Package 'Rcupcake'

June 27, 2017

Type Package

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

Version 0.99.0

Description
Package for querying and analyzing biomedical data through the BD2K PIC-SURE RESTful API

Depends R (>= 3.2.3), stringr

License MIT + file LICENSE

LazyData true

Suggests parallel, testthat

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, 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 performed, for the subset of population under specific conditions of age, gender and variation status (exposure, gene mutation). It generates a `cupcakeResults` object.

Usage

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

Arguments

`input`	A `cupcakeData` object, obtained after applying the my.data function.
`pth`	Determines the path where the required file with phenotype data is located. This file is generated applying the `phenotype.summary` function.
`ageRange`	Determines what is the age range of interest for performing the co-occurrence analysis. By default it is set from 0 to 100 years old.
`aggregate`	By default FALSE. Change it to TRUE if you want to analyze the co-occurrence aggregating the values of several phenotypes as the same category.
`gender`	Determine what is the gender of interest for performing the co-occurrence analysis. By default `ALL`. Change it to the gender of interest for your co-occurrence analysis if interested.
`variation`	Determine what is the variation of interest for performing the co-occurrence analysis. By default `c("", "")`. Change it to the values of interest for your co-occurrence 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 co-occurrence score is a measure based on the observed co-occurrence 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 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 co-occurrence association.

cores
: By default 1. To run parallel computations on machines with multiple 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", "RcupcakeExData.RData", package="Rcupcake"))
cooccurrenceExample <- co.occurrence(
             input         = RcupcakeExData,
             pth           = system.file("extdata", package="Rcupcake"),
             aggregate     = FALSE,
             ageRange      = c(0,16),
             gender        = "male",
             )
```

---

comparison2b2 *Comparison by pairs of variables in a* cupcakeData *object*

Description

Given an object of type cupcakeData and two variables of the data.frame a statistically comparison is performed. According to the type of variable different test will be run (fisher test, t-test or phearson test). As a result the p-value and some other information will be return.

Usage

```
comparison2b2(input, variable1, variable2, nfactor = 10, verbose = FALSE)
```

Arguments

input
: A cupcakeData object, obtained after applying the my.data function.

variable1
: The name of the first variable that has to be the same that the colname of the variable in the input data.frame.

variable2
: The name of the second variable that has to be the same that the colname of the variable in the input data.frame.

nfactor
: By default 10. Change it into other number if you consider there is any continuous variable with less than nfactor values.

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", "RcupcakeExData.RData", package="Rcupcake"))
comparison <- comparison2b2(
              input         = RcupcakeExData,
              variable1     = "Age",
              variable2     = "Gender"
              )
```

---

cooc.heatmap *Plot the co-occurrence analysis results in a heatmap.*

Description

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

Usage

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

Arguments

input	A cupcakeResults object, obtained by applying the co.occurrence 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'. 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 heatmap 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 "yellow".
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="Rcupcake"))
htmp <- cooc.heatmap( input        = cupcakeResults,
              representedVariable  = "patientsPhenoAB",
              variableCutOff       = 1,
              coocPatients         = 1
              )
htmp
```

`cooc.network` *Plot the co-occurrence analysis results in a network*

Description

Given an object of class `cupcakeResults` obtained from a co-occurrence 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 `cupcakeResults` object, obtained by applying the `co.occurrence` function
`layout`	By default `'layout.circle'`. 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'`. 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 heatmap 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.

Value

A network

Examples

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

---

dataframe2cupcake *Transform a data.frame into a* cupcakeData *object*

Description

Given a tabulated file that contains the data in the correct format and generates a cupcakeData object.

Usage

```
dataframe2cupcake(input, phenotypes, variants, age, gender, 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 the "patient_id", two demographic variables, "Gender" and "Age", and a list of phenotypes and variations
phenotypes	Vector that contains the phenotype variables
variants	Vector that contains the variants names
age	Vector that contains the age variable
gender	Vector that contains the gender variable
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"),
                                   age        = "Age",
                                   gender     = "Gender",
                                phenotypes = "Diabetes|Arthritis|LiverCancer|AnyCancer",
                                   variants   = "PCB153",
                                   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, a barplot with age distribution, a boxplot representing age distribution by gender and a pie chart representing 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", "RcupcakeExData.RData", package="Rcupcake"))
demographic.summary( input = RcupcakeExData,
                     maleCode   = "male",
                     femaleCode = "female"
           )
```

extract *Obtain the raw data from a* cupcakeData *and* cupcakeResults *object.*

Description

Obtain the raw data 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.

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 JSON query, created with my.query function or contained in a text file.
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 vector generated with the paths obtained after applying the getchildren function, it returns a JSON query

Usage

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

Arguments

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

Value

A JSON query.

`n.phenotype` *Get the phenotypic variables from* `cupcakeData`.

Description

Obtain the phenotypes in a `cupcakeData`.

Usage

```
n.phenotype(object)
```

Arguments

`object`	Object of class `cupcakeData`.

Examples

```
#Being qr an cupcakeData object
#n.phenotype(qr)
```

`n.variation` *Get the alteration variables from* `cupcakeData`.

Description

Obtain the alteration variables in a `cupcakeData`.

Usage

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

Arguments

`object`	Object of class `cupcakeData`.

Examples

```
#Being qr an cupcakeData object
#n.variation(qr)
```

patient.selection *Patients selection*

Description

Given an object of type cupcakeData and two phenotypes of interest, the 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"),
  verbose = FALSE, warnings = TRUE)
```

Arguments

input	A cupcakeData object, obtained with the my.data function.
pth	Determines the path where the required input file with the phenotypic data is located.
ageRange	Determines what is the age range of interest for performing the co-occurrence 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("Arthritis", "Yes")
phenotypeB	The second phenotype of interest and the value in which user is interested. For example, c("Diabetes", "Yes")
aggregate	By default TRUE. Change it to FALSE if you want to analyze the co-occurrence taking into all the values of each phenotype.
gender	Determine what is the gender of interest for performing the co-occurrence analysis. By default ALL. Change it to the gender of interest for your co-occurrence analysis.
variation	Determine what is the variation value of interest for performing the co-occurrence analysis. By default c("ALL", "ALL"). Change it to the value of interest for your co-occurrence analysis. For example, c("CHD8", "yes")
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", "RcupcakeExData.RData", package="Rcupcake"))
ex1 <- patient.selection(
             input         = RcupcakeExData,
             pth           = system.file("extdata", package="Rcupcake"),
             phenotypeA    = c("Arthritis", "Yes"),
             phenotypeB    = c("Diabetes", "Yes"),
             aggregate     = FALSE,
```

```
ageRange      = c(0,100),
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"))
phenoPrevalenceExample <- pheno.prevalence(
             input            = cupcakeResults
             )
```

---

phenotype.summary *Describes the phenotypic characteristics for the whole study population. If variation status is selected, the summary will contain the results regarding the selected variation*

Description

Given an object of class cupcakeData, a file with the different values for each phenotype, and the prevalence of each one in general population and according to the variation 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 cupcakeData 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", "RcupcakeExData.RData", package="Rcupcake"))
phenotype.summary( input       = RcupcakeExData,
                   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, verbose = FALSE)
```

Arguments

input	A cupcakeResults object, obtained by applying the co.occurrence function
layout	By default 'layout.circle'. It can be set to any other of the possible igraph layouts.
variation	Determine what is the variation value of interest for performing the co-occurrence analysis.
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", "RcupcakeExResult.RData", package="Rcupcake"))
ntwk <- prevalence.plot(
              input            = cupcakeResults
              )
```

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

Description

Given a URL and a key access it starts 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", "RcupcakeExData.RData", package="Rcupcake"))
z.score( input      = RcupcakeExData,
         verbose    = FALSE
       )
```

Index