Package ‘spTimer’

August 29, 2013

Type Package

Title Spatio-Temporal Bayesian Modelling Using R

Version 0.8

Date 2013-07-04

Author K. Shuvo Bakar & Sujit K. Sahu

Maintainer Shuvo Bakar <Shuvo.Bakar@csiro.au>

URL http://www.southampton.ac.uk/~sks/research/papers/spTimeRpaper.pdf

Depends R (>= 2.14.0), coda, lattice, forecast

Description The package is able to fit, spatially predict and temporally forecast large amounts of space-time data using [1] Bayesian Gaussian Process (GP) Models, [2] Bayesian Auto-Regressive (AR) Models, and [3] Bayesian Gaussian Predictive Processes (GPP) based AR Models.

License GPL (>= 2)

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-07-04 07:52:37

R topics documented:

`spTimer-package` *Spatio-Temporal Bayesian Modelling using R*

Description

This package uses different hierarchical Bayesian spatio-temporal modelling strategies, namely the gaussian processes (GP) models, the autoregressive (AR) models, and models using Gaussian predictive processes (GPP) approximation to analyse space-time observations.

Details

Package:	spTimer
Type:	Package

The back-end code of this package is built under c language.
Main functions used:
> `spT.Gibbs`
> `predict.spT`
Some other important functions:
> `spT.priors`
> `spT.initials`
> `spT.decay`
> `spT.time`
Data descriptions:
> `NYdata`

Author(s)

K.S. Bakar & S.K. Sahu
Maintainer: K.S. Bakar <shuvo.bakar@csiro.au>

References

Bakar, K. S. and Sahu, S. K. (2013) spTimer: Spatio-Temporal Bayesian Modelling Using R. Technical Report, University of Southampton, UK.
Sahu, S. K. and Bakar, K. S. (2012) A comparison of Bayesian Models for Daily Ozone Concentration Levels Statistical Methodology , 9, 144-157.
Sahu, S. K. and Bakar, K. S. (2012) Hierarchical Bayesian auto-regressive models for large space time data with applications to ozone concentration modelling. Applied Stochastic Models in Business and Industry, 28, 395-415.
Sahu, S. K., Bakar, K. S. and Awang, N. (2013) Bayesian Forecasting Using Hierarchical Spatio-temporal Models with Applications to Ozone Levels in the Eastern United States. Technical Report, University of Southampton.
Sahu, S.K., Gelfand, A.E., & Holland, D.M. (2007). High-Resolution Space-Time Ozone Modelling for Assessing Trends. Journal of the American Statistical Association, 102, 1221-1234.

See Also

Packages 'forecast'; 'spBayes'; 'maps'; 'MBA'; 'coda'; website: `http://www.r-project.org/`.

`as.forecast.object` *Conversion of spT object into forecast object*

Description

This function is used to convert the spT object from the temporal prediction output of "spTimer" into forecast object of the package "forecast".

Usage

```
as.forecast.object(object, site=1, level=c(80,95), ...)
```

Arguments

`object`	Object of class inheriting from "spT".
`site`	Selection of location/site for which the time-series will convert into forecast object.
`level`	Confidence level for temporal prediction intervals, see details in `forecast`.
`...`	Other arguments, see details in `forecast`.

Value

An object class `"forecast"`

`model`	Name of the fitted model.
`method`	Name of the forecasting method.

`mean`	MCMC mean for the temporal predictions.
`lower`	Lower limits for the temporal prediction intervals obtained from the MCMC samples.
`upper`	Upper limits for the temporal prediction intervals obtained from the MCMC samples.
`level`	Prediction interval limits.
`x`	The data used for model fitting in time-series format, see `ts`.
`residuals`	Residuals of the fiited model.
`fitted`	Fitted MCMC mean.

See Also

`spT.Gibbs`, `predict.spT`.

Examples

```
## Not run:
##

# 'out' is the output of spT class obtained from temporal prediction
fobj<-as.forecast.object(out)
class(fobj)
summary(fobj)
plot(fobj)

##

## End(Not run)
```

`NYdata` *Observations of ozone concentration levels, maximum temperature and wind speed.*

Description

This data set contains values of daily 8-hour maximum average ozone concentrations (parts per billion (ppb)), maximum temperature (in degree Celsius), wind speed (knots), and relative humidity, obtained from 28 monitoring sites of New York, USA.

Total 4 seperate subset datasets are created from NYdata that contains same variables as NYdata.

1. DataFit: This data is used for model fitting, contains 20 monitoring locations with 62 days observations; total 1200 rows.

2. DataValPred: This data is used for model validation for spatial prediction, contains 8 monitoring locations with 60 days observations; total 480 rows.

3. DataValFore: This data is used for model validation for temporal prediction in the unobserved locations, contains 8 monitoring locations with 2 days observations; total 8 rows.

4. DataFitFore: This data is used for model validation for temporal prediction in the observed locations, contains 20 monitoring locations with 2 days observations; total 40 rows.

We also include a dataset that contains gridded observations over NY state.

NYgrid: This dataset contains total 6200 rows for 62 days of observations for 10x10 = 100 grid points.

Usage

```
NYdata
```

Format

Columns for NYdata: each contains 1798 observations.

- 1st col = Site index (s.index),
- 2nd col = Longitude,
- 3rd col = Latitude,
- 4th col = Year,
- 5th col = Month,
- 6th col = Day,
- 7th col = Ozone (o8hrmax),
- 8th col = Maximum temperature (cMAXTMP),
- 9th col = Wind speed (WDSP).
- 10th col = Relative humidity (RH).

Source

US EPA

See Also

DataFit, DataFitFore, DataValFore, DataValPred, NYgrid.

Examples

```
## Not run:
##
  library("spTimer")
# NY data
  data(NYdata)
  head(NYdata)
# plots in NY map
  NYsite<-unique(cbind(NYdata[,1:3]))
  head(NYsite)
# map
  library(maps)
  map(database="state",regions="new york")
  points(NYsite[,2:3],pch=19)
```

```
# DataFit
  data(DataFit)
  head(DataFit)
# DataValPred
  data(DataValPred)
  head(DataValPred)
# DataValFore
  data(DataValFore)
  head(DataValFore)
# DataFitFore
  data(DataFitFore)
  head(DataFitFore)

# Plot fitted and validation locations in map
  fit.coords<-unique(cbind(DataFit[,1:3]))
  val.coords<-unique(cbind(DataValPred[,1:3]))

  library(maps)
  map(database="state",regions="new york")
  points(fit.coords[,2:3],pch=19,col=2)
  points(val.coords[,2:3],pch=7,col=1)
  legend(x=-78,y=41.5,pch=c(19,7),col=c(2,1),bty="n",
  legend=c("Fitted locations", "Validation locations"))

# Grid data
  data(NYgrid)
  head(NYgrid)
  grid.coords<-unique(cbind(NYgrid[,8:9]))
  library(maps)
  plot(grid.coords,pch=19,col=1)
  map(database="state",regions="new york",add=TRUE)

##

## End(Not run)
```

`plot.spT` *Plots for spTimer output.*

Description

This function is used to obtain MCMC summary and residual plots.

Usage

```
## S3 method for class 'spT'
## S3 method for class 'spT'
plot(x, residuals=FALSE, ...)

##
```

Arguments

`x`	Object of class inheriting from "spT".
`residuals`	If TRUE then plot residual vs. fitted and normal qqplot of the residuals. If FALSE then plot MCMC samples of the parameters using coda package. Defaults value is FALSE.
`...`	Other arguments.

See Also

`spT.Gibbs`.

Examples

```
## Not run:
##

plot(out) # where out is the output from spT class
plot(out, residuals=TRUE) # where out is the output from spT class

##

## End(Not run)
```

---

predict.spT — *Spatial and temporal predictions for the spatio-temporal models.*

Description

This function is used to obtain spatial predictions in the unknown locations and also to get the temporal forecasts using MCMC samples.

Usage

```
## S3 method for class 'spT'
## S3 method for class 'spT'
predict(object, newdata, newcoords, foreStep=NULL, type="spatial",
        nBurn, tol.dist, predAR=NULL, ...)
```

Arguments

`object`	Object of class inheriting from "spT".
`newdata`	The data set for the covariate values for spatial prediction or temporal forecasts. This data should have the same space-time structure as the original data frame.
`newcoords`	The coordinates for the prediction or forecast sites. The locations are in similar format to coords.

`foreStep`	Number of K-step (time points) ahead forecast, K=1,2, ...; Only applicable if type="temporal".
`type`	If "spatial" the do spatial prediction and if "temporal" the do temporal prediction or forecast. Default value is "spatial".
`nBurn`	Number of burn-in. Initial MCMC samples to discard before making inference.
`tol.dist`	Minimum tolerance distance limit between fitted and predicted locations.
`predAR`	The prediction output, if forecasts are in the prediction locations. Only applicable if type="forecast" and data fitted with the "AR" model.
`...`	Other arguments.

Value

`pred.samples or fore.samples`	Prediction or forecast MCMC samples.
`pred.coords or fore.coords`	prediction or forecast coordinates.
`Mean`	Average of the MCMC predictions
`Median`	Median of the MCMC predictions
`SD`	Standard deviation of the MCMC predictions
`Low`	Lower limit for the 95 percent CI of the MCMC predictions
`Up`	Upper limit for the 95 percent CI of the MCMC predictions
`computation.time`	The computation time.
`model`	The model method used for prediction.
`type`	"spatial" or "temporal".
`...`	Other values "obsData", "fittedData" and "residuals" are provided only for temporal prediction. This is to analyse the codespTimer forecast output using package `forecast` through function `as.forecast.object`.

References

Bakar, K. S. and Sahu, S. K. (2013) spTimer: Spatio-Temporal Bayesian Modelling Using R. Technical Report, University of Southampton, UK.
Sahu, S. K. and Bakar, K. S. (2012) A comparison of Bayesian Models for Daily Ozone Concentration Levels Statistical Methodology , 9, 144-157.
Sahu, S. K. and Bakar, K. S. (2012) Hierarchical Bayesian auto-regressive models for large space time data with applications to ozone concentration modelling. Applied Stochastic Models in Business and Industry, 28, 395-415.
Sahu, S. K., Bakar, K. S. and Awang, N. (2013) Bayesian Forecasting Using Hierarchical Spatiotemporal Models with Applications to Ozone Levels in the Eastern United States. Technical Report, University of Southampton.

See Also

`spT.Gibbs`, `as.forecast.object`.

Examples

```
## Not run:
##

###########################
## The GP models:
###########################

##
## Spatial prediction/interpolation
##

# Read data
data(DataValPred)

# Define prediction coordinates
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))

# Spatial prediction using spT.Gibbs output
set.seed(11)
pred.gp <- predict(post.gp, newdata=DataValPred, newcoords=pred.coords)
print(pred.gp)
names(pred.gp)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(pred.gp$Mean))

##
## Temporal  prediction/forecast
## 1. In the unobserved locations
##

# Read data
data(DataValFore);

# define forecast coordinates
fore.coords<-as.matrix(unique(cbind(DataValFore[,2:3])))

# Two-step ahead forecast, i.e., in day 61 and 62
# in the unobserved locations using output from spT.Gibbs
set.seed(11)
fore.gp <- predict(post.gp, newdata=DataValFore, newcoords=fore.coords,
           type="temporal", foreStep=2)
print(fore.gp)
names(fore.gp)

# Forecast validations
spT.validation(DataValFore$o8hrmax,c(fore.gp$Mean))

# Use of "forecast" class
tmp<-as.forecast.object(fore.gp, site=1) # default for site 1
plot(tmp)
```

```
##
## Temporal  prediction/forecast
## 2. In the observed/fitted locations
##

# Read data
data(DataFitFore)

# Define forecast coordinates
fore.coords<-as.matrix(unique(cbind(DataFitFore[,2:3])))

# Two-step ahead forecast, i.e., in day 61 and 62,
# in the fitted locations using output from spT.Gibbs
set.seed(11)
fore.gp <- predict(post.gp, newdata=DataFitFore, newcoords=fore.coords,
           type="temporal", foreStep=2)
print(fore.gp)
names(fore.gp)

# Forecast validations
spT.validation(DataFitFore$o8hrmax,c(fore.gp$Mean)) #

# Use of "forecast" class
tmp<-as.forecast.object(fore.gp, site=5) # for site 5
plot(tmp)

##
## Fit and spatially prediction simultaneously
##

# Read data
data(DataFit);
data(DataValPred)

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))

# MCMC via Gibbs will provide output in *.txt format
# from C routine to avoide large data problem in R
set.seed(11)
post.gp.fitpred <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="GP", coords=coords,
         newcoords=pred.coords, newdata=DataValPred,
         scale.transform="SQRT")
print(post.gp.fitpred)
summary(post.gp.fitpred)
coef(post.gp.fitpred)
plot(post.gp.fitpred,residuals=TRUE)
names(post.gp.fitpred)

# validation criteria
```

```
spT.validation(DataValPred$o8hrmax,c(post.gp.fitpred$prediction[,1]))

###########################
## The AR models:
###########################

##
## Spatial prediction/interpolation
##

# Read data
data(DataValPred)

# Define prediction coordinates
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))

# Spatial prediction using spT.Gibbs output
set.seed(11)
pred.ar <- predict(post.ar, newdata=DataValPred, newcoords=pred.coords)
print(pred.ar)
names(pred.ar)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(pred.ar$Mean))

##
## Temporal  prediction/forecast
## 1. In the unobserved locations
##

# Read data
data(DataValFore);

# define forecast coordinates
fore.coords<-as.matrix(unique(cbind(DataValFore[,2:3])))

# Two-step ahead forecast, i.e., in day 61 and 62
# in the unobserved locations using output from spT.Gibbs
set.seed(11)
fore.ar <- predict(post.ar, newdata=DataValFore, newcoords=fore.coords,
           type="temporal", foreStep=2, predAR=pred.ar)
print(fore.ar)
names(fore.ar)

# Forecast validations
spT.validation(DataValFore$o8hrmax,c(fore.ar$Mean))

# Use of "forecast" class
tmp<-as.forecast.object(fore.ar, site=1) # default for site 1
plot(tmp)


##
```

```
## Temporal  prediction/forecast
## 2. In the observed/fitted locations
##

# Read data
data(DataFitFore)

# Define forecast coordinates
fore.coords<-as.matrix(unique(cbind(DataFitFore[,2:3])))

# Two-step ahead forecast, i.e., in day 61 and 62,
# in the fitted locations using output from spT.Gibbs
set.seed(11)
fore.ar <- predict(post.ar, newdata=DataFitFore, newcoords=fore.coords,
           type="temporal", foreStep=2)
print(fore.ar)
names(fore.ar)

# Forecast validations
spT.validation(DataFitFore$o8hrmax,c(fore.ar$Mean)) #

# Use of "forecast" class
tmp<-as.forecast.object(fore.ar, site=1) # default for site 1
plot(tmp)

##
## Fit and spatially prediction simultaneously
##

# Read data
data(DataFit);
data(DataValPred)

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))

# MCMC via Gibbs will provide output in *.txt format
# from C routine to avoide large data problem in R
set.seed(11)
post.ar.fitpred <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="AR", coords=coords,
         newcoords=pred.coords, newdata=DataValPred,
         scale.transform="SQRT")
print(post.ar.fitpred)
summary(post.ar.fitpred)
coef(post.ar.fitpred)
names(post.ar.fitpred)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(post.ar.fitpred$prediction[,1]))

#################################
```

```
## The GPP approximation models:
##################################

##
## Spatial prediction/interpolation
##

# Read data
data(DataValPred)

# Define prediction coordinates
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))

# Spatial prediction using spT.Gibbs output
set.seed(11)
pred.gpp <- predict(post.gpp, newdata=DataValPred, newcoords=pred.coords)
print(pred.gpp)
names(pred.gpp)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(pred.gpp$Mean))

##
## Temporal  prediction/forecast
## 1. In the unobserved locations
##

# Read data
data(DataValFore);

# define forecast coordinates
fore.coords<-as.matrix(unique(cbind(DataValFore[,2:3])))

# Two-step ahead forecast, i.e., in day 61 and 62
# in the unobserved locations using output from spT.Gibbs
set.seed(11)
fore.gpp <- predict(post.gpp, newdata=DataValFore, newcoords=fore.coords,
           type="temporal", foreStep=2)
print(fore.gpp)
names(fore.gpp)

# Forecast validations
spT.validation(DataValFore$o8hrmax,c(fore.gpp$Mean))

# Use of "forecast" class
tmp<-as.forecast.object(fore.gpp, site=1) # default for site 1
plot(tmp)

##
## Temporal  prediction/forecast
## 2. In the observed/fitted locations
##
```

```
# Read data
data(DataFitFore)

# Define forecast coordinates
fore.coords<-as.matrix(unique(cbind(DataFitFore[,2:3])))

# Two-step ahead forecast, i.e., in day 61 and 62,
# in the fitted locations using output from spT.Gibbs
set.seed(11)
fore.gpp <- predict(post.gpp, newdata=DataFitFore, newcoords=fore.coords,
           type="temporal", foreStep=2)
print(fore.gpp)
names(fore.gpp)

# Forecast validations
spT.validation(DataFitFore$o8hrmax,c(fore.gpp$Mean)) #

# Use of "forecast" class
tmp<-as.forecast.object(fore.gpp, site=1) # default for site 1
plot(tmp)

##
## Fit and spatially prediction simultaneously
##

# Read data
data(DataFit);
data(DataValPred)

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))
knots.coords<-spT.grid.coords(Longitude=c(max(coords[,1]),
              min(coords[,1])),Latitude=c(max(coords[,2]),
              min(coords[,2])), by=c(4,4))

# MCMC via Gibbs will provide output in *.txt format
# from C routine to avoide large data problem in R
set.seed(11)
post.gpp.fitpred <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="GP", coords=coords, knots.coords=knots.coords,
         newcoords=pred.coords, newdata=DataValPred,
         scale.transform="SQRT")
print(post.gpp.fitpred)
summary(post.gpp.fitpred)
coef(post.gpp.fitpred)
plot(post.gpp.fitpred, residuals=TRUE)
names(post.gpp.fitpred)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(post.gpp.fitpred$prediction[,1]))

##
```

```
## End(Not run)
```

---

spT.check.locations *Distance Monitoring Function*

Description

This function is used to check the minimum distance between two locations.

Usage

```
spT.check.locations(fit.locations, pred.locations,
   method="geodetic:km", tol = 5)

spT.check.sites.inside(coords, method)
```

Arguments

fit.locations	The locations for the fitted observations.
pred.locations	The locations for the predicted observations.
method	The distance measurement. The available methods are "geodetic:km", "geodetic:mile", "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".
tol	The tolerance limit for the distance.
coords	The longitude and latitude positions in a matrix format.

Details

`spT.check.locations` is used to check minimum distance between two locations, e.g., fitted and prediction. `spT.check.sites.inside` is used to check distance within the location points. Here, the tol. limit is 0.01. If output shows nothing then we can say distances are alright. These functions are used to avoid of occuring non-positive definite correlation matrix.

See Also

`spT.geodist`, `dist`, `spT.data.selection`.

Examples

```
## Not run:
##

data(NYdata)
head(NYdata)
NYsite<-unique(NYdata[,1:3])

# Sample 4 sites randomly from the data NYdata.
```

```
r4<-spT.data.selection(data=NYsite, random=TRUE, num.rs=4)

# Choose purposively sites numbered as 2, 8, and 12, 15.

p4<-spT.data.selection(data=NYsite, random=FALSE, s=c(2,8,12,15))

# Check locations of datasets r4 and p4

spT.check.locations(fit.locations=r4, pred.locations=p4,
    method="geodetic:km", tol=5)

#
spT.check.sites.inside(NYsite[,2:3],method="geodetic:km")

# if nothing appears then distances are alright

##

## End(Not run)
```

`spT.data.selection` *Selection of Spatial data from a big dataset.*

Description

This command selects a part of the big spatial dataset using the site numbers.

Usage

```
spT.data.selection(data, random = TRUE, num.rs = NULL,
                      s = NULL, reverse=FALSE)
```

Arguments

`data`	The dataset.
`random`	Logical value: if TRUE then the `num.rs` sites are randomly sampled, if FALSE then we need to provide the value for `s`.
`num.rs`	The number of sites to be selected, e.g., 3.
`s`	The site numbers to be selected, e.g., c(2,8,12).
`reverse`	Logical value: if TRUE then `num.rs` will be discarded from the data.

See Also

`NYdata`.

Examples

```
## Not run:
##

# Load ozone concentration data for New York.

data(NYdata)
NYdata

# Sample 4 sites randomly from the data NYdata.

r4<-spT.data.selection(data=NYdata, random=TRUE, num.rs=4)

# Choose purposively defined sites numbered as 2, 8, and 12.

p4<-spT.data.selection(data=NYdata, random=FALSE, s=c(2,8,12))

# Donot choose purposively defined sites numbered as 2, 8, and 12.

p4<-spT.data.selection(data=NYdata, random=FALSE, s=c(2,8,12), reverse=TRUE)

##

## End(Not run)
```

`spT.decay` *Choice for sampling spatial decay parameter ϕ.*

Description

This function initialises the sampling method for the spatial decay parameter ϕ.

Usage

```
spT.decay(type="MH", tuning=NULL, limit=NULL, segments=NULL, value=NULL)
```

Arguments

type	The sampling method, currently available methods are, DISCRETE and MH. One can also used FIXED value for ϕ parameter.
tuning	If MH type is used then need to define the tuning parameter for ϕ.
limit	If DISCRETE type is used then need to define the lower and upper limits.
segments	If DISCRETE type is used then need to define the number of segments for the range of limits.
value	If FIXED type is used then need to define the value for ϕ.

See Also

spT.Gibbs.

Examples

```
## Not run:
##

# input for random-walk Metropolis within Gibbs
# sampling for phi parameter
spatial.decay<-spT.decay(type="MH", tuning=0.08)

# input for discrete sampling of phi parameter
spatial.decay<-spT.decay(type="DISCRETE",
           limit=c(0.01,0.02),segments=5)

# input for spatial decay if FIXED is used
spatial.decay<-spT.decay(type="FIXED", value=0.01)

##

## End(Not run)
```

---

spT.geodist *Geodetic/geodesic Distance*

Description

This geodetic distance provides the distance between the locations in Kilometers (k.m.) and Miles, using spherical law of Cosines.

Usage

```
spT.geodist(Lon, Lat, KM = TRUE)

spT.geo.dist(point1, point2)
spT.geo_dist(points)
```

Arguments

Lon	The longitude position.
Lat	The latitude position.
KM	A logical value, if 'TRUE' then output is in 'kilometers', otherwise in 'miles'.
point1	In the form of (longitude, latitude) position.
point2	In the form of (longitude, latitude) position.
points	In the form of points 1:(longitude, latitude) 2:(longitude, latitude) positions.

Details

spT.geodist is used to get geodetic distance in both miles and kilometers. spT.geo.dist is only used to get geodetic distance in kilometers with a different format. spT.geo_dist is only used to get geodetic distance in kilometers with a different format.

See Also

NYdata, spT.grid.coords.

Examples

```
## Not run:
##

# Load 28 ozone monitoring locations of New York.
data(NYdata)
head(NYdata)
NYsite<-unique(NYdata[,1:3])

# Find the geodetic distance in km
spT.geodist(Lon=NYsite$Longitude, Lat=NYsite$Latitude, KM=TRUE)

# Find the geodetic distance in miles
spT.geodist(Lon=NYsite$Longitude, Lat=NYsite$Latitude, KM=FALSE)

##

# using spT.geo.dist
point1<-c(-73.757,42.681)
point2<-c(-73.881,40.866)
spT.geo.dist(point1,point2)

# using spT.geo_dist
points<-c(point1,point2)
spT.geo_dist(points)

##

## End(Not run)
```

---

spT.Gibbs *MCMC sampling for the spatio-temporal models.*

Description

This function is used to draw MCMC samples using the Gibbs sampler.

Usage

```
spT.Gibbs(formula, data=parent.frame(), model, time.data,
coords, knots.coords, newcoords, newdata, priors, initials,
nItr, nBurn, report, tol.dist, distance.method, cov.fnc,
scale.transform, spatial.decay, annual.aggrn)
```

Arguments

`formula`	The symnbolic description of the model equation of the regression part of the space-time model.
`data`	An optional data frame containing the variables in the model. If omitted, the variables are taken from environment(formula), typically the environment from which spT.Gibbs is called. The data should be ordered first by the time and then by the sites specified by the `coords` below. One can also supply coordinates through this argument, where coordinate names should be "Latitude"/"Longitude" or "xcoords"/"ycoords".
`model`	The spatio-temporal models to be fitted, current input: "GP", "AR", and "GPP".
`time.data`	Defining the segments of the time-series set up using the function `spT.time`.
`coords`	The n by 2 matrix defining the locations (e.g., longitude/easting, latitude/northing) of the fitting sites, where n is the number of fitting sites. One can also supply coordinates through the argument data, where coordinate names should be "Latitude"/"Longitude" or "xcoords"/"ycoords".
`knots.coords`	The locations of the knots in similar format to coords above, only required if `model="GPP"`.
`newcoords`	The locations of the prediction sites in similar format to coords above, only required if fit and predictions are done simultaneously.
`newdata`	The covariate values at the prediction sites specified by `pred.coords`. This should have same space-time structure as the original data frame.
`priors`	The prior distributions for the parameters. Default distributions are specified if these are not specified. If priors=NULL a flat prior distribution will be used with large variance. See details in `spT.priors`.
`initials`	The preferred initial values for the parameters. If omitted, default values are provided automatically. Further details are provided in `spT.initials`.
`nItr`	Number of MCMC iterations. Default value is 13000.
`nBurn`	Number of burn-in samples. This number of samples will be discarded before making any inference. Default value is 3000.
`report`	Number of reports to display while running the Gibbs sampler. Defaults to number of iterations.
`distance.method`	The preferred method to calculate the distance between any two locations. The available options are "geodetic:km", "geodetic:mile", "euclidean", "maximum", "manhattan", and "canberra". See details in `dist`.

tol.dist — Minimum separation distance between any two locations out of those specified by coords, knots.coords and pred.coords. The default is 0.005. The programme will exit if the minimum distance is less than the non-zero specified value. This will ensure non-singularity of the covariance matrices.

cov.fnc — Covariance function for the spatial effects. The available options are "exponential", "gaussian", "spherical" and "matern". If "matern" is used then by default the smooth parameter (ν) is estimated from (0,1) uniform distribution using discrete samples.

scale.transform — The transformation method for the response variable. Currently implemented options are: "NONE", "SQRT", and "LOG" with "NONE" as the deault.

spatial.decay — Provides the options for sampling the spatial decay parameter ϕ. Currently implemented options are "DISCRETE", "MH" or "FIXED" and further options for each of these are specified by spT.decay. The default is "MH".

annual.aggrn — This provides the options for calculating annual summary statistics by aggregating different time segments (e.g., annual mean). Currently implemented options are: "NONE", "ave" and "an4th", where "ave" = annual average, "an4th"= annual 4th highest. Only applicable if spT.time inputs more than one segment and when fit and predict are done simultaneously.

Value

accept — The acceptance rate for the ϕ parameter if the "MH" method of sampling is chosen.

phip — MCMC samples for the parameter ϕ.

nup — MCMC samples for the parameter ν. Only available if "matern" covariance function is used.

sig2eps — MCMC samples for the parameter σ_ϵ^2.

sig2etap — MCMC samples for the parameter σ_η^2.

betap — MCMC samples for the parameter β.

op — MCMC samples for the true observations.

fitted — MCMC summary (mean and sd) for the fitted values.

tol.dist — Minimum tolerance distance limit between the locations.

distance.method — Name of the distance calculation method.

cov.fnc — Name of the covariance function used in model fitting.

scale.transform — Name of the scale.transformation method.

sampling.sp.decay — The method of sampling for the spatial decay parameter ϕ.

covariate.names — Name of the covariates used in the model.

Distance.matrix — The distance matrix.

`coords` The coordinate values.

`n` Total number of sites.

`r` Total number of segments in time, e.g., years.

`T` Total points of time, e.g., days within each year.

`p` Total number of model coefficients, i.e., β's including the intercept.

`initials` The initial values used in the model.

`priors` The prior distributions used in the model.

`PMCC` The predictive model choice criteria obtained by minimising the expected value of a loss function, see Gelfand and Ghosh (1998). Results for both goodness of fit and penalty are given.

`iterations` The number of samples for the MCMC chain, without burn-in.

`nBurn` The number of burn-in period for the MCMC chain.

`computation.time` The computation time required for the fitted model.

`model` The spatio-temporal model used for analyse the data.

`Text Output` This option is only applicable when fit and predictions are done simultaneously.

For GP models:
OutGP_Values_Parameter.txt: (nItr x parameters matrix) has the MCMC samples for the parameters, ordered as: beta's, sig2eps, sig2eta, and phi.
OutGP_Stats_FittedValue.txt: (N x 2) matrix of fitted summary, with 1st column as mean and 2nd column as standard deviations, where N=nrT.
OutGP_Stats_PredValue.txt: ((predsites*r*T) x 2) matrix of prediction summary, with 1st column as mean and 2nd column as standard deviations.
OutGP_Values_Prediction.txt: (nItr x (predsites*r*T)) matrix of MCMC predicted values in the predicted sites.
If `annual.aggregation="ave"` then we get text output as:
OutGP_Annual_Average_Prediction.txt: (nItr x (predsites*r)) matrix.
If `annual.aggregation="an4th"` then we get text output as:
OutGP_Annual_4th_Highest_Prediction.txt: (nItr x (predsites*r)) matrix.

For AR models:
OutAR_Values_Parameter.txt: (nItr x parameters matrix) has the MCMC samples for the parameters, ordered as: beta's, rho, sig2eps, sig2eta, mu_l's, sig2l's and phi.
OutAR_Stats_TrueValue.txt: (N x 2) matrix of true summary values, with 1st column as mean and 2nd column as standard deviations.
OutAR_Stats_FittedValue.txt: (N x 2) matrix of fitted summary, with 1st column as mean and 2nd column as standard deviations.
OutAR_Stats_PredValue.txt: ((predsites*r*T) x 2) matrix of prediction summary, with 1st column as mean and 2nd column as standard deviations.
OutAR_Values_Prediction.txt: (nItr x (predsites*r*T)) matrix of MCMC predicted values in the predicted sites.
If `annual.aggregation="ave"` then we get text output as:
OutAR_Annual_Average_Prediction.txt: (nItr x (predsites*r)) matrix.

If annual.aggregation="an4th" then we get text output as:
OutAR_Annual_4th_Highest_Prediction.txt: (nItr x (predsites*r)) matrix.

For models using GPP approximations:
OutGPP_Values_Parameter.txt: (nItr x parameters matrix) has the MCMC samples for the parameters, ordered as: beta's, rho, sig2eps, sig2eta, and phi.
OutGPP_Stats_FittedValue.txt: (N x 2) matrix of fitted summary, with 1st column as mean and 2nd column as standard deviations.
OutGPP_Stats_PredValue.txt: ((predsites*r*T) x 2) matrix of prediction summary, with 1st column as mean and 2nd column as standard deviations.
OutGPP_Values_Prediction.txt: (nItr x (predsites*r*T)) matrix of MCMC predicted values in the predicted sites.
If annual.aggregation="ave" then we get text output as:
OutGPP_Annual_Average_Prediction.txt: (nItr x (predsites*r)) matrix.
If annual.aggregation="an4th" then we get text output as:
OutGPP_Annual_4th_Highest_Prediction.txt: (nItr x (predsites*r)) matrix.

References

Bakar, K. S. and Sahu, S. K. (2013) spTimer: Spatio-Temporal Bayesian Modelling Using R. Technical Report, University of Southampton, UK.
Sahu, S. K. and Bakar, K. S. (2012) A comparison of Bayesian Models for Daily Ozone Concentration Levels Statistical Methodology , 9, 144-157.
Sahu, S. K. and Bakar, K. S. (2012) Hierarchical Bayesian auto-regressive models for large space time data with applications to ozone concentration modelling. Applied Stochastic Models in Business and Industry, 28, 395-415.
Sahu, S. K., Bakar, K. S. and Awang, N. (2013) Bayesian Forecasting Using Hierarchical Spatio-temporal Models with Applications to Ozone Levels in the Eastern United States. Technical Report, University of Southampton.

See Also

spT.priors, spT.initials, spT.geodist, dist, summary.spT, plot.spT, predict.spT.

Examples

```
## Not run:
##

###########################
## Attach library spTimer
###########################

library(spTimer)

###########################
## The GP models:
###########################
```

```
##
## Model fitting
##

# Read data
data(DataFit);

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))

# MCMC via Gibbs using default choices
set.seed(11)
post.gp <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="GP", coords=coords,
         scale.transform="SQRT")
print(post.gp)

# MCMC via Gibbs not using default choices
# define the time-series
time.data<-spT.time(t.series=60,segment=1)

# hyper-parameters for the prior distributions
priors<-spT.priors(model="GP",var.prior=Gam(2,1),
        beta.prior=Nor(0,10^4))

# initial values for the model parameters
initials<-spT.initials(model="GP", sig2eps=0.01,
            sig2eta=0.5, beta=NULL, phi=0.001)

# input for spatial decay, any one approach from below
#spatial.decay<-spT.decay(type="FIXED", value=0.01)
spatial.decay<-spT.decay(type="MH", tuning=0.08)
#spatial.decay<-spT.decay(type="DISCRETE",limit=c(0.01,0.02),segments=5)

# Iterations for the MCMC algorithms
nItr<-5000

# MCMC via Gibbs
set.seed(11)
post.gp <- spT.Gibbs(formula=o8hrmax ~ cMAXTMP+WDSP+RH,
         data=DataFit, model="GP", time.data=time.data,
         coords=coords, priors=priors, initials=initials,
         nItr=nItr, nBurn=0, report=nItr,
         tol.dist=2, distance.method="geodetic:km",
         cov.fnc="exponential", scale.transform="SQRT",
         spatial.decay=spatial.decay)
print(post.gp)

# Summary and plots
summary(post.gp)
summary(post.gp,pack="coda")
plot(post.gp)
plot(post.gp,residuals=TRUE)
```

```
coef(post.gp)
terms(post.gp)
formula(post.gp)
model.frame(post.gp)
model.matrix(post.gp)

# Model selection criteria
post.gp$PMCC

##
## Fit and spatially prediction simultaneously
##

# Read data
data(DataFit);
data(DataValPred)

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))

# MCMC via Gibbs will provide output in *.txt format
# from C routine to avoide large data problem in R
set.seed(11)
post.gp.fitpred <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="GP", coords=coords,
         newcoords=pred.coords, newdata=DataValPred,
         scale.transform="SQRT")
print(post.gp.fitpred)
summary(post.gp.fitpred)
coef(post.gp.fitpred)
plot(post.gp.fitpred)
names(post.gp.fitpred)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(post.gp.fitpred$prediction[,1]))

###########################
## The AR models:
###########################

##
## Model fitting
##

# Read data
data(DataFit);

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))

# MCMC via Gibbs using default choices
```

```
set.seed(11)
post.ar <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="AR", coords=coords,
         scale.transform="SQRT")
print(post.ar)

# MCMC via Gibbs not using default choices
# define the time-series
time.data<-spT.time(t.series=60,segment=1)

# hyper-parameters for the prior distributions
priors<-spT.priors(model="AR",var.prior=Gam(2,1),
        beta.prior=Nor(0,10^4))

# initial values for the model parameters
initials<-spT.initials(model="AR", sig2eps=0.01,
            sig2eta=0.5, beta=NULL, phi=0.001)

# Input for spatial decay
#spatial.decay<-spT.decay(type="FIXED", value=0.01)
spatial.decay<-spT.decay(type="MH", tuning=0.08)
#spatial.decay<-spT.decay(type="DISCRETE",limit=c(0.01,0.02),segments=5)

# Iterations for the MCMC algorithms
nItr<-5000

# MCMC via Gibbs
set.seed(11)
post.ar <- spT.Gibbs(formula=o8hrmax~cMAXTMP+WDSP+RH,
         data=DataFit, model="AR", time.data=time.data,
         coords=coords, priors=priors, initials=initials,
         nItr=nItr, nBurn=0, report=nItr,
         tol.dist=2, distance.method="geodetic:km",
         cov.fnc="exponential", scale.transform="SQRT",
         spatial.decay=spatial.decay)
print(post.ar)

# Summary and plots
summary(post.ar)
plot(post.ar)

# Model selection criteria
post.ar$PMCC

##
## Fit and spatially prediction simultaneously
##

# Read data
data(DataFit);
data(DataValPred)

# Define the coordinates
```

```
coords<-as.matrix(unique(cbind(DataFit[,2:3])))
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))

# MCMC via Gibbs will provide output in *.txt format
# from C routine to avoide large data problem in R
set.seed(11)
post.ar.fitpred <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="AR", coords=coords,
         newcoords=pred.coords, newdata=DataValPred,
         scale.transform="SQRT")
print(post.ar.fitpred)
summary(post.ar.fitpred)
names(post.ar.fitpred)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(post.ar.fitpred$prediction[,1]))

#################################
## The GPP approximation models:
#################################

##
## Model fitting
##

# Read data
data(DataFit);

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))
# Define knots
knots<-spT.grid.coords(Longitude=c(max(coords[,1]),
              min(coords[,1])),Latitude=c(max(coords[,2]),
              min(coords[,2])), by=c(4,4))

# MCMC via Gibbs using default choices
set.seed(11)
post.gpp <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="GPP", coords=coords,
         knots.coords=knots, scale.transform="SQRT")
print(post.gpp)

# MCMC via Gibbs not using default choices
# define the time-series
time.data<-spT.time(t.series=60,segment=1)

# hyper-parameters for the prior distributions
priors<-spT.priors(model="GPP",var.prior=Gam(2,1),
        beta.prior=Nor(0,10^4))

# initial values for the model parameters
initials<-spT.initials(model="GPP", sig2eps=0.01,
            sig2eta=0.5, beta=NULL, phi=0.001)
```

```
# input for spatial decay
#spatial.decay<-spT.decay(type="FIXED", value=0.001)
spatial.decay<-spT.decay(type="MH", tuning=0.05) #
#spatial.decay<-spT.decay(type="DISCRETE",limit=c(0.001,0.009),segments=10)

# Iterations for the MCMC algorithms
nItr<-5000

# MCMC via Gibbs
set.seed(11)
post.gpp <- spT.Gibbs(formula=o8hrmax~cMAXTMP+WDSP+RH,
         data=DataFit, model="GPP", time.data=time.data,
         coords=coords, knots.coords=knots,
         priors=priors, initials=initials,
         nItr=nItr, nBurn=0, report=nItr,
         tol.dist=2, distance.method="geodetic:km",
         cov.fnc="exponential", scale.transform="SQRT",
         spatial.decay=spatial.decay)
print(post.gpp)

# Summary and plots
summary(post.gpp)
plot(post.gpp)

# Model selection criteria
post.gpp$PMCC

##
## Fit and spatially prediction simultaneously
##

# Read data
data(DataFit);
data(DataValPred)

# Define the coordinates
coords<-as.matrix(unique(cbind(DataFit[,2:3])))
pred.coords<-as.matrix(unique(cbind(DataValPred[,2:3])))
knots<-spT.grid.coords(Longitude=c(max(coords[,1]),
              min(coords[,1])),Latitude=c(max(coords[,2]),
              min(coords[,2])), by=c(4,4))

# MCMC via Gibbs will provide output in *.txt format
# from C routine to avoide large data problem in R
set.seed(11)
post.gpp.fitpred <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="GP", coords=coords, knots.coords=knots,
         newcoords=pred.coords, newdata=DataValPred,
         scale.transform="SQRT")
print(post.gpp.fitpred)
summary(post.gpp.fitpred)
plot(post.gpp.fitpred)
```

```
names(post.gpp.fitpred)

# validation criteria
spT.validation(DataValPred$o8hrmax,c(post.gpp.fitpred$prediction[,1]))

##

## End(Not run)
```

---

spT.grid.coords *Grid Coordinates*

Description

This function is used to obtain Longitude/x and Latitude/y coordinates in a grid set.

Usage

```
spT.grid.coords(Longitude = c(max, min),
   Latitude = c(max, min), by = c(NA,NA))
```

Arguments

Longitude	The maximum and minimum longitude position.
Latitude	The maximum and minimum latitude position.
by	The number of x and y points in each axis.

See Also

spT.geodist.

Examples

```
## Not run:
##

# Load 29 ozone monitoring locations in New York.

data(NYsite)
NYsite
coords <- as.matrix(NYsite[,c(3,2)])

# Find the knots coordinates

knots.coords <- spT.grid.coords(Longitude=c(max(coords[,1]),
          min(coords[,1])), Latitude=c(max(coords[,2]),
          min(coords[,2])),by=c(4,4))
knots.coords
```

```
##

## End(Not run)
```

spT.initials *Initial values for the spatio-temporal models.*

Description

This command is useful to assign the initial values of the hyper-parameters of the prior distributions.

Usage

```
spT.initials(model, sig2eps=0.01, sig2eta=NULL,
   rho=NULL, beta=NULL, phi=NULL)
```

Arguments

model	The spatio-temporal models, current options are: "GP", "AR", and "GPP".
sig2eps	Initial value for the parameter σ^2_ϵ.
sig2eta	Initial value for the parameter σ^2_η.
rho	Initial value for the parameter ρ.
beta	Initial value for the parameter β.
phi	Initial value for the parameter ϕ.

Note

Initial values are automatically given if the user does not provide these.

See Also

spT.Gibbs, spT.priors.

Examples

```
## Not run:
##

initials<-spT.initials(model="GPP", sig2eps=0.01,
        sig2eta=0.5, beta=NULL, phi=0.001)
initials

##

## End(Not run)
```

`spT.keep.morethan.dist` *Present one coordinate in a defined area for presentation*

Description

This function is used to present one coordinate in a defined area to avoid cutter.

Usage

```
spT.keep.morethan.dist(coords, tol.dist=100)
```

Arguments

coords	X and Y axes/ longitude and latitude values.
tol.dist	The tolerance limit for the distance.

See Also

spT.geodist, dist, spT.data.selection.

Examples

```
## Not run:
##

data(NYdata)
head(NYdata)
NYsite<-unique(NYdata[,2:3])
head(NYsite)
spT.keep.morethan.dist(NYsite,tol.dist=100)

# Including values
dat<-cbind(NYsite,value=rnorm(dim(NYsite)[[1]]))
head(dat)
spT.keep.morethan.dist(dat,tol.dist=100)

##

## End(Not run)
```

spT.pCOVER *Nominal Coverage*

Description

This function is used to obtain nominal coverage.

Usage

```
spT.pCOVER(z=NULL,zup=NULL,zlow=NULL,zsample=NULL,level=95)
```

Arguments

z	The original values (matrix or vector).
zup	The predicted values for upper interval (matrix or vector).
zlow	The predicted values for lower interval (matrix or vector).
zsample	Predicted MCMC samples.
level	Level of coverages.

See Also

spT.validation.

Examples

```
## Not run:
##

# Create ‘x’: the true values.
# Create ‘yup’: the upper interval.
# Create ‘ylow’: the lower interval.

x <- rnorm(1000,5,0.1)
yup <- rnorm(1000,7,2)
ylow <- rnorm(1000,3,2)

# The pCOVER is:

spT.pCOVER(z=x, zup=yup, zlow=ylow)

# create predicted MCMC samples

y <- matrix(rnorm(1000*5000,5,1),1000,5000)

# The pCOVER is:

spT.pCOVER(z=x, zsamples=y)
spT.pCOVER(z=x, zsamples=y, level=50)
```

```
##

## End(Not run)
```

spT.priors *Priors for the spatio-temporal models.*

Description

This command is useful to assign the hyper-parameters of the prior distributions.

Usage

```
spT.priors(model,var.prior=Gam(a=2,b=1),
  beta.prior=Nor(0,10^10), rho.prior=Nor(0,10^10),
  phi.prior=Gam(a=2,b=1))
```

Arguments

`model`	The spatio-temporal models, current input: "GP", "AR", and "GPP".
`var.prior`	The hyper-parameter for the Gamma prior distribution (with mean = a/b) of the variance model parameters (e.g., $\sigma 2\_\epsilon$, $\sigma 2\_\eta$).
`beta.prior`	The hyper-parameter for the Normal prior distribution of the β model parameters.
`rho.prior`	The hyper-parameter for the Normal prior distribution of the ρ model parameter.
`phi.prior`	The hyper-parameter for the Gamma prior distribution (with mean = a/b) of the spatial decay (ϕ) parameter.

Note

If no prior information are given (assigned as NULL), then it use flat prior values of the corresponding distributions.
`Gam` and `Nor` refers to Gamma and Normal distributions respectively.

See Also

`spT.Gibbs`, `predict.spT`, `spT.initials`.

Examples

```
## Not run:
##

priors<-spT.priors(model="GPP",var.prior=Gam(2,1),
      beta.prior=Nor(0,10^4))
priors
```

```
##

## End(Not run)
```

spT.segment.plot *Utility plot for prediction/forecast*

Description

This function is used to obtain scatter plots with 95 percent CI for predictions/forecasts.

Usage

```
spT.segment.plot(obs, est, up, low, limit = NULL)
```

Arguments

obs	Observed values.
est	Estimated values.
up	Upper limit of the estimated values.
low	Lower limit of the estimated values.
limit	x-axis and y-axis limits.

See Also

summary.spT, plot.spT.

Examples

```
## Not run:
##

obs<-rnorm(10,15,1)
est<-rnorm(10,15,1.5)
up<-rnorm(10,25,0.5)
low<-rnorm(10,5,0.5)
spT.segment.plot(obs,est,up,low,limit=c(0,30))

##

## End(Not run)
```

spT.time *Timer series information.*

Description

This function defines the time series in the spatio-temporal data.

Usage

```
spT.time(t.series, segment=1)
```

Arguments

t.series	Number of times within each segment in each series. This should be a constant. Currently it is not possible to make it a variable? Like its 30 for April, and 31 for May etc.
segment	Number of segments in each time series. This should be a constant.

See Also

spT.Gibbs.

Examples

```
## Not run:
##

time.data<-spT.time(t.series=31,segment=2)

##

## End(Not run)
```

spT.validation *Validation Commands*

Description

The following function is used to validate the predicted observations with the actual values.

Usage

```
spT.validation(z, zhat)
```

Arguments

z	The original values (matrix or vector).
zhat	The predicted values (matrix or vector).

Value

MSE	Mean Squared Error.
RMSE	Root Mean Squared Error.
MAE	Mean Absolute Error.
MAPE	Mean Absolute Percentage Error.
BIAS	Bias.
rBIAS	Relative Bias.
rMSEP	Relative Mean Separation.

See Also

spT.pCOVER.

Examples

```
## Not run:
##

# Create 'x', which is the true values.
# Create 'y', which is the predicted values.

x <- rnorm(10,5,0.1)
y <- rnorm(10,5,1)
spT.validation(x, y)

##

## End(Not run)
```

---

summary.spT *Summary statistics of the parameters.*

Description

This function is used to obtain MCMC summary statistics.

Usage

```
## S3 method for class 'spT'
## S3 method for class 'spT'
summary(object, digits=4, package="spTimer", ...)

##
```

Arguments

`object`	Object of class inheriting from "spT".
`digits`	Rounds the specified number of decimal places (default 4).
`package`	If "coda" then summary statistics are given using coda package. Defaults value is "spTimer".
`...`	Other arguments.

Value

`sig2eps`	Summary statistics for σ_ϵ^2.
`sig2eta`	Summary statistics for σ_η^2.
`phi`	Summary statistics for spatial decay parameter ϕ, if estimated using `spT.decay`.
`...`	Summary statistics for other parameters used in the models.

See Also

`spT.Gibbs`.

Examples

```
## Not run:
##

summary(out) # where out is the output from spT class
summary(out, digits=2) # where out is the output from spT class
summary(out, pack="coda") # where out is the output from spT class

##

## End(Not run)
```

Index