Package ‘spTimer’

September 8, 2024

Type Package

Title Spatio-Temporal Bayesian Modelling

Version 3.3.3

Date 2024-09-05

Depends R (>= 4.4.0)

Description Fits, spatially predicts and temporally forecasts 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 for spatio-temporal big-n problems. Bakar and Sahu (2015) <doi:10.18637/jss.v063.i15>.

License GPL (>= 2)

Imports coda, sp, spacetime, extraDistr, grDevices, graphics, stats, utils

LazyData yes

NeedsCompilation yes

Author K. Shuvo Bakar [aut, cre] (<https://orcid.org/0000-0003-3215-4496>), Sujit K. Sahu [ctb] (<https://orcid.org/0000-0003-2315-3598>)

Maintainer K. Shuvo Bakar <shuvo.bakar@gmail.com>

Repository CRAN

Date/Publication 2024-09-08 09:10:02 UTC

Contents

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

Description

This package uses different hierarchical Bayesian spatio-temporal modelling strategies, namely:
(1) Gaussian processes (GP) models,
(2) Autoregressive (AR) models,
(3) Gaussian predictive processes (GPP) based autoregressive models for big-n problem.

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@gmail.com>

References

1. Bakar, K. S., & Sahu, S. K. (2015). sptimer: Spatio-temporal bayesian modelling using r. Journal of Statistical Software, 63(15), 1-32.
2. Sahu, S.K. & Bakar, K.S. (2012). Hierarchical Bayesian Autoregressive Models for Large Space Time Data with Applications to Ozone Concentration Modelling. Applied Stochastic Models in Business and Industry, 28, 395-415.
3. 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.
4. Bakar, K.S. (2012). Bayesian Analysis of Daily Maximum Ozone Levels. PhD Thesis, University of Southampton, Southampton, United Kingdom.

See Also

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

`confint.spT` *Credible intervals for model parameters.*

Description

This function is used to obtain credible intervals for model parameters from the MCMC samples.

Usage

```
## S3 method for class 'spT'
confint(object, parm, level = 0.95, ...)

##
```

Arguments

`object`	Object of class inheriting from "spT".
`parm`	a specification of which parameters are to be given credible intervals, a vector of names. If missing, all parameters are considered.
`level`	The required credible interval.
`...`	other arguments.

See Also

`spT.Gibbs`.

Examples

```
## Not run:
##

confint(out) # where out is the output from spT class

##

## End(Not run)
```

---

fitted.spT *Extract model fitted values.*

Description

Extract average fitted values and corresponding standard deviations from model.

Usage

```
## S3 method for class 'spT'
fitted(object, ...)

##
```

Arguments

object	Object of class inheriting from "spT".
...	Other arguments.

Value

Mean	Fitted mean values obtained from the MCMC samples.
SD	Corresponding standard deviations.

See Also

spT.Gibbs.

Examples

```
## Not run:
##

fitted(out) # where out is the output from spT class

##

## 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.

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

`NYgrid`, `spT.Gibbs`, `spT.subset`.

Examples

```
##
  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)

  # 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)

##
```

---

plot.spT *Plots for spTimer output.*

Description

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

Usage

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

##
```

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.
`coefficient`	Only applicable for package "spTDyn" (see details: https://cran.r-project.org/web/packages/spTDyn/index
`...`	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'
predict(object, newdata, newcoords, foreStep=NULL, type="spatial",
        nBurn, tol.dist, predAR=NULL, Summary=TRUE, ...)
```

Arguments

`object`	Object of class inheriting from "spT".
`newdata`	The data set providing 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`, see spT.Gibbs.
`foreStep`	Number of K-step (time points) ahead forecast, K=1,2, ...; Only applicable if type="temporal".
`type`	If the value is "spatial" then only spatial prediction will be performed at the `newcoords` which must be different from the fitted sites provided by `coords`. When the "temporal" option is specified then forecasting will be performed and in this case the `newcoords` may also contain elements of the fitted sites in which case only temporal forecasting beyond the last fitted time point will be performed.

`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.
`Summary`	To obtain summary statistics for the posterior predicted MCMC samples. Default is TRUE.
`...`	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 `spTimer` forecast output using package `forecast` through function `as.forecast.object`.

References

Bakar, K. S. and Sahu, S. K. (2014) spTimer: Spatio-Temporal Bayesian Modelling Using R. Technical Report, University of Southampton, UK. To appear in the Journal of Statistical Software.
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.

See Also

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

Examples

```
##

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

##
## Spatial prediction/interpolation
##

# Read data
data(NYdata)
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))

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

# 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
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))

# 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
#library(forecast)
#tmp<-as.forecast.object(fore.gp, site=1) # default for site 1
#plot(tmp)
#summary(tmp)

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

# Read data
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

# 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
#library(forecast)
#tmp<-as.forecast.object(fore.gp, site=5) # for site 5
#plot(tmp)

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

##
## Spatial prediction/interpolation
##
```

```
# Read data
data(NYdata)
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))

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

# 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
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))

# 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
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

# 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)

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

##
## Spatial prediction/interpolation
##

# Read data
data(NYdata)
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))

# 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=~Longitude+Latitude,
         knots.coords=knots, scale.transform="SQRT")
print(post.gpp)

# 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
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))

# 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
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
```

```
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

# 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)

##

######################################################
## The Truncated/Censored GP models:
######################################################

##
## Model fitting
##

data(NYdata)

# Truncation at 30 (say)
NYdata$o8hrmax[NYdata$o8hrmax<=30] <- 30

# Read data
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

#
nItr <- 5000 # number of MCMC samples for each model
nBurn <- 1000 # number of burn-in from the MCMC samples
# Truncation at 30
# fit truncated GP model
```

```
out <- spT.Gibbs(formula=o8hrmax~cMAXTMP+WDSP+RH,data=DataFit,
  model="truncatedGP",coords=~Longitude+Latitude,
  distance.method="geodetic:km",nItr=nItr,nBurn=nBurn,report=5,
  truncation.para = list(at = 30,lambda = 4),
  fitted.values="ORIGINAL")
#
summary(out)
head(fitted(out))
plot(out,density=FALSE)
#
head(cbind(DataFit$o8hrmax,fitted(out)[,1]))
plot(DataFit$o8hrmax,fitted(out)[,1])
spT.validation(DataFit$o8hrmax,fitted(out)[,1])

##
## prediction (spatial)
##

pred <- predict(out,newdata=DataValPred, newcoords=~Longitude+Latitude, tol=0.05)
names(pred)
plot(DataValPred$o8hrmax,c(pred$Mean))
spT.validation(DataValPred$o8hrmax,c(pred$Mean))
#pred$prob.below.threshold

##
## forecast (temporal)
##

# unobserved locations
fore <- predict(out,newdata=DataValFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataValFore$o8hrmax,c(fore$Mean))
plot(DataValFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold

# observed locations
fore <- predict(out,newdata=DataFitFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataFitFore$o8hrmax,c(fore$Mean))
plot(DataFitFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold


######################################################
## The Truncated/Censored GPP models:
######################################################

##
## Model fitting
##

data(NYdata)
```

```
# Define the coordinates
coords<-as.matrix(unique(cbind(NYdata[,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))

# Truncation at 30 (say)
NYdata$o8hrmax[NYdata$o8hrmax<=30] <- 30

# Read data
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

#
nItr <- 5000 # number of MCMC samples for each model
nBurn <- 1000 # number of burn-in from the MCMC samples
# Truncation at 30
# fit truncated GPP model
out <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="truncatedGPP",coords=~Longitude+Latitude,
         knots.coords=knots, distance.method="geodetic:km",
         nItr=nItr,nBurn=nBurn,report=5,fitted="ORIGINAL",
         truncation.para = list(at = 30,lambda = 4))
#
summary(out)
head(fitted(out))
plot(out,density=FALSE)
#
head(cbind(DataFit$o8hrmax,fitted(out)[,1]))
plot(DataFit$o8hrmax,fitted(out)[,1])
spT.validation(DataFit$o8hrmax,fitted(out)[,1])

##
## prediction (spatial)
##

pred <- predict(out,newdata=DataValPred, newcoords=~Longitude+Latitude, tol=0.05)
names(pred)
plot(DataValPred$o8hrmax,c(pred$Mean))
spT.validation(DataValPred$o8hrmax,c(pred$Mean))
#pred$prob.below.threshold

##
## forecast (temporal)
```

```
##

# unobserved locations
fore <- predict(out,newdata=DataValFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataValFore$o8hrmax,c(fore$Mean))
plot(DataValFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold

# observed locations
fore <- predict(out,newdata=DataFitFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataFitFore$o8hrmax,c(fore$Mean))
plot(DataFitFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold


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

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(distribution=Gamm(a=2,b=1), tuning=NULL, npoints=NULL, value=NULL)
```

Arguments

distribution	Prior distribution for ϕ. Currently available methods are, Gamm(a,b) and Unif(low,up). One can also used "FIXED" value for ϕ parameter.
tuning	If the Gamma prior distribution is used then we need to define the tuning parameter for sampling ϕ. The `tuning` is the standard deviation for the normal proposal distribution of the random-walk Metropolis algorithm used to sample ϕ on the log-scale.
npoints	If Unif distribution is used then need to define the number of segments for the range of limits by npoints. Default value is 5.
value	If distribution="FIXED" type is used then need to define the value for ϕ. The default value is 3/dmax where dmax is the maximum distance between the fitting sites provided by `coords`.

See Also

spT.Gibbs.

Examples

```
##

# input for random-walk Metropolis within Gibbs
# sampling for phi parameter
spatial.decay<-spT.decay(distribution=Gamm(2,1), tuning=0.08)

# input for discrete sampling of phi parameter
# with uniform prior distribution
spatial.decay<-spT.decay(distribution=Unif(0.01,0.02),npoints=5)

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

##
```

---

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

```
##

# 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)

##
```

---

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 = "GP", time.data = NULL,
coords, knots.coords, newcoords = NULL, newdata = NULL, priors = NULL,
initials = NULL, nItr = 5000, nBurn = 1000, report = 1, tol.dist = 0.05,
```

```
distance.method = "geodetic:km", cov.fnc = "exponential",
scale.transform = "NONE", spatial.decay = spT.decay(distribution = "FIXED"),
truncation.para = list(at = 0,lambda = 2), annual.aggrn = "NONE",
fitted.values="TRANSFORMED")
```

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" and "Longitude".
`model`	The spatio-temporal models to be fitted, current choices are: "GP", "truncatedGP", "AR", "GPP", and "truncatedGPP", with the first one as the default.
`time.data`	Defining the segments of the time-series set up using the function `spT.time`.
`coords`	The n by 2 matrix or data frame 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 a formula argument such as ~Longitude+Latitude.
`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 to be performed simultaneously. If omitted, no predictions will be performed.
`newdata`	The covariate values at the prediction sites specified by `newcoords`. 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 provided. 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 5000.
`nBurn`	Number of burn-in samples. This number of samples will be discarded before making any inference. Default value is 1000.
`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`. The default is "geodetic:km".

`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 prior distribution for the spatial decay parameter ϕ. Currently implemented options are "FIXED", "Unif", or "Gamm". Further details for each of these are specified by `spT.decay`.

`truncation.para` Provides truncation parameter λ and truncation point "at" using list.

`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.

`fitted.values` This option provides calculating fitted values and corresponding sd in the original scale. Currently implemented options are: "ORIGINAL" and "TRANSFORMED". Only applicable if `scale.transform` inputs "SQRT" or "LOG". Note that the PMCC (model validation criteria) values will be changed accordingly.

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.

`rhop` MCMC samples for ρ for the AR or GPP model.

`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. (2015) spTimer: Spatio-Temporal Bayesian Modelling Using R. Journal of Statistical Software, 63(15). 1–32.

Sahu, S. K. and Bakar, K. S. (2012a) A comparison of Bayesian Models for Daily Ozone Concentration Levels Statistical Methodology, 9, 144-157.

Sahu, S. K. and Bakar, K. S. (2012b) 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.

See Also

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

Examples

```
##

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

```
library(spTimer)

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

##
## Model fitting
##

# Read data
data(NYdata)

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

# MCMC via Gibbs not using default choices
# Read data
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))

# define the time-series
time.data<-spT.time(t.series=60,segment=1)

# hyper-parameters for the prior distributions
priors<-spT.priors(model="GP",inv.var.prior=Gamm(2,1),
        beta.prior=Norm(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(distribution="FIXED", value=0.01)
spatial.decay<-spT.decay(distribution=Gamm(2,1), tuning=0.08)
#spatial.decay<-spT.decay(distribution=Unif(0.01,0.02),npoints=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=~Longitude+Latitude, 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)
confint(post.gp)
terms(post.gp)
formula(post.gp)
model.frame(post.gp)
model.matrix(post.gp)

# Model selection criteria
post.gp$PMCC


######################################
## The GP model for sp class data
######################################

# Creating sp class data
library(sp)
data(meuse)
summary(meuse)
coordinates(meuse) <- ~x+y
class(meuse)
out<-spT.Gibbs(formula=zinc~sqrt(dist),data=meuse,
               model="GP", scale.transform="LOG")
summary(out)

# Create a dataset with spacetime class
library(spTimer)
site<-unique(NYdata[,c("Longitude","Latitude")])
library(spacetime)
row.names(site)<-paste("point",1:nrow(site),sep="")
site <- SpatialPoints(site)
ymd<-as.POSIXct(seq(as.Date("2006-07-01"),as.Date("2006-08-31"),by=1))
# introduce class STFDF
newNYdata<-STFDF(sp=site, time=ymd, data=NYdata) # full lattice
class(newNYdata)
out <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
      data=newNYdata, model="GP", scale.transform="SQRT")
summary(out)


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

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

##
## Model fitting
##

# Read data
data(NYdata)

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

# MCMC via Gibbs using default choices
set.seed(11)
post.ar <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=NYdata, 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=62,segment=1)

# hyper-parameters for the prior distributions
priors<-spT.priors(model="AR",inv.var.prior=Gamm(2,1),
        beta.prior=Norm(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(distribution="FIXED", value=0.01)
spatial.decay<-spT.decay(distribution=Gamm(2,1), tuning=0.08)
#spatial.decay<-spT.decay(distribution=Unif(0.01,0.02),npoints=5)

# Iterations for the MCMC algorithms
nItr<-5000

# MCMC via Gibbs
set.seed(11)
post.ar <- spT.Gibbs(formula=o8hrmax~cMAXTMP+WDSP+RH,
         data=NYdata, 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

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

##
## Model fitting
##

# Read data
data(NYdata);

# Define the coordinates
coords<-as.matrix(unique(cbind(NYdata[,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=NYdata, 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=62,segment=1)

# hyper-parameters for the prior distributions
priors<-spT.priors(model="GPP",inv.var.prior=Gamm(2,1),
        beta.prior=Norm(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(distribution="FIXED", value=0.001)
spatial.decay<-spT.decay(distribution=Gamm(2,1), tuning=0.05)
#spatial.decay<-spT.decay(distribution=Unif(0.001,0.009),npoints=10)

# Iterations for the MCMC algorithms
nItr<-5000

# MCMC via Gibbs
set.seed(11)
post.gpp <- spT.Gibbs(formula=o8hrmax~cMAXTMP+WDSP+RH,
```

```
         data=NYdata, 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


######################################################
## The Truncated/Censored GP models:
######################################################

##
## Model fitting
##

data(NYdata)

# Truncation at 30 (say)
NYdata$o8hrmax[NYdata$o8hrmax<=30] <- 30

# Read data
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

#
nItr <- 5000 # number of MCMC samples for each model
nBurn <- 1000 # number of burn-in from the MCMC samples
# Truncation at 30
# fit truncated GP model
out <- spT.Gibbs(formula=o8hrmax~cMAXTMP+WDSP+RH,data=DataFit,
  model="truncatedGP",coords=~Longitude+Latitude,
  distance.method="geodetic:km",nItr=nItr,nBurn=nBurn,report=5,
  truncation.para = list(at = 30,lambda = 2),
  fitted.values="ORIGINAL")
#
```

```
summary(out)
head(fitted(out))
plot(out,density=FALSE)
#
head(cbind(DataFit$o8hrmax,fitted(out)[,1]))
plot(DataFit$o8hrmax,fitted(out)[,1])
spT.validation(DataFit$o8hrmax,fitted(out)[,1])

##
## prediction (spatial)
##

pred <- predict(out,newdata=DataValPred, newcoords=~Longitude+Latitude, tol=0.05)
names(pred)
plot(DataValPred$o8hrmax,c(pred$Mean))
spT.validation(DataValPred$o8hrmax,c(pred$Mean))
#pred$prob.below.threshold

##
## forecast (temporal)
##

# unobserved locations
fore <- predict(out,newdata=DataValFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataValFore$o8hrmax,c(fore$Mean))
plot(DataValFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold

# observed locations
fore <- predict(out,newdata=DataFitFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataFitFore$o8hrmax,c(fore$Mean))
plot(DataFitFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold


######################################################
## The Truncated/Censored AR models:
######################################################

##
## Model fitting
##

data(NYdata)

# Truncation at 30 (say)
NYdata$o8hrmax[NYdata$o8hrmax<=30] <- 30

# Read data
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
```

```
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

#
nItr <- 5000 # number of MCMC samples for each model
nBurn <- 1000 # number of burn-in from the MCMC samples
# Truncation at 30
# fit truncated AR model
out <- spT.Gibbs(formula=o8hrmax~cMAXTMP+WDSP+RH,data=DataFit,
  model="truncatedAR",coords=~Longitude+Latitude,
  distance.method="geodetic:km",nItr=nItr,nBurn=nBurn,report=5,
  truncation.para = list(at = 30,lambda = 2),
  fitted.values="ORIGINAL")
#
summary(out)
head(fitted(out))
plot(out,density=FALSE)
#
head(cbind(DataFit$o8hrmax,fitted(out)[,1]))
plot(DataFit$o8hrmax,fitted(out)[,1])
spT.validation(DataFit$o8hrmax,fitted(out)[,1])

##
## prediction (spatial)
##

pred <- predict(out,newdata=DataValPred, newcoords=~Longitude+Latitude, tol=0.05)
names(pred)
plot(DataValPred$o8hrmax,c(pred$Mean))
spT.validation(DataValPred$o8hrmax,c(pred$Mean))
#pred$prob.below.threshold

##
## forecast (temporal)
##

# unobserved locations
fore <- predict(out,newdata=DataValFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataValFore$o8hrmax,c(fore$Mean))
plot(DataValFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold

# observed locations
fore <- predict(out,newdata=DataFitFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataFitFore$o8hrmax,c(fore$Mean))
```

```
plot(DataFitFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold


#####################################################
## The Truncated/Censored GPP models:
#####################################################

##
## Model fitting
##

data(NYdata)

# Define the coordinates
coords<-as.matrix(unique(cbind(NYdata[,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))

# Truncation at 30 (say)
NYdata$o8hrmax[NYdata$o8hrmax<=30] <- 30

# Read data
s<-c(8,11,12,14,18,21,24,28)
DataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
DataFit<-subset(DataFit, with(DataFit, !(Day %in% c(30, 31) & Month == 8)))
DataValPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
DataValPred<-subset(DataValPred, with(DataValPred, !(Day %in% c(30, 31) & Month == 8)))
DataValFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28))
DataValFore<-subset(DataValFore, with(DataValFore, (Day %in% c(30, 31) & Month == 8)))
DataFitFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(8,11,12,14,18,21,24,28),
reverse=TRUE)
DataFitFore<-subset(DataFitFore, with(DataFitFore, (Day %in% c(30, 31) & Month == 8)))

#
nItr <- 5000 # number of MCMC samples for each model
nBurn <- 1000 # number of burn-in from the MCMC samples
# Truncation at 30
# fit truncated GPP model
out <- spT.Gibbs(formula=o8hrmax ~cMAXTMP+WDSP+RH,
         data=DataFit, model="truncatedGPP",coords=~Longitude+Latitude,
         knots.coords=knots, distance.method="geodetic:km",
         nItr=nItr,nBurn=nBurn,report=5,fitted="ORIGINAL",
         truncation.para = list(at = 30,lambda = 2))
#
summary(out)
head(fitted(out))
plot(out,density=FALSE)
#
head(cbind(DataFit$o8hrmax,fitted(out)[,1]))
plot(DataFit$o8hrmax,fitted(out)[,1])
```

```
spT.validation(DataFit$o8hrmax,fitted(out)[,1])

##
## prediction (spatial)
##

pred <- predict(out,newdata=DataValPred, newcoords=~Longitude+Latitude, tol=0.05)
names(pred)
plot(DataValPred$o8hrmax,c(pred$Mean))
spT.validation(DataValPred$o8hrmax,c(pred$Mean))
#pred$prob.below.threshold

##
## forecast (temporal)
##

# unobserved locations
fore <- predict(out,newdata=DataValFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataValFore$o8hrmax,c(fore$Mean))
plot(DataValFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold

# observed locations
fore <- predict(out,newdata=DataFitFore, newcoords=~Longitude+Latitude,
   type="temporal", foreStep=2, tol=0.05)
spT.validation(DataFitFore$o8hrmax,c(fore$Mean))
plot(DataFitFore$o8hrmax,c(fore$Mean))
#fore$prob.below.threshold


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

---

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

```
##

# Load 29 ozone monitoring locations in New York.

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

# 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

##
```

`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, predict.spT, spT.priors.

Examples

```
##

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

##
```

---

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

```
##

# 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, zsample=y)
spT.pCOVER(z=x, zsample=y, level=50)

##
```

`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="GP", inv.var.prior=Gamm(a=2,b=1),
  beta.prior=Norm(0,10^10), rho.prior=Norm(0,10^10))
```

Arguments

`model`	The spatio-temporal models, current input: "GP", "AR", and "GPP".
`inv.var.prior`	The hyper-parameter for the Gamma prior distribution (with mean = a/b) of the precision (inverse variance) model parameters (e.g., $1/\sigma2\_\epsilon$, $1/\sigma2\_\eta$).
`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.

Note

If no prior information are given (assigned as NULL), then it use flat prior values of the corresponding distributions.
`Gamm` and `Norm` refers to Gamma and Normal distributions respectively.

See Also

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

Examples

```
##

priors<-spT.priors(model="GPP",inv.var.prior=Gamm(2,1),
      beta.prior=Norm(0,10^4))
priors

##
```

`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

```
##

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))

##
```

---

spT.subset *Select a subset of Spatial data.*

Description

This command selects a subset of the dataset using the site numbers.

Usage

```
spT.subset(data, var.name, s = NULL, reverse = FALSE)
```

Arguments

`data`	The dataset.
`var.name`	The name of the variable for which data will be sub-setted, e.g., "s.index".
`s`	The site numbers to be selected/deselected based on the argument `reverse`, e.g., c(2,8,12).
`reverse`	Logical value: if TRUE then `num.rs` will be discarded from the data.

See Also

`NYdata`.

Examples

```
##

# Load ozone concentration data for New York.
data(NYdata)
NYdata
# Choose sites 2, 8, and 12.
subdata<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(2,8,12))
# Do not choose purposively defined sites numbered as 2, 8, and 12.
subdata<-spT.subset(data=NYdata, var.name=c("s.index"), s=c(2,8,12), reverse=TRUE)

##
```

---

spT.time *Timer series information.*

Description

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

Usage

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

Arguments

`t.series`	Number of times within each segment in each series. It could be either a scalar or a vector. It should be a scalar if the segments are of equal length and should be a vector of length `segments` whose entries give the length of the segments.
`segments`	Number of segments in each time series. This should be a scalar.

See Also

`spT.Gibbs`.

Examples

```
##

# Equal length time-series in each of 3 years
time.data<-spT.time(t.series=365,segments=3)

# Un-equal length time-series in 5 years
time.data<-spT.time(t.series=c(366, 365, 365, 365, 366),segments=5)

##
```

---

spT.validation *Validation Commands*

Description

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

Usage

```
spT.validation(z, zhat, names=FALSE)
```

Arguments

`z`	The original values (matrix or vector).
`zhat`	The predicted values (matrix or vector).
`names`	Logical, if TRUE then print the names of the validation statistics.

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`, `spT.validation2`.

Examples

```
##

# 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)

##
```

`spT.validation2` *Validation Commands*

Description

The following function is used to validate the predicted observations with the actual values based on some threshold.

Usage

```
spT.validation2(z,zhat,cutoff,names=FALSE)
```

Arguments

z	The original values (matrix or vector).
zhat	The predicted values (matrix or vector).
cutoff	The threshold value or cut-off point.
names	Logical, if TRUE then print the names of the validation statistics.

Value

TPR	True Positive Rate, Sensitivity, Hit rate, Recall
FPR	False Positive Rate, False alarm
FNR	False Negative Rate, Miss rate
TNR	True Negative Rate, Specificity
Prevalence	Prevalence
Accuracy	Accuracy
Precision	Precision, Positive Predictive Value
FOR	False Ommission Rate
LRp	Positive Likelihood Ratio
LRn	Negative Likelihood Ratio
FDR	False Discovery Rate
NPV	Negative Predictive Value
DOR	Diagnostic Odds Ratio
F1score	F1 score
Heidke.Skill	Heidke Skill

See Also

spT.pCOVER, spT.validation.

Examples

```
##

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

x <- rnorm(100,0,0.1)
y <- rnorm(100,0,1)
spT.validation2(x, y, cutoff=0,names=TRUE)

##
```

summary.spT — *Summary statistics of the parameters.*

Description

This function is used to obtain MCMC summary statistics.

Usage

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

##
```

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". One can also use "spTDyn" for obtaining spatially varying and temporal dynamic models (see details: https://cran.r-project.org/web/packages/spTDyn/index.html)
`coefficient`	Only applicable for package "spTDyn".
`...`	Other arguments.

Value

`sig2eps`	Summary statistics for σ^2_ϵ.
`sig2eta`	Summary statistics for σ^2_η.
`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, digit=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