

```
icd9ChildrenShort("391", onlyReal = TRUE)

## [1] "3910" "3911" "3912" "3918" "3919"
```

Without the onlyReal flag, all syntactically correct ICD-9 codes are returned, even if not defined. This is relevant because of minor coding errors, or coding in a different year to the master list. A planned feature is to allow testing of an ICD-9 code against the valid codes for the year it was entered, but at present only the 2014 master list is used. This means that some older valid codes may not longer be on the list.

```
icd9ChildrenShort("391", onlyReal = FALSE)[1:10]

##  [1] "391"  "3910" "3911" "3912" "3913" "3914" "3915" "3916" "3917" "3918"
```

5 Human-readable ICD-9

There are various ways of extracting the description of the condition described by an ICD-9 code. the icd9Explain... functions return a data frame with a column for the ICD-9 code, a column for the full length Diagnosis, and a column for the short Description.

```
icd9ExplainDecimal("1.0")

## [1] "Cholera due to vibrio cholerae"

icd9ExplainShort("0019")

## [1] "Cholera, unspecified"
```

```
icd9Explain("1.0", isShort = FALSE)

## Error in icd9DecimalToParts(icd9Decimal, invalidAction = invalidAction):  could not find function
"trim"

icd9Explain(c("0010","4131"), isShort = TRUE)

## Error in icd9ShortToParts(icd9Short, invalidAction = invalidAction):  could not find function
"strip"

# combine with some conversions
icd9ExplainDecimal(icd9ShortToDecimal("0019"))

## Error in icd9ShortToParts(icd9Short):  could not find function "strip"

"4139" %>% icd9ShortToDecimal() %>% icd9ExplainDecimal()

## Error in icd9ShortToParts(icd9Short):  could not find function "strip"

"413.1" %>% icd9DecimalToShort() %>% icd9ExplainShort()

## Error in icd9DecimalToParts(icd9Decimal):  could not find function "trim"

#explain top level code with children
"391" %>% icd9ChildrenShort(onlyReal = TRUE)

## Error in icd9ShortToParts(icd9Short, minorEmpty = ""):  could not find function "strip"
```

```
"391" %>% icd9ExplainShort()

## Error in icd9ShortToParts(icd9Short, invalidAction = invalidAction):  could not find function
"strip"

# default is to condense down to three-digit "major" level
"391" %>% icd9ChildrenShort() %>% icd9ExplainShort()

## Error in icd9ShortToParts(icd9Short, minorEmpty = ""):  could not find function "strip"

"391" %>% icd9ChildrenShort() %>% icd9ExplainShort(doCondense = FALSE)

## Error in icd9ShortToParts(icd9Short, minorEmpty = ""):  could not find function "strip"
```

Arbitrary named list(s) of codes:

```
icd9ExplainDecimal(list(cholera = c("001", "001.0", "001.1", "001.9")))

## $cholera
## [1] "Cholera"

# same using decimal codes without a list
icd9ExplainDecimal(c("001", "001.0", "001.1", "001.9"))

## [1] "Cholera"
```

Now try to explain on a non-existent (but 'valid') ICD-9 code:

```
icd9ExplainDecimal("001.5")

## character(0)
```

6 Chaining commands

With the

`magrittr`

package installed, commands can be chained together in a convenient and readable manner:

```
c("001.1", "391") %>% icd9DecimalToShort %>% icd9ExplainShort

## [1] "Cholera due to vibrio cholerae el tor"  "Rheumatic fever with heart involvement"
```

Find all ICD-9 codes matching 'heart' or 'cardiac' in the short or long descriptions:

```
cardiac <- unique(c(
  icd9Hierarchy[
    grepl(
      pattern="(heart)|(cardiac)",
      x = icd9Hierarchy[["descLong"]],
      ignore.case = TRUE
    ),
    "icd9"],
  icd9Hierarchy[
    grepl(
      pattern="(heart)|(cardiac)",
```

```
      x = icd9Hierarchy[["descShort"]],
      ignore.case = TRUE
    ),
    "icd9"]
))
```

then explain the list, just showing the first ten:

```
cardiac %>% icd9ExplainShort %>% extract(2) %>% head(10)

## [1] "Other gonococcal heart disease"
```

More examples, this time also demonstrating automatic summarization of a long list of ICD-9 codes into the minimum set of explanatory parent codes:

```
quanDeyoComorbid[["Dementia"]] %>%
  icd9ExplainShort() %>%
  extract(c("ICD-9","Description"))

## [1] NA NA
```

Use a range with more than two hundred ICD-9 codes:

```
length("390" %i9d% "392.1")

## [1] 244

"390" %i9d% "392.1" %>%
  icd9DecimalToShort() %>%
  icd9ExplainShort()

## [1] "Rheumatic fever without mention of heart involvement"

"390" %i9d% "392.1" %>% icd9ExplainDecimal()

## [1] "Rheumatic fever without mention of heart involvement"
```

7 Filtering from Present-on-Arrival

Present-on-arrival (POA) is typically a factor, or vector of values such as "Y", "N", "X", "E", or NA. Intermediate codes, such as "exempt", "unknown" and NA mean that "yes" is not the same as "not no." This requires four functions to cover the possibilities stored in icd9PoaChoices:

```
icd9PoaChoices

## [1] "yes"    "no"     "notYes" "notNo"
```

Magrittr allows chaining of the filtering very nicely. Set up some data to demonstrate this:

```
myData <- data.frame(
  visitId = c("v1", "v2", "v3", "v4"),
  icd9 = c("39891", "39790", "41791", "4401"),
  poa = c("Y", "N", NA, "Y"),
  stringsAsFactors = FALSE)
```

Filter for present-on-arrival being "Y"

```
myData %>% icd9FilterPoaYes()

##   visitId  icd9
## 1      v1 39891
## 4      v4  4401
```

Show that 'yes' is not equal to 'not no':

```
myData %>% icd9FilterPoaNotNo()

##   visitId  icd9
## 1      v1 39891
## 3      v3 41791
## 4      v4  4401
```

Chain commands together to get a few columns of comorbidities. This is showing off how magrittr works in the context of this package.

```
myData %>%
  icd9FilterPoaNotNo() %>%
  icd9ComorbidAhrq(isShort = TRUE) %>%
  extract(1:9)

## Error in `[.data.frame`(icd9df, is.na(icd9df[[poaField]]) | icd9df[[poaField]] %nin% :  could
not find function "%nin%"
```

Can fill out some named fields, if we want:

```
myData %>% icd9FilterPoaYes(poaField = "poa") %>%
  icd9ComorbidAhrq(visitId = "visitId", isShort = TRUE) %>%
  extract(1:9)

##   visitId   CHF Valvular  PHTN   PVD   HTN Paralysis NeuroOther Pulmonary
## 1      v1  TRUE    FALSE FALSE FALSE FALSE     FALSE      FALSE     FALSE
## 2      v4 FALSE    FALSE FALSE  TRUE FALSE     FALSE      FALSE     FALSE
```

Call the core icd9Comorbid function with a named mapping:

```
myData %>%
  icd9FilterPoaYes() %>%
  icd9Comorbid(
    icd9Field = "icd9", visitId = "visitId",
    isShort = TRUE, icd9Mapping = quanElixComorbid,
    validateMapping = TRUE, isShortMapping = TRUE
  )  %>%
  extract(1:9)

##   visitId   CHF Arrhythmia Valvular  PHTN   PVD   HTN HTNcx Paralysis
## 1      v1  TRUE      FALSE    FALSE FALSE FALSE FALSE FALSE     FALSE
## 2      v4 FALSE      FALSE    FALSE FALSE  TRUE FALSE FALSE     FALSE
```

8 Arbitrary ICD-9 mapping

The user can provide any ICD-9 mapping they wish. Included in this package is a small data set called icd9Chapters, which lists the ICD-9-CM (and indeed ICD-9) Chapters. These can easily be expanded out and used as a mapping

```
icd9Chapters[1:5]
```

```
## $`Infectious And Parasitic Diseases`
## start   end
## "001" "139"
##
## $Neoplasms
## start   end
## "140" "239"
##
## $`Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders`
## start   end
## "240" "279"
##
## $`Diseases Of The Blood And Blood-Forming Organs`
## start   end
## "280" "289"
##
## $`Mental Disorders`
## start   end
## "290" "319"
```

The next expression is obsolete since all children are now included in the packaged mappings. The user may provide their own mapping which needs this operation.

```
myMap <- icd9:::icd9ChaptersToMap(icd9Chapters[c(1,2:4)])

system.time(
  patientChapters <- icd9Comorbid(
    icd9df = patientData,
    isShort = TRUE,
    icd9Mapping = myMap)
)

##    user  system elapsed
##  50.604   0.000  50.648

# much faster the second time because the internal lookup is memoised:
system.time(
  patientChapters <- icd9Comorbid(
    icd9df = patientData,
    isShort = TRUE,
    icd9Mapping = myMap)
)

##    user  system elapsed
##   0.010   0.000   0.009

patientChapters

##   visitId Infectious And Parasitic Diseases Neoplasms
## 1    1000                             FALSE     FALSE
## 2    1001                             FALSE     FALSE
## 3    1002                             FALSE     FALSE
##   Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders
## 1                                                                 TRUE
## 2                                                                FALSE
## 3                                                                FALSE
##   Diseases Of The Blood And Blood-Forming Organs
## 1                                          FALSE
## 2                                          FALSE
## 3                                          FALSE
```

9 AHRQ comorbidity classification

The AHRQ keeps an updated version of the Elixhauser classification of ICD-9-CM codes into comorbidities, useful for research. They provide the data in the form of SAS code. This package provides just enough code to parse the SAS source code provided by the AHRQ (but probably not much other SAS code), and generate a list of ICD-9 codes for each comorbidity.

```
ahrqComorbid <- parseAhrqSas()
```

```
head(summary(ahrqComorbid))
```

```
##          Length Class    Mode
## CHF      " 120" "-none-" "character"
## Valvular " 545" "-none-" "character"
## PHTN     " 130" "-none-" "character"
## PVD      " 585" "-none-" "character"
## HTN      "  27" "-none-" "character"
## HTNcx    "  72" "-none-" "character"
```

SAS source code has a strong whiff of the 1970s about it. A fragment of a recent AHRQ SAS comorbidity mapping SAS FORMAT is as follows. Note the mix of character and numeric-only ranges, isolated values, all in 'short' ICD-9 code form.

```
PROC FORMAT LIB=library fmtlib;
VALUE $RCOMFMT
"2780 ",
"27800",
"27801",
"27803",
"64910"-"64914",
"V8530"-"V8539",
"V8541"-"V8545",
"V8554",
"79391"          = "OBESE"     /* Obesity      */

"3004 ",
"30112",
"3090 ",
"3091 ",
"311  "          = "DEPRESS"
```

This is parsed using:

```
parseAhrqSas()
```

resulting in a named list. Here is an extract.

```
ahrqComorbid[c("Obesity", "Depression")]
```

```
## $Obesity
##  [1] "2780"  "27800" "27801" "27802" "27803" "27804" "27805" "27806" "27807" "27808"
## [11] "27809" "V8554" "79391" "64910" "64911" "64912" "64913" "64914" "V8530" "V8531"
## [21] "V8532" "V8533" "V8534" "V8535" "V8536" "V8537" "V8538" "V8539" "V8541" "V8542"
## [31] "V8543" "V8544" "V8545"
##
## $Depression
##   [1] "3004"  "30040" "30041" "30042" "30043" "30044" "30045" "30046" "30047" "30048"
##  [11] "30049" "30112" "3090"  "30900" "30901" "30902" "30903" "30904" "30905" "30906"
##  [21] "30907" "30908" "30909" "3091"  "30910" "30911" "30912" "30913" "30914" "30915"
##  [31] "30916" "30917" "30918" "30919" "311"   "3110"  "3111"  "3112"  "3113"  "3114"
##  [41] "3115"  "3116"  "3117"  "3118"  "3119"  "31100" "31110" "31120" "31130" "31140"
##  [51] "31150" "31160" "31170" "31180" "31190" "31101" "31111" "31121" "31131" "31141"
##  [61] "31151" "31161" "31171" "31181" "31191" "31102" "31112" "31122" "31132" "31142"
##  [71] "31152" "31162" "31172" "31182" "31192" "31103" "31113" "31123" "31133" "31143"
##  [81] "31153" "31163" "31173" "31183" "31193" "31104" "31114" "31124" "31134" "31144"
##  [91] "31154" "31164" "31174" "31184" "31194" "31105" "31115" "31125" "31135" "31145"
## [101] "31155" "31165" "31175" "31185" "31195" "31106" "31116" "31126" "31136" "31146"
## [111] "31156" "31166" "31176" "31186" "31196" "31107" "31117" "31127" "31137" "31147"
## [121] "31157" "31167" "31177" "31187" "31197" "31108" "31118" "31128" "31138" "31148"
## [131] "31158" "31168" "31178" "31188" "31198" "31109" "31119" "31129" "31139" "31149"
## [141] "31159" "31169" "31179" "31189" "31199"
```

The icd9Condense functions can be used to make a minimal set of parent codes which describes each group:

```
lapply(ahrqComorbid[c("Obesity", "Depression")], icd9CondenseToMajor, onlyReal = T)

## Error in icd9ShortToParts(icd9Short, invalidAction = invalidAction):  could not find function
"strip"

ahrqComorbid[c("Obesity", "Depression")] %>% icd9ExplainShort(doCondense = FALSE)

## Error in icd9ShortToParts(icd9Short = icd9, invalidAction):  could not find function "strip"
```

10 Elixhauser co-morbidities

Elixhauser originally devleoped this set of co-morbidities to predict long term mortality based on hospital ICD-9-CM coding records. The AHRQ comorbidities are an updated version of this, however the original Elixhauser have been used in many publications. The ICD-9-CM codes have changed slightly over the years.

```
names(elixComorbid)

## Error in eval(expr, envir, enclos):  object 'elixComorbid' not found
```

11 Quan

Quan's paper looked at indices using both ICD-10 and ICD-9-CM. Quan generated updated ICD-9-CM codes for all 30 of Elixhauser and all 17 of Charlson/Deyo's co-morbidities. Thus there are two 'Quan' comorbidity mappings.

```
names(quanDeyoComorbid)
```

```
##  [1] "MI"          "CHF"         "PVD"         "Stroke"      "Dementia"    "Pulmonary"
##  [7] "Rheumatic"   "PUD"         "LiverMild"   "DM"          "DMcx"        "Paralysis"
## [13] "Renal"       "Cancer"      "LiverSevere" "Mets"        "HIV"
```

```
names(quanElixComorbid)
```

```
## Error in eval(expr, envir, enclos):  object 'quanElixComorbid' not found
```