A Structured Tagging approach to interoperability of Health Indicators Management Systems

Grégoire Lurton

July 2015

Introduction

Background

Criteria of success for the project

We would consider we have been successful in the project if we finish with a format of classification that : 1. allows for coding of a variety of types of health output indicators 2. allows a good differentiation of indicators 3. allows a good matching of similar indicators

Vision of interoperability framework

OpenHIE

Not Taxonomy. Need multiple levels. Simple text mining does not capture differences in definitions.

Benefits of this exercise outside of the interoperability domain

Format definition

There still are a few problems regarding how to distribute different categories.

One of those is the fact that some categories can be services, but are then used as additional population definition for more refined indicators (eg *Patients treated for TB* and *Patients treated for TB who have been tested for HIV*).

Tables of additional dimension definition needs

Next steps

Finalization of the format, and redaction of a method / manual to help reproduce this. Example of codification of some standard indicators.

Matching

First loading the listings. The user should just input the entry files of the formatted indicators list (for now we will assume that the columns names are properly standardized), and names he wants to use to follow each of these data sources.

```r
setwd('c://Users/grlurton/Drive/Indicators_Taxonomy/')


data_1 <- read.csv('rbf_benin_coded.csv' , stringsAsFactors = FALSE)
data_2 <- read.csv('hmis_benin_coded.csv' , stringsAsFactors = FALSE)

name_data_1 <- 'rbf'
name_data_2 <- 'hmis'

##formatting of string variables
```

Il y a plusieurs entrées dans le matching qui peuvent marcher. On peut prioriser en fonction de ce qui paraît important. Le matching se fait par convergence successive.

For now, let's make a mapping in two stages :

1. First matching on ICD10 and type of service
2. Second level of matching will be different depending on the type of service considered
3. The user is then given a return that he has to parse based on population characteristics

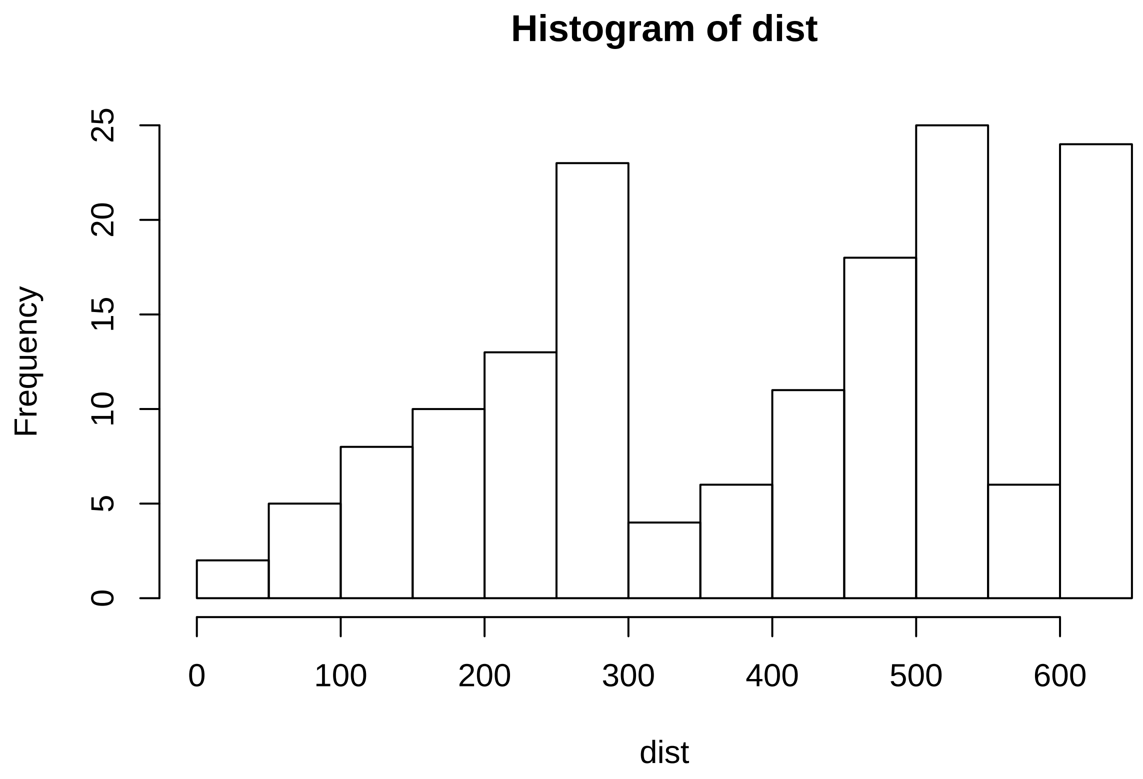
This can change. A user may want to just match a first step on ICD10 and check which services are matchable at that point. Also we should be able to read ICD10 codes that are subsets of others.

```r
#Make a first level merge of data sources. Not much more than a glorified merge
merge_first_level <- function(data_1 , data_2 , dimensions){
  merge(data_1 , data_2 , by = dimensions , suffixes = c('_1' , '_2'))
}
data_m <- merge_first_level(data_1 , data_2 , c('icd10','service'))

table(data_m$service)
```

```
##
## Accouchement    Chirurgie Consultation    Depistage Distribution
##            3            2            9          155            2
##   Traitement  Vaccination
##          259          207
```

```r
data_dep <- subset(data_m , service == 'Depistage')

##### Set of functions used for computing the distance between indicators


#Function to select variable in two data sets once it is passed :
get_both_variables <- function(data , variable_name){
  vars <- list(data[paste0(variable_name , '_1')] , data[paste0(variable_name , '_2')])
  vars
}

#Function make NAs in empty strings (comparisons don't work when there are NAs)
NAs_to_str <- function(variable){
  variable[is.na(variable)]  <- ""
  variable
```

```r
}

#Function to get distance on a given discrete variable
complement_distance <- function(variable_list , distance){
  distance <- distance * (variable_list[[1]] != variable_list[[2]])
  distance
}

#Wrapping function
dist_compute <- function(data , variable_name , distance = 100){
  var_list <- get_both_variables(data , variable_name)
  var_list <- lapply(var_list , NAs_to_str)
  distance_out <- complement_distance(var_list , distance)
  distance_out
}

## Complement
dist_dep_fait <- dist_compute(data_dep , 'depistage_fait' , 100)
dist_dep_result <- dist_compute(data_dep , 'depistage_resultat' , 100)

## Population
dist_sex <- dist_compute(data_dep , 'sex' , 50)
dist_enceinte <- dist_compute(data_dep , 'enceinte' , 50)
dist_autre <- dist_compute(data_dep , 'autre' , 50)

## Function to impute missing Age borns
NAs_to_age <- function(variable , multiplier){
  variable[is.na(variable)]  <- multiplier*12*365
  variable
}

## Function to get Age in Days
age_to_days <- function(age){
  age <- as.character(unlist(age))
  age_unit <- substr(age , 1 , 1 )
  age_value <- substr(age , 2 , nchar(age) )
  age_day <- as.numeric(age_value)
  age_day[age_unit == 'Y'] <- as.numeric(age_value[age_unit == 'Y']) * 12 * 30
  age_day[age_unit == 'M'] <- as.numeric(age_value[age_unit == 'M']) * 30
  age_day[age_unit == 'J'] <- as.numeric(age_value[age_unit == 'J'])
  age_day
}

#Function to get distance on a given continuous variable
continuous_distance <- function(variable_list , normalization){
  distance <- abs(variable_list[[1]] - variable_list[[2]]) / normalization
  distance
}

#Wrapping function
continuous_dist_compute <- function(data , variable_name , normalization = 100 , multiplier){
  var_list <- get_both_variables(data , variable_name)
  var_list <- lapply(var_list , NAs_to_age , multiplier = multiplier)
  var_list <- lapply(var_list , age_to_days)
  var_list <- lapply(var_list , NAs_to_age , multiplier = multiplier)
  distance_out <- continuous_distance(var_list , normalization)
  distance_out
}

dist_agemin <- continuous_dist_compute(data_dep , 'age_min' , normalization = 100 , multiplier = 0)
dist_agemax <- continuous_dist_compute(data_dep , 'age_max' , normalization = 100 , multiplier = 100)


dist <- dist_dep_fait + dist_dep_result + dist_sex + dist_enceinte + dist_autre + dist_agemin + dist_age

hist(dist , breaks = 10)
```



```
data_dep$dist <- dist


head(data_dep[order(dist) , c('indic_lib_1' , 'indic_lib_2' , 'dist')] , n = 10)
```

```
##                                                                                              indic_lil
## 290                                Diagnostic et traitement des cas de paludisme simple chez les enfa
## 325                        Diagnostic et traitement des cas de paludisme grave chez les femmes encein
## 345                                 Diagnostic et traitement des cas de paludisme grave chez les enfa
## 365                       Diagnostic et traitement des cas de paludisme simple chez les femmes encein
## 46  Femmes enceintes dépistées séropositive et mise sous traitement ARV (tri prophylaxie / trithérap
## 346                                Diagnostic et traitement des cas de paludisme grave chez les enfan
## 359                      Diagnostic et traitement des cas de paludisme simple chez les femmes encein
## 288                               Diagnostic et traitement des cas de paludisme simple chez les enfan
## 298                               Diagnostic et traitement des cas de paludisme simple chez les enfan
## 315                       Diagnostic et traitement des cas de paludisme grave chez les femmes encein
##                                               indic_lib_2 depistage_fait_1
## 290             Paludisme simple teste positif (TDR+GE)               36
## 325                     Paludisme grave testes positifs               36
## 345                     Paludisme grave testes positifs               68
## 365             Paludisme simple teste positif (TDR+GE)               68
## 46  Tuberculose pulmonaire (TPB+) testee positive au VIH              100
## 346                     Paludisme grave testes positifs              100
## 359             Paludisme simple teste positif (TDR+GE)              100
## 288              Total des cas toutes causes confondues              136
## 298                     Paludisme grave testes positifs              136
## 315              Total des cas toutes causes confondues              136
```

```
#Faire tourner la fonction libelle par libelle from RBF
```

A faire : definir un format de sortie pour les tables pour meilleure lisibilite

User Interactions and output of final matching format