STAT-6494 Advanced Statistical Computing with R – HW 1

Wenjie Wang

24 February 2016

Exercise: Comorbidity Measures

Comorbidity measures are important in health-care research. They can be obtained from hospital discharge records using the diagnosis coding of ICD-9-CM (International Classification of Diseases, Ninth Edition, Clinical Modifications). An influential, widely cited method was proposed by Elixhauser et al. (Comorbidity Measures for Use with Administrative Data. Medical Care, 1998;36:8–27). The method has been refined by the Healthcare Cost and Utilization Project HCUP. Their implementation is in SAS. Construction of the variables is summarized in a PDF table under the section "Variables Created". For simplicity, let us ignore the DRG code for this assignment. For real usage, one would need to consider the DRG code. An R implementation is available in package comorbidities. We will focus on the function ahrq in this package. The input of the function is a data frame where rows are patients and columns are ICD-9-CM codes. Each patient does not necessarily have all columns filled; some ICD-9-CM codes can be NA or empty. The output of the functions is a list, and we are interested in the data frame where rows are patients and columns are comorbidity measures. This function can be made much more efficient by vectorizing. Please write a function `cmbd`. Its input is the same as ahrq and its output is a data frame with only first two comorbidity measures: CHF and VALVE (again, for simplicity). We have a sample input of 100 patients in csv format for your testing. The function will be graded with another sample input randomly selected from my real data. Comment: For real implementation, We had to clean the data: 1) trim the leading and trailing spaces; 2) add zero to the end for codes with length smaller than 5; 3) consider DRG code; and 4) automate the generation of functions.

Implementation: Function `cmbd`

The function body of `cmbd` is shown as follows:

```
cmbd <- function (data) {
    mat <- as.matrix(data)
    numMat <- gsub("^[Vv]", "32", mat)
    numMat <- gsub("^[Ee]", "15", numMat)
    numMat <- as.numeric(numMat)
    dim(numMat) <- dim(mat)
    funList <- codeFun()
    outDat <- sapply(seq_along(funList),
                     function (a, numMat) {
                         funList[[a]](numMat)
                     }, numMat = numMat)
    colnames(outDat) <- c("CHF", "VALVE")
    data.frame(outDat)
}

codeFun <- function () {
    ## internal function for CHF measures
```

```
    chf <- function (numMat) {
        indVec1 <- numMat %in% c(39891, 40201, 40211, 40291, 40401,
                                 40403, 40411, 40413, 40491, 40493)
        indVec2 <- numMat >= 42800 & numMat <= 42890
        indMat <- indVec1 | indVec2
        dim(indMat) <- dim(numMat)
        as.integer(rowSums(indMat, na.rm = TRUE) > 0)
    }
    ## internal function for VALVE measures
    valve <- function (numMat) {
        indVec1 <- numMat >= 9320 & numMat <= 9324
        indVec2 <- numMat >= 39400 & numMat <= 39710
        indVec3 <- numMat >= 42400 & numMat <= 42499
        indVec4 <- numMat >= 74630 & numMat <= 74660
        indVec5 <- numMat %in% c(324220, 324330, 39790)
        indMat <- indVec1 | indVec2 | indVec3 | indVec4 | indVec5
        dim(indMat) <- dim(numMat)
        as.integer(rowSums(indMat, na.rm = TRUE) > 0)
    }
    list(chf = chf, valve = valve)
}
```

Result of Sample Dataset

By using the function `cmbd` defined in last section, the diagnosed result for each patient by ICD9 code in sample dataset is shown as follows.

```
sampleDat <- read.csv("icd9sample.csv")
(myRes <- cmbd(sampleDat))
```

```
##     CHF VALVE
## 1     0     0
## 2     1     0
## 3     0     0
## 4     0     0
## 5     0     0
## 6     0     0
## 7     0     1
## 8     0     0
## 9     0     0
## 10    0     0
## 11    0     0
## 12    0     0
## 13    0     0
## 14    0     0
## 15    0     0
## 16    0     0
## 17    0     0
## 18    1     0
## 19    0     0
## 20    0     1
```

```
## 21    0     0
## 22    0     0
## 23    0     0
## 24    0     0
## 25    0     0
## 26    0     0
## 27    0     0
## 28    0     0
## 29    0     1
## 30    0     0
## 31    1     0
## 32    0     0
## 33    0     0
## 34    0     0
## 35    0     0
## 36    0     0
## 37    0     0
## 38    0     0
## 39    0     0
## 40    0     0
## 41    0     0
## 42    0     0
## 43    0     0
## 44    0     0
## 45    0     0
## 46    0     0
## 47    0     0
## 48    0     1
## 49    0     0
## 50    0     0
## 51    0     0
## 52    0     0
## 53    0     0
## 54    0     0
## 55    0     0
## 56    0     0
## 57    1     1
## 58    0     0
## 59    1     1
## 60    0     0
## 61    0     0
## 62    0     0
## 63    0     0
## 64    0     0
## 65    0     0
## 66    0     0
## 67    0     0
## 68    0     0
## 69    0     0
## 70    0     0
## 71    0     0
## 72    1     0
## 73    0     0
## 74    0     0
```

```
## 75    0     0
## 76    0     0
## 77    0     0
## 78    0     0
## 79    0     0
## 80    0     0
## 81    1     0
## 82    0     0
## 83    0     0
## 84    1     0
## 85    0     0
## 86    0     0
## 87    0     0
## 88    0     0
## 89    0     0
## 90    0     0
## 91    0     0
## 92    0     0
## 93    0     0
## 94    0     0
## 95    1     0
## 96    0     0
## 97    0     0
## 98    0     0
## 99    0     0
## 100   0     0
```

```
## compare the results with the one from package comorbidities
library(comorbidities)
pkgRes <- (ahrq(sampleDat)[[2]])[, 1:2]
all.equal(pkgRes, myRes) # match if TRUE
```

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

Performance

Although function `cmbd` only generate the diagnosis result for CHF and VALVE, from the comparison of computing performance by package `microbenchmark`, we may find that the computing speed of `cmbd` is much faster than function `ahrq` in pacakge `comorbidities`.

```
library(microbenchmark)
microbenchmark(ahrq(sampleDat), cmbd(sampleDat), times = 5)
```

```
## Unit: milliseconds
##              expr          min           lq         mean       median
##  ahrq(sampleDat) 10747.527499 10759.559865 10783.141101 10794.797174
##  cmbd(sampleDat)     1.284635     1.306382     1.352108     1.323691
##            uq          max neval cld
##  10795.949809 10817.871157     5   b
##      1.412956     1.432876     5  a
```