genophenoR: An R package for querying and analyzing EHR data through the BD2K PIC-SURE RESTful API

# Introduction

The genophenoR package contains functions to query different databases through the BD2K RESTful API, a programmatic interface that provides access to different data sources, making easier data accessibility, analysis reproducibility and scalability. The genophenoR package includes analysis functions to study demographic population, phenotype descriptive analysis, prevalence and co-occurrence, according to patient age, gender and mutation status if data available. A special focus is made on visualization of the results, providing a variety of representation formats such as networks, heatmaps and barplots (Table @ref(tab:viz-opt)).

## Background

In the era of big data and precision medicine, the number of databases containing clinical and genomic information is increasing in an exponential way. Enables the experts to focus on their research questions rather than in the computation data merging, access and management is one of the biggest challenges nowadays.

Towards the aim of developing reproducible and scalable analysis, the genophenoR package has been developed. This R package allows to make queries to different databases through the PIC-SURE API, simplifying the data connection and analysis. Moreover it allows to easily reproducible analysis between different databases as well as its integra-tion with other packages available in R to develop bioinformatics analy-sis workflows.

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

1. query through the BD2K PIC-SURE API RESTful API
2. Demographic descriptive analysis based on age and gender
3. Phenotype descriptive analysis taking into account continuous and categorical variables.
4. Phenotype prevalence and co-occurrence analysis according to age, gender and gene mutation status if available.
5. Extract those patients that have been diagnosed with a determined pair of phenotypes.

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

## Installation

The package is provided through GitHub. To install the user must type the two following commands in an R session:

install\_github("hms-dbmi/genophenoR")

library(genophenoR)

## S4 objects

### genopheno

genopheno object is obtained when queryGenoPheno and genophenoZscore functions are applied. This object is used as input for demographicSummary and phenotypeSummary descriptive functions. The genopheno object contains the table dataset and shows the main characteristic:

* The number of mutation variables (N. Mutations)
* The number of phenotype variables (N. Phenotypes)
* The number of patients (N. Patients)

genophenoExData

## Object of class 'genopheno'  
## . N. Mutations: 0   
## . N. Phenotypes: 3   
## . N. Patients: 41474

class( genophenoExData )

## [1] "genopheno"  
## attr(,"package")  
## [1] "genophenoR"

This object comes with a series of functions to allow users to interact with the information retrieved. These functions are nphenotype, nmutation and extract.

The nphenotype function returns the phenotypes present in the output of the given query.

nphenotypeExample <- genophenoR::nphenotype( genophenoExData )  
class( nphenotypeExample )

## [1] "character"

nphenotypeExample

## [1] "Herpes" "HIV" "pcb153"

The nmutation function returns the mutations present in the output of the given query.

extractExample <- genophenoR::extract( genophenoExData )  
class( extractExample )

## [1] "data.frame"

kable( head( extractExample ) )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| patient\_id | P.Herpes | P.HIV | Gender | Age | P.pcb153 |
| 10997 |  |  | male | 61 | NA |
| 10998 | Yes | No | male | 48 | NA |
| 10999 |  |  | male | 0 | NA |
| 22529 | Yes | No | female | 21 | 0.128 |
| 22528 |  |  | male | 16 | NA |
| 22527 |  |  | female | 10 | NA |

The extract function returns a formatted data.frame with the complete set of information contained in the object, the query results.

nmutationExample <- genophenoR::nmutation( genophenoExData )  
class( nmutationExample )

## [1] "character"

nmutationExample

## character(0)

### genophenoComor

genophenoComor object is obtained when genoPhenoComorbidity function is applied. This object is used as input for genoPhenoPrevalence, genoPhenoHeatmapand genoPhenoNetwork functions. The genophenoComor object contains the comorbidity results as well as the comorbidity analysis results and a summary of the different measurements:

* Minimum age of the patients in the dataset (Age Min)
* Minimum age of the patients in the dataset (Age Max)
* Patients gender
* Mutation, if selected
* Number of patients in the age and gender interval in the database
* Number of patients in the age and gender interval with the mutation select
* The prevalence of the mutation in the dataset
* Odds ratio range
* Relative risk range
* Phi value range
* Number of resultant comorbidities

genophenoComor

## Object of class 'genophenoComor'  
## . Age Min : 4   
## . Age Max : 18   
## . Gender : ALL   
## . Mutation : NONE   
## . Patients in the age and gender interval: 14575   
## . Patients with mutation selected: 14575   
## . Prevalence: 100   
## . Odds ratio range: [0.03 , Inf]   
## . Relative risk range: [0.098 , 6.096]   
## . Phi range: [-0.352 , 0.352]   
## . Number of comorbidities: 7

class( genophenoComor )

## [1] "genophenoComor"  
## attr(,"package")  
## [1] "genophenoR"

The genophenoComor object come with the extract function. The extract function allows the user to retrieve data stored in the object. The extract function returns a formatted data.frame with the complete set of results obtained from the comorbidity analysis.

# Retrieve data from the database

The first step in order to perform any analysis is extracting the data related to the patients of interest, according to a set of variables of interest, demographic variables, as well as phenotypic or genotypic variables.

The irctquerty function allows the user to extract the data from a specific database and store it in a data.frame object. genophenoR package allows exploring the database information using the specific url, the key and a JSON query in text format. It retrieves the information that is available in the database according to the user query and allows the analysis and results visualization in different ways.

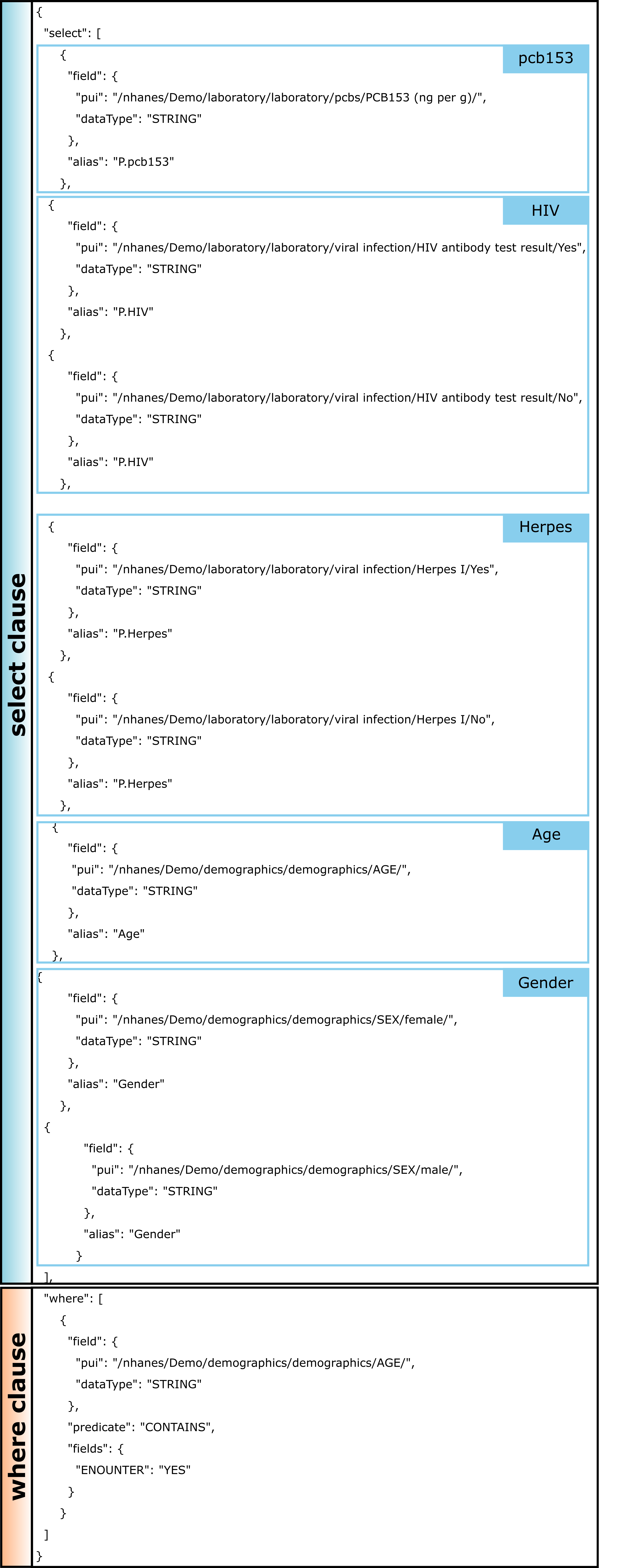
In order to perform a query in a database, the user can apply the irctquerty function. As input the irctquerty function requires:

* *url*: the ulr of the database
* *key*: the personal key to access to the data
* *query*: a text file containing the JSON query body
* *outputPath*: path where the output file will be saved. By default it will be saved in your working directory

As an example data from NHANES database [1] will be retrieve applying the irctquerty function to:

* **url**: "<https://nhanes.hms.harvard.edu/>"
* **key**: to access to the data
* **JSON query**: \_ select the variables pcb153, HIV, herpes, patient age and gender from all those patients in NHANES database that have age information\_. The example of the JSON query provided as input is shown in Figure 1.

query <- irctquery(   
 url = "https://nhanes.hms.harvard.edu/",   
 apiKey = "a77l42anvcgdtkfcvbl7hnp2v9",   
 query = system.file("extdata", "jsonQueryNhanes", package="genophenoR")  
 )

Figure 1, Query example in NHANES database: 

The result is a data.frame object:

class( query )

## [1] "data.frame"

kable( head( query ) )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| patient\_id | P.Herpes | P.HIV | Gender | Age | P.pcb153 |
| 2 |  |  | male | 85 | NA |
| 3 |  |  | female | 0 | NA |
| 4 | Yes | No | female | 49 | 0.109 |
| 5 | Yes | No | male | 18 | NA |
| 6 |  |  | female | 4 | NA |
| 7 | Yes | No | male | 31 | NA |

In the particular example used, by inspecting the data.frame object, we can see that there are 41474 patients in the database that have age data, and the pcb153, hiv and herpes information for these patients have been extracted correctly.

# Transform the data in a genopheno object

Once the data is in a data.frame format, it has to be transformed into a genopheno class object. Note that to perform a correct analysis, the data frame should contain:

* 3 demographic variables: patient\_id, Gender and Age.
* phenotype variables name should start with a "P"
* mutation variables name, if present, should start with a "M"

In order to transform the data.frame into a genopheno object, the queryGenoPheno function has to be applied. The only input needed is the inputDataFile, that determines the file with the complete path where the required input file is located.

genophenoExData <- queryGenoPheno( inputDataFile = paste0(system.file("extdata", package="genophenoR"), "/queryOutput.txt"),   
 verbose = TRUE )

## Loading the input datasets

## Checking the inputData file structure

## Removing duplicated data

## There are 41474 patients in your input data with complete information for all your variables, from the initial 41474 patients in your list.

## Checking the number of mutations in the inputData file

## Generating the result object

As a result a genopheno object class is obtained.

class( genophenoExData )

## [1] "genopheno"  
## attr(,"package")  
## [1] "genophenoR"

genophenoExData

## Object of class 'genopheno'  
## . N. Mutations: 0   
## . N. Phenotypes: 3   
## . N. Patients: 41474

head( genophenoR::extract( genophenoExData ) )

## patient\_id P.Herpes P.HIV Gender Age P.pcb153  
## 1 10997 male 61 <NA>  
## 2 10998 Yes No male 48 <NA>  
## 3 10999 male 0 <NA>  
## 4 22529 Yes No female 21 0.128  
## 5 22528 male 16 <NA>  
## 6 22527 female 10 <NA>

# Transform continuous in categorical variables: zscore

Phenotype variables can be continuous or categorical. Co-occurrence analysis will be only applied to those variables that are classified as categorical, like for example, P.Herpes, which has yes or no values. Variables like P.pcb153 are continuous. To add it to the analysis it should be transformed into a categorical.

If desired, user can apply the genophenoZscore function, that given an object of class genopheno, it transforms continuous into categorical variable applying Z-score. As a result a new genopheno object is generated.

Note that if the number of individuals is lower than 5000 a Saphiro test [2] will be done to test the normal distribution, otherwise a Kolmogorov-Smirnov test [3] will be performed. The steps followed in this process are the next:

1. Checking is the variable follows a normal distribution
2. If the variable does not follow a normal distribution the Z-score will not be estimated.
3. If the variable follows a normal distribution (p-value of the shapiro or Kolmogorov-Smirnov test is lower than 0.05) then: 3.1. it is checked if there is correlation between the age and the variable. When correlation is observed, it is fitted a linear model.
4. Z-score will be estimated for this variable

As an input genophenoZscore function requires:

* input: a genopheno object, obtained with the queryGenoPheno function.
* cutOff: determines Z-score cut-off to categorize the continuous variable. By default it is set to -2 and 2.
* nfactor: determines the maximum number of values that a variable could have to be defined as categorical. By default it is 10.

zscoreExample <- genophenoZscore( input = genophenoExData,  
 cutOff = c( -2, 2 ),   
 nfactor = 10,   
 verbose = TRUE )

## Checking the input object

## Herpes phenotype is considered as a categorical variable

## Z-score will not be applied to Herpes variable

## HIV phenotype is considered as a categorical variable

## Z-score will not be applied to HIV variable

## pcb153 phenotype is considered as a continuous variable

## Checking is the variable follows a normal distribution

## The variable pcb153 does not follow a normal distribution

## Z-score will not be estimated for this variable

As a result a new genopheno object is obtained. Note that the transformation cannot always be done.

head( genophenoR::extract( genophenoExData ) )

## patient\_id P.Herpes P.HIV Gender Age P.pcb153  
## 1 10997 male 61 <NA>  
## 2 10998 Yes No male 48 <NA>  
## 3 10999 male 0 <NA>  
## 4 22529 Yes No female 21 0.128  
## 5 22528 male 16 <NA>  
## 6 22527 female 10 <NA>

head( genophenoR::extract( zscoreExample ) )

## patient\_id P.Herpes P.HIV Gender Age P.pcb153  
## 1 10997 male 61 <NA>  
## 2 10998 Yes No male 48 <NA>  
## 3 10999 male 0 <NA>  
## 4 22529 Yes No female 21 0.128  
## 5 22528 male 16 <NA>  
## 6 22527 female 10 <NA>

# Demographic analysis

The genophenoR R package allows the user to analyze and characterize the population under study. To have a general idea about the population main characteristics, the user can apply the demographicSummary function.

The function demographicSummary describes the demographic characteristics (sex, age) of the population under study. The demographicSummary function requires three arguments:

* *input*: the genopheno object.
* *maleCode*: the symbol which denotes males in the table (e.g., 0, M, Male...etc)
* *femaleCode*: the symbol which denotes females in the table (e.g., 1, F, Female...etc)

The demographicSummary function output includes:

* A barplot with the age distribution of the whole study population
* A boxplot showing the age distribution by gender.
* A pie chart representing the gender distribution.

genophenoR::demographicSummary ( input = genophenoExData,   
 maleCode = "male",   
 femaleCode ="female",   
 verbose = TRUE )

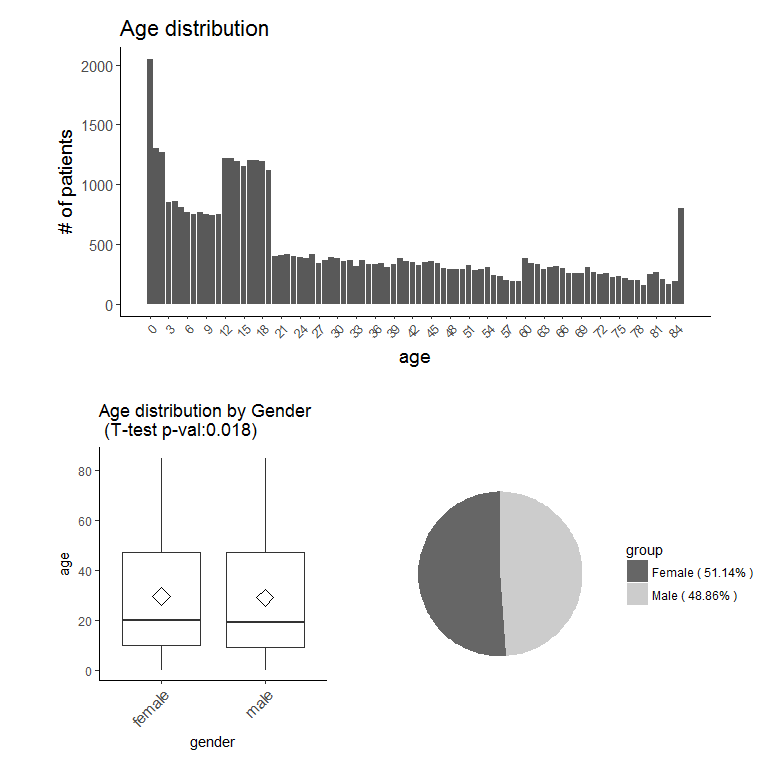
## Checking the input object

## Creating a summary table with the main characteristics of population

## Generating the sex distribution plot

## Generating the age distribution plot

## Generating the plot showing age distribution according to gender



# Phenotype analysis

The genophenoR R package includes the phenotypeSummary function that describes the phenotypic characteristics for the whole study population, according to the status regarding one selected mutation if genotype information available. otherwise a general summary of the different phenotype values is displayed.

The phenotypeSummary function requires as input the next arguments:

* input: the genopheno object.
* mutation: the mutation of interest
* nfactor: determines the maximum number of values that a variable could have to be defined as categorical. By default it is 10.
* showTable: the table is displayed before the barplot.
* showFigures: results are visualized in barplot or boxplot, depending on the kind of variable.
* path: define the path where the output file generated will be saved.

The phenotypeSummary function output could include:

* A barplot for each categorical phenotype variable
* A boxplot for each continuous phenotype variable
* A table if the showTable argument is TRUE

phenotypeSummary( input = genophenoExData,  
 showTable = TRUE,   
 showFigures = FALSE)

## No genomic information will be taken into account

## No plot is available for this variable

## phenotype phenotypeValue P\_AllPatients confint yesno  
## Herpes 62.87 [62.4-63.3] NA  
## No Herpes No 15.14 [14.8-15.5] NA  
## Yes Herpes Yes 21.99 [21.6-22.4] NA  
## 1 HIV 71.26 [70.8-71.7] NA  
## No1 HIV No 28.58 [28.1-29] NA  
## Yes1 HIV Yes 0.16 [0.1-0.2] NA

# Phenotypes co-occurrence and prevalence using

Having a general overview about the demographics and the phenotypes included in our data, the next step is to perform the phenotype co-occurrence analysis.

The user can estimate the statistically significant comorbidities by applying the genoPhenoComorbidity function to the genophenoobject previously generated with the queryGenoPheno function.

### Comorbidity Measurements

The genoPhenoComorbidity identifies all the possible phenotype co-occurrence and quantify them. Five different quantification measures are estimated:  
- Fisher test - Comorbidity score - Relative risk (RR) - Phi value (Pearson's correlation for binary variables)  
- Odds ratio

##### 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 defined in order to perform the statistical testing: patients with phenotype A and phenotype B, patients with phenotype A but not phenotype B, patients with phenotype B but not phenotype A and patients without any of the two phenotypes. The Fisher exact test is then applied to estimated the p-value for each pair of diseases. The Benjamini-Hochberg false discovery rate method is applied on the ranked list to correct for multiple testing.

##### Comorbidity score

This score is defined in Roque et al. [4] as follows:

$$comorbidity score =log\_2 \left( \cfrac{observed + 1}{expected + 1} \right)$$

$$expected = \cfrac{n\_{A} n\_{B}}{N}$$

where *observed* stands for the number of disease-disease associations (disease A and disease B), and *expected* is estimated based on the occurrence of each disease (number of patients diagnosed with disease A, *nA*, multiplied by the number of patients diagnosed with the comorbid disorder B, *nB* , and divided by the total number of patients, *N*). Since logarithm is applied, a comorbidity score of 1.0 means that the observed comorbidities are higher than two fold (approximately) than expected.

#### Relative risk (RR)

The relative risk (RR) expresses the relationship between the rate of prevalence of a phenotype among the patients exposed and those patients that are not exposed to a certain risk factor. Here the risk factor is other phenotype. The RR value moves from 0 to infinite.

* RR = 1: The risk is the same for patients exposed to the factor of risk than for those patients that are not exposed.
* RR > 1: The patients exposed to the factor of risk are more likely to suffer the disease. It is a risk factor.
* RR < 1: The patients exposed to the factor of risk are less likely to suffer the disease. It is a protection factor.

The RR is estimated as the fraction between the number of patients diagnosed with both diseases and random expectation based on disease prevalence [5]. The RR of observing a pair of diseases A and B affecting the same patient is given by:

$$RR\_{AB} = \cfrac{C\_{AB} N}{P\_A P\_B}$$

where *CAB* is the number of patients affected by both diseases, *N* is the total number of patients in the population and *PA* and *PB* are the prevalence of diseases A and B.

#### Phi value (Pearsons correlation for binary variables)

Phi value measures the robustness of the comorbidity association. It can be expressed mathematically as:

$$\phi\_{AB} = \cfrac{C\_{AB} N - P\_{A} P\_{B}}{\sqrt{P\_{A} P\_{B}(N - P\_{A})(N - P\_{A})}}$$

where *N* is the total number of patients in the population, *PA* i and *PB* are prevalence of phenotypes A and B respectively. *CAB* is the number of patients that have been diagnosed with both phenotypes A and B, and *PA PB* is the random expectation based on disease prevalence. The Pearson correlation coefficient, can take a range of values from +1 to -1. A value of 0 indicates that there is no correlation between the two diseases; a value greater than 0 indicates a positive correlation between the two diseases and a value less than 0 indicates a negative correlation.

#### Odds ratio

The odds ratio represents the increased chance that someone suffering phenotypes A will have the comorbid phenotypes B. It shows the extent to which suffering a phenotypes increases the risk of developing another phenotype. The odds ratio is derived from a comparison of rates of the phenotypes among individuals who do and do not 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 appearance of the phenotype

\*\* These measures allow the user to quantify the co-occurrence of phenotype pairs compared with the random expectation. The user can select the measure and the cut-off value in order to asses phenotype co-occurrence.\*\*

The genoPhenoComorbidity function requires 6 arguments:

* input: the genopheno object
* pth: the path where the file with the phenotype values generated previously is located.
* aggregate: if all the possible phenotypes values want to be considered individually, aggregate must be FALSE. otherwise, the phenotype values should be manually completed by the user (column yes/no) and aggregate must be TRUE.
* ageRange: determines what is the age range of interest for performing the analysis.
* gender: determine what is the gender of interest for performing the co-occurrence analysis.
* mutation: determine the mutation of interest and the value of this mutation.

genophenoComor <- genoPhenoComorbidity(   
 input = genophenoExData,  
 pth = system.file("extdata", package="genophenoR"),   
 aggregate = FALSE,   
 ageRange = c(4,18),  
 gender = "ALL",   
 )

As a result, a genophenoComor object is obtained. This object contains a summary of the phenotype co-occurrences that have been found.

class( genophenoComor )

## [1] "genophenoComor"  
## attr(,"package")  
## [1] "genophenoR"

genophenoComor

## Object of class 'genophenoComor'  
## . Age Min : 4   
## . Age Max : 18   
## . Gender : ALL   
## . Mutation : NONE   
## . Patients in the age and gender interval: 14575   
## . Patients with mutation selected: 14575   
## . Prevalence: 100   
## . Odds ratio range: [0.03 , Inf]   
## . Relative risk range: [0.098 , 6.096]   
## . Phi range: [-0.352 , 0.352]   
## . Number of comorbidities: 7

## Prevalence

The function genophenoPrevalence displays in a data frame: \* Phenotype \* Number of patients (N\_patients) \* Prevalence \* Confidence interval

prevalence <- genoPhenoPrevalence( input = genophenoComor )  
  
kable( prevalence, row.names = FALSE)

|  |  |  |  |
| --- | --- | --- | --- |
| phenotype | N\_patients | Prevalence(%) | ConfidenceInterval |
| HIV: (yes) | 13457 | 92.33 | [91.9-92.8] |
| Herpes: (yes) | 9436 | 64.74 | [64-65.5] |
| Herpes: No (yes) | 2748 | 18.85 | [18.2-19.5] |
| Herpes: Yes (yes) | 2391 | 16.40 | [15.8-17] |
| HIV: No (yes) | 1116 | 7.66 | [7.2-8.1] |
| HIV: Yes (yes) | 2 | 0.01 | [0-0] |

## Plotting the co-occurrence results

genophenoR offers several options to visualize the results from the phenotype co-occurrence analysis, in networks and heatmaps.

### Network

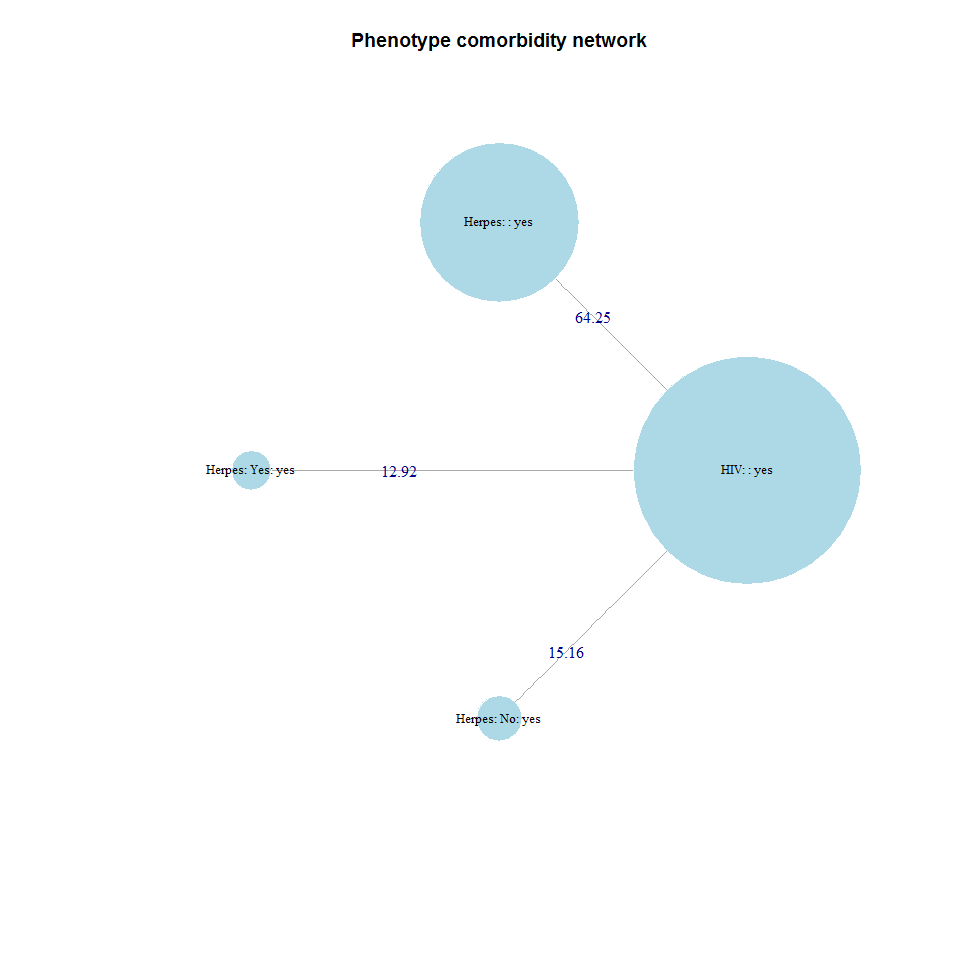
By applying the genophenoNetwork function. A network showing the phenotypes comorbidities is obtained. By default genophenoR package shows a network with layout.fruchterman.reingold layout where the phenotypes are joined according to the number of patients that have both phenotypes ( selectValue = patientsPhenoAB). The user can change this value to the different comorbidity measures - fisher, oddsRatio, relativeRisk, phi, expect, score, fdr, PercentagePhenoAB- that are estimated for each pair of phenotypes.

The genoPhenoNetwork function requires 3 arguments as input:

* input: the genophenoComor object
* selectValue: the co-occurrence measurement
* cutOff: the numeral value of the cutOff

The genoPhenoNetowrk function output is a network. Node size is proportional to the phenotype prevalence. Edge numbers represent the selected value.

genoPhenoNetwork ( input = genophenoComor,  
 selectValue = "PercentagePhenoAB",   
 cutOff = 10)



### Heatmap

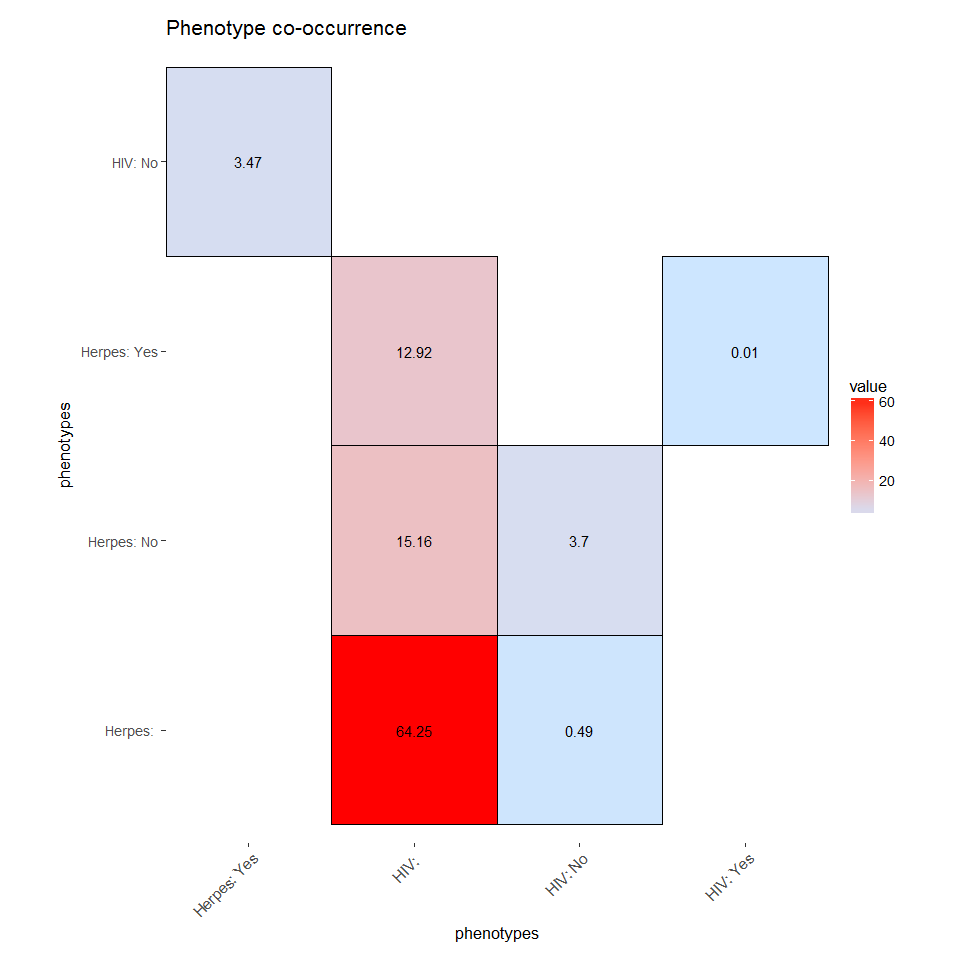
By applying the genophenoHeatmap function, a heatmap is displayed. This function is homologous to the genophenoNetwork one. The heatmap represents the selected value that by default will be patientsPhenoAB. The user can change this value to the different comorbidity measures - fisher, oddsRatio, relativeRisk, phi, expect, score, fdr, PercentagePhenoAB- that are estimated for each pair of phenotypes.

The genoPhenoHeatmap function requires 3 arguments:

* input: the genophenoComor object
* selectValue: the co-occurrence measurement
* cutOff: the numeral value of the cutOff

The genoPhenoHeatmap function output is a heatmap. Blue color represents the lower values, red color represents the upper values. The co-occurrence measurement value is showed in each cell.

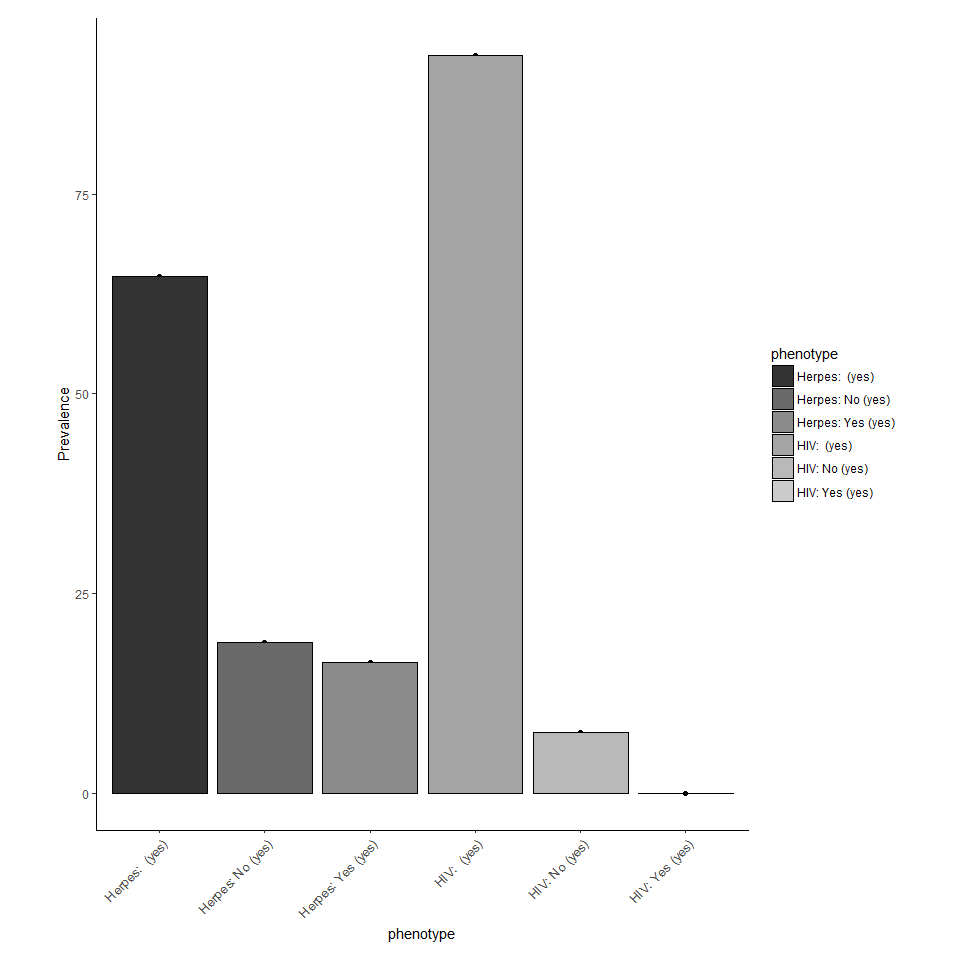
genoPhenoHeatmap ( input = genophenoComor,  
 selectValue = "PercentagePhenoAB",   
 cutOff = 0)



## Plotting the prevalence results

prevalencePlot( input = genophenoComor,   
 title = "Phenotype prevalence",   
 verbose = TRUE  
 )

## Checking the input object



# Summary of genophenoR functions available

(#tab:viz-opt) Functions in genophenoR R package

|  |  |  |
| --- | --- | --- |
| Input Object | genophenoR function | Output Generated |
| - | irctquery | Given an url, a key and a JSON object, it generates a data.frame object with the output of the query |
| - | queryGenoPheno | Given a tabulated file, checks if it contains the data in the correct format and generates a genopheno objec |
|  |
| genopheno | demographicSummary | 3 plots: age and gender distribution, relation between age and gender distribution |
|  | phenotypeSummary | A data.frame with the prevalence of each phenotype according to the value it takes, is retrieved. A figure showing the results can be also obtained. |
|  | extract | It generates a data.frame containing the raw data from a query |
|  | genophenoZscore | It transforms continuous into categorical variable applying Z-score |
|  | genoPhenoComorbidity | It generates a genophenoComor object |
|  | genoPhenoPatientsSelection | It retrieves a list of patients suffering two phenotypes of interest |
|  | nmutation | It retrieves the mutations of your data |
|  | nphenotype | It retrieves the phenotypes of your data |
| genophenoComor | genoPhenoHeatmap | It generates a heatmap showing the comorbidity analysis results |
|  | genoPhenoNetwork | It generates a network showing the comorbidity analysis results |
|  | extract | It generates a data.frame containing the comorbidity results |
|  | genoPhenoPrevalence | It generates a data.frame containing the prevalence of each phenotype |
|  | prevalencePlot | It generates a barplot or a network showing the results of the phenotype's prevalence |

# Bibliography

1. Patel CJ, Pho N, McDuffie M, Easton-Marks J, Kothari C, Kohane IS, et al. A database of human exposomes and phenomes from the us national health and nutrition examination survey. Scientific Data. 2016;3. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5079122/>.

2. Shapiro SS, Wilk MB. An analysis of variance test for normality (complete samples). Biometrika. 1965;52:591–611.

3. Razali NM, Wah YB, others. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. Journal of statistical modeling and analytics. 2011;2:21–33.

4. Roque FS, Jensen PB, Schmock H, Dalgaard M, Andreatta M, Hansen T, et al. Using electronic patient records to discover disease correlations and stratify patient cohorts. PLoS Comput Biol. 2011;7:e1002141. <http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002141>.

5. Hidalgo CA, Blumm N, Barabási A-L, Christakis NA. A dynamic network approach for the study of human phenotypes. PLoS computational biology. 2009;5:e1000353. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2661364/>.