Package ‘cepp’

October 12, 2022

Type Package
Title Context Driven Exploratory Projection Pursuit
Version 1.7
Date 2016-01-16
Author Mohit Dayal
Maintainer Mohit Dayal <mohitdayal2000@gmail.com>
Description Functions and Data to support Context Driven Exploratory Projection Pursuit.
Imports randtoolbox
Depends R (>= 2.10), trust
License GPL-3
NeedsCompilation no
Repository CRAN
Date/Publication 2016-01-30 00:19:38

R topics documented:

cep-packet ......................................................... 2
bases . ............................................................. 2
caller ............................................................. 3
Colon ............................................................... 4
CvM ................................................................. 5
evaluator .......................................................... 6
geodesic ........................................................... 7
Olive oil measurements ........................................... 8
pp ................................................................. 9

Index 11
Description

Functions and Data to support Context Driven Exploratory Projection Pursuit.

Details

The DESCRIPTION file:

Package: cepp
Type: Package
Title: Context Driven Exploratory Projection Pursuit
Version: 1.7
Date: 2016-01-16
Author: Mohit Dayal
Maintainer: Mohit Dayal <mohitdayal2000@gmail.com>
Description: Functions and Data to support Context Driven Exploratory Projection Pursuit.
Imports: randtoolbox
Depends: trust
License: GPL-3

Author(s)

Mohit Dayal
Maintainer: Mohit Dayal <mohitdayal2000@gmail.com>

bases

Create random bases

Description

Generate bases.

Usage

basis_random(n, d = 2)
basis_nearby(alpha = 0.75, method = 'geodesic', d = 2)

Arguments

n
The number of rows.

d
The number of columns.
alpha  How "far" away should the new matrix be generated?
method  How should be new matrix be found? One of linear or geodesic.

Details

basis_random returns a new orthonormal matrix of specified dimensions.
basis_nearby generates a function. Calling this function with a matrix, hybridizes it with a new
(randomly generated via basis_random) orthonormal matrix, and returns it.

Value

For basis_random, a random orthonormal matrix of specified dimensions.
For basis_nearby, a function that can be used to generate new matrices "near" the current matrix.

Author(s)

Both functions were originally taken from the tourr package. The basis_nearby function was modified so the parameters alpha and method can be set more conveniently during optimization.

caller  

Function to optimize the projection index

Description

This function provides an alternative way to optimize the projection index, by moving along a
geodesic path.

Usage

caller(start, index, n, bases)

Arguments

start  The Starting Projection for the optimization.
index  The Projection Index function. Typically generated by a call to the pp function.
n  The number of new bases to try at every stage of the optimization. Needs to be an array of the same length as bases. Typically, you either pass a constant vector, or you use a vector with ascending entries, so that you can try more matrices as the optimization proceeds.
bases  The number of new bases desired. Actual number generated may be lesser if optimization stalls.

Details

This function provides an alternative way to optimize the projection index. It moves the index along
geodesic paths between randomly generated nearby matrices, in hopes of uncovering peaks of the
index function. By experience, one can say that it can often reveal structure missed by Simulated
Annealing optimization.
Value

A list of basis matrices, of length bases or shorter (if the optimization stalls).

Author(s)

Mohit Dayal

See Also

Colon

Description

Gene expression data (2000 genes for 62 samples) from the microarray experiments of Colon tissue samples of Alon et al. (1999).

Usage

data(Colon)

Details

This data set contains 62 samples with 2000 genes: 40 tumor tissues, coded 2 and 22 normal tissues, coded 1.

Value

A list with the following elements:

- **X**: a (62 x 2000) matrix giving the expression levels of 2000 genes for the 62 Colon tissue samples. Each row corresponds to a patient, each column to a gene.
- **Y**: a numeric vector of length 62 giving the type of tissue sample (tumor or normal).
- **gene.names**: a vector containing the names of the 2000 genes for the gene expression matrix X.

Source

The data are described in Alon et al. (1999) and can be freely downloaded from http://microarray.princeton.edu/oncology/affydata/index.html.

References

Examples

```r
# how many samples and how many genes?
data(Colon)
dim(Colon$X)
norm <- Colon$X[Colon$Y == 1,]
tumor <- Colon$X[Colon$Y == 2,]
gene1 <- pp(r=2,n=50,oth=tumor,data=norm,k=2)
F1 <- basis_random(2000)
gene1(F1)
t1 <- caller(start=F1, index=gene1, n=rep(3,5), bases=5)
```

CvM

**Projection Pursuit Indices based on the bivariate empirical distribution function.**

Description

This function can be used to compute the projection pursuit indices described in Perisic and Posse (2005).

Usage

```r
ecdf.indices(A, sphered = FALSE)
```

Arguments

- **A** The projected data.
- **sphered** Whether the data has already been sphered or not. If set to FALSE (default), the function will sphere the data before computing the indices.

Details

The two-dimensional empirical distribution function is defined as,

\[ F_n(x, y) = \frac{1}{n} \#\{(x_j, y_j) : x_j \leq x \text{ and } y_j \leq y\} \]

The indices described in Perisic and Posse (2005) use this function to construct the following four indices.

Cramer-von-Mises:

\[ \sum_i (F_n(x_i, y_i) - \Phi(x_i)\Phi(y_i))^2 \]

Kolmogorov-Smirnov:

\[ \max_i |F_n(x_i, y_i) - \Phi(x_i)\Phi(y_i)| \]

D2:

\[ \sum_i (F_n(x_i, y_i) - F_n(y_i, x_i))^2 \]
D-infinity:
\[
\max_i |F_n(x_i, y_i) - F_n(y_i, x_i)|
\]

where \(\Phi(.\)) is the cumulative distribution function of the standard normal distribution.

When using any of these indices, the original authors recommended rotating the data projection several times to obtain rotational invariance. In simulations, the indices performed well even without rotations.

**Value**

A named numeric vector with the values of the following indices: the Cramer-von-Mises index, the Kolmogorov-Smirnov index, the D2 Symmetry index, and the D-infinity Symmetry index.

**Author(s)**

Mohit Dayal

**References**


---

**evaluator**

Function to evaluate spatial quantiles

**Description**

This provides an objective function whose minimization yields the spatial quantiles.

**Usage**

`evaluator(n, p)`

**Arguments**

- `n` The number of rows in the data
- `p` The number of columns in the data

**Details**

Returns another function suitable for passing to an optimizer like `nlm` or `trust`.

**Value**

A function that should be passed to an optimizer.
Author(s)
Mohit Dayal

References

Examples

```r
x <- rnorm(500)
dim(x) <- c(250,2)
ev <- evaluator(250,2)
##The Spatial Median
trust(ev, parinit=c(median(x[1,]), median(x[2,])), u=c(0,0),
     rinit=0.5, rmax=2e5, samp = x)
##Quantile for vector (0.2,0.3)
trust(ev, parinit=c(median(x[1,]), median(x[2,])), u=c(0.2,0.3),
     rinit=0.5, rmax=2e5, samp = x)
```

---

### geodesic

Functions for geodesic search

**Description**

This function provides an alternative way to optimize the projection index, by moving along a geodesic path.

**Usage**

```r
search_geodesic(current, alpha = 1, index, max.tries = 5, n = 5)
```

**Arguments**

- **current** The starting projection.
- **alpha** Maximum distance to travel (currently ignored).
- **index** The projection index.
- **max.tries** Maximum number of failed attempts before giving up.
- **n** Number of random steps to take to find best direction.

**Details**

The function `search_geodesic` finds only one basis at a time. The caller is a wrapper function that calls `search_geodesic` bases number of times.

**Value**

Returns the basis found.
Olive oil measurements

Author(s)
The function has been copied as is from the tourr package.

Olive oil measurements

Olive oil samples from Italy

Description
This data is from a paper by Forina, Armanino, Lanteri, Tiscornia (1983) Classification of Olive Oils from their Fatty Acid Composition, in Martens and Russwurm (ed) Food Research and Data Analysis. We thank Prof. Michele Forina, University of Genova, Italy for making this dataset available.

- region Three super-classes of Italy: North, South and the island of Sardinia
- area Nine collection areas: three from North, four from South and 2 from Sardinia
- palmitic, palmitoleic, stearic, oleic, linoleic, linolenic, arachidic, eicosenoic fatty acids percent \( \times 100 \)

Usage
data(olive)

Format
A 572 x 10 numeric array

Examples
data(olive)
head(olive)

#Permutation
OlivesT <- as.matrix(olive[, -c(1:2)])
OlivesF <- OlivesT

#You should set.seed here so as to “fix” the benchmark
OlivesF[, 'palmitic'] <- OlivesF[sample(572,572), 'palmitic']
OlivesF[, 'palmitoleic'] <- OlivesF[sample(572,572), 'palmitoleic']
OlivesF[, 'stearic'] <- OlivesF[sample(572,572), 'stearic']
OlivesF[, 'oleic'] <- OlivesF[sample(572,572), 'oleic']
OlivesF[, 'linoleic'] <- OlivesF[sample(572,572), 'linoleic']
OlivesF[, 'linolenic'] <- OlivesF[sample(572,572), 'linolenic']
OlivesF[, 'arachidic'] <- OlivesF[sample(572,572), 'arachidic']
OlivesF[, 'eicosenoic'] <- OlivesF[sample(572,572), 'eicosenoic']

##
oil1 <- pp(r=2, n=50, oth=OlivesF, data=OlivesT, k=2)
##In practice try at least >10 starting values
F1 <- basis_random(8)
##Increase iterations to >2000 for useful results
o1 <- optim(par=F1, fn=oil1, gr=basis_nearby(), method='SANN',
control=list(fnscale=-1, maxit=50, trace=6))
pp

Creates the projection pursuit function.

Description

These functions encapsulate everything, that is, the data, the benchmark and the index parameters, needed to compute the projection index.

Usage

pp(r = 0.8, n, data, oth, k)

Arguments

r
The radius multiplier. Values between 0.5 and 3 seem to work well.

n
Number of Monte-Carlo Evaluations to approximate the integral. Values as low as 25 can be used.

data
The data for which structure needs to be found.

oth
The benchmark dataset.

k
The target dimension.

Details

pp is for projection pursuit.

Value

The actual index function, which takes a single matrix argument, and returns the index value for that projection.

Author(s)

Mohit Dayal

Examples

## Exploring structure in the RANDU data
## Or using the MINSTD generator
randu <- as.matrix(randu)

randtoolbox::setSeed(570)
w <- randtoolbox::congruRand(1200)
dim(w) <- c(3, 400)
w <- t(w)
m <- 'geodesic'
a <- 0.50
ranif1 <- pp(r=1, n=50, data=randu, oth=w, k=2)

set.seed(50)
F1 <- basis_random(3)
o1 <- optim(par=F1, fn=ranif1, gr=basis_nearby(), method='SANN',
             control=list(fnscale=-1, maxit=100, trace=1))
plot(randu %*% o1$par)

##How accurate are the values?
ranif1hi <- pp(r=1, n=500, data=randu, oth=w, k=2)
ranif1hi(o1$par)
Index

* datasets
  Colon, 4
  Olive oil measurements, 8
* projection pursuit
  cepp-package, 2

bases, 2
basis_nearby (bases), 2
basis_random (bases), 2

caller, 3
cepp (cepp-package), 2
cepp-package, 2
Colon, 4
CvM, 5
D1 (CvM), 5
D2 (CvM), 5
ecdf.indices (CvM), 5
evaluator, 6
geodesic, 7
KS (CvM), 5
olive (Olive oil measurements), 8
Olive oil measurements, 8
pp, 9
search_geodesic (geodesic), 7