Package 'emery'

May 8, 2026

Title Accuracy Statistic Estimation for Imperfect Gold Standards

Version 0.7.1

Description Produce maximum likelihood estimates of common accuracy statistics for multiple measurement methods when a gold standard is not available. An R implementation of the expectation maximization algorithms described in Zhou et al. (2011) <doi:10.1002/9780470906514> with additional functions for creating simulated data and visualizing results. Supports binary, ordinal, and continuous measurement methods.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

URL https://github.com/therealcfdrake/emery

BugReports https://github.com/therealcfdrake/emery/issues

Imports stats, dplyr, ggplot2, purrr, tibble, tidyr, utils, methods, mvtnorm, stringr, Rdpack

RdMacros Rdpack

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Corie Drake [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1517-7103>)

Maintainer Corie Drake <therealcfdrake@gmail.com>

Depends R (>= 4.2.0)

Repository CRAN

Date/Publication 2026-02-25 11:50:21 UTC

Contents

`aggregate_boot_ML` *Aggregate bootstrapped ML estimates*

Description

`aggregate_boot_ML()` rearranges the bootstrap results from `boot_ML()` by statistic instead of bootstrap iteration.

Usage

```
aggregate_boot_ML(boot_ML_result = NULL)
```

Arguments

`boot_ML_result`	a list returned by `bootML()`.

Value

a named list of long format data frames containing aggregated statistic estimates.

boot_id	index of bootstrap sample which resulted in value
col_id	value identifier
row_id	optional value identifier used when the result has more than 1 dimension
value	statistic value

Examples

```
# Set seed for this example
set.seed(11001101)

# Generate data for 4 binary methods
my_sim <- generate_multimethod_data(
  "binary",
  n_obs = 75,
  n_method = 4,
  se = c(0.87, 0.92, 0.79, 0.95),
  sp = c(0.85, 0.93, 0.94, 0.80),
  method_names = c("alpha", "beta", "gamma", "delta"))

# Bootstrap ML results
boot_ex <- boot_ML(
  "binary",
  data = my_sim$generated_data,
  n_boot = 20)

# Aggregate Bootstrap ML results by statistic
aggregate_boot_ML(boot_ex)

# Plot Bootstrap ML estimate distributions
plot(boot_ex)
```

---

bin_auc *Calculate AUC for single Se/Sp pair*

Description

Calculate AUC

Usage

```
bin_auc(se, sp)
```

Arguments

se	Sensitivity
sp	Specificity

Value

Area under ROC curve

`boot_ML` *Bootstrap ML accuracy statistic estimation for multi-method data*

Description

`boot_ML()` is a function used to generate bootstrap estimates of results generated by `estimate_ML()` primarily for use in creating nonparametric confidence intervals.

Usage

```
boot_ML(
  type = c("binary", "ordinal", "continuous"),
  data,
  freqs = NULL,
  n_boot = 100,
  n_study = NULL,
  randomize_init = FALSE,
  max_iter = 1000,
  tol = 1e-07,
  seed = NULL,
  ...
)
```

Arguments

`type`	A string specifying the data type of the methods under evaluation.
`data`	An `n_obs` by `n_method` `matrix` containing the observed values for each method. If the dimensions are named, row names will be used to name each observation (`obs_names`) and column names will be used to name each measurement method (`method_names`).
`freqs`	A vector of elements `n_obs` long representing the number of times each row in `data` was observed. Used when `data` is a summary of unique response combinations.
`n_boot`	number of bootstrap estimates to compute
`n_study`	sample size to select for each bootstrap estimate
`randomize_init`	boolean defining whether `init` values should be randomize with each bootstrap.
`max_iter`	The maximum number of EM algorithm iterations to compute before reporting a result.
`tol`	The minimum change in statistic estimates needed to continue iterating the EM algorithm.
`seed`	optional seed for RNG
`...`	Additional arguments

Value

a list containing accuracy estimates, v, and the parameters used.

v_0	result from original data
v_star	list containing results from each bootstrap resampling
params	list containing the parameters used

Examples

```
# Set seed for this example
set.seed(11001101)

# Generate data for 4 binary methods
my_sim <- generate_multimethod_data(
  "binary",
  n_obs = 75,
  n_method = 4,
  se = c(0.87, 0.92, 0.79, 0.95),
  sp = c(0.85, 0.93, 0.94, 0.80),
  method_names = c("alpha", "beta", "gamma", "delta"))

# Bootstrap ML results
boot_ex <- boot_ML(
  "binary",
  data = my_sim$generated_data,
  n_boot = 20)

# Aggregate Bootstrap ML results by statistic
aggregate_boot_ML(boot_ex)

# Plot Bootstrap ML estimate distributions
plot(boot_ex)
```

---

censor_data *Censor data randomly rowwise*

Description

Censor data randomly rowwise

Usage

```
censor_data(
  n_obs = dis$n_obs,
  first_reads_all = first_reads_all,
  n_method_subset = n_method_subset,
  n_method = n_method
)
```

Arguments

n_obs	An integer representing the number of observations to simulate.
first_reads_all	Used for binary methods. A logical which forces method 1 to have a result for every observation
n_method_subset	Used for binary methods. An integer defining how many methods to select at random to produce a result for each observation
n_method	An integer representing the number of methods to simulate.

---

define_disease_state *Define the True disease state of a simulated sample*

Description

Define the True disease state of a simulated sample

Usage

```
define_disease_state(D = NULL, n_obs = NULL, prev = NULL)
```

Arguments

D	Optional binary vector representing the true classification of each observation.
n_obs	An integer representing the number of observations to simulate.
prev	A value between 0-1 which represents the proportion of "positive" results in the target population.

Value

A list of features defining the true disease status of each observation

---

dmvnorm *Multivariate Normal Densities*

Description

Return the density of a point in a multivariate normal distribution

Usage

```
dmvnorm(x, mu, sigma)
```

Arguments

x	matrix of observations
mu	vector of method means
sigma	method covariance matrix

`estimate_ML` *Estimate maximum likelihood accuracy statistics by expectation maximization*

Description

`estimate_ML()` is a general function for estimating the maximum likelihood accuracy statistics for a set of methods with no known reference value, i.e. "truth", or "gold standard".

Usage

```
estimate_ML(
  type = c("binary", "ordinal", "continuous"),
  data,
  freqs = NULL,
  init = list(NULL),
  max_iter = 1000,
  tol = 1e-07,
  save_progress = TRUE,
  ...
)

estimate_ML_binary(
  data,
  freqs = NULL,
  init = list(prev_1 = NULL, se_1 = NULL, sp_1 = NULL),
  max_iter = 1000,
  tol = 1e-07,
  save_progress = TRUE
)

estimate_ML_continuous(
  data,
  freqs = NULL,
  init = list(prev_1 = NULL, mu_i1_1 = NULL, sigma_i1_1 = NULL, mu_i0_1 = NULL,
    sigma_i0_1 = NULL),
  max_iter = 100,
  tol = 1e-07,
  save_progress = TRUE
)
```

```
estimate_ML_ordinal(
  data,
  freqs = NULL,
  init = list(pi_1_1 = NULL, phi_1ij_1 = NULL, phi_0ij_1 = NULL, n_level = NULL),
  level_names = NULL,
  max_iter = 1000,
  tol = 1e-07,
  save_progress = TRUE
)
```

Arguments

type	A string specifying the data type of the methods under evaluation.
data	An n_obs by n_method matrix containing the observed values for each method. If the dimensions are named, row names will be used to name each observation (obs_names) and column names will be used to name each measurement method (method_names).
freqs	A vector of elements n_obs long representing the number of times each row in data was observed. Used when data is a summary of unique response combinations.
init	An optional list of initial values used to seed the EM algorithm. If initial values are not provided, the pollinate_ML() function will be called on the data to estimate starting values. It is recommended to try several sets of starting parameters to ensure that the algorithm converges to the same results. This is to verify that the result does not represent a local extrema.
max_iter	The maximum number of EM algorithm iterations to compute before reporting a result.
tol	The minimum change in statistic estimates needed to continue iterating the EM algorithm.
save_progress	A logical indication of whether to save interim calculations used in the EM algorithm.
...	Additional arguments
level_names	An optional, ordered, character vector of unique names corresponding to the levels of the methods.

Details

The lack of an infallible reference method is referred to as an imperfect gold standard (GS). Accuracy statistics which rely on a GS method, such as sensitivity, specificity, and AUC, can be estimated using imperfect gold standards by iteratively estimating the maximum likelihood values of these statistics while the conditional independence assumption holds. estimate_ML() relies on a collection of expectation maximization (EM) algorithms to achieve this. The EM algorithms used in this function are based on those presented in *Statistical Methods in Diagnostic Medicine, Second Edition* (Zhou et al. 2011) and have been validated on several examples therein. Additional details about these algorithms can be found for binary (Walter and Irwig 1988), ordinal (Zhou et al. 2005), and continuous (Hsieh et al. 2009) methods. Minor changes to the literal calculations have been made for efficiency, code readability, and the like, but the underlying steps remain functionally unchanged.

Value

`estimate_ML()` returns an S4 object of class "MultiMethodMLEstimate" containing the maximum likelihood accuracy statistics calculated by EM.

References

Zhou X, Obuchowski NA, McClish DK (2011). *Statistical Methods in Diagnostic Medicine*. Wiley. doi:10.1002/9780470906514.

Walter SD, Irwig LM (1988). "Estimation of test error rates, disease prevalence and relative risk from misclassified data: a review." *J. Clin. Epidemiol.*, **41**(9), 923–937. doi:10.1016/0895-4356(88)901102.

Zhou X, Castelluccio P, Zhou C (2005). "Nonparametric estimation of ROC curves in the absence of a gold standard." *Biometrics*, **61**(2), 600–609. doi:10.1111/j.15410420.2005.00324.x.

Hsieh H, Su H, Zhou X (2009). "Interval Estimation for the Difference in Paired Areas under the ROC Curves in the Absense of a Gold Standard Test." *Statistics in Medicine*. https://doi.org/10.1002/sim.3661.

Examples

```
# Set seed for this example
set.seed(11001101)

# Generate data for 4 binary methods
my_sim <- generate_multimethod_data(
  "binary",
  n_obs = 75,
  n_method = 4,
  se = c(0.87, 0.92, 0.79, 0.95),
  sp = c(0.85, 0.93, 0.94, 0.80),
  method_names = c("alpha", "beta", "gamma", "delta"))

# View the data
my_sim$generated_data

# View the parameters used to generate the data
my_sim$params

# Estimate ML accuracy values by EM algorithm
my_result <- estimate_ML(
  "binary",
  data = my_sim$generated_data,
  save_progress = FALSE # this reduces the data stored in the resulting object
)

# View results of ML estimate
my_result@results
```

`generate_multimethod_data` *Create data sets which simulate paired measurements of multiple methods*

Description

`generate_multimethod_data()` is a general function for creating a data set which simulates the results one might see when using several different methods to measure a set of objects.

Usage

```
generate_multimethod_data(
  type = c("binary", "ordinal", "continuous"),
  n_method = 3,
  n_obs = 100,
  prev = 0.5,
  D = NULL,
  method_names = NULL,
  obs_names = NULL,
  ...
)

generate_multimethod_binary(
  n_method = 3,
  n_obs = 100,
  prev = 0.5,
  D = NULL,
  se = rep(0.9, n_method),
  sp = rep(0.9, n_method),
  method_names = NULL,
  obs_names = NULL,
  n_method_subset = n_method,
  first_reads_all = FALSE
)

generate_multimethod_ordinal(
  n_method = 3,
  n_obs = 100,
  prev = 0.5,
  D = NULL,
  n_level = 5,
  pmf_pos = matrix(rep(1:n_level - 1, n_method), nrow = n_method, byrow = TRUE),
  pmf_neg = matrix(rep(n_level:1 - 1, n_method), nrow = n_method, byrow = TRUE),
  method_names = NULL,
  level_names = NULL,
  obs_names = NULL,
```

```
  n_method_subset = n_method,
  first_reads_all = FALSE
)

generate_multimethod_continuous(
  n_method = 2,
  n_obs = 100,
  prev = 0.5,
  D = NULL,
  mu_i1 = rep(12, n_method),
  sigma_i1 = diag(n_method),
  mu_i0 = rep(10, n_method),
  sigma_i0 = diag(n_method),
  method_names = NULL,
  obs_names = NULL,
  n_method_subset = n_method,
  first_reads_all = FALSE
)
```

Arguments

type	A string specifying the data type of the methods being simulated.
n_method	An integer representing the number of methods to simulate.
n_obs	An integer representing the number of observations to simulate.
prev	A value between 0-1 which represents the proportion of "positive" results in the target population.
D	Optional binary vector representing the true classification of each observation.
method_names	Optional vector of names used to identify each method.
obs_names	Optional vector of names used to identify each observation.
...	Additional parameters
se, sp	Used for binary methods. A vector of length n_method of values between 0-1 representing the sensitivity and specificity of the methods.
n_method_subset	Used for binary methods. An integer defining how many methods to select at random to produce a result for each observation
first_reads_all	Used for binary methods. A logical which forces method 1 to have a result for every observation
n_level	Used for ordinal methods. An integer representing the number of ordinal levels each method has
pmf_pos, pmf_neg	Used for ordinal methods. A n_method by n_level matrix representing the probability mass functions for positive and negative results, respectively
level_names	Used for ordinal methods. Optional vector of names used to identify each level

`mu_i1`, `mu_i0` Used for continuous methods. Vectors of length n_method of the method mean values for positive (negative) observations

`sigma_i1`, `sigma_i0` Used for continuous methods. Covariance matrices of method positive (negative) observations

Details

The function supports binary measurement methods, e.g., Pass/Fail; ordinal measurement methods, e.g., the Likert scale; and continuous measurement methods, e.g., height. The data are generated under the assumption that the underlying population consists of a mixture of two groups. The primary application of this is to simulate a sample from a population which has some prevalence of disease.

Value

A list containing a simulated data set and the parameters used to create it

Examples

```
# Set seed for this example
set.seed(11001101)

# Generate data for 4 binary methods
my_sim <- generate_multimethod_data(
  "binary",
  n_obs = 75,
  n_method = 4,
  se = c(0.87, 0.92, 0.79, 0.95),
  sp = c(0.85, 0.93, 0.94, 0.80),
  method_names = c("alpha", "beta", "gamma", "delta"))

# View the data
my_sim$generated_data

# View the parameters used to generate the data
my_sim$params

# Estimate ML accuracy values by EM algorithm
my_result <- estimate_ML(
  "binary",
  data = my_sim$generated_data,
  save_progress = FALSE # this reduces the data stored in the resulting object
)

# View results of ML estimate
my_result@results
```

getNames *Return names from MultiMethodMLEstimate object*

Description

Returns the names slot from a MultiMethodMLEstimate object

Usage

```
getNames(x, name)
```

Arguments

x	An object of class MultiMethodMLEstimate.
name	type of names to extract, e.g. "method_names", "obs_names", "level_names"

Value

Contents of names slot of the MultiMethodMLEstimate object.

getNames,MultiMethodMLEstimate-method *Return names from MultiMethodMLEstimate object*

Description

Returns the names slot from a MultiMethodMLEstimate object

Usage

```
## S4 method for signature 'MultiMethodMLEstimate'
getNames(x, name)
```

Arguments

x	An object of class MultiMethodMLEstimate.
name	type of names to extract, e.g. "method_names", "obs_names", "level_names"

Value

Contents of names slot of the MultiMethodMLEstimate object.

`getResults` *Return result from MultiMethodMLEstimate object*

Description

Returns the result slot from a MultiMethodMLEstimate object

Usage

```
getResults(x)
```

Arguments

`x`	An object of class MultiMethodMLEstimate.

Value

Contents of `results` slot of the MultiMethodMLEstimate object.

`getResults,MultiMethodMLEstimate-method`
Return result from MultiMethodMLEstimate object

Description

Returns the result slot from a MultiMethodMLEstimate object

Usage

```
## S4 method for signature 'MultiMethodMLEstimate'
getResults(x)
```

Arguments

`x`	An object of class MultiMethodMLEstimate.

Value

Contents of `results` slot of the MultiMethodMLEstimate object.

MultiMethodMLEstimate-class *S4 object containing the results of multi-method ML accuracy estimates*

Description

S4 object containing the results of multi-method ML accuracy estimates

Slots

`results` a list of estimated accuracy statistics

`data` a matrix containing the raw data used for estimation

`freqs` a vector containing the frequencies at which each row in `data` was observed

`names` a list containing vectors of names of various dimensions

`data` a copy of the data used to generate the estimated values

`iter` an integer number of iterations needed for the EM algorithm to converge

`prog` a list containing the values calculated during each iteration of the EM algorithm

`type` a string describing the data type

name_thing *Create unique names for a set of things*

Description

Create unique names for a set of things

Usage

```
name_thing(thing = "", n = 1)
```

Arguments

`thing`	a string that describes the set of items to name
`n`	an integer number of unique names to create

Value

a vector of unique names

`new_boot_ML` *Create new boot_ML class object*

Description

Wrapper for creating boot_ML class object.

Usage

```
new_boot_ML(
  v_0,
  v_star,
  data,
  freqs = NULL,
  n_boot,
  n_study,
  max_iter,
  tol,
  n_obs,
  seed
)
```

Arguments

`v_0`	MultiMethodMLEstimate S4 object
`v_star`	results slot of bootstrapped MultiMethodMLEstimate objects
`data`	An `n_obs` by `n_method` `matrix` containing the observed values for each method. If the dimensions are named, row names will be used to name each observation (`obs_names`) and column names will be used to name each measurement method (`method_names`).
`freqs`	A vector of elements `n_obs` long representing the number of times each row in `data` was observed. Used when `data` is a summary of unique response combinations.
`n_boot`	number of bootstrap estimates to compute
`n_study`	sample size to select for each bootstrap estimate
`max_iter`	The maximum number of EM algorithm iterations to compute before reporting a result.
`tol`	The minimum change in statistic estimates needed to continue iterating the EM algorithm.
`n_obs`	Number of observations in data
`seed`	optional seed for RNG

Value

a boot_ML object

plot,MultiMethodMLEstimate-method *Create plots from a MultiMethodMLEstimate object*

Description

Create a list of plots visualizing the expectation maximization process and resulting accuracy statistics stored in a MultiMethodMLEstimate object.

Usage

```
## S4 method for signature 'MultiMethodMLEstimate'
plot(x, y, ...)
```

Arguments

x	a MultiMethodMLEstimate S4 object
y	not used
...	Arguments passed on to plot_ML params A list of population parameters. This is primarily used to evaluate results from a simulation where the target parameters are known, but can be used to visualize results with respect to some True value.

Value

A list of ggplot2 plots

plot.boot_ML *Plot univariate distributions of bootstrapped ML estimates*

Description

plot.boot_ML() creates univariate plots of bootstrap results from boot_ML().

Usage

```
## S3 method for class 'boot_ML'
plot(x, probs = c(0.1, 0.5, 0.9), ...)
```

Arguments

x	a result created by calling boot_ML on a MultiMethodMLEstimate object.
probs	a vector of distribution quantile values to indicate with vertical lines.
...	additional arguments.

Value

a named list of named plots.

`plot_ML` *Create plots visualizing the ML estimation process and results.*

Description

`plot_ML()` is a general function for visualizing results generated by `estimate_ML()`.

Usage

```
plot_ML(ML_est, params = NULL)

plot_ML_binary(
  ML_est,
  params = list(prev = NULL, se = NULL, sp = NULL, D = NULL)
)

plot_ML_ordinal(
  ML_est,
  params = list(pi_1_1 = NULL, phi_1ij_1 = NULL, phi_0ij_1 = NULL, D = NULL)
)

plot_ML_continuous(
  ML_est,
  params = list(prev_1 = NULL, mu_i1_1 = NULL, sigma_i1_1 = NULL, mu_i0_1 = NULL,
    sigma_i0_1 = NULL, D = NULL)
)
```

Arguments

`ML_est`	A MultiMethodMLEstimate class object
`params`	A list of population parameters. This is primarily used to evaluate results from a simulation where the target parameters are known, but can be used to visualize results with respect to some True value.

Value

A list of ggplot2 plots.

Binary:

`prev`	A plot showing how the prevalence estimate changes with each iteration of the EM algorithm
`se`	A plot showing how the sensitivity estimates of each method change with each iteration of the EM algorithm

sp	A plot showing how the specificity estimates of each method change with each iteration of the EM algorithm
qk	A plot showing how the q values for each observation k change over each iteration of the EM algorithm
qk_hist	A histogram of q values. Observations, k, can be colored by True state if it is passed by params$D.
se_sp	A plot showing the path the sensitivity and specificity estimates for each method follows during the EM algorithm. True sensitivity and specificity values can be passed by params$se and params$sp, respectively. This is useful for comparing algorithm results when applied to simulation data where True parameter values are known.

Ordinal:

ROC	The Receiver Operator Characteristic (ROC) curves estimated for each method
q_k1	A plot showing how the q values for each observation, k, change when d=1 over each iteration of the EM algorithm. Observations can be colored by True state if it is passed (params$D).
q_k0	A plot showing how the q values for each observation, k, change when d=0 over each iteration of the EM algorithm. Observations can be colored by True state if it is passed by params$D.
q_k1_hist	A histogram of q_1 values. Observations, k, can be colored by True state if it is passed by params$D.
phi_d	A stacked bar graph representing the estimated CMFs of each method when d=0 and d=1.

Continuous:

ROC	The Receiver Operator Characteristic (ROC) curves estimated for each method
z_k1	A plot showing how the z_k1 values for each observation change over each iteration of the EM algorithm. Observations can be colored by True state if it is passed (params$D).
z_k0	A plot showing how the z_k0 values for each observation change over each iteration of the EM algorithm. Observations can be colored by True state if it is passed (params$D).
z_k1_hist	A histogram of z_k1 values. Observations can be colored by True state if it is passed (params$D).

Examples

```
# Set seed for this example
set.seed(11001101)

# Generate data for 4 binary methods
my_sim <- generate_multimethod_data(
  "binary",
  n_obs = 75,
  n_method = 4,
```

```
  se = c(0.87, 0.92, 0.79, 0.95),
  sp = c(0.85, 0.93, 0.94, 0.80),
  method_names = c("alpha", "beta", "gamma", "delta"))

# View the data
my_sim$generated_data

# View the parameters used to generate the data
my_sim$params

# Estimate ML accuracy values by EM algorithm
my_result <- estimate_ML(
  "binary",
  data = my_sim$generated_data,
  save_progress = FALSE # this reduces the data stored in the resulting object
)

# View results of ML estimate
my_result@results
```

`pollinate_ML` *Generate seed values for EM algorithm*

Description

`pollinate_ML()` is a general helper function which can be used to generate starting values, i.e. seeds, for the `estimate_ML()` function from a multi-method data set.

Usage

```
pollinate_ML(
  type = c("binary", "ordinal", "continuous"),
  data,
  freqs = NULL,
  ...
)

pollinate_ML_binary(data, freqs = NULL, ...)

pollinate_ML_ordinal(
  data,
  freqs = NULL,
  n_level = NULL,
  threshold_level = ceiling(n_level/2),
  level_names = NULL,
  ...
)
```

```
pollinate_ML_continuous(
  data,
  freqs = NULL,
  prev = 0.5,
  q_seeds = c((1 - prev)/2, 1 - (prev/2)),
  high_pos = TRUE,
  ...
)
```

Arguments

type	A string specifying the data type of the methods under evaluation.
data	An n_obs by n_method matrix containing the observed values for each method. If the dimensions are named, row names will be used to name each observation (obs_names) and column names will be used to name each measurement method (method_names).
freqs	A vector of elements n_obs long representing the number of times each row in data was observed. Used when data is a summary of unique response combinations.
...	Additional arguments
n_level	Used for ordinal methods. Integer number of levels each method contains
threshold_level	Used for ordinal methods. A value from 1 to n_level which indicates the initial threshold used to define positive and negative disease states.
level_names	Used for ordinal methods. Optional vector of length n_level containing names for each level.
prev	A double between 0-1 representing the proportion of positives in the population
q_seeds	Used for continuous methods. A vector of length 2 representing the quantiles at which the two groups are assumed to be centered
high_pos	Used for continuous methods. A logical indicating whether larger values are considered "positive"

Value

a list of EM algorithm initialization values

---

random_start *Generate a random initial starting point for EM algorithm*

Description

random_start() is a general function for creating a random, plausible starting point for the EM algorithm to begin iterating from. Varying the starting points of repeated calculations can help the researcher detect local extremes.

Creates random initial sensitivity and specificity values for n_method methods. Values are generated from a random beta distribution with shape parameters a=3 and b=1. Prevalence is a random value from a uniform distribution.

Usage

```
random_start(type, n_method, method_names)

random_start_binary(n_method = NULL, method_names = NULL)

random_start_ordinal(n_method = NULL, method_names = NULL)
```

Arguments

type	A string specifying the data type of the methods being simulated.
n_method	An integer representing the number of methods to simulate.
method_names	Optional vector of names used to identify each method.

Value

List containing initial values to be passed to `init` argument

`setFreqs` *Set frequencies of MultiMethodMLEstimate objects*

Description

Set the freqs slot of a MultiMethodMLEstimate object. This can be used to update objects created using `emery` versions < 0.6.1 to work with newer versions.

Usage

```
setFreqs(x, freqs = NULL)
```

Arguments

x	An object of class MultiMethodMLEstimate.
freqs	A vector of non-negative integers

Value

A copy of the MultiMethodMLEstimate object with `freqs` in the appropriate slot.

setFreqs,MultiMethodMLEstimate-method

Set frequencies of MultiMethodMLEstimate objects

Description

Set the freqs slot of a MultiMethodMLEstimate object. This can be used to update objects created using `emery` versions < 0.6.1 to work with newer versions.

Usage

```
## S4 method for signature 'MultiMethodMLEstimate'
setFreqs(x, freqs = NULL)
```

Arguments

`x`	An object of class MultiMethodMLEstimate.
`freqs`	A vector of non-negative integers

Value

A copy of the MultiMethodMLEstimate object with `freqs` in the appropriate slot.

show,MultiMethodMLEstimate-method

Show a MultiMethodMLEstimate S4 object

Description

Print the accuracy statistic estimates stored in a MultiMethodMLEstimate object.

Usage

```
## S4 method for signature 'MultiMethodMLEstimate'
show(object)
```

Arguments

`object`	An object of class MultiMethodMLEstimate.

Value

A list containing relevant accuracy statistic estimates. This is a subset of the list stored in `results` slot of the MultiMethodMLEstimate object.

`unique_obs_summary` *Reduce Data by Summarizing Observations*

Description

Extract unique rows from a data frame. Collect row names of duplicate observations and frequencies of each.

Usage

```
unique_obs_summary(data)
```

Arguments

`data`	An `n_obs` by `n_method` `matrix` containing the observed values for each method. If the dimensions are named, row names will be used to name each observation (`obs_names`) and column names will be used to name each measurement method (`method_names`).

Value

list containing matrix of unique rows with names and frequencies of duplicates.

Index