Package ‘emcAdr’

February 5, 2026

Type Package

Title Evolutionary Version of the Metropolis-Hastings Algorithm

Version 1.3

Date 2026-02-04

Author Jules Bangard [aut, cre] (ORCID:
<https://orcid.org/0009-0007-4670-7860>)

Maintainer Jules Bangard <jules.bangard@etu.unistra.fr>

Description Provides computational methods for detecting adverse high-order drug interactions
from individual case safety reports using statistical techniques,
allowing the exploration of higher-order interactions among drug cocktails.

License GPL-3

Imports Rcpp (>= 1.0.7), ggplot2, dplyr, umap, dbscan, stats, logistf

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

LazyData true

Suggests knitr, rmarkdown, gridExtra

VignetteBuilder knitr

NeedsCompilation yes

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2026-02-05 12:10:02 UTC

Contents

`emcAdr-package` *Evolutionary Version of the Metropolis-Hastings Algorithm*

Description

Provides computational methods for detecting adverse high-order drug interactions from individual case safety reports using statistical techniques, allowing the exploration of higher-order interactions among drug cocktails.

Author(s)

Jules Bangard [aut, cre] (<https://orcid.org/0009-0007-4670-7860>)

Maintainer: Jules Bangard <jules.bangard@etu.unistra.fr>

See Also

Rcpp, RcppArmadillo

`ATCtoNumeric` *Convert ATC Code for each patients to the corresponding DFS number of the ATC tree*

Description

Convert ATC Code for each patients to the corresponding DFS number of the ATC tree

Usage

```
ATCtoNumeric(patientATC, tree)
```

Arguments

`patientATC`	: patients observations, for each patient we got a string containing taken medications (ATC code)
`tree`	: ATC tree (we assume that there is a column 'ATCCode')

Value

a matrix of the same size as patientATC but containing integer that are the index of the corresponding ATC code.

Examples

```
ATC_code <- c('A01AA30 A01AB03', 'A10AC30')
ATCtoNumeric(ATC_code, ATC_Tree_UpperBound_2024)
```

`ATC_Tree_UpperBound_2024` *ATC Tree Upper Bound 2024*

Description

Example dataset representing the ATC tree structure, sourced from the WHO website (2024-02-23). This dataset is provided for demonstration and testing purposes with the package.

Usage

```
ATC_Tree_UpperBound_2024
```

Format

A data frame with 4 variables:

ATCCode The code of ATC nodes

Name The name of ATC nodes

ATC_length The number of characters in the ATCCode

upperBound The index of the last child node in the tree

Source

World Health Organization, ATC classification register

`calculate_divergence` *Calculate the divergence between 2 distributions (the true Distribution and the learned one)*

Description

Calculate the divergence between 2 distributions (the true Distribution and the learned one)

Usage

```
calculate_divergence(
  empirical_distribution,
  true_distribution,
  method = "TV",
  Filtered = FALSE
)
```

Arguments

`empirical_distribution`	A numeric vector of values representing the empirical distribution (return value of DistributionAproximation function)
`true_distribution`	A numeric vector of values representing the true distribution computed by the trueDistributionSizeTwoCocktail function
`method`	A string, either "TV" or "KL" to respectively use the total variation distance or the Kullback-Leibler divergence. (default = "TV")
`Filtered`	Should we use the filtered distribution or the normal one

Value

A numeric value representing the divergence of the 2 distributions

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

estimated_score_distribution = DistributionApproximation(epochs = 10,
            ATCtree = ATC_Tree_UpperBound_2024,
            observations = FAERS_myopathy[1:100,], Smax =2)

true_score_distribution = trueDistributionSizeTwoCocktail(ATCtree = ATC_Tree_UpperBound_2024,
            observations = FAERS_myopathy[1:100,], beta = 4)

divergence <- calculate_divergence(empirical_distribution = estimated_score_distribution,
                true_distribution = true_score_distribution)
```

`clustering_genetic_algorithm`

Clustering of the solutions of the genetic algorithm using the hclust algorithm

---

Description

Clustering of the solutions of the genetic algorithm using the hclust algorithm

Usage

```
clustering_genetic_algorithm(
  genetic_results,
  ATCtree,
  dist.normalize = TRUE,
  umap_config = NULL
)
```

Arguments

`genetic_results`	A list of cocktails in the form of integer vector
`ATCtree`	ATC tree with upper bound of the DFS
`dist.normalize`	Do we normalize the distance (so it belongs to [0;1])
`umap_config`	The configuration to use in order to project the cocktails in a smaller space (umap::umap.defaults by default)

Value

A dataframe containing UMAP 1/2 the two coordinates of each cocktails in the plane as well as the cluster number of each cocktails

Examples

```
data("ATC_Tree_UpperBound_2024")

results = GeneticAlgorithm(epochs = 10, nbIndividuals = 10,
           ATCtree = ATC_Tree_UpperBound_2024,
           observations = FAERS_myopathy)

hclust_genetic_solution(genetic_results = results,
                ATCtree = ATC_Tree_UpperBound_2024)
```

`combination_data_frame`
Generate Matrix for Drug Combinations

Description

This function creates a logical data frame where each column represents a specific sub-combination of drugs derived from a given "cocktail." For each patient in the input data, it indicates (TRUE/FALSE) whether they were taking that specific combination based on the ATC hierarchy.

Usage

```
combination_data_frame(cocktail, upperBound, data)
```

Arguments

`cocktail`	An integer vector of drug indices representing the full combination to be analyzed.
`upperBound`	An integer vector defining the ATC tree hierarchy bounds.
`data`	A `Rcpp::DataFrame` containing patient records. It must include a column `"patientATC"` which is a list of integer vectors representing the drugs each patient is taking.

Details

The function first generates all possible non-empty power-set combinations of the `cocktail` (e.g., for $\{1, 2\}$, it generates $\{1\}, \{2\}, \{1, 2\}$).

Value

A `Rcpp::DataFrame` where:

- Each column corresponds to a sub-combination of the input `cocktail`.
- Each row corresponds to a patient in the input `data`.
- Values are boolean indicators (represented as integers/logicals in R).

Note

The column names of the resulting data frame are strings of comma-separated drug indices (e.g., "888,659,").

`computeMetrics_size2` *Function used in the reference article to compare diverse Disproportionality Analysis metrics*

Description

Function used in the reference article to compare diverse Disproportionality Analysis metrics

Usage

```
computeMetrics_size2(CocktailList, ATCtree, observations, num_thread = 1L)
```

Arguments

`CocktailList`	: A list of cocktails on which the Disproportionality analysis metrics should be computed
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`observations`	: observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default

Value

Multiple DA metrics computed on CocktailList cocktails

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

cocktails = list(c(561, 904),
                 c(1902, 4585)) # only size 2 cocktails allowed for this function

scores_of_cocktails = computeMetrics_size2(CocktailList = cocktails,
                                           ATCtree = ATC_Tree_UpperBound_2024,
                                           observations = FAERS_myopathy[1:100,])
```

compute_hypergeom_cocktail

Function used to compute the Hypergeometric score on a cocktail

Description

Function used to compute the Hypergeometric score on a cocktail

Usage

```
compute_hypergeom_cocktail(
  cocktail,
  upperBounds,
  ADRCount,
  observationsADR,
  observationsMedication,
  num_thread = 1L
)
```

Arguments

`cocktail`	: A cocktail in the form of vector of integers (ATC index)
`upperBounds`	: ATC tree upper bound of the DFS (without the root)
`ADRCount`	: number of patient experiencing ADR in dataset
`observationsADR`	: observation of the ADR for each patients (a vector containing the ADR on which we want to compute the risk distribution)
`observationsMedication`	: observation of the drug intake for each patients on which we want to compute the risk distribution
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default

Value

Hypergeometric score of the "cocktail" parameter

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

ADRCount = sum(FAERS_myopathy$patientADR)
cocktail = c(561, 904)

Hypergeom_of_cocktail = compute_hypergeom_cocktail(cocktail = cocktail,
                                   upperBounds = ATC_Tree_UpperBound_2024$upperBound,
                                   ADRCount =  ADRCount,
                                   observationsADR = FAERS_myopathy$patientADR,
                                   observationsMedication = FAERS_myopathy$patientATC,
                                   num_thread=8)
```

compute_hypergeom_on_list

Function used to compute the Hypergeometric score on a list of cocktails

Description

Function used to compute the Hypergeometric score on a list of cocktails

Usage

```
compute_hypergeom_on_list(cocktails, ATCtree, observations, num_thread = 1L)
```

Arguments

`cocktails`	: A list containing cocktails in the form of vector of integers (ATC index)
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`observations`	: observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default

Value

Hypergeometric score among "cocktails" parameters

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

cocktails = list(c(561, 904),
               c(1902, 4585))

Hypergeom_of_cocktails = compute_hypergeom_on_list(cocktails = cocktails,
                              ATCtree = ATC_Tree_UpperBound_2024,
                              observations = FAERS_myopathy)
```

`compute_RR_on_list` *Function used to compute the Relative Risk on a list of cocktails*

Description

Function used to compute the Relative Risk on a list of cocktails

Usage

```
compute_RR_on_list(cocktails, ATCtree, observations, num_thread = 1L)
```

Arguments

`cocktails`	: A list containing cocktails in the form of vector of integers (ATC index)
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`observations`	: observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default

Value

RR score among "cocktails" parameters

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

cocktails = list(c(561, 904),
                 c(1902, 4585))

RR_of_cocktails = compute_RR_on_list(cocktails = cocktails,
                                     ATCtree = ATC_Tree_UpperBound_2024,
                                     observations = FAERS_myopathy)
```

`csv_to_population` *Function used to convert your genetic algorithm results that are stored into a .csv file to a Data structure that can be used by the clustering algorithm*

Description

Function used to convert your genetic algorithm results that are stored into a .csv file to a Data structure that can be used by the clustering algorithm

Usage

```
csv_to_population(ATC_name, filename, sep = ";")
```

Arguments

`ATC_name`	the ATC_name column of the ATC tree
`filename`	Name of the file where the results are located
`sep`	the separator to use when opening the csv file (';' by default)

Value

An R List that can be used by other algorithms (e.g. clustering algorithm)

Examples

```
data("ATC_Tree_UpperBound_2024")
genetic_results = csv_to_population(ATC_Tree_UpperBound_2024$Name,
                  "path/to/output.csv")
```

`DistributionApproximation` *The MCMC method that runs the random walk on a single cocktail in order to estimate the distribution of score among cocktails of size Smax.*

Description

The MCMC method that runs the random walk on a single cocktail in order to estimate the distribution of score among cocktails of size Smax.

Usage

```
DistributionApproximation(
  epochs,
  ATCtree,
  observations,
  temperature = 1L,
  nbResults = 5L,
  Smax = 2L,
  p_type1 = 0.01,
  beta = 4L,
  max_score = 500L,
  num_thread = 1L,
  verbose = FALSE
)
```

Arguments

`epochs`	: number of steps for the MCMC algorithm
`ATCtree`	: ATC tree with upper bound of the DFS (without the root, also see on the github repo for an example)
`observations`	: real observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second)
`temperature`	: starting temperature, default = 1 (denoted T in the article)
`nbResults`	: Number of returned solution (Cocktail of size Smax with the best oberved score during the run), 5 by default
`Smax`	: Size of the cocktail we approximate the distribution from
`p_type1`	: probability to operate type1 mutation. Note : the probability to operate the type 2 mutation is then 1 - P_type1. P_type1 must be in [0;1]. Default is .01
`beta`	: filter the minimum number of patients that must have taken the cocktail for his risk to be taken into account in the DistributionScoreBeta default is 4
`max_score`	: maximum number the score can take. Score greater than this one would be added to the distribution as the value max_score. Default is 500
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default
`verbose`	: Output summary (default is false)

Value

I no problem, return a List containing : - ScoreDistribution : the distribution of the score as an array with each cells representing the number of risks = (index-1)/ 10 - Outstanding_score : An array of the score greater than max_score, - Best_cocktails : the nbResults bests cocktails encountered during the run. - Best_scores : Score corresponding to the bestCocktails. - Filtered_score_distribution : Distribution containing score for cocktails taken by at least beta patients. - Best_cocktails_beta : the nbResults bests cocktails taken by at least beta patients encountered during the run. - Best_scores_beta : Score corresponding to the bestCocktailsBeta. - cocktailSize : Smax parameter used during the run. ; Otherwise the list is empty

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

estimation = DistributionApproximation(epochs = 10, ATCtree = ATC_Tree_UpperBound_2024,
             observations = FAERS_myopathy)
```

`FAERS_myopathy` *FAERS Myopathy Dataset*

Description

Example dataset representing drug intake and adverse event reports from FAERS. This dataset is provided to demonstrate the functionality of genetic and MCMC algorithms in the package.

Usage

```
FAERS_myopathy
```

Format

A data frame with 2 columns:

patientATC Drug intake for each patient as a vector of ATC tree indices

patientADR Indicates if the patient experienced myopathy as an adverse event

Source

Food & Drug Administration Event Reporting System (FAERS)

`GeneticAlgorithm` *Genetic algorithm, trying to reach riskiest cocktails (the ones which maximize the fitness function, Hypergeometric score in our case)*

Description

Genetic algorithm, trying to reach riskiest cocktails (the ones which maximize the fitness function, Hypergeometric score in our case)

Usage

```
GeneticAlgorithm(
  epochs,
  nbIndividuals,
  ATCtree,
  observations,
  num_thread = 1L,
  diversity = FALSE,
  p_crossover = 0.8,
  p_mutation = 0.01,
  nbElite = 0L,
  tournamentSize = 2L,
  alpha = 1,
  summary = TRUE
)
```

Arguments

`epochs`	: number of step or the algorithm
`nbIndividuals`	: size of the population
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`observations`	: real observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second)
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default
`diversity`	: enable the diversity mechanism of the algorithm (favor the diversity of cocktail in the population), default is false
`p_crossover`	: probability to operate a crossover on the crossover phase. Default is 80%
`p_mutation`	: probability to operate a mutation after the crossover phase. Default is 1%
`nbElite`	: number of best individual we keep from generation to generation. Default is 0
`tournamentSize`	: size of the tournament (select the best individual between tournamentSize sampled individuals)
`alpha`	: when making a type 1 mutation you have (alpha / size of cocktail) chance to add a drug.
`summary`	: print the summary of population at each steps ?

Value

If no problem, return a List : - meanFitnesses : The mean score of the population at each epochs of the algorithm. - BestFitnesses : The best score of the population at each epochs of the algorithm. - FinalPopulation : The final population of the algorithm when finished (medications and corresponding scores)

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

results = GeneticAlgorithm(epochs = 10, nbIndividuals = 10,
            ATCtree = ATC_Tree_UpperBound_2024,
            observations = FAERS_myopathy)
```

---

get_dissimilarity_from_cocktail_list

Recover the square matrix of distance between cocktails where the index (i,j) of the matrix is the distance between cocktails i and j in an arbitrary cocktail list

---

Description

Recover the square matrix of distance between cocktails where the index (i,j) of the matrix is the distance between cocktails i and j in an arbitrary cocktail list

Usage

```
get_dissimilarity_from_cocktail_list(cocktails, ATCtree, normalization = TRUE)
```

Arguments

`cocktails`	: A list of cocktails in the form of a vector of integer
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`normalization`	: Do we keep the distance between cocktail in the range [0;1] ?

Value

The square matrix of distances between cocktails

Examples

```
data("ATC_Tree_UpperBound_2024")

cocktails = list(c(561, 904),
              c(1902, 4585)) # only size 2 cocktails allowed for this function

distance_matrix = get_dissimilarity_from_cocktail_list(cocktails = cocktails,
                                ATCtree = ATC_Tree_UpperBound_2024,
                                normalization = TRUE)
```

---

get_dissimilarity_from_genetic_results

Recover the square matrix of distance between cocktails where the index (i,j) of the matrix is the distance between cocktails i and j in the genetic_results list.

Description

Recover the square matrix of distance between cocktails where the index (i,j) of the matrix is the distance between cocktails i and j in the genetic_results list.

Usage

```
get_dissimilarity_from_genetic_results(genetic_results, ATCtree, normalization)
```

Arguments

`genetic_results`	the List returned by the genetic algorithm.
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`normalization`	: Do we keep the distance between cocktail in the range [0;1] ?

Value

The square matrix of distances between cocktails

Examples

```
 data("ATC_Tree_UpperBound_2024")
 data("FAERS_myopathy")

 genetic_results = GeneticAlgorithm(epochs = 10, nbIndividuals = 10,
           ATCtree = ATC_Tree_UpperBound_2024,
           observations = FAERS_myopathy)
distance_matrix = get_dissimilarity_from_genetic_results(genetic_results = genetic_results,
                       ATCtree = ATC_Tree_UpperBound_2024, normalization = TRUE)
```

`get_dissimilarity_from_txt_file` *Recover the square matrix of distance between cocktails where the index (i,j) of the matrix is the distance between cocktails i and j in the csv file containing results of genetic algorithm*

Description

Recover the square matrix of distance between cocktails where the index (i,j) of the matrix is the distance between cocktails i and j in the csv file containing results of genetic algorithm

Usage

```
get_dissimilarity_from_txt_file(filename, ATCtree, normalization = TRUE)
```

Arguments

`filename`	: the name of the file returned by the print_csv function.
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`normalization`	: Do we keep the distance between cocktail in the range [0;1] ?

Value

The square matrix of distances between cocktails

Examples

```
data("ATC_Tree_UpperBound_2024")

distance_matrix = get_dissimilarity_from_txt_file(filename = '250e_700ind_0.2mr_0ne_2alpha.txt',
                          ATCtree = ATC_Tree_UpperBound_2024, normalization = TRUE)
```

`hclust_genetic_solution` *Clustering of the solutions of the genetic algorithm using the hclust algorithm*

Description

Clustering of the solutions of the genetic algorithm using the hclust algorithm

Usage

```
hclust_genetic_solution(
  genetic_results,
  ATCtree,
  dist.normalize = TRUE,
  method = "complete"
)
```

Arguments

`genetic_results`	The return value of the genetic algorithm
`ATCtree`	ATC tree with upper bound of the DFS
`dist.normalize`	Do we normalize the distance (so it bellongs to [0;1])
`method`	(from hclust function) the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

Value

the hierarchical clustering of the results of the genetic algorithm

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

results = GeneticAlgorithm(epochs = 10, nbIndividuals = 10,
           ATCtree = ATC_Tree_UpperBound_2024,
            observations = FAERS_myopathy)

hclust_genetic_solution(genetic_results = results,
                 ATCtree = ATC_Tree_UpperBound_2024)
```

histogramToDitribution

Convert the histogram returned by the DistributionApproximation function, to a real number distribution (that can be used in a test for example)

Description

Convert the histogram returned by the DistributionApproximation function, to a real number distribution (that can be used in a test for example)

Usage

```
histogramToDitribution(vec)
```

Arguments

`vec`	: distribution returned by the DistributionAproximationFunction

Value

A vector containing sampled risk during the MCMC algorithm

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

 DistributionApproximationResults = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy)
  histogramToDitribution(DistributionApproximationResults$ScoreDistribution)
```

`hyperparam_test_genetic_algorithm` *This function can be used in order to try different set of parameters for the genetic algorithm in a convenient way. This will run each combination of mutation_rate, nb_elite and alphas possible nb_test_desired times. For each sets of parameters, results will be saved in a file named according to the set of parameter. One can regroup the results of each run in a csv file by using the print_csv function specifying the names of each file that needs to be treated and the number of performed runs on each parameter set*

Description

This function can be used in order to try different set of parameters for the genetic algorithm in a convenient way. This will run each combination of mutation_rate, nb_elite and alphas possible nb_test_desired times. For each sets of parameters, results will be saved in a file named according to the set of parameter. One can regroup the results of each run in a csv file by using the print_csv function specifying the names of each file that needs to be treated and the number of performed runs on each parameter set

Usage

```
hyperparam_test_genetic_algorithm(
  epochs,
  nb_individuals,
  ATCtree,
  observations,
  nb_test_desired,
  mutation_rate,
  nb_elite,
  alphas,
  path = "./",
  num_thread = 1L
)
```

Arguments

`epochs`	: the number of epochs for the genetic algorithm
`nb_individuals`	: the size of the population in the genetic algorithm
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`observations`	: observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
`nb_test_desired`	: number of genetic algorithm runs on each sets of parameters
`mutation_rate`	: a vector with each mutation_rate to be tested

`nb_elite`	: a vector with each nb_elite to be tested
`alphas`	: a vector with each alphas to be tested
`path`	: the path where the resulting files should be written
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default

Value

No return value, this function should output results of the runs of the genetic algorithm in a specific format supported by function print_csv and p_value_csv_file. The files are outputed in path which is current directory by default.

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

# different parameter to test for
mutation_rate = c(.1,.2,.3)
nb_elite = c(0,1,2)
alphas = c(0.5,1,2)
hyperparam_test_genetic_algorithm(epochs = 2, nb_individuals = 2,
                                  ATCtree = ATC_Tree_UpperBound_2024,
                                  observations = FAERS_myopathy,
                                  nb_test_desired = 5, mutation_rate = mutation_rate,
                                  nb_elite = nb_elite, alphas = alphas)
```

`int_cocktail_to_string_cocktail` *Function used to convert integer cocktails (like the one outputed by the distributionApproximation function) to string cocktail in order to make them more readable*

Description

Function used to convert integer cocktails (like the one outputed by the distributionApproximation function) to string cocktail in order to make them more readable

Usage

```
int_cocktail_to_string_cocktail(cocktails, ATC_name)
```

Arguments

`cocktails`	cocktails vector to be converted (index in the ATC tree)
`ATC_name`	The ATC_name column of the ATC tree

Value

The name of integer cocktails in cocktails

Examples

```
data("ATC_Tree_UpperBound_2024")
int_list = list(c(561, 904),
                c(1902, 4585))
int_cocktail_to_string_cocktail(int_list, ATC_Tree_UpperBound_2024$Name)
```

OutsandingScoreToDistribution

Output the outstanding score (Outstanding_score) outputed by the MCMC algorithm in a special format

---

Description

Output the outstanding score (Outstanding_score) outputed by the MCMC algorithm in a special format

Usage

```
OutsandingScoreToDistribution(outstanding_score, max_score)
```

Arguments

`outstanding_score`	: Outstanding_score outputed by MCMC algorithm to be converted to the Score-Distribution format
`max_score`	: max_score parameter used during the MCMC algorithm

Value

outstanding_score in a format compatible with MCMC algorithm output

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

 DistributionApproximationResults = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy)
OutsandingScoreToDistribution(DistributionApproximationResults$Outstanding_score, max_score = 100)
```

`plot_evolution` *Plot the evolution of the mean and the best value of the population used by the GeneticAlgorithm*

Description

Plot the evolution of the mean and the best value of the population used by the GeneticAlgorithm

Usage

```
plot_evolution(
  list,
  mean_color = "#F2A900",
  best_color = "#008080",
  xlab = "Epochs",
  ylab = "Score"
)
```

Arguments

`list`	A list with 2 elements returned by the GeneticAlgorithm: "mean" and "best", containing the numeric vectors representing the mean and best fitness of the population
`mean_color`	A string specifying the color of the mean values
`best_color`	A string specifying the color of the best values
`xlab`	A string specifying the label for the x-axis
`ylab`	A string specifying the label for the y-axis

Value

no returned value, should plot the evolution of the genetic algorithm results (mean/max score for each epoch).

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

results = GeneticAlgorithm(epochs = 10, nbIndividuals = 10,
           ATCtree = ATC_Tree_UpperBound_2024,
           observations = FAERS_myopathy)

plot_evolution(list = results)
```

plot_frequency *Plot the histogram of the approximation of the RR distribution*

Description

Plot the histogram of the approximation of the RR distribution

Usage

```
plot_frequency(
  estimated,
  sqrt = FALSE,
  binwidth = 0.1,
  hist_color = "#69b3a2",
  density_color = "#FF5733",
  xlab = "Score"
)
```

Arguments

`estimated`	The ScoreDistribution element in the list returned by the DistributionApproximation function
`sqrt`	A Boolean to specify whether we normalize the estimated or not, it is recommended on large random walk.
`binwidth`	The width of the histogram bins
`hist_color`	The fill color for the histogram bars
`density_color`	The color for the density curve
`xlab`	Label of X axis

Value

no returned value, should plot the histogram of the estimated distribution (estimated).

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

estimation = DistributionApproximation(epochs = 10, ATCtree = ATC_Tree_UpperBound_2024,
           observations = FAERS_myopathy)

plot_frequency(estimated = estimation$ScoreDistribution)
```

`print_csv`	*Print every cocktails found during the genetic algorithm when used with the hyperparam_test_genetic_algorithm function. This enables to condense the solutions found in each files by collapsing similar cocktail in a single row by cocktail.*

Description

Print every cocktails found during the genetic algorithm when used with the hyperparam_test_genetic_algorithm function. This enables to condense the solutions found in each files by collapsing similar cocktail in a single row by cocktail.

Usage

```
print_csv(
  input_filenames,
  observations,
  repetition,
  ATCtree,
  csv_filename = "solutions.csv"
)
```

Arguments

`input_filenames`	: A List containing filename of hyperparam_test_genetic_algorithm output file
`observations`	: observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
`repetition`	: The parameter nb_test_desired used in the hyperparam test function
`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`csv_filename`	: Name of the output file, "solutions.csv" by default

Value

No return value, should process the output of the genetic algorithm in files produced by hyperparam_test_genetic_algorithm and output a summary csv file. The csv file is outputed in current directory and named after the csv_filename variable (solutions.csv by default).

Examples

```
 data("ATC_Tree_UpperBound_2024")
 data("FAERS_myopathy")
files = c('250e_700ind_0.2mr_0ne_2alpha.txt') # results of hyperparam_test_genetic_algorithm

 print_csv(input_filenames = files, observations = FAERS_myopathy,
         repetition = 5, ATCtree = ATC_Tree_UpperBound_2024)
```

p_value_cocktails *Used to add the p_value to each cocktail of cocktail list*

Description

Used to add the p_value to each cocktail of cocktail list

Usage

```
p_value_cocktails(
  distribution_outputs,
  cocktails,
  ATCtree,
  observations,
  num_thread = 1L,
  filtred_distribution = FALSE
)
```

Arguments

distribution_outputs	A list of distribution of cocktails of different sizes in order to compute the p_value for multiple cocktail sizes
cocktails	A list containing cocktails in the form of vector of integers (ATC index)
ATCtree	ATC tree with upper bound of the DFS (without the root)
observations	observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
num_thread	Number of thread to run in parallel if openMP is available, 1 by default
filtred_distribution	Does the p-values have to be computed using filtered distribution or normal distribution (filtered distribution by default)

Value

A real valued number vector representing the p-value of the inputed cocktails computed on the distribution_outputs List.

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

 DistributionApproximationResults_size2 = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy, Smax = 2)

 DistributionApproximationResults_size3 = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy, Smax = 3)

 score_distribution_list = list(DistributionApproximationResults_size2,
                             DistributionApproximationResults_size3)

 cocktails = list(c(561, 904),
              c(1902, 4585))

 p_value_cocktails(score_distribution_list, cocktails, ATC_Tree_UpperBound_2024,
                   FAERS_myopathy)
```

p_value_csv_file — *Used to add the p_value to each cocktail of a csv_file that is an output of the genetic algorithm*

Description

Used to add the p_value to each cocktail of a csv_file that is an output of the genetic algorithm

Usage

```
p_value_csv_file(
  distribution_outputs,
  filename,
  filtred_distribution = FALSE,
  sep = ";"
)
```

Arguments

`distribution_outputs`	A list of distribution of cocktails of different sizes in order to compute the p_value for multiple cocktail sizes
`filename`	The file name of the .csv file containing the output
`filtred_distribution`	Does the p-values have to be computed using filtered distribution or normal distribution (filtered distribution by default)
`sep`	The separator used in the csv file (';' by default)

Value

A real valued number vector representing the p-value of the inputed csv file filename, computed on the distribution_outputs List.

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

 DistributionApproximationResults_size2 = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy, Smax = 2)

 DistributionApproximationResults_size3 = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy, Smax = 3)

 score_distribution_list = list(DistributionApproximationResults_size2,
                             DistributionApproximationResults_size3)
 p_value_csv_file(score_distribution_list, "path/to/output.csv")
```

`p_value_genetic_results`
Used to add the p_value to each cocktail of an output of the genetic algorithm

---

Description

Used to add the p_value to each cocktail of an output of the genetic algorithm

Usage

```
p_value_genetic_results(
  distribution_outputs,
  genetic_results,
  filtred_distribution = FALSE
)
```

Arguments

`distribution_outputs`	A list of distribution of cocktails of different sizes in order to compute the p_value for multiple cocktail sizes
`genetic_results`	outputs of the genetic algorithm
`filtred_distribution`	Does the p-values have to be computed using filtered distribution or normal distribution (filtered distribution by default)

Value

A real valued number vector representing the p-value of the inputed genetic algorithm results (genetic_results) computed on the distribution_outputs List.

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")
 DistributionApproximationResults_size2 = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy, Smax = 2)

 DistributionApproximationResults_size3 = DistributionApproximation(epochs = 10,
           ATCtree = ATC_Tree_UpperBound_2024, observations = FAERS_myopathy, Smax = 3)

 score_distribution_list = list(DistributionApproximationResults_size2,
                             DistributionApproximationResults_size3)
 genetic_results = GeneticAlgorithm(epochs = 10, nbIndividuals = 20,
           ATCtree = ATC_Tree_UpperBound_2024,
           observations = FAERS_myopathy)
 p_value_genetic_results(score_distribution_list, genetic_results)
```

`p_value_on_sampled` *Calculate p-value of sampled value*

Description

Calculate p-value of sampled value

Usage

```
p_value_on_sampled(
  empirical_distribution,
  sampled_values,
  isFiltered = FALSE,
  includeZeroValue = FALSE
)
```

Arguments

`empirical_distribution`	A numeric vector of values representing the empirical distribution (return value of DistributionAproximation function)
`sampled_values`	A scalar or a vector of real valued number representing the sampled value (score to be tested)
`isFiltered`	A boolean representing if we want to use the filtered distribution or the distribution as is (False by default)
`includeZeroValue`	A boolean that indicate if you want to take into account the null score (False by default)

Value

A numeric value representing the empirical p-value

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

cocktails = list(c(561, 904),
                 c(1902, 4585))

estimated_score_distribution = DistributionApproximation(epochs = 10,
                ATCtree = ATC_Tree_UpperBound_2024,
                observations = FAERS_myopathy)

Hypergeom_of_cocktails = compute_hypergeom_on_list(cocktails = cocktails,
                                ATCtree = ATC_Tree_UpperBound_2024,
                                observations = FAERS_myopathy)

p_value = p_value_on_sampled(empirical_distribution = estimated_score_distribution,
        sampled_values = Hypergeom_of_cocktails)
```

`qq_plot_output` *Make a Quantile-Quantile diagram from the output of the MCMC algorithm (DistributionAproximation) and the algorithm that exhaustively calculates the distribution*

Description

Make a Quantile-Quantile diagram from the output of the MCMC algorithm (DistributionAproximation) and the algorithm that exhaustively calculates the distribution

Usage

```
qq_plot_output(estimated, true, filtered = FALSE, color = "steelblue")
```

Arguments

`estimated`	Outputed object of DistributionApproximation function
`true`	Outputed object of either DistributionApproximation function or True distribution computation function
`filtered`	Make use of the classic distributuion estimation or of the filtred one (number of patient taking the cocktail > beta)
`color`	The color of the dashed line of the qq-plot

Value

no returned value, should plot the quantile-quantile plot of the estimated distribution (estimated) vs the true distribution (true).

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

estimated_score_distribution = DistributionApproximation(epochs = 10,
            ATCtree = ATC_Tree_UpperBound_2024,
            observations = FAERS_myopathy[1:100,], Smax =2)

true_score_distribution = trueDistributionSizeTwoCocktail(ATCtree = ATC_Tree_UpperBound_2024,
            observations = FAERS_myopathy[1:100,], beta = 4)

qq_plot_output(estimated = estimated_score_distribution,
                true = true_score_distribution)
```

---

remove_higher_cocktails

Filter out drug cocktails with high-level ATC classifications

---

Description

This function iterates through a collection of drug combinations (cocktails) and filters out those that have a ratio of "high-level" nodes (ATC codes with length <= 3) exceeding the specified threshold. This is useful for removing overly generic drug categories from results.

Usage

```
remove_higher_cocktails(
  solutions,
  ATC_name,
  ATC_length,
  find_last_occurence = TRUE,
  max_height_ratio = 0.5
)
```

Arguments

`solutions`	A `Rcpp::DataFrame` containing the results to filter. Must include columns: "score", "RR", "p_value", "n.patient.taking.C", "n.patient.taking.C.and.having.AE", and "Cocktail".
`ATC_name`	A vector of strings containing the ATC codes/names used for mapping.
`ATC_length`	An integer vector where each element represents the length (hierarchy level) of the corresponding ATC code in `ATC_name`.

`find_last_occurence`
: Logical. If `true` (default), the mapping logic will look for the last occurrence of a drug name in the reference list.

`max_height_ratio`
: A double (default 0.5) representing the maximum allowable proportion of high-level nodes (length <= 3) in a cocktail. Cocktails exceeding this ratio are removed.

Value

A `Rcpp::DataFrame` with the same columns as `solutions`, containing only the cocktails that met the `max_height_ratio` criteria.

`run_firth_regression` *Firth Penalized Logistic Regression for Drug Cocktails*

Description

This function prepares a specific "cocktail" (a set of drugs) and performs a Firth's penalized logistic regression to estimate the interaction effect between the drug present in the combination detected as "at risk".

Usage

```
run_firth_regression(
  cocktail,
  upper_bound,
  patient_data,
  adr_column = "patientADR"
)
```

Arguments

`cocktail`	An integer vector representing the ATC indices of drugs in the combination.
`upper_bound`	A list or vector defining the hierarchy/bounds (upper_bound column of the ATC_tree).
`patient_data`	A data frame containing patient-level data, including the ADR outcome.
`adr_column`	A string specifying the column name in `patient_data` used as the dependent variable (Y). Defaults to "patientADR".

Details

Firth's method is preferred here as it handles "separation" issues common in sparse clinical data (where a drug combination might perfectly predict an ADR).

Value

An object of class `logistf` containing the regression results, including coefficients, p-values, and confidence intervals.

Examples

```
## Not run:
# Example using indices for drugs 888, 659
results <- run_firth_regression(
  cocktail = c(888, 659),
  upper_bound = ATC_Tree_UpperBound_2024$upperBound,
  patient_data = FAERS_myopathy
)
summary(results)

## End(Not run)
```

---

string_list_to_int_cocktails
Function used to convert a string vector of drugs in form "drug1:drug2" to a vector of index of the ATC tree ex: c(ATC_index(drug1), ATC_index(drugs2))

Description

Function used to convert a string vector of drugs in form "drug1:drug2" to a vector of index of the ATC tree ex: c(ATC_index(drug1), ATC_index(drugs2))

Usage

```
string_list_to_int_cocktails(ATC_name, lines, last_element = FALSE)
```

Arguments

`ATC_name`	the ATC_name column of the ATC tree
`lines`	A string vector of drugs cocktail in the form "drug1:drug2:...:drug_n"
`last_element`	A boolean to indicate whether we are matching the drug to the first matching occurrence in the tree or the last one. Default is false

Value

An R List that can be used by other algorithms (e.g. clustering algorithm)

Examples

```
  data("ATC_Tree_UpperBound_2024")
  string_list = c('hmg coa reductase inhibitors:nervous system',
                  'metformin:prasugrel')
  string_list_to_int_cocktails(ATC_Tree_UpperBound_2024$Name,
                               string_list)
```

trueDistributionDrugs *The true distribution of the score among every single nodes of the ATC*

Description

The true distribution of the score among every single nodes of the ATC

Usage

```
trueDistributionDrugs(
  ATCtree,
  observations,
  beta,
  max_score = 1000L,
  nbResults = 100L,
  num_thread = 1L
)
```

Arguments

`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`observations`	: observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
`beta`	: minimum number of person taking the cocktails in order to consider it in the beta score distribution
`max_score`	: maximum number the score can take. Score greater than this one would be added to the distribution as the value max_score. Default is 1000
`nbResults`	: Number of returned solution (Cocktail with the best oberved score during the run), 100 by default
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default

Value

Return a List containing : - ScoreDistribution : the distribution of the score as an array with each cells representing the number of risks = (index-1)/ 10 - Filtered_score_distribution : Distribution containing score for cocktails taken by at least beta patients. - Outstanding_score : An array of the score greater than max_score, - Best_cocktails : the nbResults bests cocktails encountered during the run. - Best_cocktails_beta : the nbResults bests cocktails taken by at least beta patients encountered during the run. - Best_scores : Score corresponding to the Best_cocktails. - Best_scores_beta : Score corresponding to the Best_cocktails_beta.

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

size_1_score_distribution = trueDistributionDrugs(ATCtree = ATC_Tree_UpperBound_2024,
           observations = FAERS_myopathy[1:100,], beta = 4)
```

---

trueDistributionSizeTwoCocktail

The true distribution of the score among every size-two cocktails

---

Description

The true distribution of the score among every size-two cocktails

Usage

```
trueDistributionSizeTwoCocktail(
  ATCtree,
  observations,
  beta,
  max_score = 100L,
  nbResults = 100L,
  num_thread = 1L
)
```

Arguments

`ATCtree`	: ATC tree with upper bound of the DFS (without the root)
`observations`	: observation of the AE based on the medications of each patients (a DataFrame containing the medication on the first column and the ADR (boolean) on the second) on which we want to compute the risk distribution
`beta`	: minimum number of person taking the cocktails in order to consider it in the beta score distribution
`max_score`	: maximum number the score can take. Score greater than this one would be added to the distribution as the value max_score. Default is 1000
`nbResults`	: Number of returned solution (Cocktail with the best oberved score during the run), 100 by default
`num_thread`	: Number of thread to run in parallel if openMP is available, 1 by default

Value

Return a List containing : - ScoreDistribution : the distribution of the score as an array with each cells representing the number of risks = (index-1)/ 10 - Filtered_score_distribution : Distribution containing score for cocktails taken by at least beta patients. - Outstanding_score : An array of the score greater than max_score, - Best_cocktails : the nbResults bests cocktails encountered during the run. - Best_cocktails_beta : the nbResults bests cocktails taken by at least beta patients encountered during the run. - Best_scores : Score corresponding to the Best_cocktails. - Best_scores_beta : Score corresponding to the Best_cocktails_beta.

Examples

```
data("ATC_Tree_UpperBound_2024")
data("FAERS_myopathy")

size_2_score_distribution = trueDistributionSizeTwoCocktail(ATCtree = ATC_Tree_UpperBound_2024,
            observations = FAERS_myopathy[1:100,], beta = 4)
```

Index