Package 'CASMI'

July 21, 2025

Type Package

Title 'CASMI'-Based Functions

Version 2.0.0

Description Contains Coverage Adjusted Standardized Mutual Information ('CASMI')-based functions. 'CASMI' is a fundamental concept of a series of methods. For more information about 'CASMI' and 'CASMI'-related methods, please refer to the corresponding publications (e.g., a feature selection method, Shi, J., Zhang, J., & Ge, Y. (2019) <doi:10.3390/e21121179>, and a dataset quality measurement method, Shi, J., Zhang, J., & Ge, Y. (2019) <doi:10.1109/ICHI.2019.8904553>) or contact the package author for the latest updates.

Imports EntropyEstimation, entropy, stats

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Jingyi (Catherine) Shi [aut, cre, cph, ctb],
Shirli Arndt [aut],
Jialin Zhang [ctb]

Maintainer Jingyi (Catherine) Shi <jshi@math.msstate.edu>

Repository CRAN

Date/Publication 2025-02-13 22:50:03 UTC

Contents

AQI *AQI Index*

Description

A quantitative measure of dataset quality. The AQI Index score indicates the degree to which features are associated with the outcome in a dataset. (Synonyms of "feature" in this document: "independent variable," "factor," "predictor.")
For more information, please refer to the corresponding publication: Shi, J., Zhang, J. and Ge, Y. (2019), "An Association-Based Intrinsic Quality Index for Healthcare Dataset Ranking" <doi:10.1109/ICHI.2019.8904553>

Usage

```
AQI(data, alpha.ind = 0.2)
```

Arguments

data	data frame with variables as columns and observations as rows. The data must include at least one feature (a.k.a., independent variable, predictor, factor) and only one outcome variable (Y). The outcome variable MUST BE THE LAST COLUMN. Both the features and the outcome MUST be categorical or discrete. If variables are not naturally discrete, you may preprocess them using the 'autoBin.binary()' function in the same package.
alpha.ind	level of significance for the mutual information test of independence in step 2 (<doi:10.1109/ICHI.2019.8904553>). By default, 'alpha.ind = 0.2'.

Value

The AQI Index score.

Examples

```
## ---- Generate a toy dataset: "data" ----
n <- 10000
set.seed(1)
x1 <- rbinom(n, 3, 0.5) + 0.2
set.seed(2)
x2 <- rbinom(n, 2, 0.8) + 0.5
set.seed(3)
x3 <- rbinom(n, 5, 0.3)
set.seed(4)
error <- round(runif(n, min=-1, max=1))
y <- x1 + x3 + error
data <- data.frame(cbind(x1, x2, x3, y))
colnames(data) <- c("feature1", "feature2", "feature3", "Y")

## ---- Calculate the AQI score of "data" ----
AQI(data)
```

autoBin.binary *Automatically Dichotomize Quantitative Variables*

Description

Automatically compute optimal cutting points (based on mutual information) to dichotomize quantitative variables. This function can be used as a pre-processing step before using the **CASMI**-based functions.

Usage

```
autoBin.binary(data, index)
```

Arguments

data	data frame with variables as columns and observations as rows. The outcome variable (Y) MUST be categorical or discrete. The outcome variable (Y) MUST be the last column.
index	index or a vector of indices of the quantitative features (a.k.a., predictors, factors, independent variables) that need to be automatically categorized.

Value

'autoBin.binary()' returns the entire data frame after automatically dichotomizing the selected quantitative variable(s).

Examples

```
## Using the "iris" dataset embedded in R
data("iris")
head(iris) # The original data

# ---- Dichotomize One Single Feature ----
# Dichotomize the column with index 1.
newData1 <- autoBin.binary(iris, 1)
head(newData1)

# ---- Dichotomize Multiple Features at a Time ----
# Dichotomize the columns with indices 1, 2, 3, and 4.
newData2 <- autoBin.binary(iris, c(1,2,3,4))
head(newData2)

# ---- Dichotomize Features Using Column Names ----
# Dichotomize the columns with the names "Sepal.Length" and "Sepal.Width".
cols_of_interest <- c("Sepal.Length", "Sepal.Width")
col_indices <- which(names(iris) %in% cols_of_interest)
newData3 <- autoBin.binary(iris, col_indices)
head(newData3)
```

`CASMI.mineCombination` *Discover Factor Combinations based on* **CASMI**

Description

The 'CASMI.mineCombination()' function is designed to suggest combinations of factors that are most strongly associated with the outcome in a dataset. This function is partially developed based on the 'CASMI.selectFeatures()' function. (Synonyms for "factor" in this document: "independent variable," "feature," and "predictor.")

Usage

```
CASMI.mineCombination(
  data,
  NumOfVar = NULL,
  NA.handle = "stepwise",
  alpha = 0.05,
  alpha.ind = 0.1,
  intermediate.steps = FALSE,
  kappa.star.cap = 1,
  NumOfComb = 3
)
```

Arguments

`data`	data frame with variables as columns and observations as rows. The data MUST include at least one feature (a.k.a., independent variable, predictor, factor) and only one outcome variable (Y). The outcome variable MUST BE THE LAST COLUMN. Both the features and the outcome MUST be categorical or discrete. If variables are not naturally discrete, you may preprocess them using the 'autoBin.binary()' function in the same package.
`NumOfVar`	the number of variables in a combination (integer). This setting is optional. If NULL, an automatically suggested number of variables will be returned.
`NA.handle`	method for handling missing values. This parameter is inherited from the 'CASMI.selectFeature()' function. There are three possible options: 'NA.handle = "stepwise"' (default), 'NA.handle = "na.omit"', or 'NA.handle = "NA as a category"'. Check the 'CASMI.selectFeature()' documentation for more details.
`alpha`	level of significance used for the confidence intervals in the results; the default is 0.05.
`alpha.ind`	level of significance used for the initial screening of features based on a test of independence; the default is 0.1. This parameter is also used in the 'CASMI.selectFeature()' function; check the 'CASMI.selectFeature()' documentation for more details.
`intermediate.steps`	setting for outputting intermediate steps while awaiting the final results. There are two possible settings: 'intermediate.steps = TRUE' or 'intermediate.steps = FALSE'.

`kappa.star.cap` threshold of 'kappa*' for halting the feature selection process. This parameter is inherited from the 'CASMI.selectFeature()' function; check the 'CASMI.selectFeature()' documentation for more details. This setting is applicable only when 'NumOfVar' is set to NULL (default).

`NumOfComb` the number of top combinations to be returned; the default is 3. This setting is used only when a 'NumOfVar' value is defined (not NULL); if 'NumOfVar == NULL', only the automatically suggested combination will be returned.

Value

'CASMI.mineCombination()' returns the following components:

- `` `Outcome` ``: Name of the outcome variable (last column) in the input dataset.
- `` `Conf.Level` ``: Confidence level used for the results.
- `` `NumOfVar` ``: The number of variables in each combination.
- `` `TopResults` ``: A results data frame. The number of combinations (rows) returned depends on the 'NumOfComb' setting.
- `` `Comb.Idx` ``: Indices of the variables in the combination.
- `` `n` ``: Number of observations used in the analysis.
- `` `kappa*` ``: A comprehensive score reflecting the association between the factor combination and the outcome. A larger 'kappa*' indicates that the factor combination has a stronger association with the outcome. For more information about 'kappa*', please refer to the paper: Shi, J., Zhang, J. and Ge, Y. (2019), "**CASMI**—An Entropic Feature Selection Method in Turing's Perspective" <doi:10.3390/e21121179>
- `` `kappa*.low` ``: Lower bound of the confidence interval for 'kappa*'.
- `` `kappa*.upr` ``: Upper bound of the confidence interval for 'kappa*'.
- `` `SMIz` ``: Standardized Mutual Information (SMI) (using the z-estimator) between the factor combination and the outcomes.
- `` `SMIz.low` ``: Lower bound of the confidence interval for 'SMIz'.
- `` `SMIz.upr` ``: Upper bound of the confidence interval for 'SMIz'.
- `` `p.MIz` ``: P-value between the factor combination and the outcome using the mutual information test of independence based on the z-estimator.
- `` `Var.Name` ``: Names of the variables in the combination.

Examples

```
# ---- Generate a toy dataset for usage examples: "data" ----
set.seed(123)
n <- 200
x1 <- sample(c("A", "B", "C", "D"), size = n, replace = TRUE, prob = c(0.1, 0.2, 0.3, 0.4))
x2 <- sample(c("W", "X", "Y", "Z"), size = n, replace = TRUE, prob = c(0.4, 0.3, 0.2, 0.1))
x3 <- sample(c("E", "F", "G", "H", "I"), size = n,
             replace = TRUE, prob = c(0.2, 0.3, 0.2, 0.2, 0.1))
x4 <- sample(c("A", "B", "C", "D"), size = n, replace = TRUE)
x5 <- sample(c("L", "M", "N"), size = n, replace = TRUE)
x6 <- sample(c("E", "F", "G", "H", "I"), size = n, replace = TRUE)
```

```
# Generate y variable dependent on x1 to x3
x1_num <- as.numeric(factor(x1, levels = c("A", "B", "C", "D")))
x2_num <- as.numeric(factor(x2, levels = c("W", "X", "Y", "Z")))
x3_num <- as.numeric(factor(x3, levels = c("E", "F", "G", "H", "I")))
# Calculate y with added noise
y_numeric <- 3*x1_num + 2*x2_num - 2*x3_num + rnorm(n,mean=0,sd=2)
# Discretize y into categories
y <- cut(y_numeric, breaks = 10, labels = paste0("Category", 1:10))

# Combine into a dataframe
data <- data.frame(x1, x2, x3, x4, x5, x6, y)

# The outcome of the toy dataset is dependent on x1, x2, and x3
# but is independent of x4, x5, and x6.
head(data)


# ---- Usage Examples ----

## Return the suggested combination with the default settings:
CASMI.mineCombination(data)

## Return combinations when the number of variables to be included
## in each combination is specified (e.g., NumOfVar = 2):
CASMI.mineCombination(data, NumOfVar = 2)

## Return combinations when the number of variables to be included
## in each combination is specified (e.g., NumOfVar = 2),
## while the number of top combinations to return is specified
## (e.g., NumOfComb = 2):
CASMI.mineCombination(data,
                      NumOfVar = 2,
                      NumOfComb = 2)
```

`CASMI.selectFeatures` **CASMI** *Feature Selection*

Description

Selects features that are associated with an outcome while taking into account a sample coverage penalty and feature redundancy. It automatically determines the number of features to be selected, and the chosen features are ranked. (Synonyms for "feature" in this document: "independent variable," "factor," and "predictor.")
For additional information, please refer to the publication: Shi, J., Zhang, J. and Ge, Y. (2019), "**CASMI**—An Entropic Feature Selection Method in Turing's Perspective" <doi:10.3390/e21121179>

Usage

```
CASMI.selectFeatures(
  data,
  NA.handle = "stepwise",
  alpha = 0.05,
  alpha.ind = 0.1,
  intermediate.steps = FALSE,
  kappa.star.cap = 1,
  feature.num.cap = ncol(data)
)
```

Arguments

data	data frame with variables as columns and observations as rows. The data MUST include at least one feature (a.k.a., independent variable, predictor, factor) and only one outcome variable (Y). The outcome variable MUST BE THE LAST COLUMN. Both the features and the outcome MUST be categorical or discrete. If variables are not naturally discrete, you may preprocess them using the 'autoBin.binary()' function in the same package.
NA.handle	options for handling NA values in the data. There are three options: 'NA.handle = "stepwise"' (default), 'NA.handle = "na.omit"', and 'NA.handle = "NA as a category"'. (1) 'NA.handle = "stepwise"' excludes NA rows only when a particular variable is being used in a sub-step. For example, suppose we have data (Feature1: A, NA, B; Feature2: C, D, E; Feature3: F, G, H; Outcome: O, P, Q); the second observation will be excluded only when a particular step includes Feature1, but will not be excluded when a step is analyzing only Feature2, Feature3, and the Outcome. This option is designed to take advantage of the maximum possible number of observations. (2) 'NA.handle = "na.omit"' excludes observations with any NA values at the beginning of the analysis. (3) 'NA.handle = "NA as a category"' treats the NA value as a new category. This is designed to be used when NA values in the data have a consistent meaning instead of being missing values. For example, in survey data asking for comments, each NA value might consistently mean "no opinion."
alpha	level of significance for the confidence intervals in final results. By default, 'alpha = 0.05'.
alpha.ind	level of significance for the mutual information test of independence in step 1 of the features selection (for an initial screening). The smaller the 'alpha.ind', the fewer features are sent to step 2 (<doi:10.3390/e21121179>). By default, 'alpha.ind = 0.1'.
intermediate.steps	setting for outputting intermediate steps while awaiting the final results. There are two possible settings: 'intermediate.steps = TRUE' or 'intermediate.steps = FALSE'.
kappa.star.cap	a threshold of 'kappa*' for halting the feature selection process. The program will automatically terminate at the first feature whose cumulative 'kappa*' value exceeds the 'kappa.star.cap' threshold. By default, 'kappa.star.cap = 1.0', which

is the maximum possible value. A lower value may result in fewer final features but reduced computing time.

`feature.num.cap`
: the maximum number of features to be selected. A lower value may result in fewer final features but less computing time.

Value

'CASMI.selectFeatures()' returns the following components:

- `` `Outcome` ``: Name of the outcome variable (last column) in the input dataset.
- `` `Conf.Level` ``: Confidence level used for the results.
- `` `KappaStar` ``: The estimated 'kappa*' of all selected features. A larger 'kappa*' indicates that the selected features have a stronger association with the outcome.
- `` `KappaStarCI` ``: The confidence interval of 'kappa*' for all selected features.
- `` `Results` ``: A results data frame. The selected features are ranked.
- `` `Var.Idx` ``: Column index of the selected feature.
- `` `n` ``: Number of observations used in the analysis.
- `` `cml.kappa*` ``: The estimated cumulative 'kappa*' score when this particular feature was added to the list. That is, the 'kappa*' score of all currently selected features.
- `` `SMIz` ``: The Standardized Mutual Information (SMI) (using the z-estimator) between this particular feature and the outcome.
- `` `SMIz.Low` ``: Lower bound of the confidence interval for 'SMIz'.
- `` `SMIz.Upr` ``: Upper bound of the confidence interval for 'SMIz'.
- `` `p.MIz` ``: P-value between this particular feature and the outcome using the mutual information test of independence based on the z-estimator.
- `` `Var.Name` ``: Column name of the selected feature.

Examples

```
# ---- Generate a toy dataset for usage examples: "data" ----
set.seed(123)
n <- 200
x1 <- sample(c("A", "B", "C", "D"), size = n, replace = TRUE, prob = c(0.1, 0.2, 0.3, 0.4))
x2 <- sample(c("W", "X", "Y", "Z"), size = n, replace = TRUE, prob = c(0.4, 0.3, 0.2, 0.1))
x3 <- sample(c("E", "F", "G", "H", "I"), size = n,
             replace = TRUE, prob = c(0.2, 0.3, 0.2, 0.2, 0.1))
x4 <- sample(c("A", "B", "C", "D"), size = n, replace = TRUE)
x5 <- sample(c("L", "M", "N"), size = n, replace = TRUE)
x6 <- sample(c("E", "F", "G", "H", "I"), size = n, replace = TRUE)

# Generate y variable dependent on x1 to x3
x1_num <- as.numeric(factor(x1, levels = c("A", "B", "C", "D")))
x2_num <- as.numeric(factor(x2, levels = c("W", "X", "Y", "Z")))
x3_num <- as.numeric(factor(x3, levels = c("E", "F", "G", "H", "I")))
# Calculate y with added noise
y_numeric <- 3*x1_num + 2*x2_num - 2*x3_num + rnorm(n,mean=0,sd=2)
```

```
# Discretize y into categories
y <- cut(y_numeric, breaks = 10, labels = paste0("Category", 1:10))

# Combine into a dataframe
data <- data.frame(x1, x2, x3, x4, x5, x6, y)

# The outcome of the toy dataset is dependent on x1, x2, and x3
# but is independent of x4, x5, and x6.
head(data)


# ---- Usage Examples ----

## Select features and provide relevant results:
CASMI.selectFeatures(data)

## Adjust 'feature.num.cap' for including fewer features:
## (Note: A lower 'feature.num.cap' value may result in fewer
## final features but less computing time.)
CASMI.selectFeatures(data, feature.num.cap = 2)
```

Index