comoRbidity: An R package to analyze disease comorbidities

Alba Gutierrez-Sacristan   Laura I. Furlong

June 21, 2016

Contents

1 Introduction

Electronic health records contains digital information about patient history, including demographics, diagnosis and treatments. The identification and bioinformatic analysis of disease comorbidities will have a great impact on the health status evolution, selection of appropriate treatments and health system costs and is key to identify new preventive and therapeutic strategies.

The `comoRbidity` R package is an analysis software that identifies comorbidity patterns at population level by processing clinical data. It allows the identification of disease comorbidities and its subsequent analysis in a clear and easy way. The `comoRbidity` R package identifies the statistically significant comorbidities using different statistic indexes stratified by age and gender. More importantly, it allows the user to provide its own clinical record data that should follow a simple tabulated format. In addition, the `comoRbidity` package allows to compute other parameters like the sex ratio parameter and temporal directionality of the co-occurring diseases. A special focus is made on the visualization of the results, providing a variety of representation formats, such as networks, heatmaps or bar plots.

1.1 Background

The tasks that can be performed with `comoRbidity` package are the following:

1. Age and gender analysis of the population suffering the disease of interest.
2. Code usage in the population under study.
3. Performing comorbidity analysis based on diagnosis data, in an specificg gender and age interval.
4. Analyzing the comorbidity temporal directionality.
5. Visualizing the results in a clear way, easily interpretable.

In the following sections the specific functions that can be used to address each one of these tasks are presented.

1.2 Installation

The package `comoRbidity` is provided through Bitbucket. To install `comoRbidity` the user must type the following commands in an R session:

```r
library( "devtools" )
install_bitbucket( "ibi_group/comoRbidity" )
library( "comoRbidity" )
```

2 Requeriments

Four documents are required in order to do the comorbidity analysis with `comoRbidity` package (1):

- patientData
- diagnositcData
- admissionData
- indexDiseaseCodes

An example of the data is shown below. This data has been obtained from an artificial medical data set of 100000 patients with 361760 admissions. This data has been obtained from EMRbots.org.

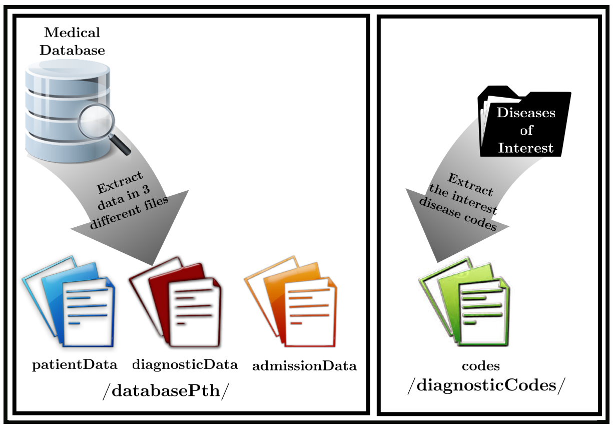

Figure 1: Data required in order to perform the comorbidity analysis

2.1 Patient Data

Patient data file must contain at least three columns with the column name as described as follows:

- patient_id: a patient identifier, that can be numeric, alphanumeric or a list of characters
- patient_gender: patient gender is required for the comorbidity analysis and for the sex ratio study. It can be numeric (0 for one gender and 1 for the other), character (M for male and F for female) and also a list of characters.
- patient_dataBirth: patient birth data is required to estimate the age when the patient suffer the disease. The structure that it must follow is year, month, and day, separated by any type of character.

The patient data file can contain other information, as the one that is shown in the next example. Note that the extra information will not be used by `comoRbidity` functions for the comorbidity analysis.

If the patientData file does not contain the required columns the next message will appear:

```
## Check the patientData file structure.  Remember that this
## + file must contain at least three columns with the column
## + names as follows:
##  -> patient_id
##  -> patient_gender
##  -> patient_dataBirth
```

```
head( patientData )

##                               patient_id patient_gender
## 1 F7CF0FE9-AFCD-49EF-BFB3-E42302FFA0D3         Female
## 2 C3935FBC-DBBA-4844-BBE4-A175FA508454           Male
## 3 1CA33F6F-2E84-4C99-AF6A-D40F7B4DB27F           Male
## 4 81606388-2471-42A4-A6F1-1868AE25CFC3           Male
## 5 E3120DE9-3361-40CF-A618-265C769E75A2         Female
## 6 5C043111-3F94-44BC-A889-97D44ACCC7F6         Female
##         patient_dataBirth      PatientRace PatientMaritalStatus
## 1 1951-07-10 07:29:47.293            Asian               Single
## 2 1956-01-27 22:46:39.380 African American               Single
## 3 1972-12-22 10:11:01.867            White              Married
## 4 1984-01-17 00:49:06.903            Asian            Separated
## 5 1978-12-21 07:24:08.957            White             Married
## 6 1974-09-25 18:38:02.440 African American             Married
##   PatientLanguage PatientPopulationPercentageBelowPoverty
## 1         English                                   13.70
## 2         English                                   15.73
## 3         English                                    7.09
## 4         Spanish                                    2.17
## 5         English                                   18.67
## 6         English                                    2.57
```

2.2 Diagnosis Data

Diagnosis data file must contain at least three columns with the column name as described as follows:

- patient_id: a patient identifier, that can be numeric, alphanumeric or a list of character. The patient identifier must be the same that is used in the patientData file.
- admission_id: an identifier related to the admission, that allows to distinguish between different entrance of the patient in the data base. It can be numeric, alphanumeric or a list of characters
- diagnoses_code: the disease code (in any format)

  The diagnosis data file can contain other information, as the one that is shown in the next example. Note that the extra information will not be used by `comoRbidity` functions for the comorbidity analysis.

  If the diagnosisData file does not contain the required columns the next message will appear:

```
## Check the diagnosisData file structure.  Remember that this
##                      file must contain at least three columns with the column
##                      names as follows:
##  -> patient_id
##  -> admission_id
##  -> diagnoses_code
```

```
head( diagnosData )

##                             patient_id admission_id diagnoses_code
## 1 54C6E968-45B3-46B1-A64F-2CE3124F2A65            3          F80.1
## 2 54C6E968-45B3-46B1-A64F-2CE3124F2A65            4         R04.81
## 3 54C6E968-45B3-46B1-A64F-2CE3124F2A65            5            I36
## 4 9DD23357-9BEB-43E4-802D-1AB7ACDD4A3A            1         H16.43
## 5 9DD23357-9BEB-43E4-802D-1AB7ACDD4A3A            2        M05.161
## 6 9DD23357-9BEB-43E4-802D-1AB7ACDD4A3A            3        M06.011
##                                      diagnosis_description
## 1                              Expressive language disorder
## 2                Acute idiopathic pulmonary hemorrhage in infants
## 3                        Nonrheumatic tricuspid valve disorders
## 4                           Localized vascularization of cornea
## 5 Rheumatoid lung disease with rheumatoid arthritis of right knee
## 6  Rheumatoid arthritis without rheumatoid factor, right shoulder
```

2.3 Admission Data

Admission data file must contain at least three columns with the column name as described as follows:

* patient_id: a patient identifier, that can be numeric, alphanumeric or a list of character. The patient identifier must be the same that is used in the patientData file.
* admission_id: an identifier related to the admission, that allows to distinguish between different entrance of the patient in the data base. It can be numeric, alphanumeric or a list of characters
* admissionStartDate:
  The admission data file can contain other information, as the one that is shown in the next example. Note that the extra information will not be used by `comoRbidity` functions for the comorbidity analysis.

  If the admissionData file does not contain the required columns the next message will appear:

```
## Check the admissionData file structure.  Remember that this
##                      file must contain at least three columns with the column
##                      names as follows:
##  -> patient_id
##  -> admission_id
##  -> admissionStartDate
```

```
head( admissiData )
##                               patient_id admission_id admissionStartDate
## 1 9380F9E3-1927-42F3-9731-03A74D4E4C6B            5         2011-03-23
## 2 0A89658C-C739-45CA-9BF1-CBDDDFB922C0            1         1974-02-10
## 3 0A89658C-C739-45CA-9BF1-CBDDDFB922C0            2         1991-05-22
## 4 0A89658C-C739-45CA-9BF1-CBDDDFB922C0            3         1995-02-26
## 5 0A89658C-C739-45CA-9BF1-CBDDDFB922C0            4         2005-03-17
## 6 0A89658C-C739-45CA-9BF1-CBDDDFB922C0            5         2008-04-12
##   admissionEndDate
## 1       2011-03-28
## 2       1974-02-16
## 3       1991-05-29
## 4       1995-02-28
## 5       2005-04-04
## 6       2008-04-16
```

2.4 Index Disease Codes

: the disease codes in which you are interested for the comorbidity analysis.

```
head( codes )
##    Code
## 1   F32
## 2 F30.9
## 3 F32.0
## 4 F32.1
## 5 F32.2
## 6 F32.3
##                                                                  Description
## 1                                    Major depressive disorder, single episode
## 2                                                   Manic episode, unspecified
## 3                              Major depressive disorder, single episode, mild
## 4                          Major depressive disorder, single episode, moderate
## 5 Major depressive disorder, single episode, severe without psychotic features
## 6    Major depressive disorder, single episode, severe with psychotic features
```

3 comoRbidity objects

3.1 comorbidity object

`comorbidity` object is obtained when `query` function is applied. This object is used as input for other functions in the package that enable the user to have an overview about the patients that suffer the disease of interest in terms of age, gender and diagnostic. `comorbidity` object is also used as input in the function that retrieves the comorbidity analysis (`comorbidityAnalysis`).

In summary, `comorbidity` object is the input for:

- `comorbidityAnalysis` function.
- `summaryDB` function.
- `populationAge` function.
- `diagnosticUse` function.

`comorbidity` object contains the information about the patients that suffer at least one of the input disorders (N. Patients), and the total number of disorders present in this subset (N. Diseases). This object also comes with a function that allows the user to retrieve the data saved in the object. This function is `extract` function. This function returns a formatted data.frame with the complete set of information obtained from the data queried.

```
comor_obj
## Loading required package:  comoRbidity
## Loading required package:  stringr
## Object of class 'comorbidity'
##  . Search:                     list
##  . Intracomorbidities:         FALSE
##  . N. Input Index Diseases:    19
##  . N. Index Diseases Present: 18
##  . N. Concomintant Diseases:  2519
##  . N. Patients:                2507
```

3.2 cAnalysis object

`cAnalysis` object is obtained when `comorbidityAnalysis` function is applied. This object is used as input for other functions in the package that enable the user to visualize the results in different graphical ways. Moreover, `cAnalysis` object is used as input for further analysis in the comorbidity results, like the sex ratio analysis and the directionality.

`cAnalysis` object is the input for:

- `network` function.
- `heatmapPlot` function.
- `sexRatio` function.
- `directionality` function.

`cAnalysis` object contains the results of the comorbidity analysis and other relevant information for the user. `cAnalysis` object shows the age interval that has been applied for the analysis (Age Min and Age Max); the gender, the number of patients that belong to this group from the total data (Subset Patients) as well as the number of them that suffer the disease of interest (Active Paients). Other interesting data such as the disease prevalence and the minimum values obtained for each parameter estimated to measure the comorbidity. Finally the number of pairs of diseases that pass the cutOff determined by the user are also shown (Final Disease Pairs).

This object also comes with a function that allows the user to retrieve the data saved in the object. This function is `extract` function, that returns a formated data.frame with the comorbidity analysis results.

```
comorMale
## Object of class 'cAnalysis'
##  . Age Min : 1
##  . Age Max : 99
##  . Gender  : Male
##  . Patients in the age and gender interval: 48000
##  . Patients suffering the disease: 1178
##  . Disease Prevalence: 2.454167
##  . Minimum odds ratio: 6.182
##  . Minimum relative risk: 6.067
##  . Minimum phi value: 0.009
##  . Number of comorbidities: 3694
class(comorMale)
## [1] "cAnalysis"
## attr(,"package")
## [1] "comoRbidity"
```

```
comorbidityData <- extract ( comorMale )
head( comorbidityData )
##      disAcode disBcode disA disB AB AnotB BnotA notAnotB
## 817    F31.11   I25.41   59   78  3    56    75    47866
## 108    F31.62      D11   64   75  3    61    72    47864
## 1639    F32.4  M05.561   77   64  3    74    61    47862
## 391     F32.4  E10.620   77   74  3    74    71    47852
## 2836   F31.77  O24.434   78   75  3    75    72    47850
## 425    F31.77    I27.0   78   82  3    75    79    47843
##                    fisher        oddsRatio     relativeRisk
## 817  0.000125671144908017 34.1619152206319 31.2907431551499
## 108  0.000142531504874312 32.6574373724672               30
## 1639 0.000154111535422692 31.7903696245923 29.2207792207792
## 391    0.0002370311720609 27.3172415397609 25.2720252720253
## 2836 0.000256200839333868 26.5808418795056 24.6153846153846
## 425  0.000333277050885006 24.2171526443886 22.5140712945591
##                     phi    expect    score       fdr
## 817  0.0428708627852663 0.0958750 1.867917 0.1270246
## 108  0.0419185894160873 0.1000000 1.862496 0.1270246
## 1639 0.0413334309640643 0.1026667 1.859003 0.1270246
## 391  0.0382304709279821 0.1187083 1.838166 0.1270246
## 2836 0.0376898516491375 0.1218750 1.834088 0.1270246
## 425  0.0359054208934747 0.1332500 1.819534 0.1270246
```

4 Data extraction

The first step in order to perform the comorbidity analysis is extracting the data related to the patients that suffer the disorder of interest. This disorder of interest is determined by the `code` file.

`query` function allows the user to extract the data and save it in a `comorbidity` class object. As input the function requires:

- `databasePth`: determine the path where the three input files required (ptientData, diagnosisData, admissionData) are located.

- `codesPth`: determine the path where the file with the diagnostic codes of interest is located.
- `admissionDataSep`: determine what is the separation symbol in the admission data.
- `birthDataSep`: determine what is the separation symbol in the birth data.

```
databasePth <- system.file("extdata", package="comoRbidity")
diagnosticCodes <- system.file("extdata", "indexDiseaseCodes.txt", package="comoRbidi
ff <- query( databasePth      = databasePth,
             codesPth         = diagnosticCodes,
             admissionDataSep = "-",
             birthDataSep     = "-"
             )
ff
```

```
## Object of class 'comorbidity'
##  . Search:                     list
##  . Intracomorbidities:         FALSE
##  . N. Input Index Diseases:    18
##  . N. Index Diseases Present:  17
##  . N. Concomitant Diseases:    2498
##  . N. Patients:                2361
```

As a result, a `comorbidity` object is obtained. This object will contain those patients that have been diagnosed with at least one of the codes presented in the `codes` file. Moreover, all the additional disorders that have been suffered by these patients will also be saved.

5 Summary DB

The `comoRbidity` R package allows the user to analyze and characterized the population of your database that suffer from a disease or diseases of interest. In order to have a general idea about the population main characteristics, the user can apply the `summaryDB` function. A set of graphics describing the age and gender distribution for the subset of patients suffering from the disorder of interest will be obtained (2).

As a input, the `summaryDB` function requires:

- `input`: a `comorbidity` object, obtained with the `query` function.
- `maleCode`: the symbol which denotes males in users' database (0, M, Male...etc)
- `femaleCode`: the symbol which denotes females in users' database (1, F, Femaale...etc)

The output of the `summaryDB` function is a plot with three different graphics:

- A barplot with the age distribution of the patients suffering the disease of interest.
- A boxplot showing the age distribution by gender.
- A pie chart representing the gender distribution.

```
summaryDB(input = ff,
          maleCode = "Male",
          femaleCode ="Female")
```

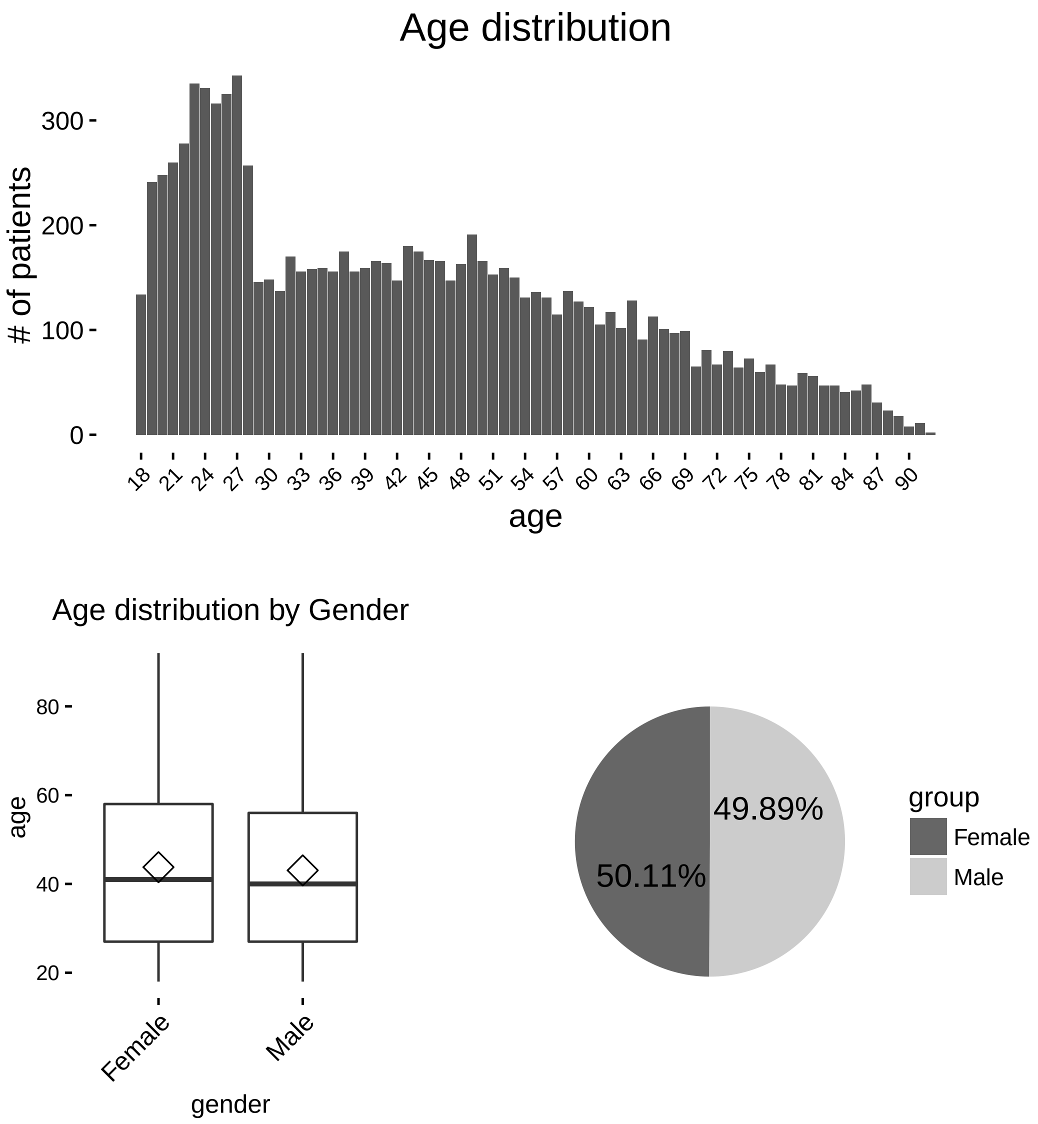

Figure 2: Summary plot, containing: age-distribution; age distribution by gender and gender distribution.

6 Population analysis based on the disease under study

Following with the age analysis, the `comoRbidity` R package also allows to compare the age distribution of patients suffering the disease under study with all the patients of the user's database. It is considered the first time in which the patient has been diagnosed with any of the diagnoses of interest (for those patients suffering the disorder of interest) and the first date the patient has an entry in the database for the rest of patients.

As an input, `populationAge` function requires:

- `input`: a `comorbidity` object, obtained with the `query` function.
- The path where the code file is located.
- The path where the three input files required (ptientData, diagnosisData, admissionData)

are located.

The `populationAge` function has two more arguments, that are optional for the user:

- `type` argument allows the user to select the output barplot. By default the type selected is "together" (3), but it can be set up to "separate" (4).
- `interactive` argument allows to create an interactive barplot, that show you the specific information of each bar in the barplot in an interactive way.

This can be visualized together (3) or separate .

```
populationAge ( input       = ff,
                codesPth    = diagnosticCodes,
                databasePth = databasePth,
                type        = "together",
                interactive = FALSE)
```

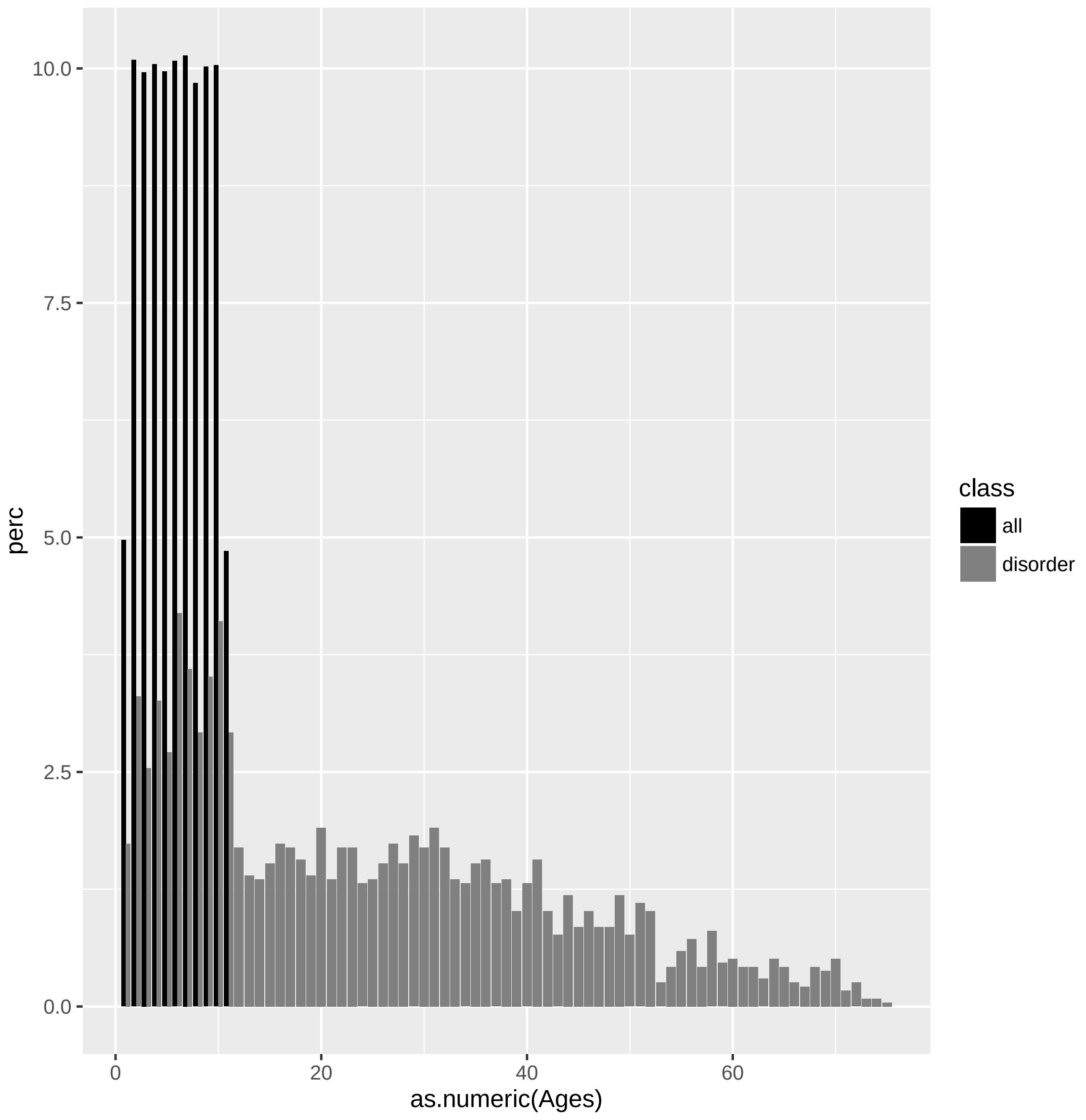

Figure 3: Age-distribution of the disorder of interest vs all population age.

```
populationAge ( input       = ff,
                codesPth    = diagnosticCodes,
                databasePth = databasePth,
                type        = "separate",
                interactive = FALSE)
```

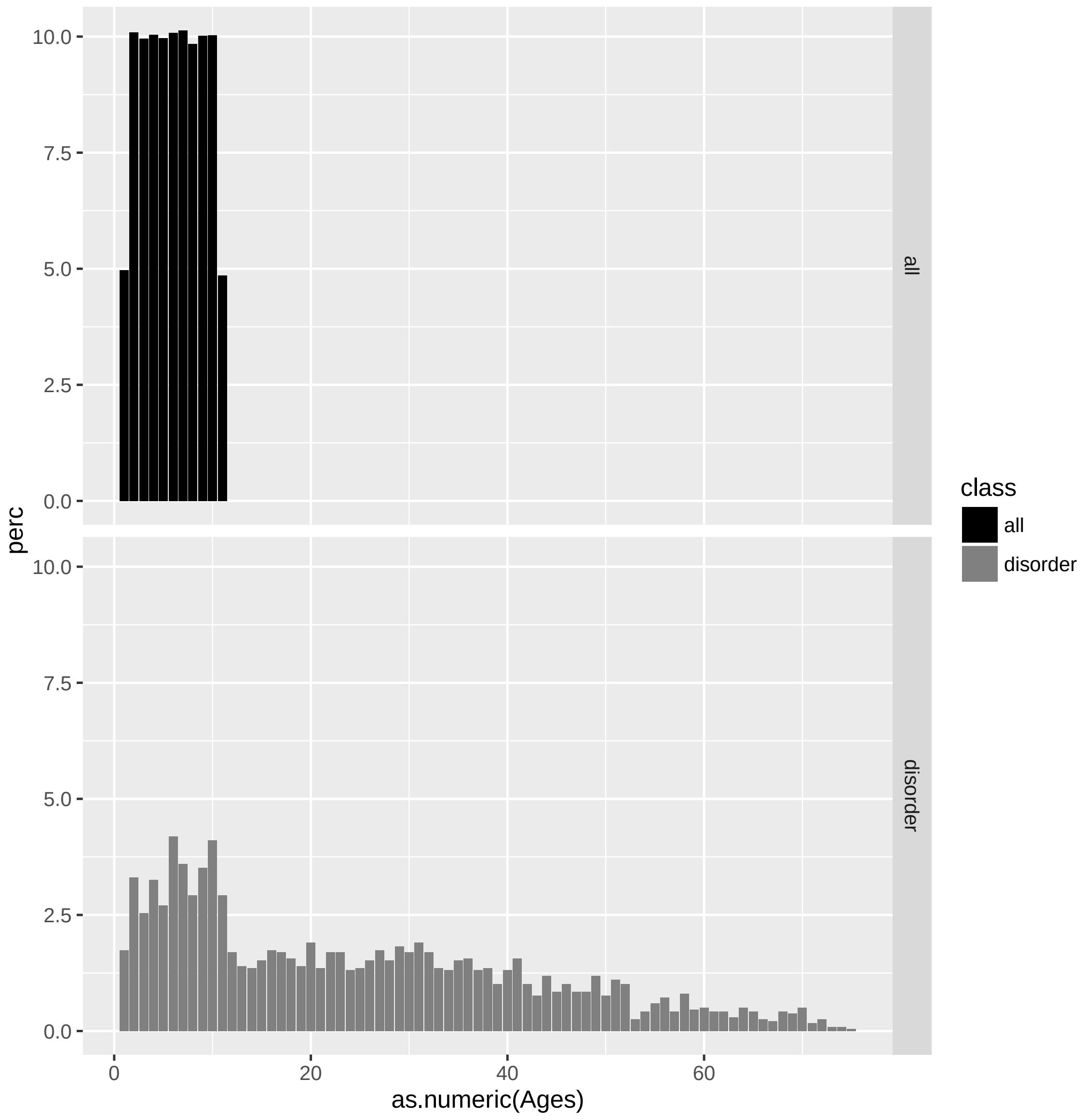

Figure 4: Age-distribution of the disorder of interest vs all population age.

7 Study disease/diseases of interest in our database

7.1 Code Use

When we are studying a disorder that is defined by more than one diagnosed code, `comoRbidity` R package allows to analyze the percentage of use of each one of the codes (5).

As an input, `diagnosticUse` function requires:

- `obj`: a `comorbidity` object, obtained with the `query` function.
- `codesPth`: The path where the code file is located.
- `databasePth`: The path where the three input files required (ptientData, diagnosisData,

admissionData) are located.

The `diagnosticUse` function has two more arguments, that are optional for the user:

- `cutOff` argument allows the user to select those diagnostic codes that will be represented in the output barplot. By default the `cutOff` selected is 0, but it can be set up to any other percentage value.
- `interactive` argument allows to create an interactive barplot, that show you the specific information of each bar in the barplot in an interactive way. By default the `interactive` argument is set up as `FALSE`.

```
diagnosticUse( codesPth     = diagnosticCodes,
               databasePth  = databasePth,
               obj          = ff,
               cutOff       = 0,
               interactive  = FALSE
               )
```

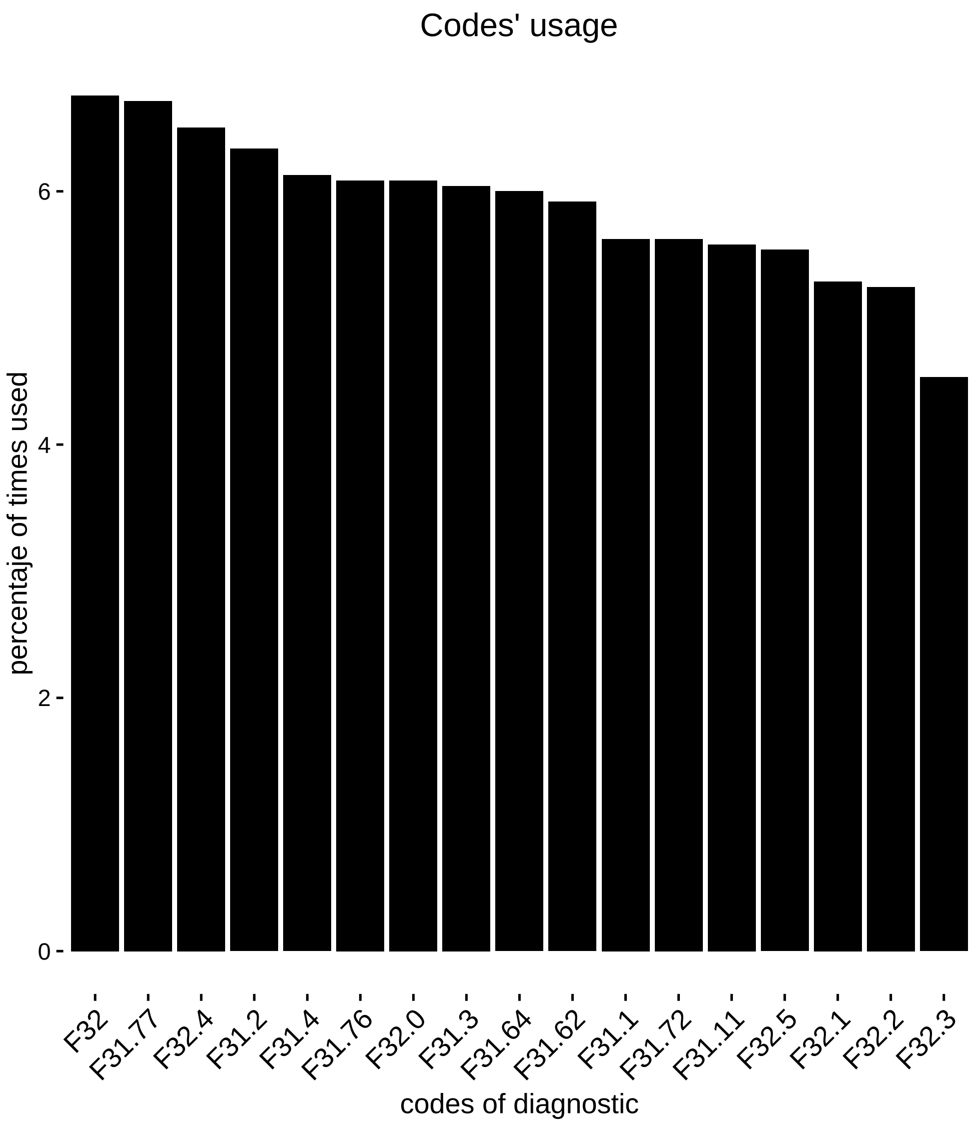

Figure 5: Disorder codes usage in percentage

7.2 Comorbidity Analysis

Once we have a general overview about the patients that suffer the disorder of interest in our datbase, and the main characteristics of this population, the next step is perform the comorbidity analysis.

Using the `comorbidity` object previously generated, it can be estimated the comorbidity pairs that are statistically significant applying the `comorbidityAnalysis` function.

7.3 Comorbidity Measurements

Two diseases are connected if they are co-expressed in a significant number of patients in a population. To estimate the correlation starting from disease co-occurrence, we need to quantify the strength of the comorbidity risk. The `comoRbidity` R package estimates several measures to quantify the strength of comorbidity associations between two diseases:

Fisher test A Fisher exact test for each pair of diseases is performed to assess the null hypothesis of independence between the two diseases. Four groups of patients are done in order to perform the statistical testing, patients suffering disease X and disease Y, patients suffering disease X but not disease Y, patients suffering disease Y but not disease X and patients not suffering disease X nor disease Y. Using these values we estimate the p-value of the Fisher exact test for each pair of diseases. The Benjamini-Hochberg false discovery rate method [1] is applied on the ranked list to correct for multiple testing.

Comorbidity score This score is defined in Roque et al. as follows [2]:

$$comorbidityscore = log_2\left(\frac{observed+1}{expected+1}\right) expected = \frac{n_x x n_y}{n_t otal} \tag{1}$$

where observed is the number of disease-disease associations, and expected is estimated based on the occurrence of each disease (nx ny).

Relative risk Fraction between the number of patients diagnosed with both diseases and random expectation based on disease prevalence.

$$RR_i j = \frac{C_i j N}{P_i P_j} \tag{2}$$

Phi value (Pearsons correlation for binary variables) measures the robustness of the comorbidity association.

$$\phi_i j = \frac{C_i j N - P_i P_j}{\sqrt{P_i P_j (N-P_i)(N-P_j)}} \tag{3}$$

where N is the total number of patients in the population, P i and P j are incidences/prevalences of diseases i and j respectively. C i j is the number of patients that have been diagnosed with both diseases i and j, and P i P j is the random expectation based on disease prevalence.

Odds ratio The odds ratio represents the increased chance that someone suffering disease X will have the comorbid disorder Y. It shows the extent to which suffering a disorder increases the risk of developing another illness or disorder. The odds ratio is derived from a comparison of rates of the illness among individuals who do and do not also exhibit the factor of interest. A statistically significant odds ratio (significantly different from 1.00 at the .05 level) indicates an appreciable risk associated with a particular factor. For example, an odds ratio of 2.00 indicates a doubled risk of the illness/disorder appearing.

These measures allows the user to quantify the co-occurrence of disease pairs compared with the random expectation. User can select the measure and the cut-off in order to asses disease comorbidity.

`comorbidity` function allows the user to perform the comorbidity analysis and save it in a `cAnalysis` class object. As input the function requires:

- `obj`: a `comorbidity` object obtained after applying the `query` function.
- `databasePth`: determine the path where the three input files required (ptientData, diagnosisData, admissionData) are located.
- `codesPth`: determine the path where the file with the diagnostic codes of interest is located.
- `ageRange`: determine what is the age range in which we are interested in performing the comorbidity analysis. By default it is set up from 0 to 100 years old.
- `gender`: determine what is the gender in which we are interested in performing the comorbidity analysis.

Moreover, the `comorbidityAnalysis` function allows us to restrict our results according to a comorbidity measurement value. The posible values that can be applied as a cut-off are:

- score
- fdr
- oddsRatio rr
- phi

User can filter the results applying all the comorbidity measurements that she considers necesary. Note that the cut-off value for these measurements must be numeric. In the next example, it is shown a query in which two of the five comorbidity measurements are applied, the `score` and the `fdr`.

```
comorFemale <- comorbidityAnalysis( obj=ff,
                              codesPth    = diagnosticCodes,
                              databasePth = databasePth,
                              score       = 0,
                              fdr         = 1,
                              ageRange    = c( 0, 100 ),
                              gender      = "Female",
                              verbose     = FALSE
                              )
save(comorFemale, file=paste0(databasePth, "comorFemale.RData"))
```

As a result, a `cAnalysis` object is obtained. This object will contain a summary of the comorbidities that have been found, showing:

```
load(system.file("extdata", "comorFemale.RData", package="comoRbidity"))
comorFemale
## Object of class 'cAnalysis'
##  . Age Min : 1
##  . Age Max : 99
##  . Gender  : Female
##  . Patients in the age and gender interval: 52000
##  . Patients suffering the disease: 1183
##  . Disease Prevalence: 2.275
##  . Minimum odds ratio: 6.542
##  . Minimum relative risk: 6.42
##  . Minimum phi value: 0.009
##  . Number of comorbidities: 3716
```

Moreover, the `extract` function can also be applied to the `cAnalysis` object. This function returns a formatted data.frame with the complete set of information obtained from the comorbidity analysis.

```
comorbidityDataFem <- extract ( comorFemale )
head( comorbidityDataFem )
```

```
##      disAcode disBcode disA disB AB AnotB BnotA notAnotB
## 328    F31.62    D37.4   77   63  3    74    60    51863
## 1521    F32.4   C40.81   78   72  3    75    69    51853
## 1191   F31.11   E10.52   74   78  3    71    75    51851
## 1017   F31.72      Z11   63   51  2    61    49    51888
## 913     F32.3    C40.3   57   59  2    55    57    51886
## 490     F32.1   D31.01   61   56  2    59    54    51885
##                      fisher        oddsRatio     relativeRisk
## 328  0.000116281237431729  35.037467798342 32.1583178726036
## 1521 0.000179713175501743 30.0454110865651 27.7777777777778
## 1191 0.000194913415546318 29.1997434140357  27.027027027027
## 1017  0.00177267055476611 34.7133596595554 32.3685029567383
## 913   0.00194038846003978 33.0411029681971 30.9247695509961
## 490   0.00200121946130541 32.5433026666394 30.4449648711944
##                       phi     expect    score       fdr
## 328    0.04178989907258 0.09328846 1.871326 0.1242194
## 1521 0.0386466254566125 0.10800000 1.852042 0.1242194
## 1191 0.0380819855006414 0.11100000 1.848141 0.1242194
## 1017 0.0342312004363577 0.06178846 1.498466 0.1242194
## 913  0.0334099311338358 0.06467308 1.494552 0.1242194
## 490  0.0331325975061247 0.06569231 1.493172 0.1242194
```

7.4 Comorbidity representation

In order to visualize the comorbidity analysis results, `comoRbidity` package provides two different options:

- Network: obtained applying `network` function.
- Heatmap: obtained applying `heatmapPlot` function.

7.4.1 Network

`network` function allows the user to visualize the data contained in the `cAnalysis` object obtained after applying the `comorbidityAnalysis` function.

As input `network` function requires:

- `input`: an object of the `cAnalysis` type, obtained after applying the `comorbidityAnalysis` function.
- `databasePth`: The path where the three input files required (ptientData, diagnosisData, admissionData) are located.
- `layout`: by deafult `"layout.fruchterman.reingold"`. It can be set up to other of the possible igraph layouts.
- `selectValue`: By default `"score"` variable will be selected. It can be set up to any of the other possible variables ('fdr', 'odds ratio', 'phi', 'rr') that makes easier the interpretation of your results.
- `cutOff`: By default '0.05'. Change it to any other numeric variable, according to the range of the selected value.
- `npairs`: By default '0'. Change it to any other numeric variable to show in the network only those comorbidities suffered by at least npairs patients.
- `prop`: By default 1. It allows to change the node size.
- `title`: By default 'Comorbidity network'.
- `interactive` argument allows to create an interactive barplot, that show you the specific information of each node in the network in an interactive way. By default the `interactive` argument is set up as `FALSE`.

```r
load(system.file("extdata", "comorMale.RData", package="comoRbidity"))
network ( input = comorMale,
          databasePth = databasePth,
          layout = "layout.fruchterman.reingold",
          selectValue = "score",
          cutOff = 1.45,
          npairs = 2,
          prop  = 1,
          title = "Male comorbidity network",
          interactive = FALSE
        )
```

Figure 6: Comorbidity network in male population

As a result, a netwok is obtained (7). Nodes in pink belong to the disorder of interest, while the blue ones correspond to the comorbidity disorders. Note that the color of the nodes can be change adding the next arguments to the `network` function:

- `diseaseColor`: By default "pink". It defines the node color for the disorder of interest.
- `comorColor`: By default "lightblue". It defines the node color for the comorbid disorders.

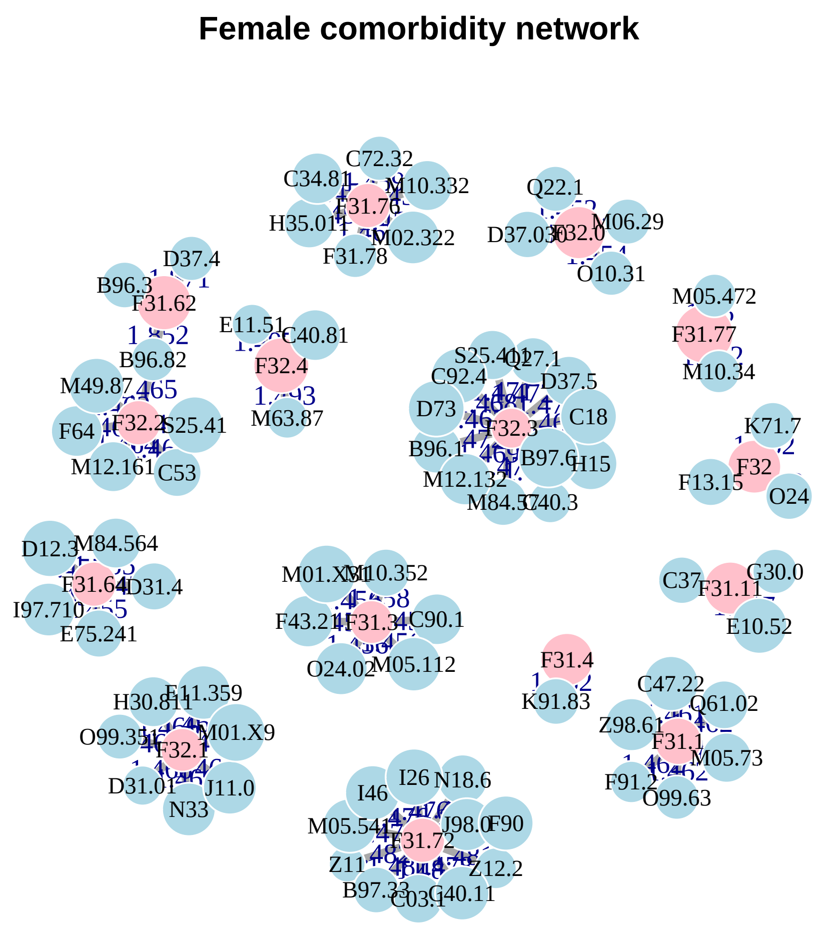

Figure 7: Comorbidity network in female population

`comoRbidity` package also allows to visualize this data in a heatmap instead in a network(8).The input required is the same than for the `network` function.

```
heatmapPlot( input        = comorFemale,
             selectValue = "score",
             npairs      = 2,
             cutOff      = 1.45,
             verbose     = FALSE,
             interactive = FALSE)
```

Note that the color of the heatmap can be change adding the next arguments to the `heatmaPlot` function:

- `lowColor`: By default "0000FF". It defines the heatmap color for the lowest value.
- `highColor`: By default "yellow". It defines the heatmap color for the highest value.

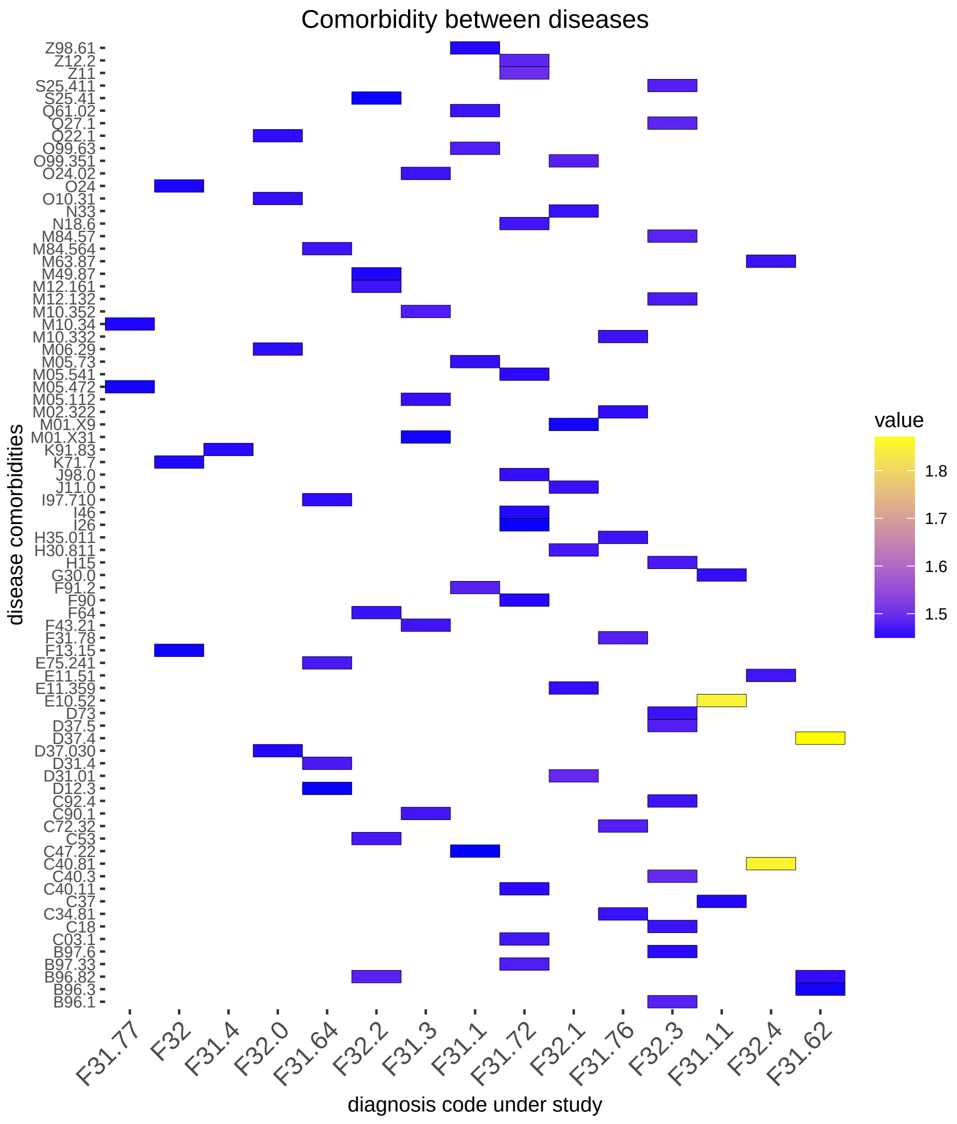

Figure 8: Comorbidity heatmap in female population

7.5 Sex ratio analysis

The `comoRbidity` package allows to compute the sex ratio (SR) parameter. The sex ratio (SR) parameter allows to see, for those comorbidities that are present in both, men and women, if the co-occurrence is equally likely for both or on the contrary if it is more likely in one group than in the other. For a comorbidity x and age group t, SR (7.5) is defined by Klimek et al. [3] as follows:

$$SR(x,t) = log\left(\frac{1+\dfrac{D_f(t)}{D_f(x,t)}}{1+\dfrac{D_m(t)}{D_m(x,t)}}\right)$$

where Dm(f)(t) stands for the number of depression patients (men or women) in age group t, and Dm(f)(x,t) denotes those depression patients who also has been diagnosed with a disease x. SR values close to 0 means that the comorbidity is equally likely for men and women. A positive value indicates that the comorbidity is more likely for women, while negative value indicates that the comorbidity is more likely in men.

In order to obtain the sex ratio heatmap, two steps should be followed:

- Apply the `sexRatio` function using as input the `cAnalysis` object obtained for both genders. Note that fisher test is performed, so it can be determined the cut-off to be applied.
- Apply the `sexRatioHeatmap` to the previous results. `interactive` argument can be set up to `TRUE` if an interactive heatmap is required.

In the next example, we apply a filter to the sex ratio results in order to show only those that have more extreme values.

```
srAnalysis <- sexRatio(female     = comorFemale,
                       male       = comorMale,
                       fisherTest = 0
                       )

srAnalysis <- srAnalysis[as.numeric(srAnalysis$SR) <= -0.5
                         | as.numeric(srAnalysis$SR) >= 0.5,]
```

As a result a heatmap (9) is obtained. Red color (positive values) in the heatmap belong to those comorbidities that are more likely for women, while blue color (negative values) belong to those that are more likely for men. These colors can be changed adding the arguments:

- maleColor
- femaleColor

```
sexRatioHeatmap(srAnalysis,
                interactive = FALSE
                )
```

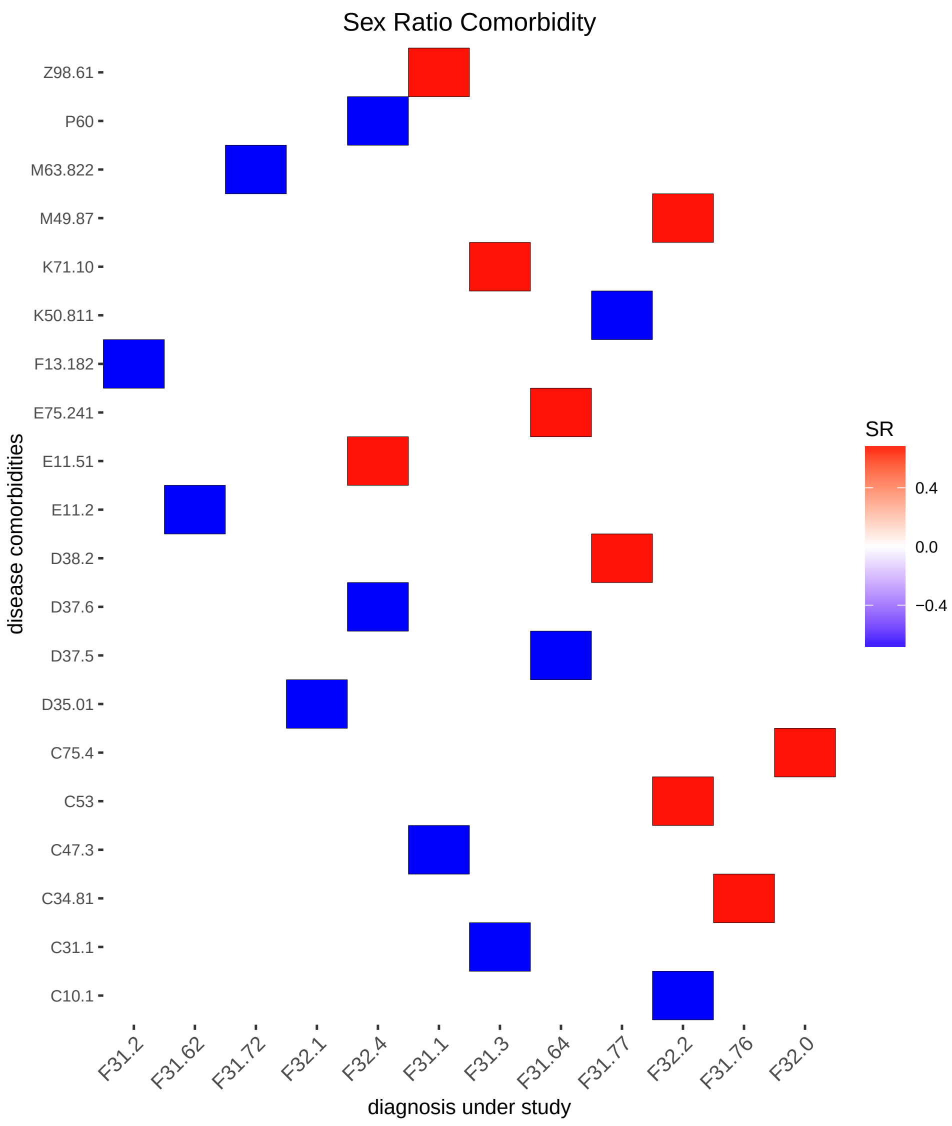

Figure 9: Sex ratio (SR) comorbidities heatmap

7.6 Directionality analysis

Additionally, the `comoRbidity` package allows to analyze of the temporal directionality of the co-occurring diseases, which allows the inference of temporal disease trajectories (10).

The temporal direction of disease association (d1 $\rightarrow$ d2 and d2 $\rightarrow$ d1) is assessed for the diagnosis pairs with a significant Bonferroni-corrected p-value. Specifically, the number of patients for whom, diagnosis d2 follows diagnosis d1, or vice versa, is calculated and an exact binomial test is, subsequently, used with a probability of success equal to 0.5. A preferred (significant) direction is determined for those diagnosis pairs that result in binomial tests with p-values < 0.05 and according to the pair that appears more often.

```
comorbidityDirection <- directionality( input        = comorFemale,
                                        databasePth = databasePth,
                                        gender      = "Female",
                                        ageRange    = c(0,100),
                                        days        = 0,
                                        minPairs    = 1,
                                        dataSep     = "-")
summary(as.factor(comorbidityDirection$result))
## No directionality
##              3716
```

```
heatmapDireccion(test,
                 interactive = FALSE)
```

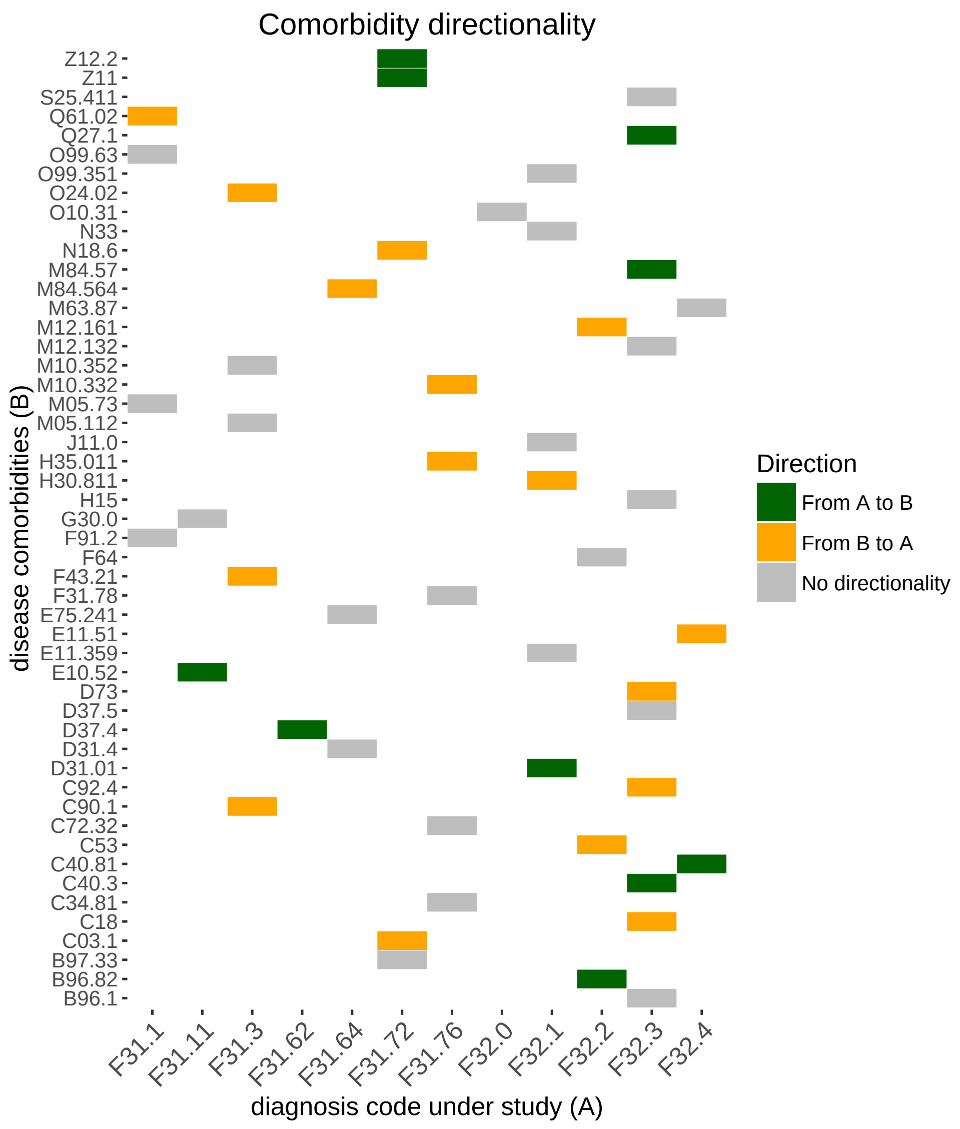

Figure 10: Comorbidities directionality heatmap

8 Bibliography

References

[1] Yoav Benjaminia; Dan Draib; Greg Elmerc; Neri Kafkafid; Ilan Golanib **Controlling the false discovery rate in behavior genetics research** Behavioural Brain Research 2001 doi:10.1016/S0166-4328(01)00297-2

[2] Francisco S. Roque; Peter B. Jensen; Henriette Schmock; Marlene Dalgaard; Massimo Andreatta; Thomas Hansen; Karen Søeby; Søren Bredkjær; Anders Juul; Thomas Werge; Lars J. Jensen; Søren Brunak **Using Electronic Patient Records to Discover Disease Correlations and Stratify Patient Cohorts** PLOS Computational Biology 2011 doi:10.1371/journal.pcbi.1002141

[3] Peter Klimek; Alexandra Kautzky-Willer; Anna Chmiel; Irmgard Schiller-Frühwirth; Stefan Thurner **Quantification of diabetes comorbidity risks across life using nation-wide big claims data.** PLoS Comput. Biol. 2015 doi: 0.1371/journal.pcbi.1004125