Package 'LatticeKrig'

June 6, 2014

Version 3.2

Date May 24, 2013

Title Multiresolution Kriging based on Markov random fields

Author Douglas Nychka [aut, cre], Dorit Hammerling [aut], Stephan Sain [aut], Nathan Lenssen [aut]

Maintainer Douglas Nychka <nychka@ucar.edu>

Description Functions for the interpolation of large spatial
datasets. This package follows a ``fixed rank Kriging'' approach using
a large number of basis functions and provides spatial estimates
that are comparable to standard families of covariance functions.
Using a large number of basis functions allows for estimates that
can come close to interpolating the observations (a spatial model
with a small nugget variance.) The covariance model for this method
can approximate the Matern covariance family but also allows for a
multi-resolution model and supports efficient computation of the
profile likelihood for estimating covariance parameters. This is
accomplished through compactly supported basis functions and a
Markov random field model for the basis coefficients. These features
lead to sparse matrices for the computations. One benefit of this
approach is the facilty to do unconditional and conditional
simulation of the field for large numbers of arbitrary points. There
is also the flexibility for estimating nonstationary covariances. Included are
generic methods for prediction, standard errors for prediction,plotting of the estimated surface and conditional and unconditional
simulation.

License GPL (>= 2)

URL http://www.r-project.org

Depends R (>= 3.0.1), spam, fields (>= 6.9.1)

R topics documented:

`LatticeKrig` *User-friendly spatial prediction and inference using a compactly supported multi-resolution basis and a lattice model for the basis coefficients.*

Description

This is a simple wrapper (or high level) function for the more basic `LKrig` function. Several default choices are made for the multi-resolution spatial covariance but the returned object from this function can then used for spatial prediction and inference. It uses the defaults that are based on a "thin-plate spline" like model for the spatial estimator and also uses `LKrigFindLambda` to estimate some covariance parameters through likelihood maximization (i.e. estimates the measurement and process variances.) Despite the simple syntax, LatticeKrig still takes advantage of the multi-resolution feature of `LKrig` and any `LKrig` parameter can be passed through the function call. See the example below for varying the range parameter.

Usage

```
LatticeKrig(x, y, Z = NULL, nu = 1, nlevel = 4, a.wght = 4.01, NC
                 = NULL, LKinfo = NULL, na.rm = TRUE, tol = 0.005, ...)
```

Arguments

`a.wght`	The "a" spatial autoregressive parameter for a first order Markov Random field. This controls the spatial dependence and must be greater than or equal to 4. Default is a large spatial similar to thin plate spline type model. See Details.
`LKinfo`	An optional list giving the full specification of the covaraince. If missing will be created internally and returned. If passed this parametrization will be used except lambda will will be reestimated by maximum likelihood.
`nlevel`	Number of levels for the multi-resolution basis. Each level increases the number of basis functions by roughly a factor of 4.
`NC`	Sets the size of the initial lattice to be NC by NC if the domain is square
`na.rm`	If TRUE NA's are removed from `y` and `x` is subsetted.
`nu`	Smoothness of the multi-resolution process. If passed, alpha vector is proportional to `exp(-(1:nlevel)*nu)` and normalized to sum to one. Default is similar to a Whittle covariance (Matern with smoothness of 1).
`tol`	Tolerance for the log likelihood used to judge convergence.
`x`	Spatial locations of observations. Or the `LKrig` object for printing.
`y`	Spatial observations. No missing values are allowed.
`Z`	Linear covariates to be included in fixed part of the model that are distinct from the default first order polynomial in `x` (i.e. the spatial drift).
`...`	Additional arguments to pass to LKrig.

Details

Keep in mind that overall LatticeKrig is just a specific type of spatial estimate that is designed to handle larger size data sets. But it incorporates a specific covariance function and the estimator is the Kriging/Multivariate Gaussian Conditional Expectation/BLUE that is standard in this field. The basic function LKrig has a very flexible covariance and at least specifying all the relevant parameters may be discouraging to a new user. LatticeKrig is a "wrapper" that generates some simplified, default model choices to call the more general function `LKrig`. It is useful for users not fully familiar with the LatticeKrig methodology or those that wish to try a default approach to get a quick look at a spatial analysis. The default values are set to give about 4 times as many basis functions as observations and to use four levels of multi-resolution. The spatial correlation range is nearly stationary and set large to mimic a thin-plate spline. The smoothness mimics the Whittle covariance function (smoothness = 1 for the Matern). (see LKrig.cov.plot to get a plot of the implied covariance function.) LatticeKrig also provides maximum likelihood estimates of the measurement error and process varaince parameters that are perhaps the parameters that most effect the shape of the estimated spatail field.

Of course this top level function is pitched with all the caveats that statistical model assumptions should always be checked and applying generic methods to a specific problems without checking the appropriateness can lead to misleading results. Details on the full computations can be found in the `LKrig` man page.

The `lambda = sigma2/ rho` parameter in `LKrig` is essential to the Lattice Krig computation and an inappropriate value will result in over or under fitting and incorrect interpolated values. The function `LKrigFindLambda` is used within `LatticeKrig` to estimate a lambda value from the data using maximum likelihood.

The main call inside `LatticeKrig` is to `LKrig` and so a `LKrig` object is returned. Thus all of the functionality that comes with `LKrig` objects such as `predict`, `summary`, `predictSurface`, etc. remain the same as described in `LKrig`. Also see the components `residuals` and `fitted.values` in the returned object for these parts of the fitted spatial model. The component `LKinfo` has all the details that specfiy the basis functions and covariance model.

Author(s)

Doug Nychka

See Also

LKrig, LKrig.setup, LKrigFindLambda, LKinfo, LKrig.sim.conditional

Examples

```
# Load ozone data set
  data(ozone2)
  x<-ozone2$lon.lat
  y<- ozone2$y[16,]

# thin plate spline-like model with the lambda parameter estimated by
# maximum likelihood. Default choices are made for a.wght, nlevel, NC
# and alpha.
  obj<- LatticeKrig( x, y)
## Not run:
# summary of fit and a plot of fitted surface
  print( obj)
  surface( obj )
  US(add=TRUE)
```

```
  points(x)
# prediction standard errors
  out.se<- predictSE( obj, xnew= x)

## End(Not run)

#############################################################################
# Including a covariate (linear fixed part in spatial model)
#############################################################################
## Not run:
  data(COmonthlyMet)

  obj  <- LatticeKrig(CO.loc,  CO.tmin.MAM.climate, Z=CO.elev)
  obj2 <- LatticeKrig(CO.loc, CO.tmin.MAM.climate)

# compare with and without linear covariates
  set.panel(1,2)
  surface(obj)
  US(add=TRUE)
  title("With Elevation Covariate")

  surface(obj2)
  US(add=TRUE)
  title("Without Elevation Covariate")


## End(Not run)
## Not run:
 data(COmonthlyMet)
# Examining a few different "range" parameters
a.wghtGrid<-  4  +  c( .1,.5, .8, 1, 2)^2

#NOTE smallest is "spline like" the largest is essentially independent
# coefficients at each level.  In this case the "independent" end is
# favored but the eff df. of the surface is very similar across models
# indicating about the same separate of the estimates into spatial
# signal and noise
#
for( k in 1:5 ){
obj  <- LatticeKrig(CO.loc,  CO.tmin.MAM.climate, Z=CO.elev,
                     a.wght=a.wghtGrid[k])
cat( "a.wght:", a.wghtGrid[k], "ln Profile Like:",
            obj$lnProfileLike, "Eff df:", obj$trA.est, fill=TRUE)
}

## End(Not run)
```

`LKinfo` *Specifying the Lattice Krig covariance object(*`LKinfo`*) and related utility functions.*

Description

The `LKinfo` object is a list that contains all the details to specify an LKrig covariance model. Typically this object is created by supplying a few parameters to the function `LKrig.setup`.

Usage

```
 LKrig.setup(x=NULL, NC=NULL, NC.buffer=5, nlevel,
grid.info=NULL, lambda=NA, sigma=NA, rho=NA, alpha=NA, nu=NULL,
a.wght=NA, overlap=2.5, normalize=TRUE, normalize.level=NULL,
edge=FALSE, rho.object=NULL, RadialBasisFunction="WendlandFunction",
distance.type = "Euclidean", V =diag(c(1, 1)), verbose=FALSE)

LKrig.make.centers( grid.info, nlevel, NC.buffer, distance.type)

LKrig.make.a.wght( a.wght, nlevel, mx, my)

LKrig.make.alpha( alpha, nu, nlevel)

LKrig.make.Normalization(mx,my, a.wght)

LKrigMRFDecomposition( mx,my,a.wght)

LKinfoUpdate(LKinfo, ... )
```

Arguments

alpha	A vector of length nlevel with the relative variances for the different mulitresolution levels.
a.wght	A vector of length nlevel that are the weights for the central point in the Markov random field specification. To be precise at level k the center point has weight a.wght[k] with the 4 nearest neighbors given weight -1. The a.wght must be greater than 4 for the fields to be stationary.
distance.type	See entry in LKrig.
edge	If FALSE no adjustments are made to edges.
grid.info	A list with components xmin, xmax, ymin, ymax, delta that specifies the range and spacing for the coarsest level grid. NOTE: If NC is not specified grid.info must be passed.
lambda	The "noise to signal ratio" or also known as the smoothing parameter it is the parameter lambda = sigma^2/rho. If specified then sigma and rho typically are estimated in LKrig by maximum likelihood. If lambda is not specified then it is set as lambda = sigma^2/ rho. Note that to evaluate the spatial process model, e.g. using the function LKrig.cov, a value of lambda is not needed and this argument can default to NA.
LKinfo	An LKinfo object as described in the Details or that created by LKrig.setup.
mx	Number of lattice points in horizontal coordinate.
my	Number of lattice points in vertical coordinate.
NC	Maximum number of grid points in either the x or y directions to determine the basis function centers. If the region is square then this will be NC grid points in each dimension within the spatial domain defined by the data locations (or grid.info) giving a total NC**2 grid points/basis functions for the coarsest level. There may be a larger number of total basis function based on how the buffer grid domain is set.
NC.buffer	Number of grid points to add beyond the spatial domain. This number is fixed at each level.

nlevel	Number of levels in multiresolution. Note that each subsequent level increases the number of basis functions within the spatial domain size by a factor of roughly 4.
overlap	The amount of overlap between basis functions. If the grid spacing in x and y is delta then the support of the basis functions will be overlap*delta. This value is hardwired to 2.5 in LKrig and also the default in LKrig.setup
normalize	If TRUE the basis functions will be normalized to give a marginal variance of one.
normalize.level	A logical vector of length number of multiresolution levels. If normalize.level[i]==TRUE then the process at level i has its basis functions "normalized" to give a constant marginal variance. Default is that all values are TRUE. This is an experimental argument to save on computation, which might be used to avoid normalizing very fine levels that have minimal artifacts, but many basis.
nu	If passed alpha vector is defined as exp(-2*(1:nlevel)*nu) but then normalized to sum to one.
RadialBasisFunction	A character string with the name of the R function used to generate the multiresolution basis. Default is WendlandFunction, the Wendland covariance (order 2).
rho.object	A prediction object to specify part of the marginal variance for the process. If omitted assumed to be the constant one. Calling predict(rho.object,x1) should return a vector with the values at the locations in x1.
rho	A scalar, the sill or marginal variance of the process.
sigma	The measurement error standard deviation.
x	Spatial locations for fitting surface. This is used to determine ranges of the grid for basis functions so only two points are required that bound the rest of the data locations. E.g. x= cbind(c(0,1), c(0,1)) will set the domain to be the unit square.
V	See entry in LKrig.
verbose	If TRUE print out intermediate information.
...	Any additional arguments that can be used in LKrig.setup argument.

Details

The basis functions are two-dimensional radial basis functions based on the Wendland covariance function, centered on regular grid points and with the scaling tied to the spacing of the grid points. These grids are created by the utility LKrig.make.centers listed above.

For a basis at the coarsest level, the grid centers for the first level are generated by expanding the sequences in the x and y coordinates into a regular grid of center points and using the spacing either determined from the domain ranges and NC or just from the delta component in grid.info. The following R code demonstrates how to create a grid without any buffer grid points (i.e. NC.buffer equal to zero) outside the spatial domain.

```
xgrid<- seq(grid.info$xmin,grid.info$xmax,grid.info$delta)
ygrid<- seq(grid.info$ymin,grid.info$ymax,grid.info$delta)
The.Lattice<- make.surface.grid( list( x=xgrid, y=ygrid))
```

Note that the same spacing `delta` is used in both directions.

The subsequent levels use the same expressions but delta is reduced by a factor 1/2 for each level. The LKinfo object has several components that help with the bookkeeping for the number of centers at each level.

The additional functions listed are typically called from the top level function `LKrig.setup` and are used to complete the LKinfo object. `LKrig.make.centers` and `LKrig.make.grid.info` setup the lattice and also the centers of the basis functions.

`LKrig.make.a.wght` and `LKrig.make.alpha` create the covariance parameter lists based on defaults and what is passed.

`LKrig.make.Normalization` computes some matrices that speedup normalizing the absis functions to have a constant marginal variance function. These matrices depend on the a.wght parameter and so need to be recomputed if a.wght is changed. The function `LKinfoUpdate` is a more rigorous way to change just a few parameter in the LKinfo object and has the advantage that when a.wght is changes the normalization matrices are recalculated.

Value

`LKrig.setup`: Returns an `LKinfo` object. This object is a "list of lists" and collects all the information to describe the covariance model and the observational model. In most case one would supply some of the important components of this and the functions `LKrig.setup` and `LKrig` will fill in the rest with their defaults. The basic design is to pass the LKinfo object to subsequent functions to avoid keeping track of many parts of the covariance specification. See for example `LKrig.cov`.

Components of the `LKinfo` object

mx my Two vectors giving the size of the grids for each level. At level `l` there are `mx[l]*my[l]` basis functions.

nlevel Number of multiresolution levels.

delta Spacing between grid point centers.

m Total number of basis functions.

offset Indices for coefficients by level. offset[k]+1 is the position of the first coefficient in level k.

grid.info A list with min and max values for x and y grids and the spacing.

grid A list with nlevel components: grid[[k]] is a list with components `x` and `y` specifying the grid for the basis function centers at level k.

overlap Amount the basis functions overlap.

alpha A list with nlevel components with alpha[[k]] being the scalar or matrix giving the values for alpha at the kth level.

a.wght A list with nlevel components with a.wght[[k]] being the scalar or matrix giving the values for a.wght at the kth level.

lambda Value of lambda. A value of NA is OK if it has not been estimated or set.

sigma rho Values of the covariance parameters for measurement standard deviation and the process marginal variance. These can be NA if not yet determined.

normalize A logical value with TRUE being default to normalize the basis function to have constant marginal variance.

normalize.level A logical vector of length nlevel to determine which levels of basis functions should be normalized.

edge A logical parameter; if TRUE edge corrections are made to precision matrix.

scalar.alpha Logical value with TRUE indicating that alpha is a scalar at each level – i.e. not a matrix.

scale.basis Logical if TRUE indicates rho is a prediction obj defining a surface.

rho.object The prediction object that defines the value of rho at every point in the domain according to the predict function.

RadialBasisFunction Name of radial basis function used as template for generating basis functions.

`LKrig.make.centers`: Returns a list whose components are a subset of those listed above:

Author(s)

Doug Nychka

See Also

LKrig, mKrig, Krig, fastTps, Wendland, LKrig.basis

Examples

```
# Load ozone data set
  data(ozone2)
  x<-ozone2$lon.lat
  y<- ozone2$y[16,]
# Find location that are not NA.
# (LKrig is not set up to handle missing observations.)
  good <-  !is.na( y)
  x<- x[good,]
  y<- y[good]
  LKinfo<- LKrig.setup( x,NC=10,nlevel=2, alpha=c(1,.5),
                            lambda= NA , a.wght=c(5,5))
```

`LKrig` *Spatial prediction and inference using a compactly supported multi-resolution basis and a lattice model for the basis coefficients.*

Description

A variation of Kriging with fixed basis functions that uses a compactly supported covariance to create a regular set of basis functions on a grid. The coefficients of these basis functions are modeled as a Gaussian Markov random field (GMRF). Although the precision matrix of the GMRF will be sparse the model can still exhibit longer ranges of spatial dependence. The multi-resolution feature of this model allows for the approximation of a wide variety of standard covariance functions.

Usage

```
LKrig(x, y = NULL, weights = rep(1, nrow(x)), Z = NULL,
                  fixedFunction = "LKrigDefaultFixedFunction",
                  fixedFunctionArgs = NULL, m = 2,
                  LKinfo = NULL, iseed = 123, NtrA = 20, use.cholesky =
                  NULL, return.cholesky = TRUE, wPHI = NULL, return.wPHI
```

```
                  = TRUE, NC, nlevel, a.wght, alpha, nu = NULL, lambda =
                  NA, sigma = NA, rho = NA, rho.object = NULL, overlap =
                  2.5, normalize = TRUE, edge = FALSE,
                  RadialBasisFunction = "WendlandFunction", V =
                  diag(c(1, 1)), distance.type = "Euclidean", verbose =
                  FALSE)

## S3 method for class LKrig
predict(object, xnew = object$x, Znew = NULL, drop.Z = FALSE,
                  return.levels = FALSE, ...)

## S3 method for class LKrig
predictSE(object, xnew = object$x, Znew = object$Z, verbose =
                  FALSE, ...)

## S3 method for class LKrig
surface( object, ...)

## S3 method for class LKrig
predictSurface(object, grid.list = NULL, extrap = FALSE, chull.mask =
                  NA, nx = 80, ny = 80, xy = c(1, 2), verbose = FALSE,
                  ZGrid = NULL, drop.Z = FALSE, ...)

## S3 method for class LKrig
print( x, digits=4, ...)
```

Arguments

alpha	Variance weights for each level of resolution. This can be a vector or a list with nlevel components. As a list each component is a matrix specifying a weight for each coefficient. See Details.
a.wght	The "a" spatial autoregressive parameter for a first order Markov Random field. This controls the spatial dependence and must be greater than or equal to 4. For a.wght = 4 normalize should be FALSE. If there are multiple levels this can be a vector. See Details.
chull.mask	An additional constraint for evaluating predictions see help(in.poly). Usually this is not needed.
distance.type	The type of distance metric (as a text string) used in determining the radial basis functions. The current choices are "Euclidean" and "cylinder" with cylinder being used to have a periodic boundary condition in the x (horizontal) coordinate.
digits	Number of digits in printed output.
drop.Z	If true the fixed part will only be evaluated at the spatial drift polynomial part of the fixed model. The contribution from the other, Z, covariates in the fixed component will be omitted.
edge	If TRUE an adjustment is made in the GMRF weights for edge effects.
extrap	If TRUE will extrapolate predictions beyond convex hull of locations.
fixedFunction	Function used to define the fixed part of the model in terms of spatial locations. Default is an m-1 degree polynomial.
fixedFunctionArgs	Additional arguments in list form to pass to the fixed function.

grid.list — A list with components x and y specifying the grid (see help(grid.list)).

iseed — Random seed used in the Monte Carlo technique for approximating the effective degrees of freedom (trace of the smoothing matrix). If NA, no seed is set.

lambda — The ratio of the nugget variance (called sigma squared in fields and LatticeKrig) to the parameter controlling the marginal variance of the process (called rho in fields and LatticeKrig). If sigma and rho are both specified then lambda == sigma^2/rho.

LKinfo — An object whose components specify the LatticeKrig spatial model. This is usually created by the function LKrig.setup. If NULL, this object is created and returned as a component of the LKrig object.

m — If the fixed is the default the polynomial will have degree m-1

nx — Number of grids in x for prediction grid.

ny — Number of grids in y for prediction grid.

NC — Controls the maximum number of grid points. For the first level there will be NC grid points in the larger dimension of the rectangular spatial domain. If the domain is square there will be NC*NC basis functions/lattice points in the coarsest level of resolution. If domain is rectangular, the smaller dimension will have less than NC points.

nlevel — Number of levels for the multiresolution basis. Each level increases the number of basis functions by roughly a factor of 4.

normalize — If TRUE basis functions are normalized so that the marginal variance of the process covariance is constant and equal to rho. This normalization avoids some of the edge and periodic artifacts from using a discrete set of basis functions.

NtrA — Number of random samples used in Monte Carlo method for determining effective degrees of freedom.

nu — If passed, alpha vector is proportional to exp(-(1:nlevel)*nu) and normalized to sum to one.

object — The LKrig object.

overlap — The overlap between basis functions. This scaling is based on centers being spaced 1 unit distance apart and the Wendland function decreasing to zero at 1 unit distance. A scaling/overlap of 2.5 (the default) implies that the support of the basis functions will extend to a disc of radius 2.5. We recommend that this parameter not be changed unless one is sure of the effect on the implied spatial covariance.

return.cholesky — If TRUE the Cholesky decomposition is included in the output list (with the name Mc). This is needed for some of the subsquent computations such as finding prediction standard errors. Set this argument to FALSE to avoid much larger objects when the decomposition is not needed. This option is often paired with a subsequent call to LKrig with use.cholesky. See the MLE.LKrig source code for details.

return.levels — If TRUE the predicted values for each level are returned as columns in a matrix.

RadialBasisFunction — An R function that defines the radial basis function to generate the multiresolution basis. Is assumed that this function is compactly supported on the interval [0,1]. See WendlandFunction for an example.

rho — Value of rho (sill variance) to use for spatial estimate. If omitted this is estimated as the MLE based on the value of lambda.

rho.object	A prediction object that defines a spatially varying component for the marginal variance of the process. The object should be such that the code `predict(rho.object, x1)` will evaluate the function at 2-d locations `x1`. See Details below.
sigma	Value of sigma (nugget variance) to use for the spatial estimate. If omitted this is estimated as the MLE based on the value of lambda.
wPHI	If not NULL then use as the "wPHI" regression matrix in the computations. This is only used when just lambda is being changed from a previous call.
return.wPHI	If TRUE return the wPHI regression matrix. This is most likely to be used for a subsequent call to LKrig that avoids recreating this matrix.
verbose	If `TRUE` print out intermediate results and messages.
use.cholesky	Use the symbolic part of the Cholesky decomposition passed in this argument.
V	A matrix used to scale and rotate coordinates to give an anisotropic basis function and covariance model. V is the 2X2 matrix describing the inverse linear transformation of the coordinates before distances are found. In R code this transformation is: `x1 %*% t(solve(V))` Default is NULL, that is the transformation is just dividing distance by the scalar value `theta`. See Details below. If one has a vector of "theta's" that are the scaling for each coordinate then just express this as `V = diag(theta)` in the call to this function.
weights	A vector that is proportional to the reciprocal variances of the errors. I.e. errors are assumed to be uncorrelated with variances sigma^2/weights.
xnew	Matrix of locations for prediction.
xy	Order of evaluating surface coordinates. This would be used if for example a lon-lat surface needed to be transposed as a "lat-lon" object. Usually not needed.
x	Spatial locations of observations. Or the `LKrig` object for printing.
y	Spatial observations.
Z	Linear covariates to be included in fixed part of the model that are distinct from the default first order polynomial in `x` (i.e. the spatial drift).
Znew	Values of covariates, distinct from the spatial drift for predictions of data locations.
ZGrid	An array or list form of covariates to use for prediction. This must match the `grid.list` argument. e.g. ZGrid and grid.list describe same grid. If ZGrid is an array then the first two indices are the x and y locations in the grid. The third index, if present, indexes the covariates. e.g. For evaluation on a 10X15 grid and with 2 covariates. `dim( ZGrid) == c(10,15, 2)`. If ZGrid is a list then the components x and y shold match those of grid.list and the z component follows the shape described above for the no list case.
...	Additional arguments to pass to generic methods or from LatticeKrig to LKrig. See help(predictSurface) for the details in calling LKrig.predictSurface.

Details

This method combines compactly supported basis functions and a Markov random field covariance model to provide spatial analysis for large data sets. The model for the spatial field (or spatial process) is

f(x) = N(x) + Z d + sum Phi.j(x) c.j.

x is a location in two dimensions, N(x) is a low order (linear) polynomial, Z is a matrix of spatial covariates and d a coefficient vector. Phi.j for 1 <= j <= m is a set of fixed basis functions and c.j

the coefficients. The variance of f(x) is given by the parameter rho throughout this package. As explained below the process f is a sum of nlevel independent processes that have different scales of spatial dependence. The alpha gives the relative weighting between these processes. Thus, the minimum set of parameters needed to describe the covariance of f are the integer NC, two scalars rho and a.wght and a vector alpha. alpha has length the number of multiresolution levels but we recommend that it be constrained sum to one. Thus in total there are a 1 + 2 + (nlevel - 1) parameters for a minimal specification of the covariance. Note that this parsimonious specification results in a covariance that is close to being stationary and isotropic when normalize is TRUE. An additional constraint on alpha is to make the weights alpha[j] proportional to exp(- 2*j*nu) where nu controls decay of the alpha weights. There is some theory to suggest that nu is analgous to the smoothness parameter from the Matern family (e.g. nu=.5 approximates the exponential). In this case the covariance model requires just four parameters, NC, rho, a.wght,nu.

The data model is

Y.k = f(x.k) + e.k

Y.k are (scalar) observations made at spatial locations x.k with e.k uncorrelated normal errors with variance sigma^2/weights. Thus there is a minimum of one new parameter from the data model: sigma. Note that prediction only depends on the ratio lambda = sigma^2/ rho and not surprisingly lambda plays a key role in specifying and fitting a spatial model. Also taken with the model for f the minimum parameters needed for a spatial prediction are still four NC,a.wght, nu and lambda. For fixed lambda there are closed form expressions for the MLEs for sigma and rho. LKrig exploits this feature by depending on lambda and then computing the MLEs for sigma and rho.

Spatial prediction: The basis functions are assumed to be fixed and the coefficients follow a multivariate Gaussian distribution. Given this spatial model for f, it is possible to compute the conditional expectation of f given Y and also maximize the likelihood for the model parameters, lambda, alpha, and a.wght. This setting is known as fixed rank Kriging and is a common strategy for formulating a spatial model. Typically fixed rank Kriging is used to reduce the dimension of the problem by limiting the number of basis functions. We take a different approach in allowing for models that might even have more basis functions than observations. This provides a spatial model that can come close to interpolating the observations and the spatial process is represented with many degrees of freedom. The key is to make sure the model ingredients result in sparse matrices where linear algebra is required for the computations. By doing so in this package it is possible to compute the estimates and likelihood for large numbers of spatial locations. This model has an advantage over covariance tapering or compactly supported covariance functions (e.g. fastTps from fields), because the implied covariance functions can have longer range correlations.

Radial basis functions (Phi.j) : The basis functions are two-dimensional radial basis functions (RBFs) that are derived from scaling and translations of a single covariance function. The default in LatticeKrig is to use the Wendland compactly supported stationary covariance (order 2 for 2 dimensions) that is scaled to be zero beyond a distance of 1. For d the distance between spatial locations, this Wendland function has the standard form:

(1 - d)^6 * (35 * d^2 + 18 * d + 3))/3 for d in [0,1]

0 otherwise.

For a single level the RBFs are centered at a regular grid of points and with radial support delta*overlap where delta is the spacing between grid points. We will also refer to this grid of centers as a lattice and the centers are also referred to as "nodes" in the RBF literature. The overlap for the Wendland has the default value of 2.5 and this represents a compromise between the number of nonzero matrix elements for computation and the shape of the covariance functions.

To create a multi-resolution basis, each subsequent level is based on a grid with delta divided by 2. See the example below and help(LKrig.basis) for more details. For multiple levels the basis functions can be grouped according to the resolution levels and the coefficients can be grouped in a similar manner. There is one important difference in the basis construction – a normalization –

and this aspect makes it different from a simple radial basis function specification and is described below.

Markov random field (GMRF) for the coefficients (c.j) : Because the coefficients are identified with locations on a lattice it is easy to formulate a Markov random field for their distribution based on the relationship to neighboring lattice points. The distribution on the basis function coefficients is a multivariate normal, with a mean of zero and the the precision matrix, Q, (inverse of Q is the covariance matrix). Q is partitioned in a block diagonal format corresponding to the organization of the basis functions and coefficients into levels of resolution. Moreover, coefficients at different levels are assumed to be independent and so Q will be block diagonal. If `nlevels` are specified, the ith block has a precision matrix based on a spatial autoregression with `a.wght[i]` being related to the spatial autoregressive parameter(s). Schematically in the simplest case the weighting for an interior lattice point and its four neighbors is

```
.   .    .       .     .

.   .   -1       .     .

.  -1  a.wght   -1     .

.   .   -1       .     .

.   .    .       .     .
Interior point
```

The fundamental idea is that these weights applied to each point in the lattice will result in a lattice of random variables that are independent. The specific precision matrix for each block (level), Q.i, is implemented by `LKrig.MRF.precision`. In the case when alpha is a scalar, let C.i be the vector of basis coefficents at the ith level then we assume that `B %*% C.i` will be independent N(0,1) random variables. By elementary properties of the covariance it follows that the precision matrix for C.i is Q.i= t(B)%*%B. Thus, given B one can determine the precision matrix and hence the covariance matrix. Each row of B, corresponding to a point in the lattice in the interior, is "a" (`a.wght[i]`) on the diagonal and -1 at each of the four nearest neighbors of the lattice points. Points on the edges and corners just have less neighbors but get the same weights.

This description is a spatial autoregressive model (SAR). The matrix Q will of course have more nonzero values than B and the entries in Q can be identified as the weights for a conditional autoregressive model (CAR). Moreover, the CAR specification defines the neighborhood such that the Markov property holds. Values for `a.wght[i]` that are greater than 4 give reasonable covariance models. Moreover setting a.wght[i] to 4 and `normalize` to FALSE in the call to LKrig will give a thin-plate spline type model that does not have a range parameter. An approximate strategy, however, is to set a.wght close to 4 and get some benefit from the normalization to reduce edge effects.

Multiresolution process Given basis functions and coefficients at each level we have defined a spatial process g.i that can be evaluated at any location in the domain. These processes are weighted by the parameter vector alpha and then added together to give the full process. It is also assumed that the coefficients at different resolution levels are independent and so the processes at each level are also independent. The block diagonal structure for Q does not appear to limit how well this model can approximate standard spatial models and simplifies the computations. If the each g.i is normalized to have a marginal variance of one then g will have a variance that is the sum of the alpha parameters. Usually it is useful to constrain the alpha parameters to sum to one and then include an additional variance parameter, rho, be the marginal variance for g. So the full model for the spatial process used in LatticeKrig is

g(x) = sqrt(rho) * sum.i sqrt(alpha[i]) * g.i(x)

The specification of the basis and GMRF is through the components of the object LKinfo, a required component for many LatticeKrig functions. High level functions such as `LKrig` only require a minimal amount of information and combined with default choices create the LKinfo list. One direct way to create the complete list is to use `LKrig.setup` as in the example below. For `nlevel==1` one needs to specify `a.wght`, `NC`, and also `lambda` related to the measurement error variance. For a multiresolution setup, besides these parameters, one should consider different values of `alpha` keeping in mind that if this vector is not constrained in some way (e.g. `sum(alpha)==1`) it will not be identifiable from `lambda`.

The covariance derived from the GMRF and basis functions: An important part of this method is that the GMRF describes the coefficients of the basis functions rather than the field itself. Thus in order to determine the covariance for the observed data one needs to take in account both the GMRF covariance and the expansion using the basis functions. The reader is referred to the function `LKrig.cov` for an explicit code that computes the implied covariance function for the process f. Formally, if P is the covariance matrix (the inverse of Q) for the coefficients then the covariance between the field at two locations x1 and x2, will be

sum_ij P_ij Phi.i(x1) Phi.j(x2)

Moreover under the assumption that coefficients at different levels are independent this sum can be decomposed into sums over the separate levels. The function `LKrig.cov` evaluates this formula based on the LKrig object (`LKinfo`) at arbitrary groups of locations returning a cross covariance matrix. `LKrig.cov.plot` is a handy function for evaluating the covariance in the x and y directions to examine its shape. The function `LKrig.cov` is also in the form to be used with conventional Kriging codes in the fields package (loaded by LatticeKrig) `mKrig` or `Krig` and can be used for checking and examining the implied covariance function.

Normalizing the basis functions The unnormalized basis functions result in a covariance that has some non-stationary artifacts (see example below). For a covariance matrix P and for any location x one can evaluate the marginal variance of the process using unnormalized basis functions for each multiresolution level. Based this computation there is a weighting function, say w.i(x), so that when the normalized basis w.i(x) Phi.i(x) is used the marginal variance for the multiresolution process at each level will be one. This makes the basis functions dependent on the choice of Q and results in some extra overhead for computation. But we believe it is useful to avoid obvious artifacts resulting from the use of a finite spatial domain (edge effects) and the discretization used in the basis function model. This is an explicit way to make the model stationary in the marginal variance with the result that the covariance also tends to be closer to a stationary model. In this way the discretization and edges effects associated with the GMRF can be significantly diminished.

The default in `LKrig` is `normalize = TRUE`. It is an open question as to whether all levels of the multi-resolution need this kind of explicit normalization. There is the opportunity within the `LKrig.basis` function to only normalize specific levels with the `normalize` being extended from a single logical to a vector of logicals for the different levels. To examine what these edge effect artifacts are like the marginal variances for a 6 by 6 basis is included at the end of the Examples Section.

Nonstationary and anisotropic modifications to the covariance Given that the process at each level has been normalized to have marginal variance of one there are several other points where the variance can be modified. The variance at level i is scaled by the parameters `alpha[i]` and the marginal variance of the process is scaled by `rho`. Each of these can been extended to have some spatial variation and thus provide a model for nonstationarity.

An option in specifying the marginal variance is to prescribe a spatially varying multiplier. This component is specified by the object `rho.object`. By default this is not included (or assumed to be identically one) but, if used, the full specification for the marginal variance of the spatial process at location `x` is formally: `rho * predict(rho.object, x) * sum( alpha)` There is then a problem of identifiability between these and a good choice is to constrain `sum(alpha) ==1` so that `rho* predict(rho.object, x)` is associated with the marginal variance of the full spatial process.

A second option is to allow the alpha variance component parameters to vary across the lattices at each level. For this case alpha is a list with `nlevel` components and each component being a matrix with the same dimensions as the lattice at that level. The SAR weight matrix is taken to be the usual weights but each row is scaled by the corresponding value in the alpha weight matrix. To be precise the weight matrix is given in psuedo R code by

```
diag( 1/sqrt( c(alpha[[i]]))%*%B
```

leading to the precision matrix at level i of

Q.i = t(B) %*% diag(c(1/alpha[[i]]))%*%B

In this code note that the matrix of weights, `alpha[[i]]` is being stacked as a larger vector to match the implicit indexing of the basis coefficients.

LKrig also has the flexibility to handle more general weights in the GMRF. This is accomplished by `a.wght` being a list with as many components as levels. If each component is a vector of length nine then these are interpreted as the weights to be applied for the lattice point and its 8 nearest neighbors (see help(LKrig.MRF.precision) and the commented source code for details). If each component is a matrix then these are interpreted as the (nonstationary) center weights for each lattice point. Finally if the component is an array with three dimensions this specifies the center and 8 nearest neighhors for every point in the lattice. At this point the choice of these weights beyond a stationary model is experimental and we will defer further documentation of these features to a future version.

The smoothing parameter lambda and effective degrees of freedom Consistent with other fields package functions, the two main parameters in the model, sigma^2 and rho are parameterized as lambda = sigma^2/rho and rho. The MLEs for rho and sigma can be written in closed form as a function of lambda and these estimates can be substituted into the full likelihood to give a concentrated version that can numerically be maximized over lambda. The smoothing parameter lambda is best varied on a log scale and is sometimes difficult to interpret independent of the particular set of locations, sample size and covariance. A more useful interpretation of lambda is through the effective degrees of freedom and an approximate value is found by default using a Monte Carlo technique. The effective degrees of freedom will vary with the dimension of the fixed regression part of the spatial model (typical 3 = constant + linear terms) and the total number of observations. It can be interpreted as the approximate number of "parameters" needed to represent the spatial prediction surface. For a fixed covariance model the spatial estimate at the observation locations can be represented as f hat = A(lambda) y where A is a matrix that does not depend on observations. The effective number of degrees of freedom is the trace of A(lambda) in analogy to the least squares regression "hat" matrix and has an inverse, monotonic relationship to lambda. The Monte Carlo method uses the fact that if e are iid N(0,1) E(t(e) A(lambda) e) = trace(A(lambda)).

Descriptions of specific functions and objects:

`LKrig`**:** Find spatial process estimate for fixed covariance specificed by `nlevel, alpha, a.wght, NC, and lambda` or this information in an `LKinfo` list.

`predict.LKrig, predictSE.LKrig`**:** These functions evaluate the model at the the data locations or at `xnew` if it is included. Note the use of the `drop.Z` argument to either include the covariates or just restrict the computation to the spatial drift and the smooth component. If `drop.Z` is FALSE then then `Znew` needs to be included for predictions off of the observation locations. The standard errors are computed under the assumption that the covariance is known, that it is the TRUE covariance for the process, and both the process and measurement errors are multivariate normal. The formula that is used involves some recondite shortcuts for efficiency but has been checked against the standard errors found from an alternative formula in the fields Krig function. (See the script Lkrig.se.tests.R in the tests subdirectory for details.)

Value

`LKrig:` An LKrig class object with components for evaluating the estimate at arbitrary locations,

describing the fit and as an option (with `Mc.return=TRUE`) the Cholesky decomposition to allow for fast updating with new values of lambda, alpha and a.wght. The "symbolic" first step in the sparse Cholesky decomposition can also be used to compute the sparse Cholesky decomposition for a different positive definite matrix that has the same pattern of zeroes. This option is useful in computing the likelihood under different covariance parameters. For the LKrig covariance the sparsity pattern will be the same if `NC`, `level`, `overlap` and the data locations `x` are kept the same. The returned component `LKinfo` has class LKinfo and is a list with the information that describes the layout of the multiresolution basis functions and the covariance parameters for the GMRF. (See help(LKinfo) and also `LK.basis` as an example.)

`predict.LKrig, predictSE.LKrig:` A vector of predictions or standard errors.

`predictSurface.LKrig:` A list in image format (i.e. having components `x,y,z`) of the surface evaluated on a regular grid. This surface can then be plotted using several different R base package and fields functions e.g. `image`, `image.plotcontour`, `persp`, `drape.plot`. The `surface` method just calls this function and then a combination of the image and contour plotting functions.

Author(s)

Doug Nychka

See Also

LatticeKrig, LKrig.sim.conditional, mKrig, Krig, fastTps, Wendland, LKrig.coef, Lkrig.lnPlike, LKrig.MRF.precision, LKrig.precision

Examples

```
# Load ozone data set
  data(ozone2)
  x<-ozone2$lon.lat
  y<- ozone2$y[16,]
# Find location that are not NA.
# (LKrig is not set up to handle missing observations.)
  good <-  !is.na( y)
  x<- x[good,]
  y<- y[good]

# fairly arbitrary choices for covariance parameters and lambda
# just to show a basic level call
  obj1<- LKrig( x,y, a.wght=5, nlevel=3, nu=1.0, NC=4, lambda=.1)

# thin plate spline-like model with the lambda parameter estimated by
# maximum likelihood. Default choices are made for a.wght, nlevel, NC
# and alpha.
## Not run:
  obj<- LatticeKrig( x, y)
# summary of fit and a plot of fitted surface
  print( obj)
  surface( obj )
  US(add=TRUE)
  points(x)
# prediction standard errors
  out.se<- predictSE( obj, xnew= x)

## End(Not run)
```

```
# breaking down the LatticeKrig function into several steps.
# also use a different covariance model that has fewer basis fucntions
# (to make the example run more quickly)

  LKinfo<- LKrig.setup( x, nlevel=1, alpha=1, NC=15, a.wght=5,
                        lambda=.01)
# maximize likelihood over lambda see help( LKrig.MLE) for details
# this assumes the value of 5 for a.wght.  In many cases the fit is not
# very sensitive to the range parameter such as a.wght in this case --
# but very sensitive to lambda when varied on a log scale.

  MLE.fit<- LKrig.MLE(x,y, LKinfo=LKinfo)
  MLE.fit$summary # summary of optimization over lambda.
# fit using MLE for lambda MLE function has added MLE value of lambda to
# the LKinfo object.
  obj<- LKrig( x,y, LKinfo=MLE.fit$LKinfo.MLE)
  print( obj)

# find prediction standard errors at locations based on fixing covariance
# at MLEs
  out.se<- predictSE( obj, xnew= x)
# one could evalute the SE on a grid to get the surface of predicted SEs
# for large grids it is better to use LKrig.sim.conditional to estimate
#  the variances by Monte Carlo

##############################################################################
# Use multiresolution model that approximates an exponential covariance
# Note that a.wght realted to a range/scale parameter
# is specified at a (seemingly) arbitrary value.
##############################################################################

  LKinfo<- LKrig.setup( x, NC=6, nu=1, nlevel=3, a.wght= 5)
# take a look at the implied covariance function solid=along x
#  and  dashed=along y
  check.cov<- LKrig.cov.plot( LKinfo)
  matplot( check.cov$d, check.cov$cov, type="l", lty=c(1,2))

##############################################################################
# Search over lambda to find MLE for ozone data with approximate exponential
# covariance
##############################################################################
## Not run:

  LKinfo.temp<-  LKrig.setup( x, NC=6, nu=1,  nlevel=3, a.wght= 5)
# starting value for lambda optimzation
  LKinfo.temp$lambda<- 1
  MLE.search<- LKrig.MLE(x,y, LKinfo=LKinfo.temp)
# this function returns an LKinfo object with the MLE for lambda included.
  MLE.ozone.fit<- LKrig( x,y,  LKinfo= MLE.search$LKinfo.MLE)

## End(Not run)
##############################################################################
# Including a covariate (linear fixed part in spatial model)
##############################################################################

  data(COmonthlyMet)
```

```
  y.CO<- CO.tmin.MAM.climate
  good<- !is.na( y.CO)
  y.CO<-y.CO[good]
  x.CO<- as.matrix(CO.loc[good,])
  Z.CO<- CO.elev[good]
# single level with large range parameter -- similar to a thin plate spline
#  lambda specified

# fit with elevation
  obj.CO.elev<- LKrig( x.CO,y.CO,Z=Z.CO, nlevel=1, NC=15, alpha=1, lambda=.005,
                         a.wght=4.1)
# BTW the coefficient for the linear term for elevation  is obj.CO$d.coef[4]
# fitted surface without the elevation term
## Not run:
   LKinfo<- LKrig.setup( x.CO, nlevel=1, NC=20,alpha=1, a.wght=4.1, lambda=1.0)
  # lambda is the starting vlaue for MLE optimization
  CO.MLE<- LKrig.MLE( x.CO,y.CO,Z=Z.CO, LKinfo=LKinfo)
  obj.CO.elev<- LKrig( x.CO,y.CO,Z=Z.CO, LKinfo= CO.MLE$LKinfo.MLE)
  CO.surface2<- predictSurface( obj.CO.elev, drop.Z=TRUE, nx=50, ny=50)
# pull off CO elevations at same locations on grid as the surface
  data( RMelevation) # a superset of elevations at 4km resolution
  elev.surface<- interp.surface.grid( RMelevation, CO.surface2)
   CO.full<- predictSurface( obj.CO.elev, ZGrid= elev.surface, nx=50, ny=50)

# for comparison fit without elevation as a linear covariate:
  CO.MLE2<- LKrig.MLE( x.CO,y.CO, LKinfo=LKinfo)
  obj.CO<- LKrig( x.CO,y.CO, LKinfo= CO.MLE2$LKinfo.MLE)
# surface estimate
  CO.surface<- predictSurface( obj.CO, nx=50, ny=50)
  set.panel( 2,1)
  coltab<- topo.colors(256)
  zr<- range( c( CO.full$z), na.rm=TRUE)
  image.plot( CO.surface,  col=coltab, zlim =zr)
    US( add=TRUE,lwd=2)
    title( "MAM min temperatures without elevation")
  image.plot( CO.full, col=coltab, zlim=zr)
    title( "Including elevation")
    US( add=TRUE,lwd=2)

## End(Not run)

#############################################################################
# for a more elaborate search over  a.wght, alpha and lambda to find
# joint MLEs see help(LKrig.MLE)
#############################################################################

#############################################################################
# A bigger problem: 26K observations and 4.6K basis functions
# fitting takes about 15 seconds on a laptop for a fixed covariance
#  LKrig.MLE to find the MLE (not included) for lambda takes abou
#  8 minutes
#############################################################################
## Not run:
  data(CO2)
  obj1<- LKrig( CO2$lon.lat,CO2$y,NC=100,nlevel=1, lambda=.088,
                       a.wght=5, alpha=1)
# 4600 basis functions 100X46 lattice  (number of basis functions
```

```
# reduced in y direction because of a rectangular domain
  obj1$trA.est # about 1040 effective degrees of freedom
#
  glist<- list( x= seq( -180,180,,200),y=seq( -80,80,,100) )
  xg<-  make.surface.grid(glist)
  fhat<- predict( obj1, xg)
  fhat <- matrix( fhat,200,100) # convert to image
#Plot data and gap-filled estimate
  set.panel(2,1)
  quilt.plot(CO2$lon.lat,CO2$y,zlim=c(373,381))
  title("Simulated CO2 satellite observations")
  world(add=TRUE,col="magenta")
  image.plot( glist$x, glist$y, fhat,zlim=c(373,381))
  world( add=TRUE, col="magenta")
  title("Gap-filled global predictions")

## End(Not run)
## same example but use a periodic lattice in longitude and
## periodic basis functions. This is the "cylinder" model.
## Not run:
  data(CO2)
  LKinfo.cyl<- LKrig.setup( cbind( c( -180,180), range( CO2$lon.lat[,2])),
                            NC=25,nlevel=1, lambda=.1,
                             a.wght=5, alpha=1,  distance.type="cylinder")
# 1127 basis functions from a 49X23 lattice
  search.CO2<- LKrig.MLE( CO2$lon.lat,CO2$y, LKinfo=LKinfo.cyl)
# MLE search over lambda
  obj2<- LKrig( CO2$lon.lat,CO2$y, LKinfo=search.CO2$LKinfo.MLE)
  surface( obj2)
  world( add=TRUE)

## End(Not run)

set.panel()
#############################################################################
#  Comparing LKrig to ordinary Kriging
#############################################################################

# Here is an illustration of using the fields function mKrig with the
# LKrig covariance to reproduce the computations of LKrig. The
# difference is that mKrig can not take advantage of any sparsity in
# the precision matrix because its inverse, the covariance matrix, is
# not sparse.  This example reinforces the concept that LKrig finds the
# the standard geostatistical estimate but just uses a particular
# covariance function defined via basis functions and the precision
# matrix.
# Load ozone data set (AGAIN)
## Not run:
  data(ozone2)
  x<-ozone2$lon.lat
  y<- ozone2$y[16,]
# Find location that are not NA.
# (LKrig is not set up to handle missing observations.)
  good <-  !is.na( y)
  x<- x[good,]
  y<- y[good]
  a.wght<- 5
```

```
  lambda <-  1.5
  obj1<- LKrig( x,y,NC=16,nlevel=1, alpha=1,  lambda=lambda, a.wght=5,
               NtrA=20,iseed=122)

# in both calls iseed is set the same so we can compare
# Monte Carlo estimates of effective degrees of freedom
  obj1$trA.est
# The covariance "parameters" are all in the list LKinfo
# to create this special list outside of a call to LKrig use
  LKinfo.test <- LKrig.setup( x, NC=16, nlevel=1, alpha=1.0,  a.wght=5)

# this call to mKrig should be identical to the LKrig results
# because it uses the LKrig.cov covariance with all the right parameters.
  obj2<- mKrig( x,y, lambda=lambda, m=2, cov.function="LKrig.cov",
                      cov.args=list( LKinfo=LKinfo.test), NtrA=20, iseed=122)
# compare the two results this is also an
# example of how tests are automated in fields
# set flag to have tests print results
  test.for.zero.flag<- TRUE
  test.for.zero( obj1$fitted.values, obj2$fitted.values,
                 tag="comparing predicted values LKrig and mKrig")
# compare standard errors.
  se1<- predictSE.LKrig( obj1)
  se2<- predictSE.mKrig(obj2)
  test.for.zero( se1, se2,
                 tag="comparing standard errors for LKrig and mKrig")

## End(Not run)
#############################################################################
#  Unnormalized marginal variance for a 6X6 basis on [-1,1]X[-1,1]
#  This is an example of why normalization seems important.
#############################################################################

## Not run:
# specify covariance without normalization note all we need is the
#corners of domains to setup the info list.
  LKinfo<- LKrig.setup(cbind( c( -1,1), c(-1,1)), NC=6, nlevel=1,
                       a.wght=4.5,alpha=1, normalize= FALSE)
# 80X80 grid of points
  xg<- make.surface.grid( list(x=seq(-1,1,,80), y= seq(-1,1,,80)))
  look<- LKrig.cov( xg, LKinfo=LKinfo,marginal =TRUE)
# surface plot of the marginal variances of the process.
  image.plot( as.surface(xg, look))
# basis function centers from the first (and only) level
  xp<- make.surface.grid( LKinfo$grid[[1]])
  points(xp)

## End(Not run)
```

LKrig Internal *Internal functions for LatticeKrig package.*

Description

Some internal functions for LKrig that estimate the coefficients of the basis functions and compute the likelihood.

Usage

```
LKrig.coef(Mc, wPHI, wT.matrix, wy, lambda, weights)
LKrig.lnPlike(Mc, Q, y, lambda, residuals, weights,
 sigma = NA, rho = NA)
LKrig.traceA( Mc, wPHI, wT.matrix, lambda, weights,NtrA, iseed=NA)
LKrigUnrollZGrid( grid.list, ZGrid=NULL)
```

Arguments

grid.list	The grid for evaluting surface
iseed	Random seed used to generate the Monte Carlo samples. Keep the same to compare results with mKrig and also for multiple values of lambda.
lambda	The ratio of the nugget variance (sigma squared) to the parameter controlling the marginal variance of the process (called rho in fields).
Mc	Cholesky decomposition of regression matrix.
NtrA	Number of Monte Carlo samples to estimate trace. Default is 20 in LKrig.
Q	Precision matrix for coefficients.
residuals	Residuals from fitting spatial process.
rho	If lambda is not specified then the value of the marginal variance of the process.
sigma	If lambda is not specified then the values of the measurement error standard deviation.
wPHI	Weighted matrix of basis functions. See LKrig source for construction.
wT.matrix	Weighted matrix of fixed part of estimate.
weights	A vector that is proportional to the reciprocal variances of the errors. I.e. errors are assumed to be uncorrelated with variances sigma^2/weights.
wy	Weighted observations.
y	Spatial observation.
ZGrid	A list or array with the covariates on the same grid as that specified by the grid.list argument.

Details

LKrig.coef and LKrig.lnPlike are two low level functions to find the basis function coefficients and to evaluate the likelihood. The coefficents (c.mKrig) are also found because they provide for shortcut formulas for the standard errors and MLE estimates. These coefficients are identical to the basis coefficients (c.coef) found for usual Kriging in the mKrig function. LKrig.lnPlike also finds the profile MLE of sigma and rho given a fixed value for lambda (and alpha and a.wght). See the source for LKrig and also MLE.LKrig to see how these functions are used.

LKrig.traceA finds an estimate of the effective degrees of freedom of the smoothing matrix based a simple Monte Carlo scheme. The smoothing matrix A is the matrix for fixed covariance parameters so that y.hat = A y, where y.hat are the predicted values at the data locations. trace(A) is the effective degrees of freedom. If e are iid N(0,1) then the expected value of t(e)% * % A % * % e is equal to

the trace of A. This is the basis for estimating the trace and the standard error for this estimate is based on NtrA independent samples.

dfind2d is a fast FORTRAN subroutine to find nearest neighbors within a fixed distance and is called by Wendland.basis. The function dfind3d is currently not used but is intended for future use to determine chordal distance between points on a sphere or cylinder.

LKrigDefaultFixedFunction Is called to construct the fixed part of the spatial model. The default is a polynomial of degree (m-1).

Value

LKrig.coef a list with components d.coef the coefficients of the spatial dirft and for covariates (Z) and c.coef the basis function coefficients. The logical vector ind.drift from the LKrig object indicates with components of d.coef are associated with the polynomial spatial drift and which are other fixed spatial covariates.

LKrig.lnPlike has the components:

lnProfileLike: the log likelihood profiled for lambda, alpha and a.wght

rho.MLE: the MLE of rho given lambda, alpha and a.wght

shat.MLE: the MLE of sigma given lambda, alpha and a.wght

quad.form: the quadratic form in the exponent of the multivariate normal likelihood

lnDetCov: the log determinant of the covariance matrix in the likelihood

LKrigDefaultFixedFunctionA matrix with dimension nrow(x) and columns of the number of polynomial terms and the number of columns of Z if given.

Author(s)

Doug Nychka

See Also

LKrig, LKrig.basis

LKrig Miscellaneous Matrix Functions

Miscellaneous internal functions for LatticeKrig package.

Description

Some utility functions used internally by higher level LKrig functions. Currently these are simple functions that perform shifts of a matrix.

Usage

```
LKrig.shift.matrix( A, shift.row=0, shift.col=0, periodic=c(FALSE, FALSE))
LKrig.rowshift.periodic( A, shift.row)
LKrig.rowshift( A,shift.row,shift.col)
which.max.matrix(z)
which.max.image(obj)
expandMatrix0( A, B)
expandMatrix( ...)
expandMList( Mlist, byrow=TRUE)
```

Arguments

A	A matrix.
byrow	If TRUE matrices will be repeated row by row. If FALSE this will be done column by column.
B	Another matrix.
Mlist	A list where each component is a matrix.
obj	An image list with the usual components x, y, and z.
periodic	A vector of two logicals pertaining to rows and columns. TRUE indicates an index where the shift will be periodic – entries shifted beyond the dimensions will be wrapped to the other side.
shift.row	An integer that specifies the number of positions that the rows of the matrix are shifted.
shift.col	An integer that specifies the number of positions that the columns of the matrix are shifted.
z	A matrix.
...	Matrices to be expanded.

Details

Shift related: These functions are used to create the nearest neighbor indices for the precision matrices.

Expand related: These functions are useful for creating a sets of covariance parametes that follow a factorial pattern. For example repeating the rows of the "alpha" paramters as the "a.wght" parameters are varied. `expandMList` is particularly useful for creating a factorial design of parameters to pass to LKrig.MLE for searching the likelihood.

Value

Shift: A matrix of shifted values. Entries that are not defined due to the shift are set to NA. A column shift is done by a combination of transpose operations and a row shift.

```
A<- matrix( 1:12,3,4)
A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

#shift of 2 for rows:
 LKrig.rowshift( A, 2)
    [,1] [,2] [,3] [,4]
[1,]   NA   NA   NA   NA
[2,]   NA   NA   NA   NA
[3,]    1    4    7   10

#periodic case
LKrig.rowshift.periodic( A, 2)
     [,1] [,2] [,3] [,4]
[1,]    2    5    8   11
[2,]    3    6    9   12
[3,]    1    4    7   10
```

Expand: ExpandMListReturns a list of matrices where the original matrices are repeated so that are combinations of rows are represented. The example below illustrates. byrow=FALSE does the repetition by columns instead of rows.

```
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> B
     [,1]
[1,]   11
[2,]   12
[3,]   13
> C
[1] 100
> expandMList( list( A=A, B=B, C=C))
$A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    1    3
[4,]    2    4
[5,]    1    3
[6,]    2    4

$B
     [,1]
[1,]   11
[2,]   11
[3,]   12
[4,]   12
[5,]   13
[6,]   13

$C
     [,1]
[1,]  100
[2,]  100
[3,]  100
[4,]  100
[5,]  100
[6,]  100
```

Author(s)

Doug Nychka

LKrig.basis	*Functions for generating a multiresolution, compactly supported basis, multiresolution covariance functions and simulating from these processes.*

Description

These functions support the LKrig function. Their main function is to create and evaluate radial basis functions of varying support on a nested set of regular grids. This series of grids forms a multiresolution basis. The Gaussian process model is an expansion in these basis functions where the basis coefficients follow a Markov random field model for each resolution level. This family of functions generate the basis using sparse matrices, evaluate the covariance function of the process and also simulate realizations of the process. LKrig.cov.plot is a useful function to get a quick plot of the covariance function implied by a LatticeKrig specification.

Usage

```
#
LKrig.cov(x1, x2 = NULL, LKinfo, C = NA, marginal = FALSE)
LKrig.cov.plot( LKinfo, NP=200, center = NULL, xlim = NULL, ylim = NULL)
LKrig.basis(x1, LKinfo, verbose = FALSE, spam.format = TRUE)
LKrig.precision(LKinfo, return.B = FALSE, level.index=NA, verbose=FALSE)
LKrig.MRF.precision(mx,my, a.wght, stationary=TRUE, edge=FALSE,
                        distance.type="Euclidean")
LKrig.normalize.basis(Level, LKinfo, PHI)
LKrig.normalize.basis.fast(Level, LKinfo, x )
LKrig.quadraticform( Q, PHI)
LKrig.spind2spam(obj, add.zero.rows=TRUE)
```

Arguments

add.zero.rows	If TRUE the conversion of the sparse matrix to spam format will have at least one element in each row. If there are no elements explicitly given in obj then a element with value zero is added. This technical detail is needed to accomodate the spam format for sparse matrices.
a.wght	A scalar, vector or list that specifies the weights of the center points and possibly the neighbor weights in the spatial autoregression. In the simplest model a.wght is a scalar and is used at all levels and at all lattice points. For this case the weights for four nearest neighbors have weight -1 and so a.wght should be greater than 4 for the fields to be stationary.
C	If passed the covariance matrix will be multiplied by this vector or matrix.
center	The point in the spatial domain that is used to evalute the covariance function. The evalution is done on x and y transects through the spatial domain intersecting at center and finding the covariance with respect to this point. If NULL defaults to the center of the spatial domain.
distance.type	The distance metric used. See doc entry under LKrig.
edge	If FALSE no adjustments are made to weights on edges. edge==TRUE at this time will generate an error because edge corrections are not supported.
Level	The level of multiresolution.
level.index	If NA then the full precision matrix is found at all levels. Otherwise the precision matrix is found just for the level specified by level.index.
LKinfo	A list with components that give the information describing a multiresolution basis with a Markov random field used for the covariance of the basis coefficients. This list is created in LKrig or by LKrig.setup and returned in the output object. (See section on returned Value below for this list's description.)
marginal	If TRUE returns the marginal variance. Currently not implemented!

mx	Number of grid/lattice points in x dimension.
my	Number of grid/lattice points in the y dimension.
NP	Number of points to evaluate the covariance function along each transect of the spatial domain.
obj	An object returned by LKrig or a sparse matrix in row/column format passed to LKrig.spind2spam.
PHI	A sparse matrix of basis functions (rows index points for evaluation, columns index basis functions).
return.B	If TRUE B is returned instead of the precision matrix t(B)%*%B.
Q	A sparse precision matrix.
spam.format	If TRUE matrix is returned in sparse matrix format.
stationary	If TRUE the precision matrix uses the same values of a.wght for each lattice point and its neighbors.
x	A two column matrix of 2-dimension locations to evaluate basis functions.
x1	A two column matrix of 2-dimension locations to evaluate basis functions or the first set of locations to evaluate the covariance function or the locations for the simulated process. Rows index the different locations: to be precise x1[i,1:2] are the "x" and "y" coordinates for the i th location.
x2	Second set of locations to evaluate covariance function.
xlim	Limits in x coordinate for evaluating the covariance model. Default is the spatial domain.
ylim	Limits in y coordinate for evaluating the covariance model.Default is the spatial domain.
verbose	If TRUE intermediate steps and other debugging information are printed.

Details

The basis functions are two-dimensional radial basis functions based on the compactly supported stationary covariance function (Wendland covariance) and centered on regular grid points with the scaling tied to the grid spacing.

For a basis at the coarest level, the grid centers are generated by expanding the two sequences

```
seq(grid.info$xmin,grid.info$xmax,grid.info$delta)
seq(grid.info$ymin,grid.info$ymax,grid.info$delta)
```

into a regular grid of center points. The same spacing delta is used in both directions. The unnormalized basis functions are evaluated at locations x1 by finding the pairwise, radial distances among centers and x1, scaling by grid.info$delta * overlap and then evaluating with the function passed as RadialBasisFunction. By default this is the 2-d Wendland covariance of order 2. Perhaps the most important point about the LKrig.basis is that it is designed to return a matrix of all basis functions as a sequence of points. There is no need to have a function that evaluates basis functions individually. In R code for a set of locations x1 and a rectangular spatial domain with ranges xmin, xmax, ymin ,ymax:

```
centers<- expand.grid(seq(xmin,xmax,delta),
                      seq(ymin,ymax,delta) )
bigD<- rdist( x1, centers)/(delta*2.5)
PHI<- Wendland.function( bigD)
```

Note that there will be `nrow(centers)` basis functions generated where the precise number depends on the range of the domain and the choice of delta. The basis functions are indexed by the columns in `PHI` and this is a convention throughout this package. There will `nrow(x1)` rows in `PHI` as each basis function will be evaluted at each 2-d location.

The basis functions are then normalized by scaling the basis functions at each location so that resulting marginal variance of the process is 1. This is done to coax the covariance model closer to a stationary representation. It is easiest to express this normalization by pseudo R code:

If `Q` is the precision matrix of the basis coefficients then in R/LatticeKrig code:

```
Omega<-  solve(Q)
process.variance <- diag(PHI%*% Omega %*%t(PHI) )
PHI.normalized <-  diag(1/sqrt(process.variance)) %*% PHI
```

where `Omega` is the unnormalized covariance matrix of the basis function coefficients.

Although accurate, the code above is not an efficient algorithm to compute the unnormalized process variance. First the normalization can be done level by level rather than dealing with the entire multiresolution process at once. Also it is important to work with the precision matrix rather than the covariance. The function `LKrig.normalize.basis` takes advantage of the sparsity of the precision matrix for the coefficients and `LKrig.normalize.basis.fast` is a more efficient version when a.wght is constant for a given level and takes advantage of the Kronecker structure on the precision matrix at each level.

The precision matrix for the basis coefficients at each resolution has the form `t(B)%*% B`. These matrices for the individual levels are assembled by `LKrig.precision` as the block diagonals of a larger precision matrix for the entire vector of coefficients. Note these matrices are also created in a sparse format. The specific entries in B, the object created by `LKrig.MRF.precision`, are a first order Markov random field: without edge adjustments the diagonal elements have the value `a.wght` and the first order neighbors have the value -1.

Below we give more details on how the weights are determined. Following the notation in Lindgren and Rue a.wght= 4 + k2 with k2 greater than or equal to 0. Some schematics for filling in the B matrix are given below (values are weights for the SAR on the lattice with a period indicating zero weights).

```
                                                 ----------
   .   -1    .          |  -1    .             |  4+k2  -1
                        |                      |
  -1  4+k2  -1          | 4+k2  -1             | -1      .
                        |                      |
   .  -1     .          |  -1    .             |  .      .

Interior point        Left edge              Upper left corner
```

Previous versions of LatticeKrig considered an edge correction to reflect other boundary conditions. We have found these corrections to be numerically unstable, however, and so prefer at this time of writing adding a buffer of lattice points and using the uncorrected weights described above.

Value

`LKrig.basis` A matrix with number of rows equal to the rows of `x1` and columns equal to the number of basis functions (LKinfo$m). Attached to the matrix is an `info` attribute that contains the list passed as `LKinfo`. Usually this value is in spam sparse matrix format.

LKrig.precision For return.B ==FALSE a sparse, square matrix with dimensions of the number of basis functions. For return.B == TRUE the "B" SAR matrix is returned. This is useful for checking this function.

LKrig.MRF.precision A sparse square matrix with dimension (mx*my by mx*my) with a.wght on the diagonal and -1 in the positions for the 4 nearest neighboring points. Note that this matrix has dimensions of the number of grid points/basis functions (mx*my by mx*my) – not mx x my. So the indexing of nearest neighbors is a little more complicated that in 1-dimensional grids.

LKrig.cov**:** If C=NA a cross covariance matrix with dimensions nrow(x1) and nrow(x2) is used. If C is passed the result of multiplying the cross covariance matrix times C is used.

LKrig.sim**:** A matrix with dimensions of nrow(x1) by M. Each column are vectors of simulated values at the locations x1.

LKrig.cov.plot Evaluates the covariance specified in the list LKinfo with respect to the point center along a transects in the x and y directions intersecting this point. Note the rectangular extent of the spatial domain is part of the grid information in LKinfo. Returns components u, d and cov. Each of these are two column matrices with the first column being results in the x direction and second column in the y direction. d are the distances of the points from the center and u are the actual x or y spatial coordinates. cov are the values of the covariance function. If normalize is TRUE these will in fact be the correlation functions. To plot the returned list use

```
out<- LKrig.cov.plot(LKinfo)
matplot( out$d, out$cov, type="l")
```

LKrig.quadraticform**:** Returns a vector that is diag(t(PHI)%*% solve(Q) %*% PHI)) closely related to the marginal variance of the process.

LKrig.normalize.basis**,** LKrig.normalize.basis.fast**:** A vector of variances corresponding to the unnormalized process at the locations.

Author(s)

Doug Nychka

See Also

LKrig, mKrig, Krig, fastTps, Wendland

Examples

```
# Load ozone data set
  data(ozone2)
  x<-ozone2$lon.lat
  y<- ozone2$y[16,]
# Find location that are not NA.
# (LKrig is not set up to handle missing observations.)
  good <-  !is.na( y)
  x<- x[good,]
  y<- y[good]
  LKinfo<- LKrig.setup( x,NC=20,nlevel=1, alpha=1, lambda= .3 , a.wght=5)
# BTW lambda is close to MLE

# What does the  LatticeKrig covariance function look like?
# set up LKinfo object
# NC=10 sets the grid for the first level of basis functions
# NC^2 = 100 grid points in first level if square domain.
```

```
# given four levels the number of basis functions
# = 10^2 + 19^2 +37^2 + 73^2 = 5329
# effective range scales as roughly kappa where a.wght =  4 + kappa^2
# or exponential decreasing marginal variances for the components.
    NC<- 10
    nlevel<- 4
    a.wght<- rep(  4 + 1/(.5)^2, nlevel)
    alpha<-  1/2^(0:(nlevel-1))
    LKinfo2<- LKrig.setup( cbind( c( -1,1), c(-1,1)), NC=NC,
                   nlevel=nlevel, a.wght=a.wght,alpha=alpha)
# evaluate covariance  along the  horizontal line through
# midpoint of region -- (0,0) in this case.
    look<- LKrig.cov.plot( LKinfo2)
# a plot of the covariance function in x and y with respect to (0,0)
    set.panel(2,1)
    plot(look$u[,1], look$cov[,1], type="l")
    title("X transect")
    plot(look$u[,2], look$cov[,2], type="l")
    title("Y transect")
    set.panel(1,1)
#
#
## Not run:
# full 2-d view of the covariance (this example follows the code
# in LKrig.cov.plot)
 x2<- cbind( 0,0)
 x1<- make.surface.grid( list(x=seq( -1,1,,40),  y=seq( -1,1,,40)))
 look<- LKrig.cov( x1,x2, LKinfo2)
 contour( as.surface( x1, look))
# Note nearly circular contours.
# of  course  plot(look[,80/2]) should look like plot above.
#

## End(Not run)

## Not run:
#Some correlation functions from different models
set.panel(2,1)
# a selection of ranges:
  hold<- matrix( NA, nrow=150, ncol=4)
  kappa<- seq( .25,1,,4)
  x2<- cbind( 0,0)
  x1<-  cbind( seq(-1,1,,150), rep( 0,150))
  for( k in 1:4){
    LKtemp<-  LKrig.setup( cbind( c( -1,1), c(-1,1)), NC=NC,
                   nlevel=nlevel,
                   a.wght= 4  + 1/(kappa[k]^2),
                   alpha=alpha)
    hold[,k]<-  LKrig.cov( x1,x2, LKinfo=LKtemp)
  }
  matplot( x1[,1], hold, type="l", lty=1, col=rainbow(5), pch=16 )
# a selection of smoothness parameters
  ktemp<- .5 # fix range
  alpha.power<- seq( 1,4,,4)
  LKtemp<- LKinfo2
  for( k in 1:4){
   LKtemp<-  LKrig.setup( cbind( c( -1,1), c(-1,1)), NC=NC,
```

```
                  nlevel=nlevel,
                  a.wght= 4  + 1/(ktemp^2),
                  alpha=alpha^alpha.power[k])
    hold[,k]<-  LKrig.cov( x1,x2, LKinfo=LKtemp)
  }
  matplot( x1[,1], hold, type="l", lty=1, col=rainbow(5) )
 set.panel()

## End(Not run)

## Not run:
# generating a basis on the domain [-1,1] by [-1,1] with 1 level
# Default number of buffer points are added to each side.
  LKinfo<- LKrig.setup(cbind( c(-1,1), c(-1,1)), NC=6,
                                nlevel=1, a.wght=4.5,alpha=1, NC.buffer=0 )
# evaluate the basis functions on a grid to look at them
  xg<- make.surface.grid( list(x=seq(-1,1,,50), y= seq(-1,1,,50)))
  PHI<- LKrig.basis( xg,LKinfo)
  dim(PHI) # should be  2500=50^2  by  36=6^2
# plot the 9th basis function  as.surface is a handy function to
# reformat the vector as an image object
# using the grid information in an attribute of the grid points
  image.plot(as.surface(xg, PHI[,9]))
  points(  make.surface.grid( LKinfo$grid[[1]]), col="grey", cex=.5)

set.panel()

## End(Not run)
#
# example of basis function indexing
#
## Not run:
# generating a basis on the domain [-1,1]X[-1,1] with 3 levels
# note that there are no buffering grid points.
  set.panel(3,2)
  LKinfo<-LKrig.setup(cbind( c(-1,1), c(-1,1)), NC=6,
                   a.wght=rep(5,3), alpha=c(1,.5,.25), nlevel=3,
                   NC.buffer=0)
# evaluate the basis functions on a grid to look at them
  xtemp<- seq(-1,1,,40)
  xg<- make.surface.grid( list(x=xtemp, y= xtemp) )
  PHI<- LKrig.basis( xg,LKinfo)
# coerce to dense matrix format to make plotting easier.
  PHI<- spam2full(PHI)
# first tenth, and last basis function in each resolution level
# basis functions centers are added
  set.panel(3,3)
  grid.info<- LKinfo$grid.info
  for(  j in 1:3){
    id1<- LKinfo$offset[j]+ 1
    id2<-  LKinfo$offset[j]+ 10
    idlast<- LKinfo$offset[j]+ LKinfo$mx[j]*LKinfo$my[j]

    centers<-  make.surface.grid( LKinfo$grid[[j]] )
    image.plot( as.surface(xg, PHI[,id1]))
    points( centers, cex=.2, col="grey")
    image.plot(as.surface(xg, PHI[,id2]))
```

```
    points( centers, cex=.2, col="grey")
    image.plot( as.surface(xg, PHI[,idlast]))
    points( centers, cex=.2, col="grey")}

  set.panel()

## End(Not run)
## Not run:
# examining the stationarity of covariance model
  temp.fun<-
     function( NC.buffer=0, NC=4,  a.wght=4.01){
        LKinfo<- LKrig.setup(cbind( c(-1,1), c(-1,1)),nlevel=1, alpha=1,
                                a.wght=a.wght, NC=NC,  NC.buffer=NC.buffer)
        cov1y<- cov1x<- cov0x<- cov0y<-  matrix( NA, nrow=200, ncol=20)
        cov1dx<- cov1dy<- cov0dx<- cov0dy<- matrix( NA, nrow=200, ncol=20)
        cgrid<- seq( 0,1,,20)
        for( k in 1:20){
            hold<- LKrig.cov.plot( LKinfo,
                           center=rbind( c(cgrid[k], cgrid[k])), NP=200)
            cov1x[,k] <- hold$cov[,1]
            cov1y[,k] <- hold$cov[,2]
            cov1dx[,k] <- hold$d[,1]
            cov1dy[,k] <- hold$d[,2]
            hold<- LKrig.cov.plot( LKinfo,
                            center=rbind( c(cgrid[k],0) ), NP=200)
            cov0x[,k] <- hold$cov[,1]
            cov0y[,k] <- hold$cov[,2]
            cov0dx[,k] <- hold$d[,1]
            cov0dy[,k] <- hold$d[,2]
                }
         matplot( cov1dx, cov1x, type="l", col= rainbow(20),
                         xlab="", ylab="correlation")
         mtext( side=1, line=-1, text="diagonal X")
         title( paste(  " buffer=",NC.buffer), cex=.5)
         matplot( cov1dy, cov1y, type="l", col= rainbow(20),
                        xlab="", ylab="")
         mtext( side=1, line=-1, text="diagonal Y")
         matplot(cov0dx, cov0x, type="l", col= rainbow(20),
                        xlab="",       ylab="")
         mtext( side=1, line=-1, text="middle X")
         matplot( cov0dy, cov0y, type="l", col= rainbow(20),
                         xlab="",   ylab="")
         mtext( side=1, line=-1, text="middle Y")
         title( paste( NC, a.wght), cex=.5)
}


 set.panel(3,4)
par(mar=c(3,4,1,0), oma=c(1,1,1,1))
temp.fun(  NC.buffer=5, NC=4, a.wght=4.05)
temp.fun(  NC.buffer=5, NC=16, a.wght=4.05)
temp.fun(  NC.buffer=5, NC=64, a.wght=4.05)

set.panel(4,4)
par(mar=c(3,4,1,0), oma=c(1,1,1,1))
temp.fun( NC.buffer=0, NC=8)
temp.fun( NC.buffer=2, NC=8)
```

```
temp.fun( NC.buffer=4, NC=8)
# this next one takes a while
temp.fun( NC.buffer=8,  NC=8)
# stationary == curves should all line up!


## End(Not run)
```

LKrig.MLE *Simple function to search over covariance parameters for Lattice Krig*

Description

Given a list of different covariance parameters for the Lattice Krig covariance model this function computes the likelihood or a profiled version (over lambda) and approximates a generalized cross-validation function at each of the parameter settings. This is an experimental function that has been productively used with a Latin hypercube design package to efficiently search through the LatticeKrig covariance parameter space.

Usage

```
LKrigFindLambda(x, y, ..., LKinfo,
                            use.cholesky=NULL,
                            verbose = FALSE,
                            lambda.profile=TRUE,
                            lowerBoundLogLambda=-16,tol = 0.005)

LKrig.MLE( x,y,..., LKinfo, use.cholesky = NULL,
                    par.grid=NULL,
                    lambda.profile=TRUE,
                    verbose=FALSE,
                    lowerBoundLogLambda = -16,
                    nTasks = 1, taskID = 1,
                    tol = 0.005)
LKrig.make.par.grid(par.grid=NULL, LKinfo = NULL)
```

Arguments

x	The spatial locations.
y	The observations.
par.grid	A list with components llambda, alpha, a.wght giving the different sets of parameters to evaluate. If M is the number of parameter setting to evaluate llambda is a vector length M and alpha and a.wght are matrices with M rows and nlevel columns. Thus, the kth trial has parameters par.grid\$llambda[k], par.grid\$alpha[k,] and par.grid\$a.wght[k,]. Currently this function does not support passing a non-stationary spatial parameterization for alpha. The LKinfo object details the other parts of the covariance specification (e.g. number of levels, grid sizes) that do not change. Note that par.grid assumes *ln* lambda not lambda. See details below for some other features of the par.grid arguments.

`lambda.profile` A logical value controlling whether the likelihood is maximized over lambda. For LKrigFindLambda if TRUE the likelihood is maximized over lambda at the covariance values in LKinfo and if FALSE the likelihood is just evaluated at LKinfo including the lambda value in this list. For LKrig.MLE if TRUE for each set of parameters in par.grid the value of lambda is found that maximizes the likelihood. In this case the llambda value is the starting value for the optimizer. If llammbda[k] is NA then the lambda value found from the k-1 maximization is used as a starting value for the k step. (In the source code this is `llambda.opt`.) Of course this only makes sense if the other parameters are ordered so that the results for k-1 make sense as a lambda starting value for k. If FALSE the likelihood is evaluated for the covariance parameters at the kth positions in the par.grid list including lambda.

`LKinfo` An LKinfo object that specifies the LatticeKrig covariance. Usually this is obtained by a call to `LKrig.setup` or as the component LKinfo from the LKrig object. The search sequentially replaces the alpha and a.wght arguments in this list by the values in par.grid but leaves everything else the same. If par.grid is not passed the parameter values in LKinfo are used to evaluate the likelihood. This option is most useful if one has fixed values of alpha and a.wght and the goal is to maximze the likelihood over lambda.

`lowerBoundLogLambda` Lower limit for lambda in searching for MLE.

`nTasks` If using Rmpi the number of slaves available.

`tol` tolerance on log likelihood use to determine convergence

`taskID` If using Rmpi the slave id.

`verbose` If TRUE prints out intermediate results.

`use.cholesky` If not NULL then this object is used as the symbolic cholesky decomposition of the covariance matrix for computing the likelihood.

`...` Any arguments to be passed to LKrig. E.g. `x`, `y`, `Z` a covariate or `weights`.

Details

`LKrigFindLambda`: Uses a simple one dimensional optimizer `optimize`. To maximize the log likelihood for log lambda over the range: llambda.start + [-8,5]. This function is used to determine lambda in `LatticeKrig`.

`LKrig.MLE`: This is a simple wrapper function to accomplish repeated calls to the LKrig function to evaluate the profile likelihood and/or to optimize the likelihood over the lambda parameters. The main point is that maximization over the lambda parameter (or equivalently for sigma and rho) is the most important and should be done before considering variation of other parameters. If lambda is specified then one has closed form expressions for sigma, rho that can then be substituted back into the log full likelihood. This operation that is the default throughout LatticeKrig (and fields) can concentrate the likelihood on a reduced set of parameters. The further refinement when `lambda.profile==TRUE` is to maximized the concentrated likelihood over lambda and report this result. This will be a profile likelihood over the remaining parameters of the covariance.

The covariance/model parameters are alpha, a.wght, and log lambda and are separate matrix or vector components of the `par.grid` list. The cleanest version of this function would just require the `par.grid` list, however, to be easier to use there are several options to give partial information and let the function itself create the master parameter list. For example, just a search over lambda should be easy and not require creating `par.grid` outside the function. To follow this option one can just give an LKinfo object. The value for the lambda component in this object will be the starting value with the default starting value being lambda =1.0.

In the second example below most of the coding is getting the grid of parameters to search in the right form. It is useful to normalize the alpha parameters to sum to one so that the marginal variance of the process is only parameterized by rho. To make this easy to implement there is the option to specify the alpha parameters in the form of a mixture model so that the components are positive and add to one. (the gamma variable below). If a component `gamma` is passed as a component of `par.grid` then this is assumed to be in the mixture model form and the alpha weights are computed from this. Note that `gamma` will be a matrix with (nlevel - 1) columns while `alpha` has nlevel columns.

For those readers that use `which.max` these functions are natural extensions and are handy for looking at interpolated surfaces of the likelihood function.

`which.max.matrix` Finds the maximum value in a matrix and returns the row/column index.

`which.max.image` Finds the maximum value in an image matrix and returns the index and the corresponding grid values.

`LKrig.make.par.grid` This is usually used as an internal function that converts the list of parameters in par.grid and the LKinfo object into an more complex data structure used by LKrig.MLE. Its returned value is a "list of lists" to make the search over different parameters combinations simple.

Value

LKrigFindLambda

`summary`	Giving information on the optimzation over lambda.
`LKinfo`	Covariance information object.
`llambda.start, lambda.MLE`	Initial and final values for lambda.
`lnLike.eval`	Matrix with all values of log likelihood that were evaluated
`call`	Calling arguments.
`Mc`	Cholesky decomposition.

LKrig.MLE

`summary`	A matrix with columns: effective degrees of freedom, ln Profile likelihood, Generalized cross-validation function, MLE sigma, MLE rho, full likelihood and number of parameter evalutations. The rows correspond to the different parameters in the rows of the `par.grid` components.
`par.grid`	List of parameters used in search. Some parameters might be filled in from the initial par.grid list passed and also from `LKinfo`.
`LKinfo`	LKinfo list that was either passed or created.
`index.MLE`	Index for case that has largest Likelihood value.
`index.GCV`	Index for case that has largest GCV value.
`LKinfo.MLE`	LKinfo list at the parameters with largest profile likelihood.
`lambda.MLE`	Value of lambda from grid with largest profile likelihood.
`call`	Calling sequence for this function.

which.max.matrix Returns a 2 column matrix with row and column index of maximum.

which.max.image Returns components x,y,z locating maximum value and component in giving the row and column of maximum in the image matrix.

LKrig.make.par.grid Returns a list with components, alpha, a.wght. Each component is a list where each component of the list is a separate set of parameters. This more general format is useful for the nonstationary case when the parameters alpha might be a list of nlevel matrices.

Author(s)

Douglas Nychka

See Also

LKrig LatticeKrig

Examples

```
#

# fitting summer precip for  sub region of North America
  data(NorthAmericanRainfall)
# rename for less typing
  x<- cbind( NorthAmericanRainfall$longitude, NorthAmericanRainfall$latitude)
# total precip in 1/10 mm for JJA
 y<- log10(NorthAmericanRainfall$precip)
# cut down the size of this data set so examples run quickly
# examples also work with  the full data set. Also try NC= 100 for a
# nontrivial model.
  ind<- x[,1] > -90 & x[,2] < 35 #
  x<- x[ind,]
  y<- y[ind]

# This is a single level smoother
  LKinfo<- LKrig.setup(x,NC=12,nlevel=1, a.wght=4.05, alpha=1.0)
  lambdaFit<- LKrigFindLambda( x,y,LKinfo=LKinfo)
  lambdaFit$summary

  NG<-5
 #NOTE: make this larger ( e.g. 15) for a better grid search on log lambda
  par.grid<- list( a.wght= rep( 4.05,NG),alpha= rep(1, NG),
                     llambda=  seq(-8,-2,,NG))
  LKinfo<- LKrig.setup(x,NC=12,nlevel=1, a.wght=5, alpha=1.0)
  lambda.search.results<-LKrig.MLE( x,y,LKinfo=LKinfo,
                                    par.grid=par.grid,
                                    lambda.profile=FALSE)
  lambda.search.results$summary
# profile likelihood
  plot( lambda.search.results$summary[,1:2],
         xlab="effective degrees freedom",
         ylab="ln profile likelihood")
# fit at largest likelihood value:
    lambda.MLE.fit<- LKrig( x,y,
                    LKinfo=lambda.search.results$LKinfo.MLE)
# optimizing  Profile likelihood over lambda using optim
# consider 3 values for a.wght (range parameter)
# in this case the log lambdas passed are the starting values for optim.
  NG<-3
  par.grid<- list( a.wght= c( 4.05,4.1,5),alpha= rep(1, NG),
                     llambda= c(-5,NA,NA))
# NOTE: NAs in llambda mean use the previous MLE for llambda as the
# current starting value.
  LKinfo<- LKrig.setup(x,NC=12,nlevel=1, a.wght=5, alpha=1.0)
  lambda.search.results<-LKrig.MLE(
                              x,y,LKinfo=LKinfo, par.grid=par.grid,
```

```
                              lambda.profile=TRUE, verbose=TRUE)
  print(lambda.search.results$summary)
# note first result a.wght = 4.05 is the optimized result for the grid
# search given above.

#########################################################################
# search over two multi-resolution levels varying the  levels of alphas
#########################################################################

# NOTE: search ranges found largely by trial and error to make this
# example work also the grid is quite coarse ( and NC is small) to
# be quick as a help file example

  Ndes<- 10  # NOTE: this is set very small just to make example run fast
  set.seed(124)
  par.grid<- list()
# create grid of alphas to sum to 1 use a mixture model parametrization
#  alpha1 = (1/(1 + exp(gamma1)) ,
# alpha2 = exp( gamma1) / ( 1 + exp( gamma1))
#
  par.grid$gamma<- cbind(runif( Ndes, -3,2), runif( Ndes, -3,2))
  par.grid$a.wght<- matrix( 4.5, nrow=Ndes, ncol=3)
# log lambda grid search values
  par.grid$llambda<- runif( Ndes,-5,-3)
   LKinfo1<- LKrig.setup( x, NC=5, nlevel=3, a.wght=5, alpha=c(1.0,.5,.25))
# NOTE: a.wght in call is not used. Also a better search is to profile over
#  llambda

 alpha.search.results<- LKrig.MLE( x,y,LKinfo=LKinfo1, par.grid=par.grid,
                                   lambda.profile=FALSE)

#########################################################################
# Viewing the search results
#########################################################################

# this scatterplot is good for a quick look because  effective degrees
# of freedom is a useful summary of fit.
  plot( alpha.search.results$summary[,1:2],
         xlab="effective degrees freedom",
         ylab="ln profile likelihood")
#
## Not run:
# a more defensible two level model search
# with profiling over lambda.
#  This takes a few minutes
  Ndes<- 40
  nlevel<-2
  par.grid<- list()
## create grid of alphas to sum to 1 use a mixture model parametrization:
#    alpha1 = (1/(1 + exp(gamma1)) ,
#   alpha2 = exp( gamma1) / ( 1 + exp( gamma1))
  set.seed(123)
  par.grid$gamma<- runif( Ndes,-3,4)
## values for range (a.wght)
  a.wght<-  4 + 1/exp(seq( 0,4,,Ndes))
  par.grid$a.wght<- cbind( a.wght, a.wght)
# log lambda grid search values (these are the starting values)
```

```
  par.grid$llambda<- rep(-4, Ndes)
  LKinfo1<- LKrig.setup( x, NC=15, nlevel=nlevel,
                         a.wght=5, alpha=c(1.0,.5,.25) )
##
## the search over the parameter list in par.grid  maximizing over lambda
  search.results<- LKrig.MLE( x,y,LKinfo=LKinfo1, par.grid=par.grid,
                              lambda.profile=TRUE)
# plotting results
set.panel(1,2)
 plot( search.results$summary[,1:2],
        xlab="effective degrees freedom",
        ylab="ln profile likelihood")
 xtemp<- matrix(NA, ncol=2, nrow=Ndes)
 for( k in 1:Ndes){
   xtemp[k,] <- c( (search.results$par.grid$alpha[[k]])[1],
                (search.results$par.grid$a.wght[[k]])[1] )
}
 quilt.plot( xtemp,search.results$summary[,2])
#  fit using Tps
 tps.out<- Tps(  xtemp,search.results$summary[,2], lambda=0)
 contour( predictSurface(tps.out), lwd=3,add=TRUE)
 set.panel()

## End(Not run)
## Not run:
# searching over nu
data(ozone2)
x<- ozone2$lon.lat
y<- ozone2$y[16,]
good<- !is.na(y)
y<- y[good]
x<- x[good,]
par.grid<- expand.grid( nu=seq(.5,1.5,,4), a.wght=c(4.01,4.1, 4.2,4.5,5))
par.grid$llambda<- rep( NA, length(par.grid$nu))
LKinfo<- LKrig.setup(x,  nlevel=3,NC=5)
out<- LKrig.MLE( x,y, LKinfo=LKinfo, par.grid=par.grid, verbose=TRUE)

## End(Not run)
```

`LKrig.sim` *Functions for simulating a multiresolution process following the Lattice Krig covariance model.*

Description

The fields are Gaussian and can be either simulated unconditionally or conditional on the field values and a set of irregular locations.

Usage

```
#
LKrig.sim(x1, LKinfo, M=1,just.coefficients = FALSE)
LKrig.sim.conditional( obj, M=1, x.grid= NULL, grid.list=NULL,
                           nx=80, ny=80,...,Z.grid=NULL)
```

Arguments

grid.list	Specifies a grid of spatial locations using the grid.list format (help(grid.list)). These are the locations used to evaluate the fields generated from conditional simulation. The default is to generate an 80X80 grid based on range of the observations.
just.coefficients	If TRUE just simulates the coefficients from the Markov Random field.
LKinfo	A list with components that give the information describing a multiresolution basis with a Markov random field used for the covariance of the basis coefficients. This list is created in LKrig and is returned as part of the output object or in a more hands on manner directly using LKrig.setup (See section on returned Value below for this list's description.)
M	Number of independent simulated fields.
nx	Number of grid points in x coordinate for output grid.
ny	Number of grid points in y coordinate for output grid.
obj	An LKrig object, i.e. the output list returned by LKrig.
x1	A two column matrix of 2-dimension locations to evaluate basis functions or the first set of locations to evaluate the covariance function or the locations for the simulated process. Rows index the different locations: to be precise x1[i,1:2] are the "x" and "y" coordinates for the i th location.
x.grid	Locations to evaluate conditional fields. This is in the form of a two column matrix where each row is a spatial location.
Z.grid	The covariates that are assoicated with the x.grid values. This is useful for conditional simulation where the fields are evaluated at x.grid locations and using covariate values Z.grid. Z.grid is matrix with columns indexing the different covariates and rows indexed by the x.grid locations.
...	Arguments to be passed to the LKrig function to specify the spatial estimate. These are components in addition to what is in the LKinfo list returned by LKrig.

Details

The simulation of the unconditiational random field is done by generating a draw from the multiresolution coefficients using a Cholesky decomposition and then multiplying by the basis functions to evaluate the field at arbitrary points. Currently there is no provision to exploit the case when one wants to simulate the field on a regular grid. The conditional distribution is a draw from the multivariate normal for the random fields conditioned on the observations and also conditioned on covariance model and covariance parameters. If the nugget/measurement error variance is zero then any draw from the conditional distribution will be equal to the observations at the observation locations. In the past conditional simulation was known to be notoriously compute intensive, but the major numerical problems are finessed here by exploiting sparsity of the coefficient precision matrix.

The conditional field is found using a simple trick based on the linear statistics for the multivariate normal. One generates an unconditional field that includes the field values at the observations. From this realization one forms a synthetic data set and uses LKrig to predict the remaining field based on the synthetic observations. The difference between the predicted field and the realization (i.e. the true field) is a draw from the conditional distribution with the right covariance matrix. Adding the conditional mean to this result one obtains a draw from the full conditional distribution. This algorithm can also be interpreted as a variant on the bootstrap to determine the estimator uncertainty. The fixed part of the model is also handled correctly in this algorithm. See the commented source for LKrig.sim.conditional for the details of this algorithm.

Value

`LKrig.sim`: A matrix with dimensions of `nrow(x1)` by `M` of simulated values at the locations `x1`.

`LKrig.sim.conditional`: A list with the components.

xgrid The locations where the simulated field(s) are evaluated.

ghat The conditional mean at the xgrid locations.

g.draw A matrix with dimensions of `nrow(x.grid)` by `M` with each column being an independent draw from the conditional distribution.

Author(s)

Doug Nychka

See Also

LKrig, mKrig, Krig, fastTps, Wendland

Examples

```
# Load ozone data set
  data(ozone2)
  x<-ozone2$lon.lat
  y<- ozone2$y[16,]
# Find location that are not NA.
# (LKrig is not set up to handle missing observations.)
  good <-  !is.na( y)
  x<- x[good,]
  y<- y[good]
  LKinfo<- LKrig.setup( x,NC=20,nlevel=1, alpha=1, lambda= .3 , a.wght=5)
# BTW lambda is close to MLE
# Simulating this  LKrig process
# simulate 4 realizations of process and plot them
# (these have unit marginal variance)
  xg<- make.surface.grid(list( x=seq( -87,-83,,40), y=seq(36.5, 44.5,,40)))
  out<- LKrig.sim(xg, LKinfo,M=4)
## Not run:
  set.panel(2,2)
  for( k in 1:4){
    image.plot( as.surface( xg, out[,k]), axes=FALSE) }

## End(Not run)
  obj<- LKrig(x,y,LKinfo=LKinfo)
  O3.cond.sim<- LKrig.sim.conditional( obj, M=3,nx=40,ny=40)
## Not run:
  set.panel( 2,2)
  zr<- range( c(  O3.cond.sim$draw,  O3.cond.sim$ghat), na.rm=TRUE)
  coltab<- tim.colors()
  image.plot( as.surface( O3.cond.sim$x.grid, O3.cond.sim$ghat), zlim=zr)
  title("Conditional mean")
  US( add=TRUE)
  for( k in 1:3){
    image( as.surface( O3.cond.sim$x.grid, O3.cond.sim$g.draw[,k]),
              zlim=zr, col=coltab)
    points( obj$x, cex=.5)
    US( add=TRUE)
```

```
  }
  set.panel()

## End(Not run)
```

LKrigDefaultFixedFunction

Creates fixed part of spatial model.

Description

Creates matrix of low order polynomial in the spatial coordinates and adds any other spatial covariates that are part of the linear model.

Usage

```
 LKrigDefaultFixedFunction(x, Z = NULL, m=2,distance.type="Euclidean")

predictLKrigFixedFunction(object, xnew, Znew = NULL, drop.Z = FALSE)
```

Arguments

drop.Z	If TRUE only spatial drift is evaluated the comtribution for covariates is omitted.
distance.type	The distance metric. See the entry in LKrig for details.
m	The order of the polynomial. Following the convention for splines the polynomial will have maximum order (m-1). Throughout LKrig m==2 is the default giving a linear polynomial.
object	An LKrig object.
x	A 2 column matrix of 2-d locations to evaluate the polynomial.
xnew	Locations for predictions.
Z	A matrix specifying additional spatial covariates.
Znew	Same as Z.

Details

LKrigDefaultFixedFunction This function creates the regression matrix for the fixed part of the spatial model. The default is a low order polynomial regression matrix of degree m-1. To this matrix are bound as columns any covaraites passed as Z. Typically one would not need to modify this function. For more exotic fixed part models one can specify create and then specify a different function. See LKrig.setup and LKrig. NOTE: If the argument for this function is passed as NULL then the subsquent computations do not include a fixed part in the model.

predictLKrigFixedFunction This function is simple, but is introduced to make the code modular and to handle the case for cylindrical geometry where only latitude should have a spatial term (to preserve periodicity in longitude).

Value

A matrix where rows index the locations and columns are the different spatial polynomial and covariates.

Author(s)

Doug Nychka

See Also

LKrig.basis, LKrig

Examples

```
x<- matrix( runif(100), nrow=50)
# linear polynomial
T.matrix<- LKrigDefaultFixedFunction(x, m=2)
# quadratic polynomial
T.matrix<- LKrigDefaultFixedFunction(x, m=3)
```

Radial.basis *Two dimensional radial basis functions based on a Wendland function.*

Description

Two dimensional radial basis functions based on a Wendland function and using sparse matrix format to reduce the storage.

Usage

```
Radial.basis(x1, centers, delta, max.points = NULL, mean.neighbor =
                50, just.distance = FALSE, RadialBasisFunction =
                "WendlandFunction",distance.type="Euclidean")
LKrig.cyl( U, R=1 )
WendlandFunction(d)
```

Arguments

x1	A 2 column matrix of 2-d locations to evaluate the basis functions. Each row of x1 is a location.
centers	A two column matrix specifying the basis function centers.
d	A vector of distances.
delta	A vector of scale parameters for the basis functions.
max.points	Maximum number of nonzero entries expected for the returned matrix.
distance.type	The distance metric. See the entry in LKrig for details.
mean.neighbor	Average number of centers that are within delta of each x1 location. For centers on a regular grid this is often easy to estimate.
just.distance	For debugging, don't evaluate the RBF just return the pairwise distance matrix.
R	Radius used for cylinder coordinates.
RadialBasisFunction	The positive definite kernel used to generate the basis. The assumption is that this will take a nonnegative argument and be zero outside [0,1].
U	A two column matrix giving the locations in cylindrical coordinates. First column is the angle in degrees and second column is the height.

Details

This function finds the pairwise distances between the points x1 and centers and evaluates the function RadialBasisFunction at these distances scaled by delta. In most applications delta is constant, but a variable delta could be useful for lon/lat regular grids. The Wendland function is for 2 dimensions and smoothness order 2. See `WendlandFunction` for the polynomial form. This code has a very similar function to the fields function `wendland.cov`.

In pseudo R code for delta a scalar `Wendland.basis` evaluates as

```
BigD<- rdist( x1,centers)
WendlandFunction(BigD/delta)
```

The actual code uses a FORTRAN subroutine to search over distances less than delta and also returns the matrix in sparse format.

The function `LKrig.cyl` transforms coordinates on a cylinder, e.g. lon/lat when taken as a Mercator projection, and returns the 3-d coordinates. It is these 3-d coordinates that are used to find distances to define the radial basis functions. For points that are close this "chordal" type distance will be close to the geodesic distance on a cylinder but not identical.

Value

For `Wendland.basis` a matrix in sparse format with number of rows equal to nrow(x1) and columns equal to nrow(center).

Author(s)

Doug Nychka

See Also

LKrig.basis

Examples

```
x<- cbind( runif(100), runif(100))
center<- expand.grid( seq( 0,1,,5), seq(0,1,,5))
# coerce to matrix
center<- as.matrix(center)

PHI<- Radial.basis(x, center, delta=.5)

# LKrig with a different radial basis function.
#
  data(ozone2)
  x<-ozone2$lon.lat
  y<- ozone2$y[16,]
# Find location that are not NA.
# (LKrig is not set up to handle missing observations.)
  good <-  !is.na( y)
  x<- x[good,]
  y<- y[good]
  obj<- LKrig(x,y,NC=30,nlevel=1, alpha=1, lambda=.01, a.wght=5)
  triweight<- function( d){
       ifelse( abs(d)<=1, (1-d^2)^3, 0)}
  obj1<- LKrig(x,y,NC=30,nlevel=1, alpha=1,
    lambda=.01, a.wght=5, RadialBasisFunction="triweight", overlap=1.8)
```

Index