Package 'spTDyn'

September 9, 2024

Type Package

Title Spatially Varying and Spatio-Temporal Dynamic Linear Models

Version 2.0.3

Date 2024-09-09

Depends R (>= 4.4.0), spTimer

Description Fits, spatially predicts, and temporally forecasts space-time data using Gaussian Process (GP): (1) spatially varying coefficient process models and (2) spatio-temporal dynamic linear models. Bakar et al., (2016). Bakar et al., (2015).

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

License GPL (>= 2)

LazyData yes

NeedsCompilation yes

Author K. Shuvo Bakar [aut, cre] (<https://orcid.org/0000-0003-3215-4496>),
Philip Kokic [ctb],
Huidong Jin [ctb] (<https://orcid.org/0000-0002-3925-0256>)

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

Repository CRAN

Date/Publication 2024-09-09 19:00:05 UTC

Contents

---

spTDyn-package *Spatially varying and spatio-temporal dynamic linear models*

Description

This package uses different hierarchical Bayesian spatio-temporal modelling strategies, namely:
(1) Spatially varying coefficient process models,
(2) Temporally varying coefficient process models, also known as the spatio-temporal dynamic linear models.

Details

Package:	spTDyn
Type:	Package

The back-end code of this package is built under c language.
Main functions used:
> `GibbsDyn`
> `predict.spTD`

Author(s)

K.S. Bakar
Maintainer: K.S. Bakar <shuvo.bakar@gmail.com>

References

Bakar, K. S., Kokic, P. and Jin, H. (2015). A spatio-dynamic model for assessing frost risk in south-eastern Australia. Journal of the Royal Statistical Society, Series C. DOI: 10.1111/rssc.12103
Bakar, K. S., Kokic, P. and Jin, H. (2015). Hierarchical spatially varying coefficient and temporal dynamic process models using spTDyn. Journal of Statistical Computation and Simulation. DOI:10.1080/00949655.2015.1038267

See Also

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

decay *Choice for sampling spatial decay parameter ϕ.*

Description

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

Usage

```
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

GibbsDyn.

Examples

```
##

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

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

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

##
```

def.time *Timer series information.*

Description

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

Usage

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

Arguments

t.series	Number of times within each segment in each series. Can take only regular time-series.
segments	Number of segments in each time series. This should be a constant.

See Also

GibbsDyn.

Examples

```
##

# regular time-series in each year
time.data<-def.time(t.series=30,segments=2)

##
```

GibbsDyn *MCMC sampling for the models.*

Description

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

Usage

```
GibbsDyn(formula, data=parent.frame(), model="GP", time.data=NULL, coords,
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=decay(distribution="FIXED"),truncation.para=list(at=0,lambda=2))
```

Arguments

formula	The symnbolic description of the model equation of the regression part of the space-time model. The terms sp and tp are used to define spatially and temporally varying parameters for the 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", and "truncated", with the first one as the default.
time.data	Defining the segments of the time-series set up using the function `def.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.
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 `priors`.
initials	The preferred initial values for the parameters. If omitted, default values are provided automatically. Further details are provided in `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 `decay`.

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

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.

sig2betap
: MCMC samples for the parameter σ_β^2, only applicable for spatially varying coefficient process model.

sig2deltap
: MCMC samples for the parameter σ_δ^2, for β_j, $j = 1, ..., u$. Only applicable for spatio-temporal DLM.

sig2op
: MCMC samples for the parameter σ_o^2, for initial variance of β_0. Only applicable for spatio-dynamic and spatio-temporal DLM.

betap
: MCMC samples for the parameter β.

rhop
: MCMC samples for ρ.

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.

References

Bakar, K. S., Kokic, P. and Jin, H. (2015). A spatio-dynamic model for assessing frost risk in south-eastern Australia. Journal of the Royal Statistical Society, Series C. Bakar, K. S., Kokic, P. and Jin, H. (2015). Hierarchical spatially varying coefficient and temporal dynamic process models using spTDyn. Journal of Statistical Computation and Simulation.

See Also

priors, initials, dist, sp, tp.

Examples

```
##

###########################
## Attach library spTDyn
###########################

library(spTDyn)

## Read Aus data ##
data(AUSdata)
# set a side data for validation
library(spTimer)
s<-c(1,4,10)
AUSdataFit<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s, reverse=TRUE)
AUSdataFit<-subset(AUSdataFit, with(AUSdataFit, !(year == 2009)))
AUSdataPred<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s)
AUSdataPred<-subset(AUSdataPred, with(AUSdataPred, !(year == 2009)))
AUSdataFore<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s)
AUSdataFore<-subset(AUSdataFore, with(AUSdataFore, (year == 2009)))

## Read NY data ##
data(NYdata)
# set a side data for validation
s<-c(5,8,10,15,20,22,24,26)
fday<-c(25:31)
NYdataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
NYdataFit<-subset(NYdataFit, with(NYdataFit, !(Day %in% fday & Month == 8)))
NYdataPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
NYdataPred<-subset(NYdataPred, with(NYdataPred, !(Day %in% fday & Month == 8)))
```

```
NYdataFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
NYdataFore<-subset(NYdataFore, with(NYdataFore, (Day %in% fday & Month == 8)))

## Code for analysing temperature data in Section: 4 ##
## Model: Spatially varying coefficient process models ##

nItr<-13000
nBurn<-3000

# MCMC via Gibbs using defaults
# Spatially varying coefficient process model

library("spTDyn", warn.conflicts = FALSE)
set.seed(11)
post.sp <- GibbsDyn(tmax ~ soi+sp(soi)+grid+sp(grid),
           data=AUSdataFit, nItr=nItr, nBurn=nBurn, coords=~lon+lat,
           spatial.decay=decay(distribution=Gamm(2,1),tuning=0.06))
print(post.sp)

## Table: 3, Section: 4.1 ##
post.sp$PMCC

# parameter summary
summary(post.sp) # without spatially varying coefficients
summary(post.sp, coefficient="spatial")

#plot(post.sp, density=FALSE)  # without spatially varying coefficients
#plot(post.sp, coefficient="spatial", density=FALSE)

## Code for Figures: 3(a), 3(b) Section: 4.1 ##
Figure_3a<-function(){
  boxplot(t(post.sp$betasp[1:9,]),pch=".",main="SOI",
          xlab="Sites",ylab="Values")
}
Figure_3b<-function(){
  boxplot(t(post.sp$betasp[10:18,]),pch=".",main="Grid",
          xlab="Sites",ylab="Values")
}
Figure_3a()
Figure_3b()

## spatial prediction
set.seed(11)
pred.sp <- predict(post.sp,newcoords=~lon+lat,newdata=AUSdataPred)

## Table: 4, Section: 4.1, validations ##
spT.validation(AUSdataPred$tmax,c(pred.sp$Mean))
plot(AUSdataPred$tmax,c(pred.sp$Mean))

## temporal prediction
set.seed(11)
pred.sp.f <- predict(post.sp,type="temporal",foreStep=12,
                     newcoords=~lon+lat, newdata=AUSdataFore)
```

```
## Table: 4, Section: 4.1, validations ##
spT.validation(AUSdataFore$tmax,c(pred.sp.f$Mean))
plot(AUSdataFore$tmax,c(pred.sp.f$Mean))

## Code for analysing Ozone data in Section: 4 ##
## Model: spatio-temporal DLM ##

# MCMC via Gibbs using defaults
# spatio-temporal DLM

library("spTDyn", warn.conflicts = FALSE)
set.seed(11)
post.tp <- GibbsDyn(o8hrmax ~ tp(cMAXTMP)-1, data=NYdataFit,
           nItr=nItr, nBurn=nBurn, coords=~Longitude+Latitude,
           initials=initials(rhotp=0), scale.transform="SQRT",
           spatial.decay=decay(distribution=Gamm(2,1),tuning=0.05))
print(post.tp)
summary(post.tp)

## Table: 5, Section: 4.2 ##
post.tp$PMCC

## Figure: 5, Section: 4.2 ##
Figure_5<-function(){
  stat<-apply(post.tp$betatp[1:55,],1,quantile,prob=c(0.025,0.5,0.975))
  plot(stat[2,],type="p",lty=3,col=1,ylim=c(min(c(stat)),max(c(stat))),
       pch=19,ylab="",xlab="Days",axes=FALSE,main="cMAXTMP",cex=0.8)
  for(i in 1:55){
    segments(i, stat[2,i], i, stat[3,i])
    segments(i, stat[2,i], i, stat[1,i])
  }
  axis(1,1:55,labels=1:55);axis(2)
  abline(v=31.5,lty=2)
  text(15,0.32,"July");  text(45,0.32,"August");
}
Figure_5()

## spatial prediction
set.seed(11)
pred.tp <- predict(post.tp, newdata=NYdataPred, newcoords=~Longitude+Latitude)

## Table 6, Section: 4.2, validation ##
spT.validation(NYdataPred$o8hrmax,c(pred.tp$Mean))

## temporal prediction
set.seed(11)
pred.tp.f <- predict(post.tp, newdata=NYdataFore, newcoords=~Longitude+Latitude,
                     type="temporal", foreStep=7)

## Table 6, Section: 4.2, validation ##
spT.validation(NYdataFore$o8hrmax,c(pred.tp.f$Mean))
```

```
######################################################
## The Truncated/Censored models:
######################################################

## Read Aus data ##
data(AUSdata)
# set the truncation point at tmax=30
AUSdata$tmax <- replace(AUSdata$tmax, AUSdata$tmax<=30, 30)

# set a side data for validation
library(spTimer)
s<-c(1,4,10)
AUSdataFit<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s, reverse=TRUE)
AUSdataFit<-subset(AUSdataFit, with(AUSdataFit, !(year == 2009)))
AUSdataPred<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s)
AUSdataPred<-subset(AUSdataPred, with(AUSdataPred, !(year == 2009)))
AUSdataFore<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s)
AUSdataFore<-subset(AUSdataFore, with(AUSdataFore, (year == 2009)))

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

## The Truncated/Censored spatially varying models:
library("spTDyn", warn.conflicts = FALSE)
set.seed(11)
out <- GibbsDyn(tmax ~ soi+sp(soi)+grid+sp(grid),model="truncated",
           data=AUSdataFit, nItr=nItr, nBurn=nBurn, coords=~lon+lat,
           spatial.decay=decay(distribution=Gamm(2,1),tuning=0.06),
           truncation.para = list(at = 30,lambda = 2))
print(out)
summary(out)
head(fitted(out))
plot(out,density=FALSE)
#
head(cbind(AUSdataFit$tmax,fitted(out)[,1]))
plot(AUSdataFit$tmax,fitted(out)[,1])
spT.validation(AUSdataFit$tmax,fitted(out)[,1])

## spatial prediction
set.seed(11)
pred.sp <- predict(out,newcoords=~lon+lat,newdata=AUSdataPred)
spT.validation(AUSdataPred$tmax,c(pred.sp$Mean))
plot(AUSdataPred$tmax,c(pred.sp$Mean))

## temporal prediction
set.seed(11)
pred.sp.f <- predict(out,type="temporal",foreStep=12,
                     newcoords=~lon+lat, newdata=AUSdataFore)
spT.validation(AUSdataFore$tmax,c(pred.sp.f$Mean))
plot(AUSdataFore$tmax,c(pred.sp.f$Mean))
```

```
## The Truncated/Censored temporal dynamic DLM models:
library("spTDyn", warn.conflicts = FALSE)
set.seed(11)
out <- GibbsDyn(tmax ~ soi+tp(soi)+grid,model="truncated",
           data=AUSdataFit, nItr=nItr, nBurn=nBurn, coords=~lon+lat,
           spatial.decay=decay(distribution=Gamm(2,1),tuning=0.06),
           truncation.para = list(at = 30,lambda = 2))
print(out)
summary(out)
head(fitted(out))
plot(out,density=FALSE)
#
head(cbind(AUSdataFit$tmax,fitted(out)[,1]))
plot(AUSdataFit$tmax,fitted(out)[,1])
spT.validation(AUSdataFit$tmax,fitted(out)[,1])

## spatial prediction
set.seed(11)
pred.tp <- predict(out,newcoords=~lon+lat,newdata=AUSdataPred)
spT.validation(AUSdataPred$tmax,c(pred.tp$Mean))
plot(AUSdataPred$tmax,c(pred.tp$Mean))

## temporal prediction
set.seed(11)
pred.tp.f <- predict(out,type="temporal",foreStep=12,
                     newcoords=~lon+lat, newdata=AUSdataFore)
spT.validation(AUSdataFore$tmax,c(pred.tp.f$Mean))
plot(AUSdataFore$tmax,c(pred.tp.f$Mean))


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

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

```
initials(sig2eps=0.01, sig2eta=NULL, sig2beta=NULL, sig2delta=NULL,
   rhotp=NULL, rho=NULL, beta=NULL, phi=NULL)
```

Arguments

sig2eps	Initial value for the parameter σ^2_ϵ.
sig2eta	Initial value for the parameter σ^2_η.
sig2beta	Initial value for the parameter σ^2_β for spatially varying model.
sig2delta	Initial value for the parameter σ^2_δ for dynamic state-space model.
rhotp	Value for the parameter ρ for dynamic state-space model. For `rhotp=1`, ρ parameters are not sampled and fixed at value 1. For `rhotp=0`, ρ parameters are sampled from the full conditional distribution via MCMC with initial value 0.
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

GibbsDyn, priors.

Examples

```
##

initials<-initials(sig2eps=0.01, sig2eta=0.5, beta=NULL, phi=0.001)
initials

##
```

ObsGridLoc *Combining observation and nearest grid locations and data.*

Description

These commands combine observation and nearest grid locations, data.

Usage

```
ObsGridLoc(obsLoc, gridLoc, distance.method="geodetic:km", plot=FALSE)
gridTodata(gridData, gridLoc=NULL, gridLon=NULL, gridLat=NULL)
ObsGridData(obsData, gridData, obsLoc, gridLoc, distance.method="geodetic:km")
```

Arguments

obsLoc	The observed/measurement locations, first column is longitude/easting/x-axis and second column is latitude/northing/y-axis.
gridLoc	Grid locations, first column is longitude/easting/x-axis and second column is latitude/northing/y-axis.
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.
plot	Logical argument, if TRUE then plot observed and nearest grid locations.
gridData	Gridded data, should be in array form with dimenstions as longitude/x-axis, latitude/y-axis, day/time1, year/time2.
gridLon	Longitude/easting/x-axis of grid locations.
gridLat	Latitude/northing/y-axis of grid locations.
obsData	Observation data in data frame.

Examples

```
##

library(spTimer)
data(NYdata)
data(NYgrid)

obsLoc<-unique(cbind(NYdata$Longitude,NYdata$Latitude))
gridLoc<-unique(cbind(NYgrid$Longitude,NYgrid$Latitude))

# find closest observed and grid locations
dat<-ObsGridLoc(obsLoc, gridLoc)
head(dat)
# with plots
dat<-ObsGridLoc(obsLoc, gridLoc, plot=TRUE)
head(dat)

# convert array gridData to spTimer data format
gridData<-array(1:(10*10*31*2),dim=c(10,10,31,2)) # lon, lat, day, year
dat<-gridTodata(gridData, gridLoc)
head(dat)

# combine observed and grid data and locations
obsData<-NYdata
gridData<-array(1:(10*10*31*2),dim=c(10,10,31,2)) # lon, lat, day, year
dat<-ObsGridData(obsData, gridData, obsLoc, gridLoc)
head(dat)

# combine observed and more than one grid datasets
obsData<-NYdata
gridData1<-array(1:(10*10*31*2),dim=c(10,10,31,2)) # lon, lat, day, year
gridData2<-array(((10*10*31*2)+1):(2*(10*10*31*2)),dim=c(10,10,31,2)) # lon, lat, day, year
```

```
gridLoc1<-unique(cbind(NYgrid$Longitude,NYgrid$Latitude))
gridLoc2<-unique(cbind(NYgrid$Longitude,NYgrid$Latitude))
dat<-ObsGridData(obsData, gridData=list(gridData1,gridData2),
        obsLoc, gridLoc=list(gridLoc1, gridLoc2))
head(dat)

##
```

---

plot.spTD *Plots for spTDyn output.*

Description

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

Usage

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

##
```

Arguments

x	Object of class inheriting from "spTD".
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	Takes values: "spatial", "temporal" and "rho" for summary statistics of spatial, temporal and rho coefficients respectively. If NULL then provides parameter plots without spatial and temporal coefficients.
...	Other arguments.

See Also

GibbsDyn.

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
plot(out, coefficient="spatial") # for spatially varying coefficients

##

## End(Not run)
```

predict.spTD *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 'spTD'
predict(object, newdata, newcoords, foreStep=NULL, type="spatial",
        nBurn, tol.dist, Summary=TRUE, ...)
```

Arguments

`object`	Object of class inheriting from "spTD".
`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.
`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.

References

Bakar, K. S., Kokic, P. and Jin, H. (2015). A spatio-dynamic model for assessing frost risk in south-eastern Australia. Journal of the Royal Statistical Society, Series C. Bakar, K. S., Kokic, P. and Jin, H. (2015). Hierarchical spatially varying coefficient and temporal dynamic process models using spTDyn. Journal of Statistical Computation and Simulation.

See Also

GibbsDyn.

Examples

```
##

library(spTDyn)

## Read Aus data ##
data(AUSdata)
# set a side data for validation
s<-c(1,4,10)
AUSdataFit<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s, reverse=TRUE)
AUSdataFit<-subset(AUSdataFit, with(AUSdataFit, !(year == 2009)))
AUSdataPred<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s)
AUSdataPred<-subset(AUSdataPred, with(AUSdataPred, !(year == 2009)))
AUSdataFore<-spT.subset(data=AUSdata, var.name=c("s.index"), s=s)
AUSdataFore<-subset(AUSdataFore, with(AUSdataFore, (year == 2009)))

## Read NY data ##
data(NYdata)
# set a side data for validation
s<-c(5,8,10,15,20,22,24,26)
fday<-c(25:31)
NYdataFit<-spT.subset(data=NYdata, var.name=c("s.index"), s=s, reverse=TRUE)
NYdataFit<-subset(NYdataFit, with(NYdataFit, !(Day %in% fday & Month == 8)))
NYdataPred<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
NYdataPred<-subset(NYdataPred, with(NYdataPred, !(Day %in% fday & Month == 8)))
```

```
NYdataFore<-spT.subset(data=NYdata, var.name=c("s.index"), s=s)
NYdataFore<-subset(NYdataFore, with(NYdataFore, (Day %in% fday & Month == 8)))

## Code for analysing temperature data in Section: 4 ##
## Model: Spatially varying coefficient process models ##

nItr<-13000
nBurn<-3000

# MCMC via Gibbs using defaults
# Spatially varying coefficient process model

library("spTDyn", warn.conflicts = FALSE)
set.seed(11)
post.sp <- GibbsDyn(tmax ~ soi+sp(soi)+grid+sp(grid),
           data=AUSdataFit, nItr=nItr, nBurn=nBurn, coords=~lon+lat,
           spatial.decay=decay(distribution=Gamm(2,1),tuning=0.06))
print(post.sp)

## Table: 3, Section: 4.1 ##
post.sp$PMCC

# parameter summary
summary(post.sp) # without spatially varying coefficients
summary(post.sp, coefficient="spatial")

#plot(post.sp, density=FALSE)  # without spatially varying coefficients
#plot(post.sp, coefficient="spatial", density=FALSE)

## Code for Figures: 3(a), 3(b) Section: 4.1 ##
Figure_3a<-function(){
  boxplot(t(post.sp$betasp[1:9,]),pch=".",main="SOI",
          xlab="Sites",ylab="Values")
}
Figure_3b<-function(){
  boxplot(t(post.sp$betasp[10:18,]),pch=".",main="Grid",
          xlab="Sites",ylab="Values")
}
Figure_3a()
Figure_3b()

## spatial prediction
set.seed(11)
pred.sp <- predict(post.sp,newcoords=~lon+lat,newdata=AUSdataPred)

## Table: 4, Section: 4.1, validations ##
spT.validation(AUSdataPred$tmax,c(pred.sp$Mean))
plot(AUSdataPred$tmax,c(pred.sp$Mean))

## temporal prediction
set.seed(11)
pred.sp.f <- predict(post.sp,type="temporal",foreStep=12,
                     newcoords=~lon+lat, newdata=AUSdataFore)
```

```
## Table: 4, Section: 4.1, validations ##
spT.validation(AUSdataFore$tmax,c(pred.sp.f$Mean))
plot(AUSdataFore$tmax,c(pred.sp.f$Mean))

## Code for analysing Ozone data in Section: 4 ##
## Model: spatio-temporal DLM ##

# MCMC via Gibbs using defaults
# spatio-temporal DLM

library("spTDyn", warn.conflicts = FALSE)
set.seed(11)
post.tp <- GibbsDyn(o8hrmax ~ tp(cMAXTMP)-1, data=NYdataFit,
           nItr=nItr, nBurn=nBurn, coords=~Longitude+Latitude,
           initials=initials(rhotp=0), scale.transform="SQRT",
           spatial.decay=decay(distribution=Gamm(2,1),tuning=0.05))
print(post.tp)
summary(post.tp)

## Table: 5, Section: 4.2 ##
post.tp$PMCC

## Figure: 5, Section: 4.2 ##
Figure_5<-function(){
  stat<-apply(post.tp$betatp[1:55,],1,quantile,prob=c(0.025,0.5,0.975))
  plot(stat[2,],type="p",lty=3,col=1,ylim=c(min(c(stat)),max(c(stat))),
       pch=19,ylab="",xlab="Days",axes=FALSE,main="cMAXTMP",cex=0.8)
  for(i in 1:55){
    segments(i, stat[2,i], i, stat[3,i])
    segments(i, stat[2,i], i, stat[1,i])
  }
  axis(1,1:55,labels=1:55);axis(2)
  abline(v=31.5,lty=2)
  text(15,0.32,"July");  text(45,0.32,"August");
}
Figure_5()

## spatial prediction
set.seed(11)
pred.tp <- predict(post.tp, newdata=NYdataPred, newcoords=~Longitude+Latitude)

## Table 6, Section: 4.2, validation ##
spT.validation(NYdataPred$o8hrmax,c(pred.tp$Mean))

## temporal prediction
set.seed(11)
pred.tp.f <- predict(post.tp, newdata=NYdataFore, newcoords=~Longitude+Latitude,
                     type="temporal", foreStep=7)

## Table 6, Section: 4.2, validation ##
spT.validation(NYdataFore$o8hrmax,c(pred.tp.f$Mean))
```

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

priors *Priors for the spatio-temporal models.*

Description

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

Usage

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

Arguments

`inv.var.prior`	The hyper-parameter for the Gamma prior distribution (with mean = a/b) of the precision (inverse variance) model parameters (e.g., $1/\sigma 2\_\epsilon$, $1/\sigma 2\_\eta$).
`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.

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

`GibbsDyn`, `initials`.

Examples

```
##
library(spTimer)
priors<-priors(inv.var.prior=Gamm(2,1), beta.prior=Norm(0,10^4))
priors

##
```

sp *Defining spatially varying coefficients in the formula*

Description

This function is used to define spatially varying coefficients within the formula for the Gaussian process spatio-dynamic and spatially varying coefficient process models.

Usage

```
sp(x)
```

Arguments

x	The variable/covariate for which spatially varying coefficient is defined.

See Also

GibbsDyn, tp

Examples

```
##

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

library(spTDyn)

###########################
## The GP 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.gp <- GibbsDyn(formula=o8hrmax ~cMAXTMP+WDSP+sp(RH),
         data=NYdata, coords=coords, scale.transform="SQRT")
print(post.gp)
```

summary.spTD *Summary statistics of the parameters.*

Description

This function is used to obtain MCMC summary statistics.

Usage

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

##
```

Arguments

object	Object of class inheriting from "spTD".
digits	Rounds the specified number of decimal places (default 4).
package	If "coda" then summary statistics are given using coda package. Defaults value is "spTDyn".
coefficient	Takes values: "spatial", "temporal" and "rho" for summary statistics of spatial, temporal and rho coefficients respectively. If NULL then provides parameter summary without spatial and temporal coefficients.
...	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 decay.
...	Summary statistics for other parameters used in the models.

See Also

GibbsDyn.

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
summary(out, coefficient="spatial") # for spatially varying coefficients
summary(out, coefficient="temporal") # for temporally varying coefficients

##

## End(Not run)
```

---

tp — *Defining dynamic time-series coefficients in the formula*

Description

This function is used to define dynamic time-series coefficients within the formula for the Gaussian process spatio-dynamic and spatio-temporal DLM.

Usage

```
tp(x)
```

Arguments

x	The variable/covariate for which time varying coefficient is defined.

See Also

GibbsDyn, sp

Examples

```

##

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

library(spTDyn)

###########################
## The GP 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.gp <- GibbsDyn(formula=o8hrmax ~cMAXTMP+WDSP+tp(RH),
         data=NYdata, coords=coords, scale.transform="SQRT")
print(post.gp)

##
```

Index