Package ‘pkmon’

October 22, 2018

Type Package

Title Least-Squares Estimator under k-Monotony Constraint for Discrete Functions

Version 1.0

Date 2018-10-18

Author Jade Giguelay

Maintainer Francois Deslandes <francois.deslandes@inra.fr>

Description We implement two least-squares estimators under k-monotony constraint using a method based on the Support Reduction Algorithm from Groeneboom et al (2008) <DOI:10.1111/j.1467-9469.2007.00588.x>. The first one is a projection estimator on the set of k-monotone discrete functions. The second one is a projection on the set of k-monotone discrete probabilities. This package provides functions to generate samples from the spline basis from Lefevre and Loisel (2013) <DOI:10.1239/jap/1378401239>, and from mixtures of splines.

License CC BY 4.0

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-22 08:10:03 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>pkmon-package</td>
<td>2</td>
</tr>
<tr>
<td>BaseNorm</td>
<td>5</td>
</tr>
<tr>
<td>Delta</td>
<td>6</td>
</tr>
<tr>
<td>estMonotone</td>
<td>7</td>
</tr>
<tr>
<td>kKnot</td>
<td>8</td>
</tr>
<tr>
<td>pEmp</td>
<td>9</td>
</tr>
<tr>
<td>Spline</td>
<td>10</td>
</tr>
</tbody>
</table>

Index 12
pkmon-package

Least-squares estimator under k-monotony constraint for discrete functions

Description

Description: This package implements two least-squares estimators under k-monotony constraint using a method based on the Support Reduction Algorithm from Groeneboom et al (2008). The first one is a projection estimator on the set of k-monotone discrete functions. The second one is a projection on the set of k-monotone discrete probabilities. This package provides functions to generate samples from the spline basis from Lefevre and Loisel (2013), and from mixtures of splines.

Details

Package: pkmon
Type: Package
Version: 0.9
Date: 2016-07-28
License: CC BY 4.0

Author(s)

Jade Giguelay <jade.giguelay@ens-paris-saclay.fr> http://maiage.jouy.inra.fr/jgiguelay

References


See Also

pMonotone, fMonotone

Examples

############################
# Example 1
# one triangular function T_j=Q_j^2, for j=supp=20 and for k=2 and k=3
n=30;
k1=2;
k2=3;
l=2;
supp=20;
p=dSpline(supp, k=1);
X=rSpline(n=n, supp, k=1);
ptilde=pEmp(X);
phat1=pMonotone(ptilde$freq, k=k1);
phat2=pMonotone(ptilde$freq, k=k2);

x.limits=c(0, max(supp+1, phat1$spi+1, phat2$spi+1));
y.limits=range(p, ptilde$freq, phat1$p, phat2$p);

plot(NULL, xlim=x.limits, ylim=y.limits, xlab="Counts", ylab="Frequencies");
points(0:supp, p, pch=16, col=1, lwd=2);
points(0:ptilde$supp, ptilde$freq, pch=4, col=2, lwd=2);
points(0:max(phat1$spi), phat1$p, pch=8, col=3, lwd=2);
points(0:max(phat2$spi), phat2$p, pch=2, col=4, lwd=2);
legend("topright", pch=c(16, 4, 8, 2), col=c(1, 2, 3, 4),
legend=c("p", expression(tilde("p")), expression(hat("p")*" - k = 2"),
expression(hat("p")*" - k = 3"));

# Example 2
# mixture of S splines Q_j^3 and for k=4 and k=3
n=30;
k1=4;
k2=3;
l=3;
supp=c(5, 10, 20);
prob=c(0.5, 0.3, 0.2);
p=dmixSpline(supp, k=1, prob=prob);
X=rmixSpline(n=n, supp, k=1, prob=prob);
ptilde=pEmp(X);
phat1=pMonotone(ptilde$freq, k=k1);
phat2=pMonotone(ptilde$freq, k=k2);

x.limits=c(0, max(supp+1, phat1$spi+1, phat2$spi+1));
y.limits=range(p, ptilde$freq, phat1$p, phat2$p);

plot(NULL, xlim=x.limits, ylim=y.limits, xlab="Counts", ylab="Frequencies");
points(0:supp, p, pch=16, col=1, lwd=2);
points(0:ptilde$supp, ptilde$freq, pch=4, col=2, lwd=2);
points(0:max(phat1$spi), phat1$p, pch=8, col=3, lwd=2);
points(0:max(phat2$spi), phat2$p, pch=2, col=4, lwd=2);
legend("topright", pch=c(16, 4, 8, 2), col=c(1, 2, 3, 4),
legend=c("p", expression(tilde("p")), expression(hat("p")*" - k = 4"),
expression(hat("p")*" - k = 3"));

########################################################################
# Example 3
# Poisson density
n=30;
k1=2;
k2=3;
supp=10;
p=dpois(0:supp, lambda=1);
X=rpois(n, lambda=1);
ptilde=pEmp(X);
phat1=pMonotone(ptilde$freq, k=k1);
phat2=pMonotone(ptilde$freq, k=k2);

x.limits=c(0, max(supp, phat1$Spi+1, phat2$Spi+1));
y.limits=range(p, ptilde$freq, phat1$p, phat2$p);

plot(NULL, xlim=x.limits, ylim=y.limits, xlab="Counts", ylab="Frequencies");
points(0:max(supp), p, pch=16, col=1, lwd=2);
points(ptilde$supp, ptilde$freq, pch=4, col=2, lwd=2);
points(0:max(phat1$Spi), phat1$p, pch=8, col=3, lwd=2);
points(0:max(phat2$Spi), phat2$p, pch=2, col=4, lwd=2);
legend("topright", pch=c(16, 4, 8, 2), col=c(1, 2, 3, 4),
      legend=c("p", expression(tilde("p")), expression(hat(p)* " - k = 2"),
      expression(hat(p)* " - k = 3"));

## Not run:

# Simulation for comparing ptilde and pHat (p is 3-monotone, k=3)
#

# cvge : percentage of non-convergence of the algorithm
# r.emp : L2-risk for the empirical estimator
# r.Hat : L2-risk for the estimator under k-monotony constraint

nSim=500;

n=30;
k=3;
l=3;
supp=20;
p=dSpline(supp, k=1);

result <- matrix(nrow=nSim,ncol=3);
dimnames(result)[[2]] <- c("cvge","r.emp","r.Hat");
for (i in 1:nSim) {
  X=rSpline(n=n, supp, k=l);
  ptilde=pEmp(X);
  phat=pMonotone(ptilde$freq, k=k);
  m <- max(supp+1,length(ptilde$freq)+1,phat$Spi+1)
  pV=c(p,rep(0,m-length(p)))
  phatV=c(phat$p,rep(0,m-length(phat$p)))
  ptildeV=c(ptilde$freq,rep(0,m-length(ptilde$freq)))
}
Computes the k-monotone discrete splines from Lefevre and Loisel (2013).

Usage

BaseNorm(k, J)

Arguments

k  Degree of monotony
J  maximum support of the splines

Value

matrix Q with J+1 rows and J+1 columns with \( Q(i, j) = Q^k(i - 1) = C_{j-i+k-1}^{k-1} \), where \( C \) represents the binomial coefficient.

Author(s)

Jade Giguelay <jade.giguelay@ens-paris-saclay.fr> http://maiage.jouy.inra.fr/jgiguelay

References


See Also

rSpline, dSpline, rmixSpline, dmixSpline
Examples

# Computing 3-monotone splines with maximum support 8
Q=BaseNorm(3, 8)
matplot(Q, type="l", main="3-monotone splines with maximum support 8");

<table>
<thead>
<tr>
<th>Delta</th>
<th>Discrete laplacian</th>
</tr>
</thead>
</table>

Description

Computes the laplacians of a discrete function

Usage

Delta(k, L, p)

Arguments

- k: Maximum order of the laplacian
- L: Support of the function
- p: Discrete function represented as a vector

Value

Returns a matrix with the laplacians \((-1)^j \Delta^j(p(l))\) of vector \(p\) for \(j\) in \(1, \ldots, k\) and \(l\) in \(0, \ldots, L\).

Author(s)

Jade Giguelay <jade.giguelay@ens-paris-saclay.fr> [http://maiage.jouy.inra.fr/jgiguelay](http://maiage.jouy.inra.fr/jgiguelay)

References


See Also

- kKnot

Examples

```
p=dSpline(k=3, supp=20)
M=Delta(3, 20, p)
```
Estimators of discrete probabilities under k-monotony constraint

Description

Estimators of discrete probabilities under k-monotony constraint. Estimation can be done on the set of k-monotone functions or on the set of k-monotone probabilities.

Usage

```r
pMonotone(ptild, t.zero = 1e-10, t.P = 1e-08, max.sn = NULL, k, verbose = FALSE)
```

```r
fMonotone(ptild, t.zero = 1e-10, t.P = 1e-08, max.sn = NULL, k, verbose = FALSE)
```

Arguments

- **ptild**: Empirical estimator
- **t.zero**: Threshold for the precision of the directionnal derivatives. (see OUTPUT below)
- **t.P**: Threshold for the precision on the stopping criterion. (see OUTPUT below)
- **max.sn**: The maximum support for the evaluation of the estimator
- **k**: Degree of monotony
- **verbose**: if TRUE, print for each iteration on the maximum support : pi, Psi and sumP (see OUTPUT below)

Details

The thresholds t.P and t.zero are used for the precision in the algorithm : in Step one (See REFERENCES below) the algorithm computes the directionnal derivatives of the current estimator and stops if all the directionnal derivarives are null that is to say if they are smaller than t.zero. In Step two (See REFERENCES below) the algorithm computes a stopping criterion and stops if and only if the stopping criterion is verified that is to say if some quantities are non-negative that is to say bigger than -t.P.

Value

- **cvge**: cvge = 0 if the criterion Psi decreases with the support of pi. cvge = 1 if Psi increases. cvge = 2 if maximum number of iterations reached
- **Spi**: Support of the positive measure pi at the last iteration
- **pi**: Values of the positive measure pi at the last iteration
- **p**: Values of pHat
- **Psi**: Scalar value of the criterion to be minimised
- **sumP**: sum(pHat) at convergence
- **history**: Data frame with components
- **L**: The maximum of the support of pi
- **Psi**: Value of the criterion for the value L
- **SumP**: Value of sum(pHat)
Author(s)

Jade Giguelay <jade.giguelay@ens-paris-saclay.fr> http://maiage.jouy.inra.fr/jgiguelay

References


See Also

pEmp, BaseNorm

Examples

```r
x=rSpline(n=50, 20, k=4)
pTild=pEmp(x);
res=pMonotone(pTild$freq, k=4)
```

<table>
<thead>
<tr>
<th>kKnot</th>
<th>k-Knot</th>
</tr>
</thead>
</table>

Description

k-Knots of a discrete function.

Usage

kKnot(p, k)

Arguments

- p: Vector
- k: Degree of the knots

Details

An integer i is a k-knot of p if $\Delta^kp(i) > 0$, where $\Delta^k$ is the k-th Laplacian of the sequence p.

Value

Vector with the k-knots of p.

Author(s)

Jade Giguelay <jade.giguelay@ens-paris-saclay.fr> http://maiage.jouy.inra.fr/jgiguelay
References


See Also

Delta

Examples

```r
p = dmixSpline(c(5, 10, 20), k=3, c(0.5, 0.25, 0.25))
knots = kKnot(p, 3)
```

---

**pEmp**

*Empirical estimator of a discrete function*

**Description**

Empirical estimator of a discrete function

**Usage**

`pEmp(x)`

**Arguments**

`x`

A random sample from a discrete probability.

**Details**

The empirical estimator is defined as

\[ p(j) = \sum_{i=1}^{n} \mathbf{1}_{x_i = j}. \]

**Value**

<table>
<thead>
<tr>
<th>support</th>
<th>The points of the support of the estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>The counts of the sample</td>
</tr>
<tr>
<td>freq</td>
<td>The normalized counts</td>
</tr>
</tbody>
</table>

**Author(s)**

Jade Giguelay <jade.giguelay@ens-paris-saclay.fr> [http://maiage.jouy.inra.fr/jgiguelay](http://maiage.jouy.inra.fr/jgiguelay)

**Examples**

```r
x = rpois(100, lambda=0.3)
ptild = pEmp(x)
```
Spline

Random generation and distribution function of k-monotone densities

Description

Random generation and distribution function for the spline of the basis from Lefevre and Loisel (2013), and mixtures of splines.

Usage

rspline(n=1, supp, k)
dSpline(supp, k)
rmixSpline(n=1, supp, k, prob)
dmixSpline(supp, k, prob)

Arguments

supp Support of the spline, or vector of the supports of the splines for the mixture of splines
n Number of random values to return
k Degree of monotony
prob Vector of probabilities for the mixture of splines

Details

See BaseNorm for details on the spline basis.

Value

rspline and rmixSpline generates random deviates from the splines and mixtures of splines.
dSpline and dmixSpline gives the distribution function.

Author(s)

Jade Giguelay <jade.giguelay@ens-paris-saclay.fr> http://maiage.jouy.inra.fr/jgiguelay

References


See Also

pEmp
Examples

```r
x <- rSpline(n=100, 20, 3)
p <- dSpline(20, 3)
xmix <- rmixSpline(n=100, c(5, 20), 3, c(0.5, 0.5))
pxmix <- dmixSpline(c(5, 20), 3, c(0.5, 0.5))
par(mfrow=c(1, 2))
hist(x, freq=FALSE)
lines(p, col="red")
hist(xmix, freq=FALSE)
lines(pmix, col="red")
```
Index

*Topic package
  pkmon-package, 2

BaseNorm, 5, 8, 10

Delta, 6, 9
dmixSpline, 5
dmixSpline (Spline), 10
dSpline, 5
dSpline (Spline), 10

estMonotone, 7

fMonotone, 2
fMonotone (estMonotone), 7

kKnot, 6, 8

pEmp, 8, 9, 10
pkmon (pkmon-package), 2
pkmon-package, 2
pMonotone, 2
pMonotone (estMonotone), 7

rmixSpline, 5
rmixSpline (Spline), 10
rSpline, 5
rSpline (Spline), 10

Spline, 10