Package ‘Rcupcake’

May 1, 2017

Type Package

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

Version 0.99.0

Description Package to integrate, analyze and visualize phenotype and genotype
information from electronic health record data.

Depends R (>= 3.2.3), stringr

License MIT + file LICENSE

LazyData true

Suggests parallel

Imports stringr, igraph, ggplot2, plotrix, plyr, reshape, parallel,
gridExtra, scales, plotly, RCurl, d3heatmap, gtools, shiny,
httr, jsonlite

NeedsCompilation no

Author Alba Gutierrez-Sacristan [aut],

Maintainer Alba Gutierrez-Sacristan <alba_gutierrez@hms.harvard.edu>

biocViews Software, BiomedicalInformatics, Genetics,
DataRepresentation

RoxygenNote 6.0.0

R topics documented:

`co.occurrence` *Co-occurrence Analysis* `cupcakeResults`

Description

Given an object of type `cupcakeData`, a co-occurrence analysis is perform, for the subset of population under specific conditions of age, gender and gene status. It generates a `cupcakeResults` object.

Usage

```
co.occurrence(input, pth, ageRange = c(0, 100), aggregate = TRUE,
  gender = "ALL", variation = c("", ""), nfactor = 10, scoreCutOff,
  fdrCutOff, oddsRatioCutOff, relativeRiskCutOff, phiCutOff, cores = 1,
  verbose = FALSE)
```

Arguments

`input`	A `cupcakeData` object, obtained with the my.data function.
`pth`	Determines the path where the required file with phenotype data is located. This file is generated applying the `phenotypeSummary` function.
`ageRange`	Determines what is the age range of interest for performing the comorbidity analysis. By default it is set from 0 to 100 years old.
`aggregate`	By default TRUE. Change it to FALSE if you want to analyze the comorbidity taking into all the values of each phenotype.
`gender`	Determine what is the gender of interest for performing the comorbidity analysis. By default `ALL`. Change it to the gender of interest for your comorbidity analysis.
`variation`	Determine what is the variation of interest for performing the comorbidity analysis. By default `c("", "")`. Change it to the value of interest for your comorbidity analysis. For example, `c("CHD8", "yes")`
`nfactor`	By default 10. Change it into other number if you consider there is any categorical variable with more than nfactor values.
`scoreCutOff`	The comorbidity score is a measure based on the observed comorbidities and the expected ones, based on the occurrence of each disease.
`fdrCutOff`	A Fisher exact test for each pair of diseases is performed to assess the null hypothesis of independence between the two diseases. The Benjamini-Hochberg false discovery rate method (FDR) is applied to correct for multiple testing.
`oddsRatioCutOff`	The odds ratio represents the increased chance that someone suffering disease X will have the comorbid disorder Y.

`relativeRiskCutOff`	The relative risk refers to the fraction between the number of patients diagnosed with both diseases and random expectation based on disease prevalence.
`phiCutOff`	The Pearsons correlation for binary variables (Phi) measures the robustness of the comorbidity association.
`cores`	By default `1`. To run parallel computations on machines with multiple cores or CPUs, the cores argument can be changed.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.

Value

An object of class `cupcakeResults`

Examples

```
load(system.file("extdata", "genophenoExData.RData", package="Rcupcake"))
cooccurrenceExample <- co.occurrence(
              input         = genophenoExData,
              pth           = system.file("extdata", package="Rcupcake"),
              aggregate     = TRUE,
              ageRange      = c(0,16),
              gender        = "male",
              )
```

`cooc.heatmap` *Plot the comorbidity analysis results in a heatmap.*

Description

Given an object of class `cupcakeResults` obtained from a comorbidity analysis, a heatmap is obtained.

Usage

```
cooc.heatmap(input, representedVariable = "patientsPhenoAB",
  variableCutOff = 0.05, coocPatients = 0, interactive = FALSE,
  lowColor = "#cde6ff", highColor = "red", verbose = FALSE)
```

Arguments

`input`	A `cupcakeResults` object, obtained by applying the `comorbidityAnalysis` function
`representedVariable`	By default `"patientsPhenoAB"` variable will be selected. Change it to any of the other possible variables (`'score'`,(`'fdr'`,`'oddsRatio'`, `'phi'`, `'relativeRisk'`, `'PercentagePhenoAB'`).
`variableCutOff`	By default `'0.05'`. The value of the argument can be changed to any other numeric variable, according to the range of the selected value.
`coocPatients`	by default `'0'`. The value of the argument can be changed to any other numeric variable to show in the network only those comorbidities suffered by at least `coocPatients` of patients.

`interactive`	Determines if the output heatmap is interactive or not. By default the `interactive` argument is set up as `FALSE`. The value of the argument can be changed to `TRUE`, as a result an interactive heatmap will be obtained.
`lowColor`	Determines the heatmap color for the lowest value. By default it is set to `"#cde6ff"`.
`highColor`	Determines the heatmap color for the highest value. By default it is set to `"red"`.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.

Value

A heatmap

Examples

```
load(system.file("extdata", "RcupcakeExResult.RData", package="genophenoR"))
htmp <- cooc.heatmap( input              = cupcakeResults,
              representedVariable  = "patientsPhenoAB",
              variableCutOff       = 1,
              coocPatients         = 1
              )
htmp
```

`cooc.network` *Plot the comorbidity analysis results in a network*

Description

Given an object of class `genophenoComor` obtained from a comorbidity analysis, a network is obtained.

Usage

```
cooc.network(input, layout = "layout.circle",
  representedVariable = "patientsPhenoAB", variableCutOff = 0,
  coocPatients = 0, nodeProportion = 1, interactive = FALSE,
  verbose = FALSE)
```

Arguments

`input`	A `genophenoComor` object, obtained by applying the `comorbidityAnalysis` function
`layout`	By default `'layout.fruchterman.reingold'`. It can be set to any other of the possible igraph layouts.
`representedVariable`	By default `"patientsPhenoAB"` variable will be selected. Change it to any of the other possible variables (`'score'`,(`'fdr'`,`'oddsRatio'`, `'phi'`, `'relativeRisk'`, `'PercentagePhenoAB'`).
`variableCutOff`	By default `'0.05'`. The value of the argument can be changed to any other numeric variable, according to the range of the selected value.

`coocPatients`	by default `'0'`. The value of the argument can be changed to any other numeric variable to show in the network only those comorbidities suffered by at least `coocPatients` of patients.
`nodeProportion`	Determines the node size proportionality. By default it is set to `1`. The value of the argument can be changed to any other numeric variable.
`interactive`	Determines if the output network is interactive or not. By default the `interactive` argument is set up as `FALSE`. The value of the argument can be changed to `TRUE`, as a result an interactive network will be obtained.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.
`databasePth`	Determines the path where the intermediate RData objects have been created. It is the same path where the three required input files (patientData, diagnosisData, admissionData) are located.

Value

A network

Examples

```
load(system.file("extdata", "RcupcakeExResult.RData", package="Rcupcake"))
cooc.network <- network(
              input              = cupcakeResults,
              representedVariable = "patientsPhenoAB",
              variableCutOff      = 1.5,
              coocPatients        = 2
              )
```

`dataframe2cupcake` *Query your data and generates a* `cupcakeData`

Description

Given a tabulated file, checks if it contains the data in the correct format and generates a `cupcakeData` object.

Usage

```
dataframe2cupcake(input, verbose = FALSE, warnings = TRUE)
```

Arguments

`input`	Determines the file with the complete path where the required input file is located. This input file must contain three columns: "patient_id" with the patient identifier, "Gender" and "Age". Variation columns must start with "M." while phenotype ones must start with a "P."
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.
`warnings`	By default `TRUE`. Change it to `FALSE` to don't see the warnings.

Value

An object of class `cupcakeData`

Examples

```
queryExample <- dataframe2cupcake( input = paste0(system.file("extdata", package="Rcupcake"),
                                                  "/queryOutput.txt"),
                                   verbose     = TRUE)
queryExample
```

---

demographic.summary *Describes the demographic characteristics (sex, age) of the population under study*

Description

Given an object of class `cupcakeData`, and the characters used to specify the gender, a graphic containing 3 plots, one of the age distribution, another one of the age distribution and the third one with the relation between age and gender distribution, is obtained.

Usage

```
demographic.summary(input, maleCode, femaleCode, verbose = FALSE,
  warnings = TRUE)
```

Arguments

`input`	Object of `cupcakeData` class.
`maleCode`	Characters(s) used to determine the male condition of a patient. Depending on the database it can be determined, for example, as `Male`, . `MALE`, `M`, with digits as `0` or `1`.
`femaleCode`	Characters(s) used to determine the female condition of a patient. Depending on the database it can be determined, for example, as `Female`, . `FEMALE`, `F`, with digits as `0` or `1`.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.
`warnings`	By default `TRUE`. Change it to `FALSE` to don't see the warnings.

Value

A multiple graph containing a barplot with age distribution, a boxplot representing age distribution by gender and a pie chart representing gender distribution.

Examples

```
load(system.file("extdata", "genophenoExData.RData", package="Rcupcake"))
demographic.summary( input = genophenoExData,
                     maleCode   = "male",
                     femaleCode = "female"
          )
```

extract — *Obtain the raw query from a* `cupcakeData` *and* `cupcakeResults` *object.*

Description

Obtain the raw query from a `cupcakeData` and `cupcakeResults` object.

Usage

```
extract(object, ...)
```

Arguments

`object`	of class `cupcakeData` or `cupcakeResults` object

Value

A `data.frame` containing the raw data (cupcakeData)or co-occurrence analysis results (cupcakeResults)

Examples

```
## Not run:
#Being x an cupcakeResults
qr <- extract(x)

## End(Not run)
```

get.children — *Get children paths*

Description

Given a url, a key and a path it returns all the children paths under the given one.

Usage

```
get.children(fieldname, url, verbose = FALSE)
```

Arguments

`fieldname`	The path in which the urser is interested.
`url`	The url.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.
`apiKey`	The key to access to the data.

Value

A vector with all the fields that are under the path that has been given as input.

Examples

```
nhanesPcbs <- get.children(
                 fieldname   = "/nhanes/Demo/laboratory/laboratory/pcbs/",
                 url         = "https://nhanes.hms.harvard.edu/"
              )
```

---

my.data *Query analysis to the API*

Description

Given an url and a JSON object, it generates a `data.frame` object with the output of the query.

Usage

```
my.data(query, url, responseFormat = "CSV", outputPath = paste0(getwd(),
  "/queryData.txt"), verbose = FALSE)
```

Arguments

`query`	A text file containing the JSON query body.
`url`	The url.
`outputPath`	Path and the file name where the output file will be saved. By default it will be saved in the working directory with the name queryData.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.

Value

An object of class `data.frame` with the query output.

Examples

```
query <- my.data(
              query  = system.file("extdata", "jsonQueryNhanes", package="Rcupcake"),
              url    = "https://nhanes.hms.harvard.edu/"
              )
```

`my.query` *Query analysis to the API*

Description

Given a vector with the fields of interest, and the paths vector generated applying the getchildren function, it returns a JSON query

Usage

```
my.query(myfields, myvector, verbose = FALSE)
```

Arguments

`myfields`	A vector with the fields of interest
`myvector`	A vector with the paths of interest, generated applying the getchildren function
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.

Value

A JSON query.

Examples

```
nhanesPcbs <- getchildren(
              url         = "https://nhanes.hms.harvard.edu/",
              apiKey      = "YOURKEY",
              fieldname   = "/nhanes/Demo/laboratory/laboratory/pcbs/"
              )
queryExample <- my.query( myfields = "AGE|PCB153",
                          myvector = nhanesPcbs
              )
```

`n.phenotype` *Getter from* `cupcakeData`.

Description

Obtain the phenotypes in a `cupcakeData`.

Usage

```
n.phenotype(object)

## S4 method for signature 'cupcakeData'
n.phenotype(object)
```

Arguments

`object`	Object of class `cupcakeData`.

Value

The number of unique phenotypes

Methods (by class)

- `cupcakeData`: get the distinct phenotypes

Examples

```
data(qr)
n.phenotype(qr)
```

`n.variation` *Getter from* `cupcakeData`.

Description

Obtain the alteration variables in a `cupcakeData`.

Usage

```
n.variation(object)
```

Arguments

`object`	Object of class `cupcakeData`.

Examples

```
data(qr)
n.variation(qr)
```

`patient.selection` *Patients selection* `genophenoPatientsSelection`

Description

Given an object of type genopheno and two phenotypes of interest, the patients identifiers of those patients having both phenotypes are selected.

Usage

```
patient.selection(input, pth, ageRange = c(0, 100), phenotypeA, phenotypeB,
  aggregate = TRUE, gender = "ALL", variation = c("ALL", "ALL"),
  nfactor = 10, cores = 1, verbose = FALSE, warnings = TRUE)
```

Arguments

`input`	A genopheno object, obtained with the queryPheno function.
`pth`	Determines the path where the required input file with the yes/no phenotype data is located.
`ageRange`	Determines what is the age range of interest for performing the comorbidity analysis. By default it is set from 0 to 100 years old.
`phenotypeA`	One of the phenotypes of interest and the value in which user is interested. For example, c("FacialExpression", "yes)
`phenotypeB`	The second phenotype of interest and the value in which user is interested. For example, c("HandMovement", "yes)
`aggregate`	By default TRUE. Change it to FALSE if you want to analyze the comorbidity taking into all the values of each phenotype.
`gender`	Determine what is the gender of interest for performing the comorbidity analysis. By default `ALL`. Change it to the gender of interest for your comorbidity analysis.
`variation`	Determine what is the variation value of interest for performing the comorbidity analysis. By default `c("ALL", "ALL")`. Change it to the value of interest for your comorbidity analysis. For example, `c("CHD8", "yes")`
`nfactor`	By default 10. Change it into other number if you consider there is any categorical variable with more than nfactor values.
`cores`	By default `1`. To run parallel computations on machines with multiple cores or CPUs, the cores argument can be changed.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.
`warnings`	By default `TRUE`. Change it to `FALSE` to don't see the warnings.

Value

An object of class `cgpAnalysis`

Examples

```
load(system.file("extdata", "cupcakeData.RData", package="Rcupcake"))
ex1 <- patient.selection(
              input         = cupcakeData,
              pth           = system.file("extdata", package="Rcupcake"),
              phenotypeA    = c("Herpes", "yes"),
              phenotypeB    = c("HIV", "yes"),
              aggregate     = TRUE,
              ageRange      = c(18,40),
              gender        = "male",
              )
```

pheno.prevalence *Generates a table with the prevalence of each phenotype*

Description

Given an object of class cupcakeResults obtained from the co-occurrence analysis, a prevalence table is obtained.

Usage

```
pheno.prevalence(input, verbose = FALSE, warnings = FALSE)
```

Arguments

input	A cupcakeResults object, obtained by applying the co.occurrence function
verbose	By default FALSE. Change it to TRUE to get an on-time log from the function.
warnings	By default TRUE. Change it to FALSE to don't see the warnings.

Value

A table

Examples

```
load(system.file("extdata", "RcupcakeExResult.RData", package="Rcupcake"))
pheno.prevalence <- genoPhenoPrevalence(
             input           = cupcakeResults
             )
```

phenotype.summary *Describes the phenotypic characteristics for the whole study population. If gene is selected, the summary will contain the results regardin the selected gene.*

Description

Given an object of class genopheno, a file with the different values for each phenotype, and the prevalence of each one in general population and according to the gene status, if present, is generated. A figure containing a barplot or boxplot for each phenotype is displayed. Each barplot shows the population percentage suffering each type of the phenotypes according to the values it takes, and distinguishing between those having or not a variation, if present. Furthermore, a data.frame with the numerical values is obtained.

Usage

```
phenotype.summary(input, variation = FALSE, nfactor = 10,
  showTable = TRUE, showFigures = FALSE, pth = getwd(), verbose = FALSE)
```

Arguments

`input`	Object of genopheno class.
`variation`	Determines the variation of interest for which you want to analyze the phenotype values. By default FALSE.
`nfactor`	By default 10. Change it into other number if you consider there is any categorical variable with more than `nfactor` values.
`showTable`	By default TRUE. Change it into FALSE in order to not visualize the table with the results.
`showFigures`	By default FALSE. Change it into TRUE in order to visualize the table with the results.
`pth`	By default the working directory. Define the pth where you want the file to be saved
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.

Value

A file .

Examples

```
load(system.file("extdata", "genophenoExData.RData", package="genophenoR"))
phenotype.summary( input       = genophenoExData,
                  showTable   = TRUE,
                  showFigures = TRUE,
                  verbose      = FALSE
          )
```

`prevalence.plot` *Plot the phenotype prevalence in a barplot (when there is not any variation) and in a network if a variation is defined*

Description

Given a `cupcakeResults` object a barplot or a network is obtained.

Usage

```
prevalence.plot(input, layout = "layout.circle", variation,
  interactive = FALSE, verbose = FALSE)
```

Arguments

`input`	A `cupcakeResults` object, obtained by applying the `comorbidityAnalysis` function
`layout`	By default `'layout.fruchterman.reingold'`. It can be set to any other of the possible igraph layouts.
`variation`	Determine what is the variation value of interest for performing the comorbidity analysis.

`interactive`	Determines if the output network is interactive or not. By default the `interactive` argument is set up as `FALSE`. The value of the argument can be changed to `TRUE`, as a result an interactive network will be obtained.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.
`warnings`	By default `TRUE`. Change it to `FALSE` to don't see the warnings.

Value

A barplot or a network

Examples

```
load(system.file("extdata", "cupcakeResultExample.RData", package="Rcupcake"))
ntwk <- prevalence.plot(
             input            = cupcakeResultExample
             )
```

`start.session` *Start the connection to the database*

Description

Given an url and a key it start the connection to the database

Usage

```
## S3 method for class 'session'
start(url, apiKey)
```

Arguments

`url`	The url.
`apiKey`	The key to access to the data.

Value

A message showing if the connection has been done or not.

Examples

```
sessionEx <- start.session(
             url        = "https://nhanes.hms.harvard.edu/",
             apiKey     = "YOURKEY"
             )
```

z.score — *Transform continuous in categorical variables and generates a new* `cupcakeData` *object.*

Description

Given an object of class `cupcakeData`, it transforms continuous into categorical variable applying Z-score. As a result a new `cupcakeData` object is generated. Note that if the number of individuals is lower than 5000 a Saphiro test is done to test the normal distribution, otherwise a Kolmogorov-Smirnov test is performed.

Usage

```
z.score(input, zscoreCutOff = c(-2, 2), nfactor = 10, verbose = FALSE)
```

Arguments

`input`	Object of `cupcakeData` class.
`zscoreCutOff`	Z-score cut-off to categorize the continuous variable. By default it is set to -2 and 2.
`nfactor`	By default 10. Change it into other number if you consider there is any categorical variable with more than nfactor values.
`verbose`	By default `FALSE`. Change it to `TRUE` to get an on-time log from the function.

Value

A `cupcakeData` class object with the continuous variable transformed into a categorical variable, if possible.

Examples

```
load(system.file("extdata", "cupcakeDataExample.RData", package="Rcupcake"))
z.score( input      = cupcakeDataExample,
         verbose    = FALSE
        )
```

Index