Package 'mcrs'

April 17, 2018

Type Package

Title mCRS Package (multiclass composite reference standard using simulation)

Version 0.1.0

Author Ty Stanford

Maintainer Ty Stanford <tyman@lbtinnovations.com>

Description Functions to perform mCRS for any ordinal outcome with 2 or more classes/categories.

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat

Imports dplyr,
tibble,
purrr,
binom

RoxygenNote 6.0.1

R topics documented:

brenton2018 *Data from Benton et al. (2018)*

Description

These are the data from from Benton et al. (2018) containing the APAS (A, the index test), St Vincent's standard workflow (S, imperfect truth) and the panel consensus (P, resolver) variables.

Usage

```
data(brenton2018)
```

Format

A data frame with 881 rows and 3 variables:

A APAS values 1, 2, 3 and 4 representing growth enumeration of 0, 10^6, 10^7 and 10^8+ CFU/L, respectively.

S St Vincent's standard workflow values 1, 2, 3 and 4 representing growth enumeration of 0, 10^6, 10^7 and 10^8+ CFU/L, respectively.

P Panel consensus values 1, 2, 3 and 4 representing growth enumeration of 0, 10^6, 10^7 and 10^8+ CFU/L, respectively. Column contains 600 NA values where the resolver values are not obtained.

col_check *Check a vector of column names are all contained in a suppled data frame*

Description

Check a vector of column names are all contained in a suppled data frame

Usage

```
col_check(df, cols)
```

Arguments

df	data frame
cols	character vector of column names to be checked whether they exist in df

Value

TRUE if all cols exist in df

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
adf <- data_frame(A = 1:2, B = 3:4, `C=1` = LETTERS[1:2])
col_check(adf, LETTERS[1:2])
col_check(adf, c("A", "B", "C=1"))
```

`get_c_ik` *Add 2D arrays of [resolver, index] observed counts and [resolver, index] simulated counts*

Description

Add 2D arrays of [resolver, index] observed counts and [resolver, index] simulated counts

Usage

```
get_c_ik(m_ik, m_st_ik)
```

Arguments

`m_ik`	a [resolver, index] 2D array of counts. Elements are $\sum_{j=1}^{K} m_{ijk}$ values.
`m_st_ik`	a [resolver, index] 2D array of counts. Elements are $\sum_{j=1}^{K} m^*_{ijk}$ values.

Value

a [resolver, index] 2D array of counts. Elements are c_{ik} values.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
ABC <- data_frame(
  A=sample(1:3, 20, replace=TRUE),
  B=sample(1:3, 20, replace=TRUE),
  C=sample(c(NA, 1:3), 20, replace=TRUE)
)
m <- get_m_ik(get_m_ijk(ABC, "A", "B", "C"))
p <- get_p_ijk(get_m_ijk(ABC, "A", "B", "C"))
s_st <- get_s_st_ij(ABC, "A", "B", "C")
m_st <- get_m_st_ik(p, s_st)
get_c_ik(m, m_st)
```

get_m_ijk *Calculate 3D array of [resolver, index, imperfect] counts*

Description

Calculate 3D array of [resolver, index, imperfect] counts from a dataset

Usage

```
get_m_ijk(dat, index, imperfect, resolver)
```

Arguments

dat	data frame that contains the index test, imperfect truth and resolver columns
index	string of column name in dat corresponding to the index test
imperfect	string of column name in dat corresponding to the imperfect truth
resolver	string of column name in dat corresponding to the resolver

Value

a [resolver, index, imperfect] 3D array of counts. Elements are m_{ijk} values. Rows where resolver are NA are removed.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
ABC <- data_frame(
  A=sample(1:3, 20, replace=TRUE),
  B=sample(1:3, 20, replace=TRUE),
  C=sample(c(NA, 1:3), 20, replace=TRUE)
)
get_m_ijk(ABC, "A", "B", "C")
```

get_m_ik *Get 2D array of [resolver, index] counts from 3D array of [resolver, index, imperfect] counts*

Description

Get 2D array of [resolver, index] counts from 3D array of [resolver, index, imperfect] counts

Usage

```
get_m_ik(m_ijk)
```

Arguments

m_ijk	a [resolver, index, imperfect] 3D array of counts

Value

a [resolver, index] 2D array of counts. Elements are $\sum_{j=1}^{K} m_{ijk}$ values.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
ABC <- data_frame(
  A=sample(1:3, 20, replace=TRUE),
  B=sample(1:3, 20, replace=TRUE),
  C=sample(c(NA, 1:3), 20, replace=TRUE)
)
get_m_ik(get_m_ijk(ABC, "A", "B", "C"))
```

---

get_m_st_ik *Simulate 2D array of [imperfect, index] counts*

Description

Simulate 2D array of [imperfect, index] counts

Usage

```
get_m_st_ik(p_ijk, s_st_ij)
```

Arguments

p_ijk	a [resolver, index, imperfect] 3D array of $p_{k\|ij}$=P(resolver=k | index=i, imperfect=j)
s_st_ij	a [imperfect, index] 2D array of counts where resolver value aren't observed (NAs).

Value

a [resolver, index] 2D array of simulated counts. Elements are $\sum_{j=1}^{K} m^*_{ijk}$ values.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
ABC <- data_frame(
  A=sample(1:3, 20, replace=TRUE),
  B=sample(1:3, 20, replace=TRUE),
  C=sample(c(NA, 1:3), 20, replace=TRUE)
)
get_s_st_ij(ABC, "A", "B", "C")
get_p_ijk(get_m_ijk(ABC, "A", "B", "C"))
get_m_st_ik(get_p_ijk(get_m_ijk(ABC, "A", "B", "C")), get_s_st_ij(ABC, "A", "B", "C"))
```

`get_p_ijk` *Calculate 3D array of $p\_k|ij$ values*

Description

Calculate 3D array of $p_{k|ij}$ values.

Usage

```
get_p_ijk(m_ijk)
```

Arguments

`m_ijk`	a [resolver, index, imperfect] 3D array of m_{ijk} values

Value

a [resolver, index, imperfect] 3D array of $p_{k|ij}$=P(resolver=k | index=i, imperfect=j).

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
ABC <- data_frame(
  A=sample(1:3, 20, replace=TRUE),
  B=sample(1:3, 20, replace=TRUE),
  C=sample(c(NA, 1:3), 20, replace=TRUE)
)
get_m_ijk(ABC, "A", "B", "C")
get_p_ijk(get_m_ijk(ABC, "A", "B", "C"))
```

get_sens_spec *Calculate sensitivity and specificity*

Description

Calculate sensitivity and specificity from a confusion matrix

Usage

```
get_sens_spec(tab, pos, alpha = 0.05)
```

Arguments

tab	a confusion matrix (object of class "matrix" or table")
pos	the levels of the margins that are considered "positive"
alpha	confidence level of CIs (default=0.05)

Value

A tibble with columns param ("sens" or "spec"), cases, correct, est, lo, up.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
AB <- data_frame(A=sample(1:3, 20, replace=TRUE), B=sample(1:3, 20, replace=TRUE))
get_sens_spec(with(AB,table(A,B)), pos = 2:3)
```

get_s_st_ij *Calculate 2D array of [imperfect, index] counts where resolver is* NA

Description

Calculate 2D array of [imperfect, index] counts where resolver is NA from dataset

Usage

```
get_s_st_ij(dat, index, imperfect, resolver)
```

Arguments

dat	data frame that contains the index test, imperfect truth and resolver columns
index	string of column name in dat corresponding to the index test
imperfect	string of column name in dat corresponding to the imperfect truth
resolver	string of column name in dat corresponding to the resolver

Value

a [imperfect, index] 2D array of counts. Elements are s^*_{ij} values. Only rows where resolver are NA are used.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
ABC <- data_frame(
  A=sample(1:3, 20, replace=TRUE),
  B=sample(1:3, 20, replace=TRUE),
  C=sample(c(NA, 1:3), 20, replace=TRUE)
)
get_s_st_ij(ABC, "A", "B", "C")
```

---

perform_mcrs *Perform mCRS using a one sample-per-row input dataset*

Description

Performs the mCRS analysis

Usage

```
perform_mcrs(dat, index, imperfect, resolver, pos, r = 1000,
  seed = 12345678, alpha = 0.05, summ = TRUE)
```

Arguments

dat	data frame that contains the index test, imperfect truth and resolver columns (see details)
index	string of column name in dat corresponding to the index test
imperfect	string of column name in dat corresponding to the imperfect truth
resolver	string of column name in dat corresponding to the resolver
pos	the levels of index/imperfect/resolver that are considered "positive"
r	the number of random simulations of m*_ik to generate (default=1000)
seed	for reporducibility of results (default=12345678)
alpha	confidence level (default=0.05)
summ	the full r results are returned when FALSE or just the summarised values (default, TRUE)

Details

Currently perform_mcrs requires dat to contain variables with levels that are consectutive integers starting at 1 (i.e. here the outcome levels are labelled 1, 2, 3, 4).

Value

A tibble with columns `param`, `cases`, `correct`, `est`, `lo`, `up`. The number of rows is determined by the value of `summ`

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
data(brenton2018)
# the sens+spec estimates using the imperfect truth
get_sens_spec(table(brenton2018[["S"]], brenton2018[["A"]]), 3:4)
# now check against mCRS (allow ~0.02 sec per repetition)
perform_mcrs(brenton2018, "A", "S", "P", pos = 3:4, r = 10)
```

`rep_as_list` *Create a list of an object repeated* `x` *times*

Description

Create a list with length `n` with the object `x` in each elelment of the list

Usage

```
rep_as_list(x, n)
```

Arguments

`x`	an object
`n`	number of repeated elements

Value

a list with length `n` with the object `x` in each elelment of the list

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
A <- matrix(c(1,2,1,0,0,0,6,3,3), nrow=3)
rep_as_list(A, 3)
```

sim_mstar *Calculate a simulated [resolver, index, imperfect] 3D array*

Description

Calculate a simulated [resolver, index, imperfect] 3D array

Usage

```
sim_mstar(p_ijk, s_st_ij)
```

Arguments

p_ijk	a [resolver, index, imperfect] 3D array of probabilities. Elements are $p_{k\|ij} = P(resolver\|index, imperfect)$.
s_st_ij	a [index, imperfect] 2D array counts requiring a simulated resolver value.

Value

a [resolver, index, imperfect] 3D array of counts. Elements are $m^*_{k|ij}$ values.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
library(tibble)
set.seed(1234)
AB <- data_frame(A=sample(1:3, 20, replace=TRUE), B=sample(1:3, 20, replace=TRUE))
get_sens_spec(with(AB,table(A,B)), pos = 2:3)
```

two_dim_col_wise_prop *Calculate column-wise proportions of counts of a matrix*

Description

Calculate column-wise proportions of counts of a matrix (2D array)

Usage

```
two_dim_col_wise_prop(x)
```

Arguments

x	a square matrix (or object of class "table")

Value

an object of the same dim as x. The elements are the original elements divided by the column sum. However, if the column sum is 0, the column is replaced by the corresponding column in the equivalent size identity matrix.

Author(s)

Ty Stanford <tyman@lbtinnovations.com>

Examples

```
A <- matrix(c(1,2,1,0,0,0,6,3,3), nrow=3)
two_dim_col_wise_prop(A)
```

Index