Package ‘telefit’

February 3, 2020

Type Package

Title Estimation and Prediction for Remote Effects Spatial Process Models

Version 1.0.3

Date 2020-02-03

Author Joshua Hewitt

Maintainer Joshua Hewitt <joshua.hewitt@duke.edu>

Description Implementation of the remote effects spatial process (RESP) model for teleconnection. The RESP model is a geostatistical model that allows a spatially-referenced variable (like average precipitation) to be influenced by covariates defined on a remote domain (like sea surface temperatures). The RESP model is introduced in Hewitt et al. (2018) <doi:10.1002/env.2523>. Sample code for working with the RESP model is available at <https://jmhewitt.github.io/research/resp_example>. This material is based upon work supported by the National Science Foundation under grant number AGS 1419558. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

License GPL-3

Depends R (>= 3.0.2)

Imports abind, coda, cowplot, dplyr, fields, itertools, mvtnorm, raster, scoringRules, stringr, foreach, ggplot2, gtable, reshape2, scales, sp

LinkingTo Rcpp (>= 0.12.4), RcppArmadillo, RcppEigen (>= 0.3.3.3.1)

RoxygenNote 7.0.2

Suggests testthat

LazyData true

SystemRequirements A system with a recent-enough C++11 compiler (such as g++-4.8 or later).

NeedsCompilation yes

Encoding UTF-8

Repository CRAN

Date/Publication 2020-02-03 21:00:02 UTC

R topics documented:

`abind3` *Convenience function for stacking matrices into an array.*

Description

This function extends the abind function from the abind package.

Usage

```
abind3(...)
```

Arguments

`...`	Any number of matrices of equal dimension to stack together into a 3d matrix

`arrayToLong` *Reshape array of data matrices into long format*

Description

Reshape array of data matrices into long format

Usage

```
arrayToLong(X, coords, yrs)
```

Arguments

`X`	3 dimensional array of matrices to extract to long format
`coords`	Spatial coordinates associated with the data (longitude in first column)
`yrs`	Vector with labels for the years

`cca.predict` *Make predictions using canonical correlation analysis (CCA)*

Description

Canonical correlation analysis (CCA) is sometimes referred to as a double-barreled principal component analysis. Loosely, it fits a linear regression model to the scores of principal component decompositions for of the predictors `X` and responses `Y`. Oftentimes, only the largest k principal components are used to make predictions.

Usage

```
cca.predict(X, Y, X.new, k.x, k.y)
```

Arguments

`X`	An `(nvars x nobs)` data frame or matrix in which each column contains all observations of measured (predictor) variables for a given timepoint or sample. For example, if X represents a spatial variable that was recorded at several timepoints, then each row of X should contain the variable's measurement for all timepoints at a single location.
`Y`	An `(nvars x nobs)` data frame or matrix in which each column contains all observations of measured (response) variables for a given timepoint or sample.
`X.new`	An `(nvars x nobs.new)` data frame or matrix of values to use to predict Y.new using CCA.
`k.x`	An integer less than `(nobs)` indicating how many eigenvectors of `(X)` to use in the CCA.
`k.y`	An integer less than `(nobs)` indicating how many eigenvectors of `(Y)` to use in the CCA.

Details

CCA has been used to predict a teleconnected response (like precipitation) using the remote field generating the teleconnection (like ocean temperatures). In this application, principal components are often referred to as empirical orthogonal functions (EOFs).

Value

Y.new Predicted values for Y.new

References

Cook, E.R., Briffa, K.R., and Jones, P.D., 1994, Spatial regression methods in dendroclimatology: A review and comparison of two techniques: International Journal of Climatology, v. 14, p. 379-402.

Glahn, H.R., 1968, Canonical Correlation and Its Relationship to Discriminant Analysis and Multiple Regression: Journal of the Atmospheric Sciences, v. 25, p. 23-31.

Examples

```
data("coprecip")
attach(coprecip)

# compute CCA predictions of Y (CO precipitation) given Z (Pacific ocean SSTs)
# using 2 principal components (aka. EOFs)
preds = cca.predict(X = Z, Y = Y, X.new = Z, k.x = 2, k.y = 2)

# compute R^2
1 - var(as.numeric(preds-Y)) / var(as.numeric(Y))
```

---

coef.stFit *Compute point estimates for parameters from posterior samples*

Description

Compute point estimates for parameters from posterior samples

Usage

```
## S3 method for class 'stFit'
coef(object, burn = 1, fun = mean, ...)
```

Arguments

`object`	stFit object containing posterior samples for model
`burn`	number of posterior samples to reject before computing estimates
`fun`	function for computing point estimates
`...`	S3 generic/method consistency

Examples

```
data("coprecip.fit")
coef(coprecip.fit, burn = 50)
```

coef.stPredict — *Compute point estimates for parameters from posterior samples*

Description

Compute point estimates for parameters from posterior samples

Usage

```
## S3 method for class 'stPredict'
coef(object, stFit, stData, burn = 1, type = "eof-alpha_knots", ...)
```

Arguments

`object`	stPredict object containing posterior estimates of alphas
`stFit`	stFit object containing posterior samples for model
`stData`	stData object containing spatial information for dataset
`burn`	number of posterior samples to reject before computing estimates
`type`	One of the following options to specify what point estimates to return **eof-alpha_knots** Remote coefficient estimates (alpha_knots) mapped onto the eof patterns of the remote covariates.
`...`	S3 generic/method consistency

Examples

```
data("coprecip")
data("coprecip.fit")
data("coprecip.predict")

coef(coprecip.predict, stFit = coprecip.fit, stData = coprecip, burn = 50)
```

coprecip — *Standardized anomalies of CO Precipitation*

Description

A dataset containing sample spatially-aggregated climate data from the ERA-Interim and PRISM datasets. The response comes from PRISM, average monthly precipitation in a DJF winter. The covariates come from ERA-Interim, Colorado and Pacific Ocean (sea) surface temperatures. All data has been converted to standardized anomalies.

Usage

```
coprecip
```

Format

A stDataobject with 3 years of observations

tLabs year labels for data columns

coords.s centers of grid cells for Colorado data

coords.r centers of grid cells for Pacific Ocean data

X Array of design matrices for Colorado covariates

Y Matrix of precipitation observations

Z Matrix of Pacific Ocean data

X.lab Label for covariate data, used by plotting functions

Y.lab Label for response data, used by plotting functions

Z.lab Label for covariate data, used by plotting functions

Source

http://prism.oregonstate.edu

https://rda.ucar.edu/datasets/ds627.0/

Examples

```
data("coprecip")
str(coprecip)
```

---

coprecip.fit *Sample MCMC output for the RESP model*

Description

An example stFit object containing output from a short run of the MCMC sampler that fits the RESP model to data.

Usage

```
coprecip.fit
```

Format

An stFit object, which is a list of several objects

parameters MCMC samples of model parameters

priors description of priors used to fit model

miles TRUE or FALSE to specify whether the spatial distances used to estimate spatial covariance parameters were in units of miles (TRUE) or kilometers (FALSE)

localOnly TRUE if remote covariates were not estimated

remoteOnly TRUE if local covariates were not estimated

varying (deprecated) TRUE if local covariates were estimated as a spatially-varying field

coords.knots coordinates of remote knot locations

Examples

```
data("coprecip.fit")
str(coprecip.fit)
```

---

coprecip.predict *Sample composition sampling output for the RESP model*

Description

An example stPredict object containing predictions from a short run of the MCMC composition sampler. The output also contains teleconnection estimates.

Usage

```
coprecip.predict
```

Format

An stPredict object, which is a list of several objects

pred A list containing summaries of posterior predictions

samples Posterior samples for predictions

coords.s centers of grid cells for Colorado data

localOnly TRUE if remote covariates were not estimated

varying (deprecated) TRUE if local covariates were estimated as a spatially-varying field

tLabs year labels for prediction timepoints

Y.lab Label for response data, used by plotting functions

cat.probs vector of probabilities for using posterior samples to return categorical predictions from the posterior prediction samples

category.breaks Breakpoints used to discretize posterior predictive distribution at each coordinate in coords.s during composition sampling.

alpha_knots Summaries of posterior estimates of teleconnection effects

eof_alpha_knots Summaries of posterior estimates of teleconnection effects after spatial basis function transformation

Examples

```
data("coprecip.predict")
str(coprecip.predict)
```

---

dgemkmm *Evaluate kron(A,B) * C without storing kron(A,B)*

Description

Evaluate kron(A,B) * C without storing kron(A,B)

Usage

```
dgemkmm(A, B, C)
```

Arguments

A	(m x n) matrix
B	(p x q) matrix
C	(nq x r) matrix

---

eof *Performs an EOF decomposition of the data*

Description

Uses the stats::prcomp function to implement EOF decompositions of data

Usage

```
eof(X, center = F, scale = F)
```

Arguments

X	[variable x observation] matrix of data for which to compute EOFs
center	TRUE/FALSE to center columns of X in call to prcomp
scale	TRUE/FALSE to scale columns of X in call to prcomp

Value

A list containing EOF patterns as columns, and their scores

Examples

```
data("coprecip")
attach(coprecip)

# compute ocean surface temperature eofs
eofs = eof(Z)

# view first EOF, which corresponds to the El-Nino pattern
coords.r.mod = coords.r
coords.r.mod[,1][coords.r.mod[,1]>0] =
  coords.r.mod[,1][coords.r.mod[,1]>0] - 360
fields::quilt.plot(coords.r.mod, eofs$patterns[,1])

# alternatively, the plot.stData function can directly compute and plot EOFs
plot(coprecip, type='eof', pattern=1)
```

`errDump` *Wrapper for a function to dump errors from C++*

Description

Wrapper for a function to dump errors from C++

Usage

```
errDump(x, fname = file.path(tempdir(), "error_samplerState.RData"))
```

Arguments

`x`	Data to save
`fname`	Path/name to save data to

`extractRegion` *Extract region from a SpatialGridDataFrame*

Description

This method is intended for use as the main helper function for extractStData.

Usage

```
extractRegion(
  sgdf,
  extent,
  type = "response",
  aggfact = NULL,
  mask = NULL,
  aspect = F,
  aspect.categories = NULL,
  slope = F
)
```

Arguments

`sgdf`	SpatialGridDataFrame containing data to extract
`extent`	raster::extent object featuring region to extract, or a SpatialPolygonsXXX object used for extracting areal data
`type`	whether to return the raw data, anomalies (data minus temporal average at each location), standardized anomalies (anomalies divided by temporal standard deviation at each location), or spatially standardized data (data minus overall spatial average divided by spatial std. dev.; each year gets its own spatial standardization)
`aggfact`	if provided, will spatially average the data
`mask`	if an sgdf is provided, the data will be masked before extraction, aggregation, and anomaly computation
`aspect`	TRUE to return the aspect of the surface at each location instead of the value of the surface itself
`aspect.categories`	if aspect==TRUE, this specifies the number of discrete categories to divide aspect numbers (0-360) into. NULL if the original scale (0-360) should be kept. By design, the aspect categories will be centered on north in the first category.
`slope`	TRUE to return the slope of the surface at each location instead of the value of the surface itself

Value

a modified SpatialGridDataFrame, sgdf, with the climatology for each location accessible via attr(sgdf@data@values, 'scaled:center') if anomalies were computed extractStData *Basic extraction of SpatialGridDataFrame data for teleconnection analysis*

Description

Basic extraction of SpatialGridDataFrame data for teleconnection analysis

Usage

```
extractStData(
  X,
  Y,
  Z,
  t = NULL,
  D.s,
  D.r,
  mask.s = NULL,
  mask.r = NULL,
  aggfact.s = NULL,
  aggfact.r = NULL,
  intercept = T,
  type.s = "response",
  type.r = "response",
  type.s.y = "response",
  X.lab = NULL,
  Y.lab = NULL,
  Z.lab = NULL,
  aspect = F,
  aspect.categories = 4,
  slope = F,
  colnames.X = NULL,
  formula = NULL
)
```

Arguments

X	SpatialGridDataFrame with local covariates. If X is a list, each SpatialGridDataFrame will be included as one covariate.
Y	SpatialGridDataFrame with response data
Z	SpatialGridDataFrame with remote covariates. If Z is a list, this function assumes each element of the list contains observations for the same covariate, but from different spatial regions. If Z is a list, D.r and mask.r must also be lists so that this function can know which regions to extract from each SpatialGridDataFrame

`t`	Timepoint from which to extract data from X, Y, and Z. If NULL, then all timepoints will be used.
`D.s`	c(xmin, xmax, ymin, ymax) region from which to extract data from X and Y, or a SpatialPolygonsXXX object containing boundaries of regions to extract areal data from.
`D.r`	c(xmin, xmax, ymin, ymax) region from which to extract data from Z
`mask.s`	SpatialGridDataFrame to be used as a mask when extracting data from X and Y. Locations in mask.s with NA values will be ignored when extracting data from X and Y.
`mask.r`	SpatialGridDataFrame to be used as a mask when extracting data from Z. Locations in mask.s with NA values will be ignored when extracting data from Z.
`aggfact.s`	If provided, will spatially average Y and X data
`aggfact.r`	If provided, will spatially average Z data
`intercept`	If TRUE, an intercept will be added to the design matrix
`type.s`	'response' 'anomaly' or 'std.anomaly' or a vector of these options depending on whether data extracted from X should be the observed data, anomalies, or standardized anomalies (where the climatology is computed from the observations as the pointwise temporal average)
`type.r`	'response' 'anomaly' or 'std.anomaly' or a vector of these options depending on whether data extracted from Z should be the observed data, anomalies, or standardized anomalies (where the climatology is computed from the observations as the pointwise temporal average)
`type.s.y`	'response' 'anomaly' or 'std.anomaly' depending on whether data extracted from Y should be the observed data, anomalies, or standardized anomalies (where the climatology is computed from the observations as the pointwise temporal average)
`X.lab`	name for X data (optional)
`Y.lab`	name for Y data (optional)
`Z.lab`	name for Z data (optional)
`aspect`	TRUE or vector of logicals (one for each X object) to return the aspect of the surface at each location instead of the value of the surface itself
`aspect.categories`	if aspect==TRUE, this specifies the number of discrete categories to divide aspect numbers (0-360) into. NULL if the original scale (0-360) should be kept. By design, the aspect categories will be centered on north in the first category.
`slope`	TRUE or vector of logicals (one for each X object) to return the slope of the surface at each location instead of the value of the surface itself
`colnames.X`	names of columns of X
`formula`	formula object to specify how to create the design matrix

Examples

```
# the extractRegion and extractStData methods create data matrices from
# SpatialGridDataFrame objects

library(sp)

data("coprecip")
attach(coprecip)


#
# build SpatialGridDataFrame objects containing some of the coprecip data
#

gt = GridTopology(cellcentre.offset = apply(coords.s, 2, min),
                  cellsize = c(.5, .5),
                  cells.dim = c(20, 12))

# Note: This is an example only; this grid will not match coprecip$coords.r
gt.Z = GridTopology(cellcentre.offset = apply(coords.r, 2, min),
                    cellsize = c(1.4, 1.4),
                    cells.dim = c(101, 52))

Xd = data.frame(`1981` = X[,2,1], `1982` = X[,2,2])
colnames(Xd) = gsub('X','', colnames(Xd))
sgdf.x = SpatialGridDataFrame(gt, Xd)


Yd = data.frame(`1981` = Y[,1], `1982` = Y[,2])
colnames(Yd) = gsub('X','', colnames(Yd))
sgdf.y = SpatialGridDataFrame(gt, Yd)


Zd = data.frame(`1981` = Z[,1], `1982` = Z[,2])
colnames(Zd) = gsub('X','', colnames(Zd))
sgdf.z = SpatialGridDataFrame(gt.Z, Zd)

# only extract a region of the coordinates
coprecip2 = extractStData(sgdf.x, sgdf.y, sgdf.z,
                    D.s = c(-105, -103, 37, 41),
                    D.r = c(-160, -100, -15, 0))
```

`forwardsolve.kron` *Solves a triangular system with a Kronecker product structure*

Description

Solves $kron(A, B)x = y$ where A and B are lower triangular matrices.

Usage

```
forwardsolve.kron(A, B, y)
```

Arguments

A	an mxn matrix
B	an pxq matrix
y	an $mpxs$ matrix

Value

x

Examples

```
set.seed(2018)

coord.s = matrix(runif(100), ncol=2)
coord.r = matrix(runif(50), ncol=2)

d.s = as.matrix(dist(coord.s))
d.r = as.matrix(dist(coord.r))

S1 = exp(-d.s)
S2 = exp(-d.r)

A = t(chol(S1))
B = t(chol(S2))

s = 15

x = matrix(runif(nrow(S1)*nrow(S2)*s), ncol=s)

y = kronecker(A,B) %*% x

x.solved = forwardsolve.kron(A, B, y)

max(abs(x - x.solved))
```

`HPDinterval.stFit` *Compute Highest posterior density intervals from posterior samples*

Description

Compute Highest posterior density intervals from posterior samples

Usage

```
## S3 method for class 'stFit'
HPDinterval(stFit, burn = 1, prob = 0.95)
```

Arguments

stFit	stFit object containing posterior samples for model
burn	number of posterior samples to reject before computing estimates
prob	The target probability content of the intervals

Examples

```
data("coprecip.fit")
HPDinterval.stFit(coprecip.fit, burn = 50)
```

---

invWSamp *Samples an Inverse-Wishart matrix*

Description

Samples $W\ IW(Psi, n)$

Usage

```
invWSamp(Psi, n)
```

Arguments

Psi	an nxn scale matrix
n	degrees of freedom parameter

Examples

```
A = matrix(c(1,.5,.5,1), ncol=2)
W = invWSamp(A, 3)
```

kronSamp — *Samples a multivariate normal with a Kronecker product covariance structure*

Description

Samples $x\ N(0, AxB)$

Usage

```
kronSamp(A, B)
```

Arguments

A	an mxn matrix
B	an pxq matrix

Examples

```
A = matrix(c(1,.5,.5,1), ncol=2)
B = diag(2)
x = kronSamp(A, B)
```

lat_trans — *Formatting for longitude scales in ggplot spatial maps*

Description

Formatting for longitude scales in ggplot spatial maps

Usage

```
lat_trans()
```

lon_trans — *Formatting for longitude scales in ggplot spatial maps*

Description

Formatting for longitude scales in ggplot spatial maps

Usage

```
lon_trans()
```

maternArray *Matern covariance*

Description

This function evaluates the Matern covariance function for the elements of a vector.

Usage

```
maternArray(d, scale = 1, range = 1, smoothness = 0.5, nugget = 0)
```

Arguments

`d`	A numeric vector of distances at which the Matern correlation function should be evaluated.
`scale`	Scales correlations to covariances.
`range`	Matern range parameter. Controls the decay of pointwise correlations as a function of distance.
`smoothness`	Matern smoothness parameter. Controls the number of process derivatives.
`nugget`	Spatial covariance nugget.

maternCov *Matern covariance*

Description

This function evaluates the Matern covariance function for the elements of a (potentially non-square) spatial distance matrix

Usage

```
maternCov(d, scale = 1, range = 1, smoothness = 0.5, nugget = 0)
```

Arguments

`d`	A numeric vector or matrix of distances at which the Matern correlation function should be evaluated.
`scale`	Scales correlations to covariances.
`range`	Matern range parameter. Controls the decay of pointwise correlations as a function of distance.
`smoothness`	Matern smoothness parameter. Controls the number of process derivatives.
`nugget`	Spatial covariance nugget.

Examples

```
data("coprecip")
attach(coprecip)

# compute spatial covariance matrix for an exponential covariance function
# using Colorado coordinates
Sigma = maternCov(fields::rdist.earth(coords.s), scale = 1, range = 250,
  smoothness = .5, nugget = 0)
```

---

maternEffectiveRange *Compute effective range for Matern correlation to drop to a specified level*

Description

The effective range for an isotropic spatial correlation function is commonly defined to be the distance beyond which the correlation becomes small, typically below .05. Given range and smoothness parameters for a Matern covariance function, this function numerically searches for this distance. Note that the scale is not important for this calculation.

Usage

```
maternEffectiveRange(cor = 0.05, range = 1, smoothness = 0.5)
```

Arguments

`cor`	Effective correlation to check for
`range`	Matern range parameter. Controls the decay of pointwise correlations as a function of distance.
`smoothness`	Matern smoothness parameter. Controls the number of process derivatives.

Examples

```
# effective range for exponential covariance function with range = 1,
# which is theoretically known to equal -ln(.05)
maternEffectiveRange(cor = .05, range = 1, smoothness = .5)
```

mergeComposition *Combine results from composition sampler*

Description

Combine results from composition sampler

Usage

```
mergeComposition(xfull, yfull)
```

Arguments

`xfull`	Raw output from one run of the Rcpp/Armadillo composition sampler
`yfull`	Raw output from another run of the Rcpp/Armadillo composition sampler

mergeCovmat *Combine sample covariance matrices from two samples*

Description

This function combines the sample covariance information from two samples (of the same phenomena) to return the sample covariance matric of the union of the two samples.

Usage

```
mergeCovmat(
  A.cov.xy,
  B.cov.xy,
  A.mean.x,
  A.mean.y,
  B.mean.x,
  B.mean.y,
  A.n,
  B.n
)
```

Arguments

`A.cov.xy`	sample covariance matrix from the first sample, 'A'
`B.cov.xy`	sample covariance matrix from the second sample, 'B'
`A.mean.x`	sample mean from the first sample, 'A'
`A.mean.y`	sample mean from the first sample, 'A'
`B.mean.x`	sample mean from the second sample, 'B'

`B.mean.y`	sample mean from the second sample, 'B'
`A.n`	sample size from the first sample, 'A'
`B.n`	sample size from the second sample, 'B'

Details

This function assumes the data is normalized by n (the MLE estimator) instead of n-1 (the unbiased estimator).

References

Pebay, P., 2008, Formulas for Robust, One-Pass Parallel Computation of Covariances and Arbitrary-Order Statistical Moments: Sandia Report.

`mergeMean` *Combine sample means from two samples*

Description

This function combines the sample mean information from two samples (of the same phenomena) to return the sample mean of the union of the two samples.

Usage

```
mergeMean(x.mean, y.mean, x.n, y.n)
```

Arguments

`x.mean`	sample mean from the first sample, 'x'
`y.mean`	sample mean from the second sample, 'y'
`x.n`	sample size from the first sample, 'x'
`y.n`	sample size from the second sample, 'y'

`mergeVar` *Combine sample variances from two samples*

Description

This function combines the sample variance information from two samples (of the same phenomena) to return the sample variance of the union of the two samples.

Usage

```
mergeVar(x.var, y.var, x.mean, y.mean, x.n, y.n)
```

Arguments

`x.var`	sample variance from the first sample, 'x'
`y.var`	sample variance from the second sample, 'y'
`x.mean`	sample mean from the first sample, 'x'
`y.mean`	sample mean from the second sample, 'y'
`x.n`	sample size from the first sample, 'x'
`y.n`	sample size from the second sample, 'y'

Details

This function assumes the data is normalized by n (the MLE estimator) instead of n-1 (the unbiased estimator).

References

Chan, T.F., Golub, G.H., and LeVeque, R.J., 1979, Updating formulae and a pairwise algorithm for computing sample variances: Technical Report, Stanford University .

`plot.stData` *Plot stData objects*

Description

This function provides basic plotting for telefit package data.

Usage

```
## S3 method for class 'stData'
plot(
  x,
  type = "response",
  t = NULL,
  p = NULL,
  map = "world",
  region = ".",
  coord.s = NULL,
  coord.r = NULL,
  zlim = NULL,
  fill.lab = NULL,
  lab.teleconnection = expression(alpha),
  fill.lab.width = 20,
  category.breaks = NULL,
  coords.knots = NULL,
  signif.telecon = F,
  dots = NULL,
  pattern = 1,
  lwd = 1.75,
  cutoff = 0.9,
  signif.level = 0.05,
  alpha = 0.2,
  zmid = 0,
  contour = c(F, F),
  ...
)
```

Arguments

`x` Object of class stData to plot.

`type` One of the following options to specify what type of plot to build

response

sd.response Plot standard deviation of response variable at each location.

cat.response

covariate

remote

teleconnection This plot only applies if the stData object contains information about teleconnection effects, i.e., if it is a simulated dataset or otherwise modified to include estimates of teleconnection effects.

remote_cor This plot shows pointwise correlations between a local coordinate and the remote covariates.

eof

eof_scores

eof_scree

eof_cor This plot shows pointwise correlations with EOF patterns.

local_cor This plot shows pointwise correlations with local covariates.

teleconnection_knot_local

`t` timepoint to plot. Will automatically plot the first timepoint if t=NULL.

`p` column index of local covariate to plot if type='covariate'. Will automatically assume the local covariate data includes an intercept and will plot the second column if p=NULL.

`map` name of map provided by the maps package. These include county, france, italy, nz, state, usa, world, world2. By default, all stData plots will include us state outlines.

`region` name of subregions to include. Defaults to . which includes all subregions. See documentation for map for more details.

`coord.s` if plot type is 'teleconnection', specifies the longitude and latitude of local coordinate for which to plot teleconnection effects. if NULL, the middle local coordinate will be plotted.

`coord.r` if plot type is 'teleconnection_local', specifes the longitude and latitude of remote coordinate for which to plot associated teleconnection effects. if NULL, the middle remote coordinate will be plotted.

`zlim` c(min, max) vector that specifies the colorscale limits

`fill.lab` Optional label to override the default fill scale labels

`lab.teleconnection`
label used for fill scale in teleconnection plot

`fill.lab.width` line width for fill scale label

`category.breaks`
[ncoords x ncats] list of breakpoints used for binning responses into categories

`coords.knots` if plot type is 'remote', specifies the longitude and latitude of knot locations to overlay on the 'remote' plot

`signif.telecon` if TRUE, will highlight significant grid cells if the plotting data contain a signif column

`dots` additional named arguments with defaults to pass to additional functions

`pattern` if type=='eof' this specifies which (remote) EOF pattern to plot or if type=='eof_scores' this (vector) specifies which (remote) EOF pattern scores to plot

`lwd` line width for when plotting with signif.telecon==T

`cutoff` Used to denote where this proportion of variance is achieved in the eof_scree plots

`signif.level` significance level for eof_cor significance highlighting

`alpha` the level of fading that should be applied to insignificant grid boxes when plotting significant effects

`zmid` number that specifies the midpoint of the colorscale

`contour` c(TRUE, TRUE) to plot local and remote responses as contours vs. observations

`...` additional arguments to pass to functions

Value

a ggplot object with the specified map

Examples

```
data("coprecip")
p = plot(coprecip)
```

---

plot.stFit *Plot stFit objects*

Description

This function provides basic plotting for telefit package data.

Usage

```
## S3 method for class 'stFit'
plot(
  x,
  type = "density",
  stData = NULL,
  coord.s = NULL,
  coord.knot = NULL,
  text.size = NULL,
  axis.text.size = NULL,
  title.text.size = NULL,
  burn = 1,
  signif.telecon = F,
  p = 1,
  local.covariate = NULL,
  lwd = NULL,
  facet.signif = 3,
  stat.smooth.bw = NULL,
  stat.smooth.degree = NULL,
  dots = NULL,
  ...
)
```

Arguments

x	Object of class stFit to plot.
type	One of the following options to specify what type of plot to build **traceplot**

density
pairs
teleconnection
teleconnection_local
teleconnection_knot
teleconnection_knot_transect
teleconnection_knot_influence
beta

`stData` Object of class stData to provide coordinate and related information for plotting estimated teleconnection effects

`coord.s` if plot type is 'teleconnection', specifies the longitude and latitude of local coordinate for which to plot estimated teleconnection effects. if NULL, the middle local coordinate will be plotted.

`coord.knot` if plot type is 'teleconnection_knot_influence' or 'teleconnection_knot_local', specifies the longitude and latitude of knot coordinate for which to plot influence of remote coefficient on remote covariates, or the teleconnection coefficients associated with coord.knot

`text.size` number specifying the size of text labels

`axis.text.size` number specifying the size of axis text labels

`title.text.size` number specifying the size of title

`burn` number of observations to exclude from graph

`signif.telecon` if TRUE, will highlight significant teleconnection effects when type=='teleconnection'

`p` If stFit was fit with spatially varying coefficients, p specifies the index of the spatially varying coefficient to plot

`local.covariate` data.frame with variables, 'lon.Y', 'lat.Y', 'x' that will be plotted against teleconnection effects if type=='teleconnection_knot_transect'

`lwd` specifies linewidth for plots that include reference lines

`facet.signif` number of significant figures to round facet latitudes and longitudes for if type=='teleconnection_knot_tra

`stat.smooth.bw` if type=='teleconnection_knot_transect' this specifies the bandwith of the nonparametric smooth of the estimates

`stat.smooth.degree` if type=='teleconnection_knot_transect' this specifies the degree of the nonparametric smooth of the estimates

`dots` additional named arguments with defaults to pass to additional functions

`...` additional arguments to pass to functions

Value

a ggplot object with the specified map

Examples

```
data("coprecip.fit")
plot(coprecip.fit, burn = 50, type = 'trace')
```

`plot.stPredict` *Plot stPredict objects*

Description

This function provides basic plotting for telefit package data.

Usage

```
## S3 method for class 'stPredict'
plot(
  x,
  type = "prediction",
  t = NULL,
  stFit = NULL,
  stData = NULL,
  err.comparison = NULL,
  err.var = NULL,
  err.lab = err.var,
  pattern = 1,
  dots = NULL,
  burn = 1,
  signif.telecon = F,
  ...
)
```

Arguments

`x` Object of class stPredict to plot.

`type` One of the following options to specify what type of plot to build

- **prediction** Spatial plot of predicted response variable for a given timepoint t.
- **residual** Spatial plot of residual for a given timepoint t. Note, this plot is only available if the model has been evaluated and the predictions have been compared to another response dataset.
- **observed** Spatial plot of observed response variable for a given timepoint t. Note, this plot is only available if the model has been evaluated and the predictions have been compared to another response dataset.
- **standard_error (or 'se')** Spatial plot of prediction standard errors for a given timepoint t.

- **local** Spatial plot of the local components of the response variable for a given timepoint t.
- **remote** Spatial plot of the remote components of the response variable for a given timepoint t.
- **w** Spatial plot of the spatial noise component of the reponse variable for a given timepoint t.
- **correlation** Scatterplot of observed vs. predicted response variables for a given timepoint t. Note, this plot is only available if the model has been evaluated and the predictions have been compared to another response dataset.
- **teleconnection** Spatial plot of remote coefficients associated with a location coord.s in the spatial response domain.
- **teleconnection_knot** Spatial plot of remote knot coefficients associated with a location coord.s in the spatial response domain.
- **teleconnection_knot_transect**
- **errors** Series of plots that measure overall prediction error across prediction timepoints.
- **cat.prediction** Spatial plot of the predicted response variable category (i.e., above/below average) for a given timepoint t.
- **truth** Note, this plot is only available if the model has been evaluated and the predictions have been compared to another response dataset.
- **residual** Note, this plot is only available if the model has been evaluated and the predictions have been compared to another response dataset.
- **eof_alpha_knots** A map of the local domain where the plotted colors show the remote influence coefficients mapped onto the eof pattern specified by the "pattern" argument.

`t`	timepoint to plot. Will automatically plot the first timepoint if t=NULL.
`stFit`	Object of class stFit to provide related information and structures for plotting estimated teleconnection effects
`stData`	Object of class stData to provide coordinate and related information for plotting estimated teleconnection effects
`err.comparison`	data.frame with Year column and a column for a variable that will be used to plot annual errors against
`err.var`	name of variable in err.comparison for plotting against
`err.lab`	label for name of variable in err.comparison for plotting against
`pattern`	if type=='eof_alpha_knots', this specified which eof the remote coefficients should be mapped onto and then plotted over the local domain
`dots`	additional named arguments with defaults to pass to additional functions
`burn`	number of observations to exclude from graph
`signif.telecon`	TRUE to highlight significant teleconnection effects
`...`	additional arguments to be passed to lower-level plotting functions

Value

a ggplot object with the specified map

Examples

```
data("coprecip.predict")
p = plot(coprecip.predict, t=1981)
```

`plot.teleCor` *Plots teleconnection correlation maps*

Description

This function provides basic plotting for analyses returned from cor.tel

Usage

```
## S3 method for class 'teleCor'
plot(
  x,
  signif = F,
  coord.s = NULL,
  map = "world",
  region = ".",
  zlim = NULL,
  dots = NULL,
  ...
)
```

Arguments

`x`	object of class teleCor, containing pointwise correlations
`signif`	if TRUE, then teleCor must have a column labeled 'signif' that indicates which correlations are significant. These correlations will be printed in bold, and the rest will be printed more lightly
`coord.s`	specifies the longitude and latitude of local coordinate for which to plot pointwise correlations (if type=='remote'). if NULL, the middle local coordinate will be plotted.
`map`	name of map provided by the maps package. These include county, france, italy, nz, state, usa, world, world2. By default, all stData plots will include us state outlines.
`region`	name of subregions to include. Defaults to . which includes all subregions. See documentation for map for more details.
`zlim`	c(min, max) vector that specifies the colorscale limits
`dots`	additional named arguments with defaults to pass to additional functions
`...`	additional arguments to be passed to lower-level plotting functions

Value

a ggplot object with the specified map

Examples

```
data("coprecip")

cors = teleCor(coprecip)
p = plot(cors, coords.s = c(-105, 39.73))
```

`rmatnorm` *Simulate matrices from matrix normal distributions*

Description

Draw random matrices from the matrix normal distribution

$$MN(M,U,V)$$

Note that an observation, X, from this equation has the following distribution when vectorized

$$vec(X)\ N(vec(M),kron(V,U))$$

Usage

```
rmatnorm(n, U, V, M = matrix(0, nrow = nrow(U), ncol = nrow(V)))
```

Arguments

`n`	Number of random matrices to simulate
`U`	Covariance matrix defining dependence between rows
`V`	Covariance matrix defining dependence between columns
`M`	average value of each entry in the sampled matrices

rwishart — *Random wishart matrix*

Description

Random wishart matrix

Usage

```
rwishart(V, n)
```

Arguments

`V`	symmetric positive definite p x p scale matrix
`n`	degrees of freedom (greater than p-1)

stEval — *Basic evaluation of fit*

Description

Provides basic measures for evalutating the fit. Includes Brier skill score against the climatology, MSPE, PPL, overall correlation, and a computation of the coverage probabilities for confidence intervals

Usage

```
stEval(forecast, Y, clim)
```

Arguments

`forecast`	stPredict object containing predictions for Y
`Y`	observed values of the response
`clim`	the climatology for the location in Y

Examples

```
data("coprecip")
data("coprecip.predict")

clim = rowMeans(coprecip$Y)
coprecip.predict = stEval(coprecip.predict, coprecip$Y, clim)
```

stFit *Fit the remote effects spatial process (RESP) model*

Description

Fit the remote effects spatial process (RESP) model

Usage

```
stFit(
  stData = NULL,
  priors,
  maxIt,
  X = stData$X,
  Y = stData$Y,
  Z = stData$Z,
  coords.s = stData$coords.s,
  coords.r = stData$coords.r,
  rw.initsd = NULL,
  returnll = T,
  miles = T,
  C = 1,
  alpha = 0.44,
  localOnly = F,
  varying = F,
  remoteOnly = F,
  coords.knots
)
```

Arguments

`stData` Object with class 'stData' containing data needed to fit this model. The data need only be manually entered if not using a stData object.

`priors` A list containing parameters for the prior distributions. The list needs to contain the following values

- **beta** list(Lambda=matrix) specifying the prior covariance matrix for the local effects if varying==F, otherwise list(Psi=matrix, nu=double) specifying the Inverse wishart prior distribution for the spatially varying coefficient process if varying==T.
- **cov.s** list(smoothness=double, range=c(min, max), variance=c(shape, rate), nugget=c(shape, rate))
- **cov.r** list(smoothness=double, range=c(min, max), variance=c(shape, rate), nugget=c(shape, rate))

`maxIt` number of iterations to run the MCMC chain for

`X` [ns, p, nt] array of design matrices with local covariates

`Y`	[ns, nt] matrix with response data
`Z`	[nr, nt] matrix with remote covariates
`coords.s`	matrix with coordinates where responses were observed (lon, lat)
`coords.r`	matrix with coordinates where remote covariates were observed (lon, lat)
`rw.initsd`	A list containing initial standard deviation parameters for the MCMC parameters requiring random walk updates **cov.s** list(range=double, nugget=double) **cov.r** list(range=double, variance=double, nugget=double)
`returnll`	TRUE to compute the model log-likelihood at each iteration
`miles`	TRUE if covariance matrix distances should be in miles, FALSE for kilometers
`C`	scaling factor used in adapting random walk proposal variances.
`alpha`	target acceptance rate for random walk proposals.
`localOnly`	TRUE to fit the model without the teleconnection effects (typically for evaluating impact of teleconnection effects)
`varying`	(depreceated) TRUE to fit the model with spatially varying local coefficients
`remoteOnly`	TRUE to fit the model without local effects. This will fit a local intercept, but will not incorporate local covariates.
`coords.knots`	matrix with coordinates where remote teleconnections will be based (lon, lat)

Examples

```
library(dplyr)
library(foreach)
library(itertools)

set.seed(2018)

data("coprecip")
data("coprecip.fit")
attach(coprecip)

coprecip.fit = stFit(stData = coprecip, priors = coprecip.fit$priors,
                     maxIt = 10, coords.knots = coprecip.fit$coords.knots)
```

`stLL` *Compute log likelihood for model*

Description

Compute log likelihood for model

Usage

```
stLL(
  stData,
  stFit,
  beta,
  sigmasq_y,
  sigmasq_r,
  sigmasq_eps,
  rho_y,
  rho_r,
  X = stData$X,
  Y = stData$Y,
  Z = stData$Z,
  coords.s = stData$coords.s,
  coords.r = stData$coords.r,
  coords.knots = stFit$coords.knots,
  miles = TRUE,
  sigmasq_r_eps
)
```

Arguments

`stData`	Object with class 'stData' containing data needed to fit this model. The data need only be manually entered if not using a stData object.
`stFit`	Object with class 'stFit' containing posterior parameter samples needed to composition sample the teleconnection effects and generate posterior predictions. The data needed from stFit need only be manually entered if not using a stData object.
`beta`	values of β at which to evaluate the likelihood
`sigmasq_y`	values of σ^2_w at which to evaluate the likelihood
`sigmasq_r`	values of σ^2_α at which to evaluate the likelihood
`sigmasq_eps`	values of σ^2_ε at which to evaluate the likelihood
`rho_y`	values of ρ_w at which to evaluate the likelihood
`rho_r`	values of ρ_α at which to evaluate the likelihood
`X`	[ns, p, nt] array of design matrices with local covariates
`Y`	[ns, nt] matrix with response data
`Z`	[nr, nt] matrix with remote covariates
`coords.s`	matrix with coordinates where responses were observed (lon, lat)
`coords.r`	matrix with coordinates where remote covariates were observed (lon, lat)
`coords.knots`	matrix with coordinates of knots for remote covariates (lon, lat)
`miles`	TRUE if distances should be computed in miles (kilometers otherwise)
`sigmasq_r_eps`	values of $\sigma^2_{\alpha_\varepsilon}$ at which to evaluate the likelihood

Examples

```
library(dplyr)
library(foreach)
library(itertools)

set.seed(2018)

data("coprecip")
data("coprecip.fit")
attach(coprecip)

ests = coef(coprecip.fit, burn = 50)

ll = stLL(stData = coprecip, stFit = coprecip.fit,
          beta = matrix(ests$beta, ncol = 2),
          sigmasq_y = ests$sigmasq_y, sigmasq_r = ests$sigmasq_r,
          sigmasq_eps = ests$sigmasq_eps,
          rho_y = ests$rho_y, rho_r = ests$rho_r,
          sigmasq_r_eps = 0)
```

---

stPredict *Compute forecasts based on posterior samples*

Description

Predict response at new timepoints by drawing samples of the response from the posterior predictive distribution. Since this requires sampling teleconnection effects, this method can return estimates of the teleconnection effects as a by-product.

Usage

```
stPredict(
  stFit,
  stData,
  stDataNew,
  burn = 1,
  prob = 0.95,
  ncores = 1,
  conf = 0.95,
  tLabs = stDataNew$tLabs,
  X = stData$X,
  Y = stData$Y,
  Z = stData$Z,
  Xnew = stDataNew$X,
  Znew = stDataNew$Z,
  coords.s = stData$coords.s,
  coords.r = stData$coords.r,
  returnAlphas = T,
```

```
  cat.probs = c(1/3, 2/3),
  returnFullAlphas = F
)
```

Arguments

- `stFit` Object with class 'stFit' containing posterior parameter samples needed to composition sample the teleconnection effects and generate posterior predictions. The data needed from stFit need only be manually entered if not using a stData object.
- `stData` Object with class 'stData' containing data needed to fit this model. The data need only be manually entered if not using a stData object.
- `stDataNew` object of class stData that includes information needed for making forecasts. If response data is included, this function will automatically run stEval using the empirical climatology as the reference forecast
- `burn` number of posterior samples to burn before drawing composition samples
- `prob` confidence level for approximate confidence intervals of teleconnection effects (only needed if returnAlphas==TRUE)
- `ncores` Since the teleconnection effects and posterior predictions can be sampled in parallel, this parameter lets users specify the number of cores to use to draw teleconnection and prediction samples
- `conf` Parameter specifying the HPD level to compute for posterior predictive samples
- `tLabs` Forecast timepoint labels
- `X` [ns, p, nt] array of design matrices with local covariates
- `Y` [ns, nt] matrix with response data
- `Z` [nr, nt] matrix with remote covariates
- `Xnew` [ns, p, nt0] array of design matrices with local covariates at forecast timepoints
- `Znew` [nr, nt0] matrix with remote covariates at forecast timepoints
- `coords.s` matrix with coordinates where responses were observed (lon, lat)
- `coords.r` matrix with coordinates where remote covariates were observed (lon, lat)
- `returnAlphas` TRUE to return the teleconnection effects sampled at knot locations. Note that only basic summary information about the teleconnection effects will be returned.
- `cat.probs` vector of probabilities for also returning categorical predictions from the posterior prediction samples; NULL otherwise
- `returnFullAlphas` TRUE to return the teleconnection effects. Note that only basic summary information about the teleconnection effects will be returned.

Examples

```
set.seed(2018)

data("coprecip")
```

```
data("coprecip.fit")

coprecip.predict = stPredict(stFit = coprecip.fit, stData = coprecip,
                             stDataNew = coprecip, burn = 90,
                             returnFullAlphas = FALSE)
```

`stSimulate` *Simulate responses from the spatio-temporal teleconnection model*

Description

This function simulates spatio-temporal data. The intention is that data Y and latent parameters alpha will be generated using provided covariates X and Z; spatial domains coords.s, coords.r, and coords.knots; and model parameters.

Usage

```
stSimulate(dat.train, dat.test, coords.knots, params, miles = T)
```

Arguments

`dat.train`	stData object with training data to simulate new Y values for
`dat.test`	stData object with test data to simulate new Y values for
`coords.knots`	matrix with coordinates of knots for remote covariates (lon, lat)
`params`	A list containing model parameters for use in simulation **beta** vector with fixed effect coefficients **cov.s** list(smoothness=double, range=double, variance=double, nugget=double) **cov.r** list(smoothness=double, range=double, variance=double, nugget=double)
`miles`	TRUE to compute distances for evaluating covariance functions in miles. This is important since the interpretations of the cov.r and cov.s parameters depend on the units with which distance is measured.

Examples

```
set.seed(2018)

data("coprecip")
data("coprecip.fit")

coprecip.predict = stPredict(stFit = coprecip.fit, stData = coprecip,
                             stDataNew = coprecip, burn = 90,
                             returnFullAlphas = FALSE)
```

`stVIF` *Computes variance inflation factors for fixed effects of the teleconnection model*

Description

VIFs will be computed at the posterior mean of all covariance parameters.

Usage

```
stVIF(stData, stFit, burn)
```

Arguments

`stData`	Object with class 'stData' containing data needed to fit this model.
`stFit`	Object with class 'stFit' containing posterior parameter samples needed to composition sample the teleconnection effects and generate posterior predictions.
`burn`	number of posterior samples to burn before drawing composition samples

Examples

```
data("coprecip")
data("coprecip.fit")

stVIF(stData = coprecip, stFit = coprecip.fit, burn = 50)
```

`summariseAlpha` *Summarize alphas*

Description

This function computes approximate normal intervals, etc. for fitted alphas.

Usage

```
summariseAlpha(alpha, prob = 0.95, coords.s, coords.r)
```

Arguments

`alpha`	structure containing posterior inference for remote coefficients
`prob`	confidence level for confidence intervals and significance
`coords.s`	matrix with coordinates where responses were observed (lon, lat)
`coords.r`	matrix with coordinates where remote covariates were observed (lon, lat)

Examples

```
## Not run:
data("coprecip")
data("coprecip.fit")
attach(coprecip)

# sample posterior predictive distributions AND estimate teleconnection effects
coprecip.precict = stPredict(stFit = coprecip.fit, stData = coprecip,
                             stDataNew = coprecip, burn = 90,
                             returnFullAlphas = TRUE)

alpha.90 = summariseAlpha(alpha = coprecip.precict$alpha, prob = .9,
                          coords.s = coords.s, coords.r = coords.r)

## End(Not run)
```

`summariseEOFAlpha` *Summarize eof-mapped alphas*

Description

This function computes approximate normal intervals, etc. for fitted eof-mapped alphas.

Usage

```
summariseEOFAlpha(eof_alpha, prob = 0.95, coords.s)
```

Arguments

`eof_alpha`	structure containing posterior inference for transformed remote coefficients
`prob`	confidence level for confidence intervals and significance
`coords.s`	matrix with coordinates where responses were observed (lon, lat)

Examples

```
data("coprecip.predict")
attach(coprecip.predict)

alpha.eof.90 = summariseEOFAlpha(eof_alpha = eof_alpha_knots, prob = .9,
  coords.s = coords.s)
```

summary.stPredict *Plot stPredict objects*

Description

This function prints basic summary info for telefit stPredict objects

Usage

```
## S3 method for class 'stPredict'
summary(object, t = NULL, digits = NULL, ...)
```

Arguments

object	Object of class stPredict to summarise
t	timepoint to plot. Will automatically plot all timepoints and overall summary if NULL.
digits	Number of digits to pass to signif, if not NULL.
...	S3 generic/method consistency

Examples

```
data("coprecip.predict")
summary(coprecip.predict)
```

svcFit *Fit a spatially varying coefficient model*

Description

Fit a spatially varying coefficient model

Usage

```
svcFit(
  y,
  X,
  z,
  coords,
  miles = T,
  priors,
  nSamples,
  thin = 1,
```

```
  rw.initsd = 0.1,
  inits = list(),
  C = 1,
  alpha = 0.44
)
```

Arguments

- `y` — vector containing responses for each timepoint. vector is blocked by timepoint.
- `X` — matrix containing local covariates for each timepoint. each row are the covariates for one location and timepoint. matrix is blocked by timepoint.
- `z` — matrix containing remote covariates. each column has remote covariates for one timepoint.
- `coords` — n x 2 matrix containing lon-lat coordinates for locations.
- `miles` — T/F for whether to compute great circle distances in miles (T) or km (F)
- `priors` — A list containing parameters for the prior distributions. The list needs to contain the following values
  - **T** list(Psi=matrix, nu=double) specifying the Inverse wishart prior distribution for the spatially varying coefficient process.
  - **beta** list(Linv=matrix) specifying the prior precision matrix for the fixed local covariates.
  - **sigmasq** list(a=double, b=double) specifying the prior shape and scale parameters for the covariance scale and nugget parameters.
  - **rho** list(L=double, U=double) specifying the lower and upper bounds for the spatial range parameter.
  - **cov** list(nu=double) specifying the smoothness for the matern covariance.
- `nSamples` — number of MCMC iterations to run
- `thin` — MCMC thinning; defaults to no thinning (thin=1)
- `rw.initsd` — Initial proposal standard deviation for RW samplers
- `inits` — optional list containing starting parameters for MCMC sampler
- `C` — scaling factor used in adapting random walk proposal variances.
- `alpha` — target acceptance rate for random walk proposals.

Examples

```
library(fields)
library(mvtnorm)

set.seed(2018)

# set key parameters
dims = list(N=100, nt=3, k=2, p=2)
params = list(sigmasq=.2, rho=.3, eps=.5, nu=.5)

# generate parameters and data
```

```
coords = matrix( runif(2 * dims$N), ncol = 2 )
X = matrix( rnorm(dims$p * dims$N * dims$nt), ncol = dims$p )
beta = c(-1, .5)
z = matrix( rnorm(dims$k * dims$nt), ncol = dims$nt)
H = maternCov(rdist.earth(coords), scale = params$sigmasq, range = params$rho,
              smoothness = params$nu, nugget = params$sigmasq * params$eps)
Hinv = solve(H)
Tm = matrix(c(.5,.2, .2, .5), ncol=2)/2
theta = kronSamp(Hinv, Tm)


# generate response
xb = X %*% beta
zt = as.numeric(apply(z, 2, function(d) {
  kronecker(diag(dims$N), t(d)) %*% theta }))
w = kronSamp(diag(dims$nt), H)
y =  xb + zt + w


# fit model
it = 100
priors = list(
  T = list(Psi = .1*diag(dims$k), nu = dims$k),
  beta = list(Linv = diag(dims$p) * 1e-2),
  sigmasq = list(a=2, b=1),
  rho = list(L=0, U=1),
  cov = list(nu=.5)
)

fit = svcFit(y=y, X=X, z=z, coords=coords, priors=priors, nSamples=it)


#
# predict at new timepoints
#

# generate parameters and data
nt0 = 3
Xn = matrix( rnorm(dims$p * dims$N * nt0), ncol = dims$p )
zn = matrix( rnorm(dims$k * nt0), ncol = nt0)

# generate response
xbn = Xn %*% beta
ztn = as.numeric(apply(zn, 2, function(d) {
  kronecker(diag(dims$N), t(d)) %*% theta }))
wn = kronSamp(diag(nt0), H)
yn =  xbn + ztn + wn

# predict responses
pred = svcPredict(fit, Xn, zn, burn = 50)
```

svcPredict *Make predictions using a fitted varying coefficient model*

Description

Make predictions using a fitted varying coefficient model

Usage

```
svcPredict(
  fit,
  Xn = NULL,
  Zn = NULL,
  stData = NULL,
  stDataNew = NULL,
  burn = 0,
  cat.probs = c(1/3, 2/3),
  conf = 0.95
)
```

Arguments

`fit`	svcFit object containing posterior samples
`Xn`	[nr*nt, p] matrix of local covariates at new timepoint
`Zn`	[nr, nt] matrix of remote covariates at new timepoints
`stData`	Object with class 'stData' containing data needed to fit this model. The data is used to compute empirical quantiles for making categorical predictions.
`stDataNew`	object of class stData that includes information needed for making forecasts.
`burn`	number of posterior samples to burn from fit
`cat.probs`	vector of probabilities for also returning categorical predictions from the posterior prediction samples; NULL otherwise
`conf`	Parameter specifying the HPD level to compute for posterior predictive samples

Examples

```
library(fields)
library(mvtnorm)

set.seed(2018)

# set key parameters
dims = list(N=100, nt=3, k=2, p=2)
params = list(sigmasq=.2, rho=.3, eps=.5, nu=.5)

# generate parameters and data
coords = matrix( runif(2 * dims$N), ncol = 2 )
```

```
X = matrix( rnorm(dims$p * dims$N * dims$nt), ncol = dims$p )
beta = c(-1, .5)
z = matrix( rnorm(dims$k * dims$nt), ncol = dims$nt)
H = maternCov(rdist.earth(coords), scale = params$sigmasq, range = params$rho,
              smoothness = params$nu, nugget = params$sigmasq * params$eps)
Hinv = solve(H)
Tm = matrix(c(.5,.2, .2, .5), ncol=2)/2
theta = kronSamp(Hinv, Tm)


# generate response
xb = X %*% beta
zt = as.numeric(apply(z, 2, function(d) {
  kronecker(diag(dims$N), t(d)) %*% theta }))
w = kronSamp(diag(dims$nt), H)
y =  xb + zt + w


# fit model
it = 100
priors = list(
  T = list(Psi = .1*diag(dims$k), nu = dims$k),
  beta = list(Linv = diag(dims$p) * 1e-2),
  sigmasq = list(a=2, b=1),
  rho = list(L=0, U=1),
  cov = list(nu=.5)
)

fit = svcFit(y=y, X=X, z=z, coords=coords, priors=priors, nSamples=it)


#
# predict at new timepoints
#

# generate parameters and data
nt0 = 3
Xn = matrix( rnorm(dims$p * dims$N * nt0), ncol = dims$p )
zn = matrix( rnorm(dims$k * nt0), ncol = nt0)

# generate response
xbn = Xn %*% beta
ztn = as.numeric(apply(zn, 2, function(d) {
  kronecker(diag(dims$N), t(d)) %*% theta }))
wn = kronSamp(diag(nt0), H)
yn =  xbn + ztn + wn

# predict responses
pred = svcPredict(fit, Xn, zn, burn = 50)
```

teleCor — *Pointwise correlations for an exploratory teleconnection analysis*

Description

Computes empirical correlations between rows of Y and Z, for use as exploratory analysis of teleconnection patterns between locations indexed by coords.s and coords.r. Optionally, an stData object containing Y and Z can be passed instead.

Usage

```
teleCor(
  stData = NULL,
  Y = stData$Y,
  Z = stData$Z,
  coords.s = stData$coords.s,
  coords.r = stData$coords.r
)
```

Arguments

stData	stData object containing data to analyze
Y	[ny x nt] a matrix composed of ny row vectors, each of which contains nt observations fom a different spatial location. Spatial locations for Y are indexed by coords.s.
Z	[nz x nt] a matrix composed of nz row vectors each of which contains nt observations from a different spatial location. Spatial locations for Z are indexed by coords.r.
coords.s	coordinates of locations in Y
coords.r	coordinates of locations in Z

Value

list with a matrix 'cor' containing correlations. The columns index remote coordinates, while the rows index the local coordinates. The returned list also includes the coordinates.

Examples

```
data("coprecip")

cors = teleCor(coprecip)
```

telefit *Tools for modeling teleconnections*

Description

The package **telefit** provides functions for fitting the remote effects spatial process (RESP) model.

Index