Package ‘KriSp’

November 21, 2006

Version 0.4

Date 2006-10-26

Title Covariance Tapering for Interpolation

Author Reinhard Furrer <rfurrer@mines.edu>

Maintainer Reinhard Furrer <rfurrer@mines.edu>

Depends fields, SparseM

Description This package provides functions for covariance tapering for interpolation of

URL http://www.mines.edu/~rfurrer/software/KriSp

R topics documented:

anomaly — *Precipitation anomalies of April 1948 in the US*

Description

The data are anomalies of aggregated monthly precipitation for April 1948 at 11,918 stations in the US.

Usage

```
data(anomaly)
```

Format

A list containing the following components. `x`: longitude-latitude position of measurement locations, `Y`: are precipitation anomalies.

Source

http://www.image.ucar.edu/GSP/Data/US.monthly.met/

References

Johns, C., Nychka, D., Kittel, T., and Daly, C. (2003). Infilling sparse records of spatial fields. Journal of the American Statistical Association, 98, 796-806.

See Also

`simple.data`.

Examples

```
data(anomaly)
plot(anomaly.data$x)
```

cite.KriSp — *Citing Package KriSp in Publications*

Description

How to cite the package `KriSp` in publications.

Usage

```
cite.KriSp()
```

Details

Execute function `cite.KriSp()` for information on how to cite KriSp in publications.

Examples

```
cite.KriSp()
```

---

covapprox.error *Error of linear covariance approximation*

Description

Evaluates different error measures of the linear covariance approximation used in the kriging approach

Usage

```
covapprox.error(n, cov.fun='expo.cov', cov.fun.args=list(range=1),
          taper.fun='spher.cov', taper.fun.args=list(range=1),
          hmax=taper.fun.args$range,nres=25)
```

Arguments

`n`	Number of knots to evaluate the approximation
`cov.fun`	Covariance function in the form of an R function, or its name as a string.
`cov.fun.args`	A list with the arguments to call the covariance function (in addition to the locations).
`taper.fun`	Taper function in the form of an R function, or its name as a string.
`taper.fun.args`	A list with the arguments to call the taper function (in addition to the locations).
`hmax`	Distance over which the error is calculated.
`nres`	Resolution over which the errors are calculated.

Details

`n=1000` usually gives a maximum error smaller than 0.1 percent of the total sill.

Value

A list with the elements:

`max`	Maximum of the error.
`ise`	Approximation of the integrated squared error.
`iae`	Approximation of the integrated absolute error.
`covapprox`	Linear approximation of the covariance.
`error`	Error committed using the linear interpolation.

Note

To obtain a 'small' tapering effect, the taper range can be set to a large value. In such a case, `hmax` should be set to the diameter of the domain.

Using the `tophat` function as taper, no tapering is done.

See Also

Covariance functions such as `expo.cov`, `mater.cov`, etc.

`Krig.sparse` and `Krig.simple.sparse`.

Examples

```
# plot the error using the default functions.
nres <- 10
n <- 25
h <- seq(0,to=1, l=n*nres)
plot( h, covapprox.error(n,nres=nres)$error, type='l')

# evaluate error for a covariance only.
expoapprox <- covapprox.error(1000,taper.fun=tophat,
               taper.fun.args=list(range=5))
# all values are negative, as expo.cov is convex.
```

`covariance` *Theoretical Distance Based Covariance Functions*

Description

Computes theoretical covariance function at supplied distance values. Models include exponential, spherical, Matern and compactly supported covariances.

Usage

```
spher.cov(distance, range, sill=1, nugget=0, eps=1.0e-7,...)
expo.cov(distance, range, sill=1, nugget=0, effect=FALSE, eps=1.0e-7,...)
mater.cov(distance, smooth, range, sill=1, nugget=0, eps=1.0e-7,...)
tri.cov(distance, range, sill=1, nugget=0, eps=1.0e-7,...)
Wu1.cov(distance, range, sill=1, nugget=0, eps=1.0e-7,...)
Wu2.cov(distance, range, sill=1, nugget=0, eps=1.0e-7,...)
Wu3.cov(distance, range, sill=1, nugget=0, eps=1.0e-7,...)
tophat(distance, range, sill=1,...)
```

Arguments

`distance`	a vector/matrix of distances to compute the covariance for.
`range`	the range value.
`smooth`	smoothness of the Matern covariance.
`sill`	the partial sill value. The absolute sill (the variance or equivalently the covariance at distance zero) is `sill + nugget`.
`nugget`	the nugget effect.
`effect`	if `TRUE`, `range` is the practial range
`eps`	any distance less than it will be set to `nugget + sill`.
`...`	see below.

Details

The tophat is NOT an actual covariance function. It is included for illustration purposes only.

The triangular, spherical and the Wu type functions are compactly supported covariance functions, vanishing beyond the range. The triangular and the first Wu type are not valid in two dimensions and up.

Value

a vector/matrix of covariance values at the supplied distances.

Note

To all the covariance functions the `...` argument is added to ensure compatibility between different types of covariance functions. Although this would not be necessary, it simplifies internal coding and usage considerably.

There is a difference between `exp.cov` and `expo.cov`, as well as `matern.cov` and `mater.cov` functions in `fields` and `KriSp`.

Examples

```
distance <- seq(0,2,length=150)
plot( distance, mater.cov(distance,smooth=1,range=.4,sill=.8,nugget=.2))
```

`datasets` *Artificial datasets used in the tutorial*

Description

The `simple.data` and `universal.data` are artifical datasets to illustrate the simple and universal kriging in the tutorial.

Usage

```
data(simple)
data(universal)
```

Format

A list containing the two components. `x`: 100 longitude-latitude position of locations, `Y`: the simulated values.

See Also

`anomaly.data`.

Examples

```
library(mvtnorm)

# simple.data:
set.seed(15)
n <- 100
x <- round(cbind(lon=runif(n)-105,lat=runif(n)+40),3)

lags <- rdist.earth(x)
C <- expo.cov( lags, range=10, sill=1)

Y <- round(c(rmvnorm(1,sigma=C)), 3)
simple.data <- list(x=x,Y=Y)

# universal.data:
set.seed(15)
n <- 100
x <- round(cbind(lon=runif(n)-105,lat=runif(n)+40),3)

lagC <- rdist.earth(x)
C <- expo.cov( lagC, range=10, sill=0.9, nugget=0.1)

Y <- round(c(rmvnorm(1,sigma=C))+4*x[,2]-100, 3)
universal.data <- list(x=x,Y=Y)
```

`Krig.simple.sparse` *Kriging surface estimate*

Description

Calculating the BLUP or kriging estimate for large two dimensional spatial datasets.

Usage

```
Krig.simple.sparse(x,Y,
        cov.fun = "expo.cov",
            cov.fun.args = list(range = 10, eps = eps),
        taper.fun = "spher.cov",
            taper.fun.args = list(range = 10),
        ncov = 10000, approxhmax = taper.fun.args$range,
        miles = TRUE, R = NULL, eps = 1e-5,
        save.sigma = TRUE, save.chol = FALSE,
        verbose = FALSE, nnzmax = 1e+05, tmpmax = 10000, ...)
```

Arguments

`x`	a `m` times `2` matrix containing the locations.
`Y`	the observed values at `x`.
`cov.fun`	Covariance function in the form of an R function, or its name as a string.
`cov.fun.args`	A list with the arguments to call the covariance function (in addition to the locations).
`taper.fun`	Taper function in the form of an R function, or its name as a string.
`taper.fun.args`	A list with the arguments to call the taper function (in addition to the locations).
`ncov`	Number of knots to evaluate the approximation
`approxhmax`	Maximum distance over which the covariance is approximated
`miles`	logical. If `TRUE` (default) distances are in statute miles if `FALSE` distances in kilometers.
`R`	the radius to use for the sphere to find spherical distances. If `NULL` the radius is either in miles or kilometers of the earth depending on the values of the miles argument. If `R=1` then distances are in radians.
`eps`	small value, everything smaller is considered zero.
`save.sigma`	should the covariance matrix be saved (in `SparseM` format).
`save.chol`	should the Cholesky factor of the covariance matrix be saved (in `SparseM` format).
`verbose`	should timing and convergence results be printed.
`nnzmax`	upper bound of non-zero elements in the covariance matrix.
`tmpmax`	working array for the Cholesky factorisation
`...`	supplementary parameters that can be given as arguments to the function `chol`.

Details

For computational reasons, we do not call simple the `solve` function but use `chol(...)` and `backsolve(...)` form the `SparseM` library. We do not allow missing values. We only consider two-dimensional domains.

Value

`Krig.simple.sparse` returns an object of `class c("sparse","Krig")`. The second is to reuse many handy functions of the library `fields`.

An object of the class `"sparse"` is a list containing at least the following components:

`call`	the matched call.
`fitted`	fitted values at observed values.
`solve`	vector used for prediction on other grid points.
`sigma`	if requested, the covariance matrix.
`sigmachol`	if requested, the Cholesky factor.
`timing`	time needed for the main calculations.
`nnz`	The number of nonzero elements in the covariance matrix and its Cholesky factor.

Additionally, most input arguments are passed to the object.

Note

For REALLY big datasets, it would be wise to dissect the functions.

The radius of the earth is assumed to be 3963.34 miles or 6378.388 kilometers.

`approxhmax` should be at least as big as the taper range or the domain of the field.

If `nnzmax` is too small, R may produce a 'core dumped'.

See Also

`Krig.simple.sparse`, `predict.sparse`, `plot.sparse`;

`chol` and `backsolve` from the `SparseM` library.

Examples

```
data(simple)
attach(simple.data)

obj <- Krig.simple.sparse( x, Y)
pre <- predict( obj, x=cbind(-104.5,40.5))

surf <- predict.surface( obj)
image.plot( surf)
```

`Krig.sparse` *Kriging surface estimate*

Description

Calculating the BLUP or kriging estimate for large two dimensional spatial datasets.

Usage

```
Krig.sparse(x,Y,
        cov.fun="expo.cov", cov.fun.args=list(range=10,eps=eps),
        taper.fun="spher.cov", taper.fun.args=list(range=10),
        ncov=10000,approxhmax=taper.fun.args$range,
        miles=TRUE,R=NULL, eps=1e-5,
        xM=NULL, m=2,scale.type = "user",x.center = rep(0, ncol(x)),
        x.scale = rep(1, ncol(x)),
        maxiter=25,epsiter=1e-3,
        save.sigma=TRUE,save.chol=FALSE,verbose=FALSE,
        nnzmax=100000,tmpmax=10000,...)
```

Arguments

`x`	a `m` times `2` matrix containing the locations.
`Y`	the observed values at `x`.
`cov.fun`	Covariance function in the form of an R function, or its name as a string.
`cov.fun.args`	A list with the arguments to call the covariance function (in addition to the locations).
`taper.fun`	Taper function in the form of an R function, or its name as a string.
`taper.fun.args`	A list with the arguments to call the taper function (in addition to the locations).
`ncov`	Number of knots to evaluate the approximation
`approxhmax`	Maximum distance over which the covariance is approximated
`miles`	logical. If `TRUE` (default) distances are in statute miles if `FALSE` distances in kilometers.
`R`	the radius to use for the sphere to find spherical distances. If `NULL` the radius is either in miles or kilometers of the earth depending on the values of the miles argument. If `R=1` then distances are in radians.
`eps`	small value, everything smaller is considered zero.
`xM`	a `m` times `2` matrix containing the values for the spatial drift. By default, the locations are used.
`m`	A polynomial function of degree (m-1) will be included in the model as the spatial trend (drift) component.

`scale.type`	A character string among: `range`, `unit.sd`, `user`, `unscaled`. The independent variables are scaled to the specified type. See below.
`x.center`	Centering values to be subtracted from each column of the x matrix.
`x.scale`	Scale values that are divided into each column after centering.
`maxiter`	Maximum number of iterations used in the backfitting iteration
`epsiter`	Stop the backfitting as soon as the sum of squares of the coefficients of two consecutive iterations is smaller.
`save.sigma`	should the covariance matrix be saved (in `SparseM` format).
`save.chol`	should the Cholesky factor of the covariance matrix be saved (in `SparseM` format).
`verbose`	should timing and convergence results be printed.
`nnzmax`	upper bound of non-zero elements in the covariance matrix.
`tmpmax`	working array for the Cholesky factorisation
`...`	supplementary parameters that can be given as arguments to the function `chol`.

Details

For computational reasons, we do not call simple the `solve` function but use `chol(...)` and `backsolve(...)` form the `SparseM` library. We do not allow missing values. We only consider two-dimensional domains.

Concerning the scaling for the spatial trend. By default no scaling is done. Scale type of `range` scales the data to the interval (0,1) by forming `(x-min(x))/range(x)` for the x- and y-axis. Scale type of `unit.sd` subtracts the mean and divides by the standard deviation. Scale type of `user` allows specification of an `x.center` and `x.scale` by the user. The default for `user` is mean 0 and standard deviation 1. Scale type of `unscaled` does not scale the data.

Value

`Krig.sparse` returns an object of `class c("sparse","Krig")`. The second is to reuse many handy functions of the library `fields`.

An object of the class `"sparse"` is a list containing at least the following components:

`call`	the matched call.
`iternorm`	norm of coefficients for each iteration.
`trend`	fitted trend surface at observed values.
`coef`	coefficients of trend surface.
`spatial`	spatial part of surface at observed values.
`solve`	vector used for prediction on other grid points.
`sigma`	if requested, the covariance matrix.
`sigmachol`	if requested, the Cholesky factor.
`timing`	time needed for the main calculations.
`nnz`	The number of nonzero elements in the covariance matrix and its Cholesky factor.

Additionally, most input arguments are passed to the object.

Note

For REALLY big datasets, it would be wise to dissect the functions.

The radius of the earth is assumed to be 3963.34 miles or 6378.388 kilometers.

`approxhmax` should be at least as big as the taper range or the domain of the field.

If `nnzmax` is too small, R may produce a 'core dumped'.

See Also

`Krig.simple.sparse`, `predict.sparse`, `plot.sparse`;

`transformx` from the `fields` library; `chol` and `backsolve` from the `SparseM` library.

Examples

```
data(universal)
attach(universal.data)

obj <- Krig.sparse(x, Y,
          cov.fun.args=list(range=10,sill=.9,nugget=.1))

summary(obj)

image.plot( predict.surface(obj))
image.plot( predict.surface(obj, trend.only=TRUE))
```

`KriSp-internal` *KriSp internal and secondary functions*

Description

Listed below are supporting functions for KriSp.

Usage

```
distprep(distance, range, effect)

version.KriSp(verbose=TRUE)

predict.surface.se.sparse(object,...)
predict.se.sparse(object,...)

expo.earth.cov(x1, x2, theta = 1, C = NA)
```

Details

distprep is an auxiliary function used in the different covariance functions. effect can be used if 'effective' ranges should be considered.

As all sparse objects are also Krig objects, we need as many methods as defined for Krig in the fields library.

The structure of the functions Krig in fields has changed drastically between versions 2.x and 3.x. To keep sample source code simple, we wrote a function expo.earth.cov that works like exp.earth.cov in versions 2.x but has the required functionality for higher versions.

KriSp.methods *Accessing Sparse Kriging Fits*

Description

These functions are all methods for class sparse objects.

Usage

```
summary(object, ...)
print(x, digits = max(3, getOption("digits") - 3),...)

residuals(object, ...)
resid(object, ...)

fitted(object, ...)
fitted.values(object, ...)

coef(object, ...)
coefficients(object, ...)
```

Arguments

object,x	an object of class sparse, typically the result of a call to Krig.sparse or Krig.simple.sparse.
digits	a non-null value for digits specifies the minimum number of significant digits to be printed in values. If digits is NULL, the value of digits set by options is used.
...	further arguments passed to or from other methods.

See Also

Krig.sparse, Krig.simple.sparse.

`KriSp` *Tools for interpolating large data sets*

Description

KriSp is a collection of functions for interpolationg large spatial datasets with covariance tapering.

Details

There are also generic functions that support these methods such as

`plot` - diagnostic plots of fit
`summary` - statistical summary of fit
`print` - shorter version of summary
`surface` - graphical display of fitted surface
`predict` - evaluation fit at arbitrary points

To get started, try some of the examples from help files for `KriSp`. See also the manual/tutorial at http://www.mines.edu/~rfurrer/software/KriSp.

The theoretical background can be found in the paper:

Furrer, R., Genton, M. G., and Nychka, D. (2006). Covariance Tapering for Interpolation of Large Spatial Datasets. Journal of Computational and Graphical Statistics, 15(3), 502–523.

The structure of the functions `Krig` in `fields` has changed drastically between versions 2.x and 3.x. To keep sample source code simple, we wrote a function `expo.earth.cov` that works like `exp.earth.cov` in versions 2.x but has the required functionality for higher versions.

The next major version of `fields` (version>3.2) will contain a sparse matrix module and extends the capacities of `KriSp`. Therefore, `KriSp` will no longer be significantly extended and future releases will most likely consist of bug fixes only.

Note

DISCLAIMER:

This is software for statistical research and not for commercial uses. The authors do not guarantee the correctness of any function or program in this package. Any changes to the software should not be made without the authors permission.

nnz — *Nonzero elements of a sparse matrix*

Description

Returns the nonzero elements of a sparse matrix.

Usage

```
nnz(x)
```

Arguments

x	matrix of class matrix.csr.

Value

nonzero elements of a sparse matrix x.

See Also

as.matrix.csr from the SparseM library.

Examples

```
nnz( as.matrix.csr( diag( 5)))
```

plot.sparse — *Diagnostic and summary plots of the Krig.sparse object*

Description

Plots a series of two diagnostic plots that summarize the fit from Krig.sparse.

Usage

```
plot.sparse(x, main=NA, which=c(TRUE,TRUE), graphics.reset=TRUE,
            ...)
```

Arguments

x	an object of class sparse, typically the result of a call to Krig.sparse or Krig.simple.sparse.
main	Title of the plot. Default is the function call.
graphics.reset	Reset to original graphics parameters after plotting. Default is TRUE.
which	A vector of 2 logical values. Controls which of the two graphs to plot.
...	Optional graphics arguments to pass to each plot.

Details

This function creates two summary plots of the `sparse` object. The default is to put these in a 1 times 2 panel. However, if the screen is already divided in some other fashion the plots will just be added according to that scheme. This option is useful to compare to compare several different model fits.

The first is a scatterplot of predicted value against observed.

The second plot is a histogram of the residuals.

See Also

`Krig.sparse`, `summary.sparse` and `plot.Krig` from the `fields` library.

Examples

```
data(universal)
attach(universal.data)
obj <- Krig.sparse(x, Y)
fit <- predict(obj)
# fitting a surface to ozone measurements
plot(fit)
```

---

predict.sparse *Evaluation of Krig spatial process estimate*

Description

Provides predictions from the spatial process estimate at arbitrary points

Usage

```
predict.sparse(object, x=NULL, trend.only=FALSE, ...)
```

Arguments

`object`	an object of class `sparse`, typically the result of a call to `Krig.sparse` or `Krig.simple.sparse`.
`x`	Matrix of x-values on which to evaluate the kriging surface. If omitted, the data x-values, i.e. `obj$x` will be used.
`trend.only`	for universal kriging, should only the trend be returned.
`...`	only for compatibility reasons.

Details

We evaluate the kriging surface on the given grid.

Value

Vector of predicted responses.

See Also

Krig.sparse, Krig.sparse; predict.surface from the fields package.

Examples

```
data(universal)
attach(universal.data)

obj <- Krig.sparse(x, Y,
          cov.fun.args=list(range=10,sill=.9,nugget=.1))

print( predict( obj, x=cbind(-104.5,40.5)))

xgrid <- expand.grid( lon=seq(min(x[,1]), max(x[,1]), l=50),
                      lat=seq(min(x[,2]), max(x[,2]), l=50))

# older verstions of fields used:
#   xgrid <- make.surface.grid( grid.list=list(lon='x',lat='y'),
#                             X=x, nx=50, ny=50)

surf <- predict( obj, x=xgrid)

image.plot( predict.surface(obj))
image.plot( predict.surface(obj, trend.only=TRUE))
```

Index