Package ‘bfslice’

Type Package
Title Bayesian nonparametric tests via sliced inverse modeling
Version 1.0.0
Date 2014-09-25
Author Chao Ye and Bo Jiang
Maintainer Chao Ye <yechao1009@gmail.com>
Description This R package implements a non-parametric dependence testing method for categorical covariates and continuous response. Instead of estimating the conditional distribution of the response given values of covariates, we model the conditional distribution of covariates given the discretized response (aka “slices”). By assigning a prior probability to each possible discretization scheme, we compute efficiently a Bayes factor (BF)-statistic for the independence (or conditional independence) test using a dynamic programming algorithm. The package also contains functions for dynamic slicing based K-sample test (see Jiang, Ye and Liu, JASA, 2014). The R package bfslice requires Rcpp and ggplot2.
License GPL (>= 2)
Depends Rcpp (>= 0.11.1), ggplot2 (>= 0.9.3.1), scales
LinkingTo Rcpp

R topics documented:

bfslice_c ......................................................... 2
bfslice_eqp_c .................................................. 3
bfslice_eqp_u .................................................. 4
bfslice_u ........................................................ 5
ds_1 ............................................................. 6
ds_eqp_1 ........................................................ 7
ds_eqp_k ........................................................ 8
ds_k .............................................................. 9
ds_test .......................................................... 10
relabel ............................................................ 12
slice_show ...................................................... 13

Index 14
bfslice_c

**Dependency and conditional dependency detection between a level k (k > 1) categorical variable and a continuous variable via Bayes factor.**

**Description**

Conditional dependency detection between a level $k_z$ ($k_z > 1$) categorical variable $x$ and a continuous variable $y$ via Bayes factor given a level $k_z$ categorical variable $z$. If $k_z = 1$, it is unconditional dependency detection method. It could be applied for non-parametric variable selection.

**Usage**

```
bfslice_c(z, x, zdim, xdim, lambda, alpha)
```

**Arguments**

- **z**: Vector: observations of given (preselected) categorical variable, 0, 1, ..., $k_z - 1$ for level $k_z$ categorical variable, should be ranked according to values of continuous variable $y$ with $x$ in advanced, either ascending or descending.
- **x**: Vector: observations of categorical variable, 0, 1, ..., $k_x - 1$ for level $k_x$ categorical variable, should be ranked according to values of continuous variable $y$ with $z$ in advanced, either ascending or descending.
- **zdim**: Level of $z$, equals $k_z$.
- **xdim**: Level of $x$, equals $k_x$.
- **lambda**: lambda corresponds to the probability that makes slice in each possible position. lambda should be greater than 0.
- **alpha**: alpha is hyper-parameter of the prior distribution of frequency in each slice. alpha should be greater than 0 and less equal than $k_x$.

**Value**

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.

**See Also**

`bfslice_u`, `bfslice_eqp_c`.

**Examples**

```
n <- 100
mu <- 0.5

## Unconditional test
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
z <- rep(0, 2*n)

## Conditional test
```
bfslice_eqp_c

Dependency and conditional dependency detection between a level $k$ ($k > 1$) categorical variable and a continuous variable via Bayes factor.

Description

Conditional dependency detection between a level $k_z$ ($k_z > 1$) categorical variable $x$ and a continuous variable $y$ via Bayes factor given a level $k_z$ categorical variable $z$ with $O(n^{1/2})$-resolution. The basic idea is almost the same as bfslice_c. The only different is that bfslice_eqp_c groups samples into approximate $O(n^{1/2})$ groups which contain approximate $O(n^{1/2})$ samples and treat the groups as a sample to calculate Bayes factor. If $k_z = 1$, it is unconditional dependency detection method. It could be applied for non-parametric variable selection.

Usage

bfslice_eqp_c(z, x, zdim, xdim, lambda, alpha)

Arguments

- **z**: Vector: observations of given (preselected) categorical variable, $0, 1, \ldots, k_z - 1$ for level $k_z$ categorical variable, should be ranked according to values of continuous variable $y$ with $x$ in advanced, either ascending or descending.
- **x**: Vector: observations of categorical variable, $0, 1, \ldots, k_x - 1$ for level $k_x$ categorical variable, should be ranked according to values of continuous variable $y$ with $z$ in advanced, either ascending or descending.
- **zdim**: Level of $z$, equals $k_z$.
- **xdim**: Level of $x$, equals $k_x$.
- **lambda**: lambda corresponds to the probability that makes slice in each possible position. **lambda** should be greater than 0.
- **alpha**: alpha is hyper-parameter of the prior distribution of frequency in each slice. **alpha** should be greater than 0 and less equal than $k_x$.

Value

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.
**See Also**

`bfslice_c`, `bfslice_eqp_u`.

**Examples**

```r
n <- 1000
mu <- 0.2

## Unconditional test
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
z <- rep(0, 2*n)

## Conditional test
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n/5), rep(1, n), rep(0, 4*n/5))
z <- c(rep(0, n), rep(1, n))
z <- z[order(y)]
x <- x[order(y)]
zd <- max(z) + 1
xd <- max(x) + 1
lambda <- 1.0
alpha <- 1.0
bfval <- bfslice_eqp_c(z, x, zd, xd, lambda, alpha)
```

---

**bfslice_eqp_u**  
Dependency detection between a level $k$ ($k > 1$) categorical variable and a continuous variable via Bayes factor with given size of each group.

**Description**

Dependency detection between a level $k$ ($k > 1$) categorical variable $x$ and a continuous variable $y$ via Bayes factor with $O(n^{1/2})$-resolution. The basic idea is almost the same as `bfslice_u`. The only different is that `bfslice_eqp_u` groups samples into approximate $O(n^{1/2})$ groups which contain approximate $O(n^{1/2})$ samples and treat the groups as a sample to calculate Bayes factor.

**Usage**

```r
bfslice_eqp_u(x, dim, lambda, alpha)
```

**Arguments**

- **x**  
  Vector: observations of categorical variable, 0, 1, ..., $k - 1$ for level $k$ categorical variable, should be ranked according to values of continuous variable $y$, either ascending or descending.

- **dim**  
  Level of $x$, equals $k$.

- **lambda**  
  Lambda corresponds to the probability that makes slice in each possible position. Lambda should be greater than 0.

- **alpha**  
  Alpha is hyper-parameter of the prior distribution of frequency in each slice. Alpha should be greater than 0 and less equal than $k$. 
Value

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.

See Also

\texttt{bfslice_u}, \texttt{bfslice_eqp_c}.

Examples

\begin{verbatim}
n <- 1000
mu <- 0.2
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
x <- x[order(y)]
dim <- max(x) + 1
lambda <- 1.0
alpha <- 1.0
bfval <- bfslice_eqp_u(x, dim, lambda, alpha)
\end{verbatim}

\begin{verbatim}
bfval
\end{verbatim}

Description

Dependency detection between a level $k$ ($k > 1$) categorical variable and a continuous variable via Bayes factor.

Usage

\texttt{bfslice_u(x, dim, lambda, alpha)}

Arguments

\begin{itemize}
\item \texttt{x} Vector: observations of categorical variable, $0, 1, \ldots, k - 1$ for level $k$ categorical variable, should be ranked according to values of continuous variable \texttt{y}, either ascending or descending.
\item \texttt{dim} Level of \texttt{x}, equals $k$.
\item \texttt{lambda} lambda corresponds to the probability that makes slice in each possible position. lambda should be greater than 0.
\item \texttt{alpha} alpha is hyper-parameter of the prior distribution of frequency in each slice. alpha should be greater than 0 and less equal than $k$.
\end{itemize}

Value

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.
See Also

\[ \text{bfslice}_c, \text{bfslice}_eqp_u. \]

Examples

\[
\begin{align*}
n & \leftarrow 100 \\
mu & \leftarrow 0.5 \\
y & \leftarrow c(rnorm(n, -mu, 1), rnorm(n, mu, 1)) \\
x & \leftarrow c(rep(0, n), rep(1, n)) \\
x & \leftarrow x[order(y)] \\
dim & \leftarrow \max(x) + 1 \\
lambda & \leftarrow 1.0 \\
alpha & \leftarrow 1.0 \\
bfval & \leftarrow \text{bfslice}_u(x, \dim, \lambda, \alpha)
\end{align*}
\]

---

\[
\text{ds}_1 \quad \text{Non-parametric one-sample hypothesis testing via dynamic slicing}
\]

Description

Non-parametric one-sample hypothesis testing via dynamic slicing. By mapping sample values to the quantile of null distribution, \[\text{ds}_1\] test whether they follow uniform distribution on \([0, 1]\) via a regularized likelihood-ratio. Its calculated is based on a dynamic programming procedure.

Usage

\[
\text{ds}_1(y, \lambda, \alpha)
\]

Arguments

- \(y\) Vector: quantiles of observations according to null distribution.
- \(\lambda\) Lambda penalizes the number of slices to avoid too many slices. \(\lambda\) should be greater than 0.
- \(\alpha\) alpha penalizes both the width and the number of slices to avoid too many slices and degenerate slice (interval). \(\alpha\) should be greater than 1.

Value

Value of dynamic slicing statistic for one-sample test. It is nonnegative. The null hypothesis that observations are from the null distribution is rejected if this statistic is greater than zero, otherwise accept the null hypothesis.

See Also

\[ \text{ds}_eqp_1. \]
Examples

```r
n <- 100
mu <- 0.5
x <- rnorm(n, mu, 1)
y <- pnorm(sort(x), 0, 1)
lambda <- 1.0
alpha <- 1.0
dsres <- ds_1(y, lambda)
```

Non-parametric one-sample hypothesis testing via dynamic slicing

Description

Non-parametric one-sample hypothesis testing via dynamic slicing with $O(n)$-resolution. The basic idea of `ds_eqp_1` is almost the same as `ds_1`. Difference between these two functions is that `ds_eqp_1` considers an equal partition on [0, 1] but `ds_1` does not. Candidate slicing boundaries in `ds_eqp_1` only depend on the total number of samples and are unrelated to sample quantiles. In `ds_1` they are immediately to the left or right of sample quantile.

Usage

```r
ds_eqp_1(y, lambda)
```

Arguments

- **y** Vector: quantiles of observations according to null distribution.
- **lambda** lambda penalizes the number of slices to avoid too many slices. Since the interval [0, 1] is divided into $n$ equal size element-slice and slicing strategy only consider boundaries of them, this version of dynamic slicing does not require penalty lambda as `ds_1`. lambda should be greater than 0.

Value

Value of dynamic slicing statistic for one-sample test. It is nonnegative. The null hypothesis that observations are from the null distribution is rejected if this statistic is greater than zero, otherwise accept the null hypothesis.

See Also

`ds_1`.

Examples

```r
n <- 100
mu <- 0.5
x <- rnorm(n, mu, 1)
y <- pnorm(sort(x), 0, 1)
lambda <- 1.0
dsres <- ds_eqp_1(y, lambda)
```
Dependency detection between level $k$ ($k > 1$) categorical variable and continuous variable via dynamic slicing with $O(n^{1/2})$-resolution. The basic idea is almost the same as $ds_k$. The only different is that $ds_{eqp_k}$ groups samples into approximate $O(n^{1/2})$ groups which contain approximate $O(n^{1/2})$ samples and performs dynamic slicing on their boundaries. This much faster version could reduce computation time substantially without too much power loss. Based on the strategy of $ds_{eqp_k}$, we recommend to apply it in large sample size problem and use $ds_k$ for ordinary problem. For more details please refer to Jiang, Ye & Liu (2015). Results contains value of dynamic slicing statistic and slicing strategy. It could be applied for non-parametric $K$-sample hypothesis testing.

### Usage

ds_eqp_k(x, xdim, lambda, slice = FALSE)

### Arguments

- **x**: Vector: observations of categorical variable, $0, 1, \ldots, k-1$ for level $k$ categorical variable, should be ranked according to values of continuous variable in advanced, either ascending or descending.
- **xdim**: Level of $x$, equals $k$.
- **lambda**: Penalty for introducing an additional slice, which is used to avoid making too many slices. It corresponds to the type I error under the scenario that the two variables are independent. $\lambda$ should be greater than 0.
- **slice**: Indicator for reporting slicing strategy or not.

### Value

- **dsval**: Value of dynamic slicing statistic. It is nonnegative. If it equals zero, the categorical variable and continuous variable will be treated as independent of each other, otherwise they will be treated as dependent.
- **slices**: Slicing strategy that maximize dynamic slicing statistic based on currently ranked vector $x$. It will be reported if slice is true. Each row stands for a slice. Each column except the last one stands for the number of observations take each value in each slice. The last column is the number of observations in each slice $i.e.$, the sum of the first column to the $k$th column.

### References


### See Also

ds_k.
**Examples**

```r
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep("1", n), rep("2", n))
x <- relabel(x)
x <- x[order(y)]
xdim <- max(x) + 1
lambda <- 1.0
dsres <- ds_eqp_k(x, xdim, lambda, slice = TRUE)
```

---

**ds_k**

*Dependency detection between level k (k > 1) categorical variable and continuous variable*

**Description**

Dependency detection between level $k$ ($k > 1$) categorical variable and continuous variable. The basic idea is that the different values of categorical variable correspond to different distribution of continuous variable if there exist dependency between this two variables, otherwise the distributions of continuous variable do not show difference conditioning on the values of categorical variable. Statistic for this dynamic slicing method is a regularized likelihood-ratio calculated via a dynamic programming procedure. For more details please refer to Jiang, Ye & Liu (2015). Results contains value of dynamic slicing statistic and slicing strategy. It could be applied for non-parametric $K$-sample hypothesis testing.

**Usage**

```r
dsk(x, xdim, lambda, slice = FALSE)
```

**Arguments**

- **x**: Vector: observations of categorical variable, 0, 1, ..., $k - 1$ for level $k$ categorical variable, should be ranked according to values of continuous variable in advanced, either ascending or descending.
- **xdim**: Level of $x$, equals $k$.
- **lambda**: Penalty for introducing an additional slice, which is used to avoid making too many slices. It corresponds to the type I error under the scenario that the two variables are independent. $lambda$ should be greater than 0.
- **slice**: Indicator for reporting slicing strategy or not.

**Value**

- **dsval**: Value of dynamic slicing statistic. It is nonnegative. If it equals zero, the categorical variable and continuous variable will be treated as independent of each other, otherwise they will be treated as dependent.
- **slices**: Slicing strategy that maximize dynamic slicing statistic based on currently ranked vector $x$. It will be reported if slice is true. Each row stands for a slice. Each column except the last one stands for the number of observations take each value in each slice. The last column is the number of observations in each slice i.e., the sum of the first column to the $k$th column.
References


See Also

ds_eqp_k.

Examples

```r
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep("1", n), rep("2", n))
x <- relabel(x)
x <- x[order(y)]
xdim <- max(x) + 1
lambda <- 1
dsres <- ds_k(x, xdim, lambda, slice = TRUE)
```

## ds_test

*Hypothesis testing via dynamic slicing*

**Description**

Perform a one- or $K$-sample ($K > 1$) hypothesis testing via dynamic slicing.

**Usage**

```r
ds_test(y, x, ..., type = c("ds", "eqp"), lambda = 1, alpha = 1, rounds = 0)
```

**Arguments**

- `y` A numeric vector of data values.
- `x` Either an integer vector of data values, from 0 to $K - 1$, or a character string naming a cumulative distribution function or an actual cumulative distribution function such as `pnorm`. Only continuous CDFs are valid.
- `...` Parameters of the distribution specified (as a character string) by `x`.
- `type` Methods applied for dynamic slicing. "ds" (default) stands for original dynamic slicing scheme. "eqp" stands for dynamic slicing scheme with $n^{1/2}$-resolution (for $K$-sample test, $K > 1$) or $n$-resolution (for one-sample test).
- `lambda` Penalty for introducing an additional slice, which is used to avoid making too many slices. It corresponds to the type I error under the scenario that the two variables are independent. `lambda` should be greater than 0.
- `alpha` Penalty required for "ds" type in one-sample test. It penalizes both the width and the number of slices to avoid too many slices and degenerate slice (interval). `alpha` should be greater than 1.
- `rounds` Number of permutations for estimating empirical $p$-value.
Details

If \( x \) is an integer vector, \( \text{ds\_test} \) performs \( K \)-sample test \((K > 1)\).

Under this scenario, suppose that there are observations \( y \) drawn from some continuous populations.
Let \( x \) be a vector that stores values of indicator of samples from different populations, i.e., \( x \) has values \( 0, 1, \ldots, K - 1 \). The null hypothesis is that these populations have the same distribution.

If \( x \) is a character string naming a continuous (cumulative) distribution function, \( \text{ds\_test} \) performs one-sample test with the null hypothesis that the distribution function which generated \( y \) is distribution \( x \) with parameters specified by \( \ldots \). The parameters specified in \( \ldots \) must be pre-specified and not estimated from the data.

Only empirical \( p \)-values are available by specifying the value of parameter \( \text{rounds} \), the number of permutation. \( \lambda \) and \( \alpha \) (for one-sample test with type "ds") contributes to \( p \)-value.

The procedure of choosing parameter \( \lambda \) was described in Jiang, Ye & Liu (2015). Refer to http://www.people.fas.harvard.edu/~junliu/DS/lambda-table.html for the empirical relationship of \( \lambda \), sample size and type I error.

Value

A list with class "htest" containing the following components:

- statistic: The value of the dynamic slicing statistic.
- p.value: The \( p \)-value of the test.
- alternative: A character string describing the alternative hypothesis.
- method: A character string indicating what type of test was performed.
- data.name: A character string giving the name(s) of the data.
- slices: Slicing strategy that maximize dynamic slicing statistic in \( K \)-sample test. Each row stands for a slice. Each column except the last one stands for the number of observations take each value in each slice. The last column is the number of observations in each slice i.e., the sum of the first column to the \( k \)th column.

References


Examples

```r
## One-sample test
n <- 100
mu <- 0.5
y <- rnorm(n, mu, 1)
lambda <- 1.0
alpha <- 1.0
dares <- ds_test(y, "pnorm", 0, 1, lambda = 1, alpha = 1, rounds = 100)
dares <- ds_test(y, "pnorm", 0, 1, type = "ds", lambda = 1, alpha = 1)
dares <- ds_test(y, "pnorm", 0, 1, type = "eqp", lambda = 1, rounds = 100)
dares <- ds_test(y, "pnorm", 0, 1, type = "eqp", lambda = 1)

## K-sample test
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
```
## generate x in this way:
x <- c(rep(0, n), rep(1, n))
x <- as.integer(x)

## or in this way:
x <- c(rep("G1", n), rep("G2", n))
x <- relabel(x)

lambda <- 1.0
dsres <- ds_test(y, x, lambda = 1, rounds = 100)
dsres <- ds_test(y, x, type = "eqp", lambda = 1, rounds = 100)

### relabel

**Reassigning values of categorical variable**

**Description**

Reassigning values of categorical variable. It is used for generating legal value of categorical variable before applying dynamic slicing.

**Usage**

relabel(x)

**Arguments**

- **x**: A vector of data values.

**Value**

An integer vector with values range from 0 to k (k > 0).

**See Also**

ds_test.

**Examples**

n <- 10
x <- c(rep("G1", n), rep("G2", n))
x <- relabel(x)

x <- c(rep(4, n), rep(5, n), rep(NA, n))
x <- relabel(x)
slice_show

Show the slicing result

**Description**

Showing slicing result and plotting counts of observations in each slice.

**Usage**

```r
slice_show(slices_obj, main="Counts in each slice", xlab="Slices", ylab="Percentage")
```

**Arguments**

- `slices_obj`: A matrix stores slicing strategy. It is a component of object returned by function `dslice_k` or `dslice_eqp_k`
- `main`: An overall title for the plot
- `xlab`: A title for the x axis
- `ylab`: A title for the y axis

**Value**

A “ggplot” object which illustrates details of slicing.

**See Also**

`ds_k`, `ds_eqp_k`

**Examples**

```r
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
x <- x[order(y)]
xdim <- max(x) + 1
lambda <- 1.0
dsres <- ds_k(x, xdim, lambda, slice = TRUE)
ds_show <- slice_show(dsres$slices)
```
Index

bfslice_c, 2, 3, 4, 6
bfslice_eqp_c, 2, 3, 3, 5
bfslice_eqp_u, 4, 4, 6
bfslice_u, 2, 4, 5, 5

d_1, 6, 7
d_eqp_1, 6, 7
d_eqp_k, 8, 8, 10, 13
d_k, 8, 9, 13
d_test, 10, 12

relabel, 12

slice_show, 13