Package 'maxLinear'

January 21, 2013

Type Package

Title Conditional Samplings for Max-Linear Models

Version 1.1

Date 2013-01-21

Author Yizao Wang

Maintainer Yizao Wang <yizao.wang@uc.edu>

Description Computational tools for conditional samplings from max-linear models.

License GPL>=2

LazyLoad yes

URL http://www.stat.lsa.umich.edu/~yizwang/software/maxLinear/maxLinear.html

Archs i386

R topics documented:

`maxLinear-package` *Conditional Sampling Tools for Max-Linear Models.*

Description

This package provides efficient computational tools for conditional samplings in max-linear models, which include a large class of max-linear random fields.

Details

Package:	maxLinear
Type:	Package
Version:	1.0
Date:	2010-04-20
License:	GPL>=2
LazyLoad:	yes

Author(s)

Yizao Wang

Maintainer: Yizao Wang <yizwang@umich.edu>

References

Yizao Wang and Stilian Stoev, Conditional Samplings for Max–Stable Random Fields. Technical Report, Department of Statistics, University of Michigan. 2010

Examples

```
n = 10; p = 100; N = 100;

## randomly generate a max-linear model.
A = matrix(runif(n*p),nrow = n) * 5;

## generate p 1-Frechet random variables.
Z = rAF(n = p);

## generate one observation from the max-linear model.
X = maxLinear(A,Z);

## get upper boundary for Z's
z = maxLinear.zHat(A,X);

## get hitting distribution HD
H = maxLinear.HD(A,X,z);

## get hitting scenarios HS
HS = maxLinear.HS(H,z,alpha,para);

## generate N conditional samplings
## based on the observation X, w.r.t. the model A.
sampledZp = maxLinear.CS(HS, z, N);
```

`grid.axis` *Generate axis for rectangle grids.*

Description

Generate axis for arbitrary rectangle grids.

Usage

```
grid.axis(grid_leftbottom, gridSize, gridNumber)
```

Arguments

grid_leftbottom	The left-bottom location of the grid.
gridSize	The lengths of horizontal and the vertical ranges of the grid. Saved in a 2-vector.
gridNumber	The number of points in each horizontal line and vertical line of the grids. Saved in a 2-vector.

Value

A list (say axis) of two vectors (axis$x and axis$y) is returned.

Author(s)

Yizao Wang

See Also

grid.locations

Examples

```
## Generate a grid of size 40 by 40 with
## left-bottom location (-2,-2)
## right-top location (1.9,1.9)

T_leftbottom = c(-2,-2); TSize = c(4,4); gridNb = c(40,40);
grid = grid.axis(T_leftbottom, TSize, gridNb);
```

---

grid.HP2d *Generate de Haan-Pereira model.*

Description

Generate a discretized version of de Haan-Pereira model. See return values.

Usage

```
grid.HP2d(rho, beta1, beta2, obs, HPgrid, HPgridSize, alpha)
```

Arguments

rho	Parameter for the de Haan-Pereira model.
beta1	Parameter for the de Haan-Pereira model.
beta2	Parameter for the de Haan-Pereira model.
obs	A 2 by n matrix representing the locations of n observations.
HPgrid	A 2 by p matrix representing the grids in the discretized de Haan-Pereira model.
HPgridSize	The size of a single block region in the grid.
alpha	The α-Frechet index.

Value

The value returned is a matrix A. This corresponds to a discretized (and truncated) version of the de Haan-Pereira model:

$$Z(t_1, t_2) = \max_{j \geq 1} \frac{\phi(X_j - t_1, W_j - t_2)}{Y_j} \text{ for } (t_1, t_2) \in R^2$$

with

$$\phi(t_1, t_2) := \frac{\beta_1 \beta_2}{2\pi\sqrt{1-\rho^2}} \exp\left\{ -\frac{1}{2(1-\rho^2)} \left[\beta_1^2 t_1^2 - 2\rho\beta_1\beta_2 t_1 t_2 + \beta_2^2 t_2^2\right] \right\}.$$

Here, h^2 equals `HPgridSize`, and $(X_i, W_i, Y_i)_{i \geq 1}$ are the points of a homogeneous Poisson point process on $R^2 \times R_+$.

To obtain a discretized version of the de Haan-Pereira model, we restrict to the region $[-M, M]^2$ and consider a uniform grid of size $h = M/q,\ q \in N$:

$$X(t_1, t_2) = \bigvee_{-q \leq j_1, j_2 \leq q-1} h^{2/\alpha} \phi(t_1 - u_{j_1}, t_2 - u_{j_2}) Z_{j_1, j_2},$$

where Z_{j_1,j_2} are i.i.d. α-Frechet random variables.

In practice, the locations of grid are saved in `gridHP`, a $2 \times p$ matrix. The observations are saved in `obs`, a $2 \times n$ matrix. Then, the return value A is an $n \times p$ matrix, with the i-th row, corresponding to the i-th observation, equal to

$$A_{i,j} = h^{2/\alpha} \phi(t_i^{obs}(1) - u_j(1), t_i^{obs}(2) - u_j(2)).$$

Here, $(t_i^{obs}(1), t_i^{obs}(2))$ corresponds to the location of the i-th observation, and $(u_j(1), u_j(2))$ corresponds to the location of the j-th point in the grid. Note that the points in grid, obtained from `grid.locations`, is in lexicographical order. See `grid.locations`.

Author(s)

Yizao Wang

References

Laurens de Haan and Teresa T. Pereira. Spatial Extremes: Models for the Stationary Case. The Annals of Statistics, 2006, Vol. 34, No. 1, 146–168.

See Also

`grid.locations`

Examples

```
## simulate from the discrete de Haan-Pereira model.

rho = -0.8; beta1 = 1.5; beta2 = 0.7;    ## parameters
alpha = 1; para = 1;

## HP grid
HPGrid_leftbottom = c(-4,-4); HPGridSize = c(8,8); HPGridNb = c(40,40);
disGrid_leftbottom = c(-2,-2); disGridSize = c(4,4); disGridNb = c(80,80);

p = HPGridNb[1]*HPGridNb[2];  ## number of Z's
```

```
HPGrid = grid.locations(HPGrid_leftbottom, HPGridSize, HPGridNb);
HPGridRegionSize = HPGridSize[1]*HPGridSize[2]/(HPGridNb[1]*HPGridNb[2]);
HPShift = 0.5*HPGridSize/HPGridNb;

## display grid
disGridAxis = grid.axis(disGrid_leftbottom, disGridSize, disGridNb);
disGrid = grid.locations(disGrid_leftbottom, disGridSize, disGridNb);
disShift = 0.5*disGridSize/disGridNb;

## generate the discretized de Haan-Pereira model.
A = grid.HP2d(rho,beta1,beta2,
              disGrid+disShift,HPGrid+HPShift,HPGridRegionSize,alpha);

Z = rAF(alpha = alpha, sigma = para, n = p);                    ## generate a sample.
X = maxLinear(A,Z);                                             ##
X = matrix(X, nrow = disGridNb[1], ncol = disGridNb[2]);

X11();
image(disGridAxis$x, disGridAxis$y, X, col = topo.colors(100), ## plot
      xlab = "", ylab = "",
      xlim = c(disGridAxis$x[1],disGridAxis$x[disGridNb[1]]),
      ylim = c(disGridAxis$y[1],disGridAxis$y[disGridNb[2]]))

contour(disGridAxis$x,disGridAxis$y, X, nlevel = 5,lwd = 2,
        axes = FALSE, add = TRUE);

mainExpr = "Sampling from the de Haan-Pereira model"
subExpr = substitute("Parameters:" * rho * "=" * rh * ","
                     * beta[1] * "=" * b1 * "," * beta[2] * "=" * b2,
                     list(rh = rho, b1 = beta1,b2 = beta2));
title(main = mainExpr, sub = subExpr, cex = 0.5,font = 1)
```

`grid.locations` *Generate locations for rectangle grids.*

Description

Generate locations for arbitrary rectangle grids.

Usage

```
grid.locations(grid_leftbottom, gridSize, gridNumber)
```

Arguments

`grid_leftbottom`	The left-bottom location of the grid.
`gridSize`	The lengths of horizontal and the vertical ranges of the grid. Saved in a 2-vector.
`gridNumber`	The number of points in each horizontal line and vertical line of the grids. Saved in a 2-vector.

Value

A 2 by p matrix is returned with p equals `gridNumber[1]` times `gridNumber[2]`. The grid locations are saved in the lexicographically increasing order. That is, in the order of

```
(1,1), (1,2), ... , (1,gridNumber[1]),
(2,1), (2,2), ... , (2,gridNumber[1]),
(3,1), ...
       ...
(gridNumber[2],1), ... , (gridNumber[1],gridNumber[1]).
```

Author(s)

Yizao Wang

See Also

`grid.axis`

Examples

```
## Generate a grid of size 40 by 40 with
## left-bottom location (-2,-2)
## right-top location (1.9,1.9)

T_leftbottom = c(-2,-2); TSize = c(4,4); gridNb = c(40,40);
grid = grid.locations(T_leftbottom, TSize, gridNb);
```

`maxLinear.CS` *Conditional Sampling for Max-Linear Models.*

Description

Given the max-linear model

$$\mathbf{X} = A \odot \mathbf{Z}$$

resampling $\mathbf{Z}$ according to the conditional distribution

$$\mathbf{Z}|\mathbf{X} = \mathbf{x}$$

for some observation $\mathbf{x}$. See Details below.

Usage

```
maxLinear.zHat(A, X)
maxLinear.HD(A, X, z, error = 10^(-15))
maxLinear.HS(H, z, alpha = 1, para = 1)
maxLinear.CS(HS, z, N, alpha = 1, para = 1)
```

Arguments

A	An $n \times p$ matrix with non-negative entries.
X	A vector of size n. An observation for the max-linear model $A \odot \mathbf{Z}$.
z	Upper boundary of $\mathbf{Z}$'s for the max-linear model $A \odot \mathbf{Z}$, with observation $\mathbf{X}$.
error	Numerical error control.
H	An n by p zero-one matrix.
HS	The hitting matrix (with blocking structure) corresponding to A and $\mathbf{X}$.
para	Parameters of $\mathbf{Z}$'s in the max-linear model. A vector of size p. By default equals 1.
alpha	α-Frechet.
N	Number of conditional samplings.

Value

The functions are realizations of `Algorithm II` in *Wang and Stoev, 2010.*

`maxLinear.zHat(A,X)` returns a p-vector $\widehat{\mathbf{z}}$, with

$$\widehat{z}_j = \min_{1 \le i \le n} x_i / a_{i,j},$$

which is the necessary upper bound of Z_j, given $\mathbf{X}$.

`maxLinear.HD(A,X,z)` returns an $n \times p$ matrix, corresponding to the hitting matrix H defined by

$$h_{i,j} = 1 \text{ if } a_{i,j}\widehat{z}_j = x_i \text{ and } 0 \text{ otherwise.}$$

In practice, instead of checking $a_{i,j}\widehat{z}_j = x_i$, we set $h_{i,j} = 1$ if $|a_{i,j}\widehat{z}_j - x_i| <$`error`.

`maxLinear.HS(H)` returns a class, saving the structure of $(I_s, J^{(s)}, \overline{J}^{(s)})_{1 \le s \le r}$ as well as the weights. See *Wang and Stoev, 2010* for details.

`maxLinear.CS(HS, z, N, para, alpha , mode )` returns an `N` by p matrix. Each row is an independent sample $\mathbf{Z}$ with the desired conditional distribution $\mathbf{Z}|\mathbf{X}$.

Author(s)

Yizao Wang

References

Yizao Wang and Stilian Stoev, Conditonal Sampling for Max–Linear Random Fields. 2010.

See Also

`grid.HP2d`

Examples

```
############################
## a simple illustration ##
############################

n = 10; p = 100; N = 100;
A = matrix(runif(n*p),nrow = n) * 5; ## randomly generate a max-linear model.
```

```
Z = rAF(n = p);                     ## generate p 1-Frechet random variables.
X = maxLinear(A,Z);                 ## generate one observation
                                    ## for the max-linear model.


z = maxLinear.zHat(A,X);            ## get upper boundary for Z's
H = maxLinear.HD(A,X,z);            ## get hitting distribution HD
HS = maxLinear.HS(H,z,alpha,para);  ## get hitting scenarios HS

sampledZp = maxLinear.CS(HS, z, N); ## generate N conditional samplings
                                    ## based on the observation X,
                                    ## w.r.t. the model A.

############################################################
## Conditional Sampling from the de Haan-Pereira model ##
############################################################

#################
## preparation ##
#################
rho = -0.8; beta1 = 1.5; beta2 = 0.7;  ## parameters of de Haan-Pereira model
alpha = 1; para = 1;
nA = 7;                                ## number of observations

## the range of HP grid and prediction grid
HPGrid_leftbottom = c(-4,-4); HPGridSize = c(8,8); HPGridNb = c(50,50);
pred_leftbottom = c(-2,-2); predGridSize = c(4,4); predGridNb = c(50,50);

## observation locations
obsGrid = matrix(0,nrow = 2,ncol = nA);
obsGrid[1,] = 0.35 * predGridSize[1] * cos(2 * pi * seq(1,nA)/nA);
obsGrid[2,] = 0.35 * predGridSize[2] * sin(2 * pi * seq(1,nA)/nA);

## the range of domain of discretization of the HP model
HPGrid = grid.locations(HPGrid_leftbottom, HPGridSize, HPGridNb);
HPGridRegionSize = HPGridSize[1]*HPGridSize[2]/(HPGridNb[1]*HPGridNb[2]);
HPShift = 0.5*HPGridSize/HPGridNb;

## the range of prediction grid
predGridAxis = grid.axis(pred_leftbottom, predGridSize, predGridNb);
predGrid = grid.locations(pred_leftbottom, predGridSize, predGridNb);
predShift = 0.5*predGridSize/predGridNb;

## grid for HP model: Creating a uniform grid
A = grid.HP2d(rho,beta1,beta2,
              obsGrid+predShift,HPGrid+HPShift,HPGridRegionSize, alpha);
B = grid.HP2d(rho,beta1,beta2,
              predGrid+predShift,HPGrid+HPShift,HPGridRegionSize,alpha);


## fix abritrary observations
X = array(5,nA);

#########################
## conditional sampling ##
#########################
z = maxLinear.zHat(A,X);              ## obtain zhat.
```

```
H = maxLinear.HD(A,X,z);               ## Obtain hitting matrix.
HS = maxLinear.HS(H,z,alpha,para);     ## Obtain hitting scenarios.

## conditional sampling
sampledZp = maxLinear.CS(HS, z, 1, alpha ,para);
Y = maxLinear(B,sampledZp); Y = matrix(Y, nrow = predGridNb[1]);

##########
## plot ##
##########
image(predGridAxis$x, predGridAxis$y, Y, col = topo.colors(100), ## plot
      xlab = "", ylab = "",
      xlim = c(predGridAxis$x[1],predGridAxis$x[predGridNb[1]]),
      ylim = c(predGridAxis$y[1],predGridAxis$y[predGridNb[2]]))
contour(predGridAxis$x,predGridAxis$y, Y, nlevel = 5,lwd = 2,
        axes = FALSE, add = TRUE);
points(obsGrid[1,],obsGrid[2,],
       pch= 4, col = "red", cex = 2, lwd = 2, xlab = "", ylab = "");
text(obsGrid[1,], obsGrid[2,],
     floor(X*100)/100, pos = 4, col = "red", xlab = "", ylab = "");

mainExpr = "Conditional sampling from the de Haan-Pereira model"
subExpr = substitute("Parameters:" * rho * "=" * rh * ","
                     * beta[1] * "=" * b1 * "," * beta[2] * "=" * b2,
                     list(rh = rho, b1 = beta1,b2 = beta2));
title(main = mainExpr, sub = subExpr, cex = 0.5,font = 1)
```

---

rAF *Generate Random Variables from the alpha-Frechet Distribution.*

Description

Generate random variables from the α-Frechet distribution

$$P(Z \leq z) = \exp\{-\sigma^{\alpha} z^{-\alpha}\}$$

and the conditional α-Frechet distribution

$$P(Z \leq z | Z \leq c) = \exp\{-\sigma^{\alpha}(z^{-\alpha} - c^{-\alpha})\}, \forall 0 < z \leq c\,.$$

Usage

```
rAF(alpha = 1, sigma = 1, n = 1)
rCAF(alpha = 1, sigma = 1, n = 1,cc)
```

Arguments

alpha	Parameter in the α-Frechet distribution.
sigma	Parameter in the α-Frechet distribution.
cc	Parameter in the conditional α-Frechet distribution.
n	Number of independent copies to return.

Value

Return n independent samples from the required α-Frechet distribution.

Author(s)

Yizao Wang

See Also

maxLinear.CS

Index