Package 'tboot'

July 22, 2025

Title Tilted Bootstrap

Version 0.2.1

Description Creates simulated clinical trial data with realistic correlation structures and assumed efficacy levels by using a tilted bootstrap resampling approach. Samples are drawn from observed data with some samples appearing more frequently than others. May also be used for simulating from a joint Bayesian distribution along with clinical trials based on the Bayesian distribution.

License GPL-3

Depends R (>= 3.4.0)

Imports stats, quadprog, kernlab

Suggests knitr, rmarkdown, testthat, MASS, ggplot2

VignetteBuilder knitr

URL https://github.com/njm18/tboot

BugReports https://github.com/njm18/tboot/issues

RoxygenNote 7.0.2

NeedsCompilation no

Author Nathan Morris [aut, cre],
William Michael Landau [ctb],
Eli Lilly and Company [cph]

Maintainer Nathan Morris <morris_nathan@lilly.com>

Repository CRAN

Date/Publication 2020-12-02 16:40:02 UTC

Contents

`tboot-package` *tboot: tilted bootstrapping and Bayesian marginal reconstruction.*

Description

tboot: tilted bootstrapping and Bayesian marginal reconstruction.

Author(s)

Nathan Morris <morris_nathan@lilly.com>

References

https://github.com/njm18/tboot

`post_bmr` *Function post_bmr*

Description

Simulates the joint posterior based upon a dataset and specified marginal posterior distribution of the mean of selected variables.

Usage

```
post_bmr(nsims, weights_bmr)
```

Arguments

`nsims`	The number of posterior simulations to draw.
`weights_bmr`	An object of class 'tweights_bmr' created using the 'tweights_bmr' function.

Value

A matrix of simulations from the posterior.

See Also

`tweights_bmr`

Examples

```
#Use winsorized marginal to keep marginal simulation within feasible bootstrap region
winsor=function(marginalSims,y)  {
  l=min(y)
  u=max(y)
  ifelse(marginalSims<l,l,ifelse(marginalSims>u,u, marginalSims))
}
#Create an example marginal posterior
marginal = list(Sepal.Length=winsor(rnorm(10000,mean=5.8, sd=.2),iris$Sepal.Length),
               Sepal.Width=winsor(rnorm(10000,mean=3,sd=.2), iris$Sepal.Width),
               Petal.Length=winsor(rnorm(10000,mean=3.7,sd=.2), iris$Petal.Length)
)

#simulate
w = tweights_bmr(dataset = iris, marginal = marginal, silent = TRUE)
post_sims = post_bmr(1000, weights = w)
```

---

tboot *Function tboot*

Description

Bootstrap `nrow` rows of `dataset` using the given row-level weights.

Usage

```
tboot(nrow, weights, dataset = weights$dataset, fillMissingAug = TRUE)
```

Arguments

`nrow`	Number of rows in the new bootstrapped dataset.
`weights`	An object of class 'tweights' output from the 'tweights' function.
`dataset`	Data frame or matrix to bootstrap. By default, the dataset will come from the tweights object. Rows of the dataset must be in the same order as was used for the 'tweights' call. However the dataset may include additional columns not included in the 'tweights' call.
`fillMissingAug`	Fill in missing augmentation with primary weights resampling.

Details

Bootstrap samples from a dataset using the tilted weights. Details are further described in the vignette.

Value

A simulated dataset with 'nrow' rows.

See Also

tweights

Examples

```
target=c(Sepal.Length=5.5, Sepal.Width=2.9, Petal.Length=3.4)
w = tweights(dataset = iris, target = target, silent = TRUE)
simulated_data = tboot(nrow = 1000, weights = w)
```

---

tboot_bmr *Function tboot_bmr*

Description

Bootstrap nrow rows of dataset using the given row-level weights.

Usage

```
tboot_bmr(nrow, weights_bmr, tol_rel_sd = 0.01)
```

Arguments

nrow	Number of rows in the new bootstrapped dataset.
weights_bmr	An object of class 'tweights_bmr' output from the 'tweights_bmr' function.
tol_rel_sd	An error will be called if for some simulation if the target is not achievable with the data. However, the error will only be called if max absolute difference releative to the marginal standard is greater than specified.

Details

Simulates a dataset by first simulating from the posterior distribution of the column means and then simulating a dataset with that underlying mean. Details a further documented in the vignette.

Value

A simulated dataset with 'nrow' rows. The underlying 'true' posterior parameter value is an attribute which can be extracted useing attr(ret, "post_bmr") where 'ret' is the matrix.

See Also

tweights

Examples

```
#Use winsorized marginal to keep marginal simulation within feasible bootstrap region
winsor=function(marginalSims,y)  {
  l=min(y)
  u=max(y)
  ifelse(marginalSims<l,l,ifelse(marginalSims>u,u, marginalSims))
}
#Create an example marginal posterior
marginal = list(Sepal.Length=winsor(rnorm(10000,mean=5.8, sd=.2),iris$Sepal.Length),
               Sepal.Width=winsor(rnorm(10000,mean=3,sd=.2), iris$Sepal.Width),
               Petal.Length=winsor(rnorm(10000,mean=3.7,sd=.2), iris$Petal.Length)
)

#simulate
w = tweights_bmr(dataset = iris, marginal = marginal, silent = TRUE)
sample_data = tboot_bmr(1000, weights = w)
```

---

tweights *Function* tweights

Description

Returns a vector p of resampling probabilities such that the column means of tboot(dataset = dataset, p = p) equals target on average.

Usage

```
tweights(dataset, target = apply(dataset, 2, mean), distance = "klqp",
  maxit = 1000, tol = 1e-08, warningcut = 0.05, silent = FALSE,
  Nindependent = 0)
```

Arguments

dataset	Data frame or matrix to use to find row weights.
target	Numeric vector of target column means. If the 'target' is named, then all elements of names(target) should be in the dataset.
distance	The distance to minimize. Must be either 'euchlidean,' 'klqp' or 'klpq' (i.e. Kullback-Leibler). 'klqp' which is exponential tilting is recommended.
maxit	Defines the maximum number of iterations for optimizing 'kl' distance.
tol	Tolerance. If the achieved mean is to0 far from the target (i.e. as defined by tol) an error will be thrown.
warningcut	Sets the cutoff for determining when a large weight will trigger a warning.
silent	Allows silencing of some messages.
Nindependent	Assumes the input also includes 'Nindependent' samples with independent columns. See details.

Details

Let $p_i = 1/n$ be the probability of sampling subject i from a dataset with n individuals (i.e. rows of the dataset) in the classic resampling with replacement scheme. Also, let q_i be the probability of sampling subject i from a dataset with n individuals in our new resampling scheme. Let $d(q, p)$ represent a distance between the two resampling schemes. The `tweights` function seeks to solve the problem:

$$q = argmin_p d(q, p)$$

Subject to the constraint that:

$$sum_i q_i = 1$$

and

$$dataset' q = target$$

where dataset is a n x K matrix of variables input to the function.

$$d_{euclidian}(q, p) = sqrt(\sum_i (p_i - q_i)^2)$$

$$d_{kl}(q, p) = \sum_i (log(p_i) - log(q_i))$$

Optimization for Euclidean distance is a quadratic program and utilizes the ipop function in kernLab. Optimization for the others utilize a Newton-Raphson type iterative algorithm.

If the original target cannot be achieved. Something close to the original target will be selected. A warning will be produced and the new target displayed.

The 'Nindependent' option augments the dataset by assuming some additional specified number of patients. These patients are assumed to made up of a random bootstrapped sample from the dataset for each variable marginally leading to independent variables.

Value

An object of type `tweights`. This object contains the following components:

weights Tilted weights for resampling

originalTarget Will be null if target was not changed.

target Actual target that was attempted.

achievedMean Achieved mean from tilting.

dataset Inputed dataset.

X Reformated dataset.

Nindependent Inputed 'Nindependent' option.

See Also

`tboot`

Examples

```
target=c(Sepal.Length=5.5, Sepal.Width=2.9, Petal.Length=3.4)
w = tweights(dataset = iris, target = target, silent = TRUE)
simulated_data = tboot(nrow = 1000, weights = w)
```

`tweights_bmr` *Function tweights_bmr*

Description

Set up the needed prerequisites in order to prepare for Bayesian marginal reconstruction (including a call to tweights). Takes as input simulations from the posterior marginal distribution of variables in a dataset.

Usage

```
tweights_bmr(dataset, marginal, distance = "klqp", maxit = 1000,
  tol = 1e-08, warningcut = 0.05, silent = FALSE, Nindependent = 1)
```

Arguments

`dataset`	Data frame or matrix to use to find row weights.
`marginal`	Must be a named list with each element a vector of simulations of the marginal distribution of the posterior mean of data in the dataset.
`distance`	The distance to minimize. Must be either 'euchlidean,' 'klqp' or 'klpq' (i.e. Kullback-Leibler). 'klqp' which is exponential tilting is recommended.
`maxit`	Defines the maximum number of iterations for optimizing 'kl' distance.
`tol`	Tolerance. If the achieved mean is too far from the target (i.e. as defined by tol) an error will be thrown.
`warningcut`	Sets the cutoff for determining when a large weight will trigger a warning.
`silent`	Allows silencing of some messages.
`Nindependent`	Assumes the input also includes 'Nindependent' samples with independent columns. See details.

Details

Reconstructs a correlated joint posterior from simulations from a marginal posterior. The algorithm is summarized more fully in the vignettes. The 'Nindependent' option augments the dataset by assuming some additional specified number of patients. These patients are assumed to made up of a random bootstrapped sample from the dataset for each variable marginally leading to independent variables.

Value

An object of type `tweights`. This object conains the following components:

Csqrt Matrix square root of the covariance.

tweights Result from the call to tweigths.

marginal Input marginal simulations.

dataset Formatted dataset.

target Attempted target.

distance,maxit,tol, Nindependent, warningcut Inputed values to 'tweights_bmr'.

Nindependent Inputed 'Nindependent' option.

augmentWeights Used for 'Nindependent' option weights for each variable.

weights Tilted weights for resampling

originalTarget Will be null if target was not changed.

marginal_sd Standard deviation of the marginals.

See Also

tweights

Examples

```
#Use winsorized marginal to keep marginal simulation within feasible bootstrap region
winsor=function(marginalSims,y)  {
  l=min(y)
  u=max(y)
  ifelse(marginalSims<l,l,ifelse(marginalSims>u,u, marginalSims))
}
#Create an example marginal posterior
marginal = list(Sepal.Length=winsor(rnorm(10000,mean=5.8, sd=.2),iris$Sepal.Length),
               Sepal.Width=winsor(rnorm(10000,mean=3,sd=.2), iris$Sepal.Width),
               Petal.Length=winsor(rnorm(10000,mean=3.7,sd=.2), iris$Petal.Length)
)

#simulate
w = tweights_bmr(dataset = iris, marginal = marginal, silent = TRUE)
post1 = post_bmr(1000, weights = w)
```

Index