Package ‘EnviroStat’

June 3, 2015

Type Package

Title Statistical Analysis of Environmental Space-Time Processes

Version 0.4-2

Date 2015-06-03

Author Nhu Le, Jim Zidek, Rick White, and Davor Cubranic, with Fortran code for Sampson-Guttorp estimation authored by Paul D. Sampson, Peter Guttorp, Wendy Meiring, and Catherine Hurley, and Runge-Kutta-Fehlberg method implementation by H.A. Watts and L.F. Shampine.

Maintainer Davor Cubranic <cubranic@stat.ubc.ca>

Description
Functions and datasets to support the book by Nhu D Le and James V Zidek, Springer (2006).

License AGPL (>= 3)

Imports MASS

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-06-03 22:38:07

R topics documented:

`EnviroStat-package` *Statistical analysis of environmental space-time processes*

Description

EnviroStat provides functions for spatio-temporal modeling of environmental processes and designing monitoring networks for them based on an approach described in Le and Zidek (2006).

Details

The modeling approach offered by this package has a number of features:

- Conditional on knowing the process parameters the environmental process is assumed to have (after a suitable transformation if necessary) to be a Gaussian random field (GRF).
- At every spatial location, the process can yield a multiplicity of random responses such as air pollutant concentrations.
- The approach used in the package lies within a Bayesian hierarchical modeling framework However for computational expediency empirical shortcuts are made at higher levels of the hierarchical setup. Thus for example most hyperparameters are fitted using a type II maximum likelihood approach, eliminating the need for the the user to specify them. Thus the package can handle large fields of monitoring networks, say with 600 or more spatial sites.
- The approach does not assume a stationary GRF. Instead it takes a nonparametric approach where the spatial covariance matrix is left completely unspecified and instead endowed with a prior distribution with a hypercovariance matrix that can be modeled at level two of the hierarchy, making the method quite robust against non-stationarity in the random field.
- It presents a approach for designing monitoring networks based on the well-known warping method of Sampson and Guttorp (1992) as developed with Wendy Meiring.
- It allows for missing data, providing that these data are missing in blocs of time, which after a regional trend is fitted, then become exchangeable. For then the blocs of residuals can be permuted the get a decreasing or increasing staircase pattern in the data matrix something that is required in the approach.
- It has been empirically assessed in a number of major applications and found to yield well calibrated prediction intervals. For example, a 95% interval will cover their predictands about 95% of the time.

Author(s)

Nhu Le, Jim Zidek, Rick White, and Davor Cubranic.

Fortran code for Sampson-Guttorp estimation authored by Paul D. Sampson, Peter Guttorp, Wendy Meiring, and Catherine Hurley.

Maintainer: Davor Cubranic <cubranic@stat.ubc.ca>

References

Le, Nhu D. and James V. Zidek. Statistical Analysis of Environmental Space-Time Processes. Springer, New York, 2006.

See Also

See the package vignette for a guided example of complete analysis using the package and the manual for details of individual functions.

---

bgrid *Create a bi-orthogonal grid*

Description

Function to create a biorthogonal grid using the thin-plate solution; the grid object created can be used with the `draw` function to plot the grid.

Usage

```
bgrid(start, xmat, coef, xlimt, iter.limit = 10, perpc = 8)
```

Arguments

`start` coordinates of the starting point to create the grid - usually near the center of region

`xmat` $n \times 2$ matrix containing xy coordinates of locations

`coef` coefficients of the thin-plate solution, for instance as returned by `sinterp`

`xlimt` vector of 4 elements, representing x- and y-ranges for the grid to be created; if not provided, the ranges of `xmat` are used.

`iter.limit` limit of iterations

`perpc` parameter to control spacing of the grid

Value

A list with following named components:

`grid` coordinates of points in the grid

`ngrid` number of points in the grid

`fldmag` the "gradient" index (see Sampson and Guttorp, 1992, for details)

`nn` internally set for the maximum number of grid points allowed

References

Sampson P.D., and Guttorp, P., "Nonparametric estimation of nonstationary spatial covariance structure", in Journal of the American Statistical Association, vol 87, pp 108-119, 1992.

See Also

`draw`, `sinterp`

`comb.all` *Enumerate all combinations*

Description

Function to enumerate all combinations of size 'k' from the set 1,.., n.

Usage

```
comb.all(n, k)
```

Arguments

`n`	number of elements in the set
`k`	number of elements to choose for each combination

Value

An (n-choose-k) by k matrix with one combination per row.

Examples

```
comb.all(5, 3)
```

`corrfit` *Estimate location correlations*

Description

Function to estimate correlations between all the locations(new+stations) using the results of the SG step

Usage

```
corrfit(crds, Tspline, sg.fit, model = 1)
```

Arguments

`crds` coordinates of all locations beginning with new locations

`Tspline` the thin-spline fit from the SG-steps

`sg.fit` the mapping resulted from the SG method

`model` variogram model; 1: exponential 2: gaussian

Value

correlation matrix among the locations

`Disp.link` *Function to plot and link points in a dispersion-distance plot and a geographic map.*

Description

User may identify points on the dispersion scatter in order to identify line segments on the coordinate plot, **and/or** identify individual sites on the coordinate plot in order to highlight the corresponding set of points on the dispersion scatter.

Usage

```
Disp.link(disp.mx, coords, dmap, ddisp, names, device = getOption('device'))
```

Arguments

`disp.mx` nxn matrix of dispersions (i.e. Var(Z(x)-Z(y))

`coords` nx2 matrix of coordinates

`dmap` device number for existing window to be used for map (optional)

`ddisp` device number for existing window to be used for dispersion plot (optional)

`names` not yet used, but might be used for labelling

`device` a character string giving the name of a function, or the function object itself, which when called creates a new graphics device

Value

Indices of station pairs selected in the dispersion plot, and indices of individual stations selected on the coordinate plot.

Note

Uses `setplot` to set up coordinates for geographic map and `Fdist` to compute distances from coords.

draw *Plot the biorthogonal grid*

Description

Function to plot the biorthogonal grid created by the bgrid function

Usage

```
draw(data, fs = FALSE, lwidth = c(1, 1), lcolor = c(1, 1),
     cutpts, limits = FALSE, optlist, pts = FALSE)
```

Arguments

data	grid object, typically as returned by the bgrid function
fs	indicator of whether non-solid lines can be used
lwidth	range of line widths to be used.
lcolor	range for colours
cutpts	cutpoints; to be created from data if not provided
limits	indicator if lower and upper limits are added to the cutpts.
optlist	list containing values of options including lwidth, lcolor, cutpts, limits; values are obtained from data if not provided.
pts	indicator whether points are also plotted.

Value

Besides drawing the grid, the function also returns optlist.

Note

See Sampson+Guttorp (1992) for detail on interpretation of the plotted grid, (E.g., solid lines indicate contraction and dashed lines indicate expansion.)

References

Sampson P.D., and Guttorp, P., "Nonparametric estimation of nonstationary spatial covariance structure", in Journal of the American Statistical Association, vol 87, pp 108-119, 1992.

See Also

bgrid

Falternate3 *Simultaneous coordinate and variogram estimation*

Description

Perform simultaneous estimation of coords and exponential or gaussian variogram by alternating weighted least squares.

Usage

```
Falternate3(disp, coords, model = 1, a0 = 0.1, t0 = 0.5,
            max.iter = 50, max.fcal = 100, alter.lim = 50,
            tol = 1e-05, prt = 0, dims = 2, lambda = 0,
            ncoords, dev.mon = NULL, verbose = FALSE)
```

Arguments

`disp`	$n \times n$ dispersion matrix, equal to $2 - 2(spatial correlation matrix)$
`coords`	$n \times 2$ coordinate matrix
`model`	type of variogram: 1 for exponential or 2 for gaussian
`a0, t0`	initial variogram parameter estimates
`max.iter, max.fcal`	control parameter for calls to non-linear optimization routines (same values used in MDS step and in variogram step)
`alter.lim`	maximum number of iterations of pairs of alternating calls to `Fmdsfit3` (coordinate estimation) and `Fvariogfit3` (variogram fitting)
`tol`	convergence criterion for coordinate estimates
`prt`	not used
`dims`	dimension of multi-dimensional scaling
`lambda`	smoothing parameter
`ncoords`	$n \times 2$ optional initial coordinates to use if not G-plane
`dev.mon`	Function to open the graphics device used for plots monitoring the convergence of objective. If `NULL`, monitoring plots will not be created.
`verbose`	if `TRUE`, display the results of each iteration to the console

Details

This version permits dimension > 2 for scaling. In the plotting we'll use a plot symbol proportional to the third coordinate.

Value

A list containing the following named components:

`variogfit`	Fitted variogram parameters with new locations
`ncoords`	$n \times 2$ matrix with coordinates of new locations

Warning

make sure that coords are scaled reasonably small before attempting to compute; otherwise matrix inversion is likely not to work in calculation of bending energy matrix.

Note

This version also passes a smoothing parameter to the optimization. This parameter probably is not scaled exactly the same as it is in sinterp and this has not been investigated yet.

`Fdist` *Interpoint distances*

Description

Function to compute interpoint distances for an $n \times p$ coordinate matrix.

Usage

```
Fdist(crds)
```

Arguments

`crds` $n \times p$ matrix of n locations, each with p coordinates

Value

$n \times n$ matrix of interdistances; (i, j) element is the distance between locations i and j

`Flamb2` *Lambert projection*

Description

Projects the geo-coordinates into rectangular ones using the Lambert projection.

Usage

```
Flamb2(geoconfig, latrf1 = NA, latrf2 = NA, latref = NA, lngref = NA)
```

Arguments

`geoconfig` $n \times 2$ matrix, containing geo-coordinates in format (`lat, -long`, in degrees) of n locations.

`latref, lngref` latitute and -longitude of the reference point (mid-point if not provided)

`latrf1, latrf2` range of latitudes used in the projection (default: `latref` $+/-.3*range(latitudes)$)

Value

A list containing the following named components:

xy $n \times 2$ matrix of Lambert projection coordinates

latref, lngref

latrf1, latrf2

Fmgrid *Generate a grid of points*

Description

Function to generate points on a grid. Points are assembled in an $n \times 2$ matrix with NA's separating series of vertical and horizontal lines in the grid.

Usage

```
Fmgrid(xlim, ylim, xn = 8, xspace, xres, yn = 8, yspace, yres)
```

Arguments

xlim range in x-axis

ylim range in y-axis

xn, yn the number of vertical and horizontal lines, respectively. This parameter is overridden by xspace and yspace if specified.

xspace, yspace the distance between successive vertical and horizontal lines, respectively

xres, yres the distance between points generated along horizontal and vertical lines, respectively; if xres and yres are not specified, then points are generated only at the nodes of intersection of the vertical and horizontal lines. Note that these nodes appear in duplicate as sequences of points are generated first for the vertical lines and then for the horizontal lines.

Value

A list containing the following named components:

grid coordinates of grid points

rl.ind Indicator of points generated from vertical or horizontal line

Ftransdraw *Interactively choose smoothing parameter value*

Description

An interactive function showing the fitted variogram and the mapping transformation from the geographical space into D-space, allowing the user to interactively choose a suitable value for the smoothing parameter ("lambda").

Usage

```
Ftransdraw(disp, Gcrds, MDScrds, gridstr, sta.names, lambda = 0,
           lsq = FALSE, eye, model = 1, a0 = 0.1, t0 = 0.5)
```

Arguments

`disp`	$n \times n$ dispersion matrix, equal to $2 - 2(spatialcorrelationmatrix)$
`Gcrds`	$n\ times2$ coordinate matrix (in G-space)
`MDScrds`	Coordinates of new locations – as calculated by `Falternate3`
`gridstr`	grid coordinates, obtained from `Fmgrid` using `Gcrds` as input
`sta.names`	names of locations; if not provided, locations will be numbered 1 to n
`lambda`	initial value of the smoothing parameter
`lsq`	logical flag used in Sampson-Guttorp method
`eye`	eye perspective; if not provided, locations will be selected using the provided data
`model`	type of variogram: 1 for exponential or 2 for gaussian
`a0, t0`	initial variogram parameter estimates

Value

A list containing the following named components:

`Dcrds`	$n \times 2$ cooridnate matrix of new locations in D-space
`Ddist`	$n \times n$ matrix of interdistances in D-space

Note

The graphic device has to be setup before engaging this function, for example using the 'setplot' function:

```
par(mfrow=c(1,2))
temp <- setplot(coords.lamb, ax=TRUE)
```

In window, after entering the function call in R console, click on the graphical device to register the curse before proceeding and this interactive function will provide instructions for using a different smoothing parameter value; ie. enter a new lambda value in R console, then click on the graphical device to see results.

Fvariogfit3 *Fit an exponential or gaussian variogram*

Description

Fit an exponential or gaussian variogram with interdistances and corresponding dispersion values.

Usage

```
Fvariogfit3(disp.lt, h.lt, model = 1, a0 = 0.1, t0 = 0.5,
            max.iter = 25, max.fcal = 100, bep = 0, verbose = FALSE)
```

Arguments

disp.lt	vector of spatial dispersion values between locations (one value for each pair)
h.lt	vector of corresponding interdistances
model	type of variogram: 1 for exponential or 2 for gaussian
a0, t0	initial variogram parameter estimates
max.iter, max.fcal	control parameters for calls to non-linear optimization routines
bep	bending energy penalty, described in Sampson-Guttorp method (JASA 1992)
verbose	if TRUE, display fitting details to the console

Details

Exponential variogram is defined as $a_1 + (2 - a_1)(1 - exp(-t_0 h))$. Gaussian variogram is defined as $a_1 + (2 - a_1)(1 - exp(-t_0 h^2))$.

Value

A list containing the following named components:

objf	sum of residual sum of squares, plus bending energy penalty
a, t0	fitted parameters
fit	fitted values of variogram

`ldet.eval` *Log determinant of sub-covariances*

Description

Function to calculate the log determinant off all sub-covariance matrices of size (k x k) from a covariance matrix.

Usage

```
ldet.eval(covmat, k, all = FALSE)
```

Arguments

`covmat` a covariance matrix (ie. non-negative definite, square and symmetric)

`k` dimension of sub-covariance matrices considered

`all` if `TRUE`, returns all combinations with corresponding $log|det|$

Value

`coord.sel` The k coordinates having the largest $log|det|$

`log.det` The $log|det|$ of the submatrix corresponding the `coord.sel`

`all.comb` all combinations and their $log|det|$ if `all = TRUE`

; NULL otherwise.

Note

Setting `all = TRUE` may need additionally a large amount of memory and so may not work for a large number of combinations!!

`location.NY` *Locations of New York ozone measuring stations.*

Description

The `location.NY` data frame consists of geographical coordinates of nine stations in New York State numbered for our purposes from 1 to 9.

Usage

```
data(location.NY)
```

Format

A data frame with 9 observations on the following 2 variables.

`lat` Station latitude

`long` Station longitude

See Also

This dataset accompanies the ozone measurements in ozone.NY.

Examples

```
data(location.NY)
plot(location.NY[,1], location.NY[,2])
```

---

ozone.NY *New York ozone concentration levels*

Description

The `ozone.NY` data frame consists of hourly O_3 concentration levels (ppb) from nine stations in New York State numbered for our purposes from 1 to 9. These data were originally downloaded from the EPA's AIRS air quality site (Environmental Protection Agency 2013). Furthermore, preliminay analysis suggested a sqrt transformation of the original concentrations to symmetrize the data distribution.

Usage

```
data(ozone.NY)
```

Format

Each row of the data set represents a daily record starting at April 1, 1995, and ending at September 30, 1995 (183 days). Measurements are recorded by station in columns, with four columns per station (for hours 8–12); there are nine stations, so there are 36 columns of measurements in total.

The last six stations have no missing observations while stations 1, 2, 3 have 2616, 2016, and 72 missing hourly observations, respectively.

`month` month of the measurement, numeric 4–9

`weekday` day of the week, numeric 2–8

`sqO3.1`–`sqO3.4` square root of the O_3 measurement for station 1 for hours 8–12 (8am to noon).

`...`

`sqO3.33`–`sqO3.36` square root of the O_3 measurement for station 9 for hours 8–12 (8am to noon).

Source

US Environmental Protection Agency's AIRS air quality site (2013), `http://www.epa.gov/ttn/airs/airsaqs/`.

pred.dist.simul *Simulate from the predictive distribution*

Description

Simulate N- replicates from the predictive distribution for a given time point (tpt) from 1 to n (length of the data).

Usage

```
pred.dist.simul(hyperest, tpt, include.obs = T, N = 1)
```

Arguments

hyperest	Output from the staircase.hyper.est functions, containing estimates of all hyperparameters
tpt	A specific time point - from 1 to n corresponding to the number of time points from the data set
include.obs	If TRUE, the observed data for time tpt, are also returned
N	Number of replicates

Value

A matrix with N rows; the number of columns depends on whether the observed data are returned

The columns are organized consistent with the observed data (ie. $u \times p$ ungauged blocks, $g1 \times p$, $g_2 \times p, \ldots$)

Note

This function could be slow if there are missing data at gauged sites correspondind to the selected time point. That is, it is fastest at time points corresponding to Block 1 and slower with higher blocks.

setplot *Set up device for locations*

Description

Function to setup the graphical device to cover the range of locations.

Usage

```
setplot(xdata, ydata, pretty.call = TRUE, maxdim, axes = FALSE)
```

Arguments

`xdata` vector x-axis coordinates of locations, or either a $n \times 2$ matrix or a list with named components `x` and `y` for x- and y-axis coordinates, respectively

`ydata` vector y-axis coordinates of locations, not needed if $n \times 2$ matrix is given for xdata

`pretty.call` if `TRUE`, coordinate locations are replaced with a grid calculated using `pretty`

`maxdim` plot dimensions, `(width, height)`, in inches; if not provided, current `par("pin")` is used

`axes` logical value indicating whether both axes should be drawn on the plot

Value

A list containing the following named components:

`xlim, ylim` plot limits for the x- and y-axis, respectively

`oldpin` old plot dimensions, as returned by `par('pin')`

`newpin` new plot dimensions

`seval` *Value estimation using thin-plate spline*

Description

Function to estimate value at a location using thin-plate spline

Usage

```
seval(x, tpsp)
```

Arguments

`x` $n \times 2$ matrix, containing xy coordinates of locations

`tpsp` thin-plate solution, typically returned by `sinterp`

Value

List containing the following named components:

`x` location coordinates, copied from the `x`

`y` estimated values at location `x`

sinterp *Fitting a thin-plate spline*

Description

Function to compute coefficients for arbitrary dimension thinplate spline smooths, that is, smooth mappings from Rq to Rp.

Usage

```
sinterp(x, y, m = 2, lam = 0, lsq = FALSE)
```

Arguments

x	An $nq \times nk$ matrix of knots, where nq = dimension of the domain space, and nk = the number of knots. Each column represents one knot. That is, the row dimension should be the dimension of the domain space, and the column dimension should be the number of knot points.
y	An $nq \times nk$ matrix of function values at each knot point, where np = dimension of the image space. Each data set is a column. (Note that the row dimension of y is the number of knots which corresponds to the column dimension of x). (As returned by `Falternate3`.)
m	An integer such that $2 *$ `m` is the order of the spline. The default value is 2.
lam	A vector of real smoothing parameters. If lam is missing or 0, sinterp performs interpolation.
lsq	if `TRUE`, subtract least squares fit from `y`, returning the polynomial coefficients as element `b` of the result

Value

Thin-plate spline solution; more details in Sampson-Guttorp (1992)

x	A copy of the `x` matrix argument passed to the function.
y	A copy of the `y` matrix argument.
m	A copy of the `m` argument.
lam	A copy of the `lam` argument.
lsq	A copy of the `lsq` argument.
b	If requested, this contains the coefficients of the least squares portion. Each column represents one `y` dataset, with the coefficients in the same order as in `sol`.
sol	A solution array. The first n components are the coefficients of $u(x - x(i))$. The next $d + m - 1$ choose $m - 1$ components are the coefficients of the interpolating polynomial. These coefficients are ordered by increasing order of the total degree of the monomial. Within a group of monomials whose total degree

is the same, the coefficient of monomial i is before the coefficient of monomial j if and only if $x(k)$ appears to a higher power in monomial i than it does in monomial j for some k such that $x(k-l)$ is not in either monomial for any l. Example: if $m = 3$ and $d = 4$, the order of the polynomial coefficients will be: $1, x1, x2, x3, x4, x1^2, x1 * x2, x1 * x3, x1 * x4, x2^2, x2 * x3, x2 * x4, x3^2, x3 * x4, x4^2$ The above mess describes each column. The array is $len \times ny \times nlam$, where nlam is the number of lambda values, ny is the number of y data vectors, and len is the length of the vector described above. In the case that nlam equals 1 (the default), "sol" is returned as 2-dimensional $len \times ny$ array without the third dimension.

`ainf` A vector of informational integers for the factored "a" matrix. Zero's indicate everything is O.K.. If the i-th entry is k, then the k-th pivot block of the a matrix for the i-th lambda value is singular. In the last case, no solution will be computed.

`linf` A vector of informational integers corresponding to the least squares solution(if requested). Zero's indicate no problems. If the i-th entry is k, then the k-th diagonal entry of the R matrix of the QR decomposition of the polynomial matrix is zero.

`f` A copy of the factorial values calculated for various internal length determinations.

`a` A copy of the internally generated factored "a" matrix.

References

Described in Sampson P.D., and Guttorp, P., "Nonparametric estimation of nonstationary spatial covariance structure", in Journal of the American Statistical Association, vol 87, pp 108-119, 1992.

`staircase.EM` *Estimate gauged sites hyperparameters*

Description

Estimate $\mathcal{H}_\}$ hyperparameters of the gauged sites using the EM algorithm, using the staircase of the missing data to determine the default block structure.

Usage

```
staircase.EM(data, p = 1, block = NULL, covariate = NULL,
             B0 = NULL, init = NULL, a = 2, r = 0.5, verbose = FALSE,
             maxit = 20, tol = 1e-06)
```

Arguments

`data` data matrix, grouped by blocks each with stations having the same number of missing observations. The blocks are organized in order of decreasing number of missing observations, ie. block 1 has more missing observations than block2. Default structure:

- Each column represent data from a station; rows are for time
- Blocks are decided based on the number of missing observations

`p`	number of pollutants measured at each stations. (first p columns of y are for p pollutants from station 1, block 1).
`block`	a vector indicating the number of stations in each block - from 1 to K
`covariate`	design matrix for covariates created with `model.matrix` with `as.factor`
`B0`	Provided if the hyperparameter β_0 (B0) is known and not estimated
`init`	Initial values for the hyperparameters; output of this function can be used for that
`a`	When p=1, the type-II MLE's for delta's are not available. Delta's are assumed to follow a gamma distribution with parameters (a,r)
`r`	When p=1, the type-II MLE's for delta's are not available. Delta's are assumed to follow a gamma distribution with parameters (a,r)
`verbose`	flag for writing out the results at each iteration
`maxit`	the default maximum number of iterations
`tol`	the convergence level.

Details

The estimated model is as follows:

- $data \sim MVN(z \times \beta, \mathrm{kronecker}(I, \Sigma))$
- $\beta \sim MVN(\beta_0, \mathrm{kronecker}(F^{-1}, \Sigma))$
- $\Sigma \sim GIW(\Theta, \delta)$

Θ is a collection of hyperparameters including $\xi_0, \Omega, \Lambda, H^{-1}$.

Value

A list with following elements:

`Delta`	The estimated degrees freedom for each of the blocks (list)
`Omega`	The estimated covariance matrix between pollutants
`Lambda`	The estimated conditional covariance matrix between stations in each block given data at stations in higher blocks (less missing data) - (list)
`Xi0`	The estimated slopes of regression between stations in each blocks and those in higher blocks (list). Note that $\tau_{0i} = \mathrm{kronecker}(\xi_0, diag(p))$ - same across stations for each pollutants.
`Beta0`	Coefficients - assumed to be the same across stations for each pollutant
`Finv`	Scale associated with β_0
`Hinv`	The estimated hyperparameters (list) - inverse of H_j
`Psi`	The estimated (marginal) covariance matrix between stations
`block`	From input
`data`	From input
`covariate`	From input
`Lambda.1K`	The inverse Bartlett decomposition (eqn 23?)

See Also

`staircase.hyper.est`

`staircase.hyper.est` *Estimate hyperparameters of ungauged sites*

Description

This function combines the results from the `staircase.EM` fit and the SG method to estimate the hyperparameters associated with the ungauged sites.

Usage

```
staircase.hyper.est(emfit, covfit, u, p, g, d0 = NULL)
```

Arguments

`emfit`	Output from the `staircase.EM` fit
`covfit`	The covariance matrix between all locations (with new locations at the beginning). This is an output from the SG fitting
`u`	number of new locations
`p`	dimension of the multivariate response
`g`	number of stations
`d0`	(optional) The degrees of freedom for the new locations (ungauged block)

Value

List with the following elements:

`Delta.0`	The degree of freedoms for the new locations. Equal to `d0`, if given (must be $> u*p+2$), else `mean(emfit$delta)` if $> u*p+2$; `u*p+ min(emfit$delta)` otherwise.
`Lambda.0`	Conditional variance between new locations given the gauged stations
`Xi0.0`	the regression slope (Note: $\tau_{0i} = \text{kronecker}(\xi_0, diag(p))$)
`H.0`	The variance matrix for the rows of $\tau^{[u]}$

Also all components of the output of the `staircase.EM` fit (for blocks 1-K).

See Also

`staircase.EM`

Index