Package 'CircSpaceTime'

June 6, 2019

Type Package

Title Spatial and Spatio-Temporal Bayesian Model for Circular Data

Version 0.9.0

BugReports https://github.com/santoroma/CircSpaceTime

Description Implementation of Bayesian models for spatial and spatio-temporal interpolation of circular data using Gaussian Wrapped and Gaussian Projected distributions. We developed the methods described in Jona Lasinio G. et al. (2012) <doi: 10.1214/12-aoas576>, Wang F. et al. (2014) <doi: 10.1080/01621459.2014.934454> and Mastrantonio G. et al. (2016) <doi: 10.1007/s11749-015-0458-y>.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/santoroma/CircSpaceTime

Suggests foreach, iterators, parallel, doParallel, gridExtra

LinkingTo Rcpp, RcppArmadillo, RInside

Imports Rcpp (>= 0.12.14), circular, RInside, coda, ggplot2

RoxygenNote 6.1.0

NeedsCompilation yes

Author Giovanna Jona Lasinio [aut] (<https://orcid.org/0000-0001-8912-5018>), Gianluca Mastrantonio [aut] (<https://orcid.org/0000-0002-2963-6729>), Mario Santoro [aut, cre] (<https://orcid.org/0000-0001-6626-9430>)

Maintainer Mario Santoro <santoro.ma@gmail.com>

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2019-06-06 15:12:15 UTC

R topics documented:

`APEcirc` *Average Prediction Error for circular Variables.*

Description

`APEcirc` computes the average prediction error (APE), defined as the average circular distance across pairs

Usage

```
APEcirc(real, sim, bycol = F)
```

Arguments

`real`	a vector of the values of the process at the test locations
`sim`	a matrix with `nrow` = the test locations and `ncol` = the number of posterior samples from the posterior distributions by `WrapKrigSp` `WrapKrigSpTi`, `ProjKrigSp`, `ProjKrigSpTi`
`bycol`	logical. It is TRUE if the columns of sim represent the observations and the rows the posterior samples, the default value is FALSE.

Value

a list of two elements

`ApePoints` a vector of APE, one element for each test point

`Ape` the overall mean

References

G. Jona Lasinio, A. Gelfand, M. Jona-Lasinio, "Spatial analysis of wave direction data using wrapped Gaussian processes", The Annals of Applied Statistics 6 (2013), 1478-1498

See Also

ProjKrigSp and WrapKrigSp for posterior spatial estimations, ProjKrigSpTi and WrapKrigSpTi for posterior spatio-temporal estimations

Other model performance indices: CRPScirc

Examples

```
library(CircSpaceTime)
## functions
rmnorm <- function(n = 1, mean = rep(0, d), varcov){
  d <- if (is.matrix(varcov))
    ncol(varcov)
  else 1
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}

#####################################
## Simulation                      ##
#####################################
set.seed(1)
n <- 20
### simulate coordinates from a unifrom distribution
coords  <- cbind(runif(n,0,100), runif(n,0,100)) #spatial coordinates
coordsT <- sort(runif(n,0,100)) #time coordinates (ordered)
Dist <- as.matrix(dist(coords))
DistT <- as.matrix(dist(coordsT))

rho     <- 0.05 #spatial decay
rhoT    <- 0.01 #temporal decay
sep_par <- 0.5 #separability parameter
sigma2  <- 0.3 # variance of the process
alpha   <- c(0.5)
#Gneiting covariance
SIGMA <- sigma2 * (rhoT * DistT^2 + 1)^(-1) * exp(-rho * Dist/(rhoT * DistT^2 + 1)^(sep_par/2))

Y <- rmnorm(1,rep(alpha, times = n), SIGMA) #generate the linear variable
theta <- c()
## wrapping step
for(i in 1:n) {
  theta[i] <- Y[i] %% (2*pi)
}
### Add plots of the simulated data
```

```
rose_diag(theta)
## use this values as references for the definition of initial values and priors
rho_sp.min <- 3/max(Dist)
rho_sp.max <- rho_sp.min+0.5
rho_t.min  <- 3/max(DistT)
rho_t.max  <- rho_t.min+0.5
val <- sample(1:n,round(n*0.2)) #validation set
set.seed(100)
mod <- WrapSpTi(
  x       = theta[-val],
  coords    = coords[-val,],
  times    = coordsT[-val],
  start   = list("alpha"      = c(.79, .74),
                 "rho_sp"     = c(.33,.52),
                 "rho_t"     = c(.19, .43),
                 "sigma2"    = c(.49, .37),
                 "sep_par"  = c(.47, .56),
                 "k"       = sample(0,length(theta[-val]), replace = TRUE)),
  priors   = list("rho_sp"      = c(0.01,3/4), ### uniform prior on this interval
                  "rho_t"      = c(0.01,3/4), ### uniform prior on this interval
                  "sep_par"  = c(1,1), ### beta prior
                  "sigma2"    = c(5,5),## inverse gamma prior with mode=5/6
                  "alpha" =  c(0,20) ## wrapped gaussian with large variance
  )  ,
  sd_prop   = list( "sigma2" = 0.1,  "rho_sp" = 0.1,  "rho_t" = 0.1,"sep_par"= 0.1),
  iter    = 7000,
  BurninThin    = c(burnin = 3000, thin = 10),
  accept_ratio = 0.234,
  adapt_param = c(start = 1, end = 1000, exp = 0.5),
  n_chains = 2 ,
  parallel = FALSE,
  n_cores = 1
)
check <- ConvCheck(mod,startit = 1 ,thin = 1)
check$Rhat ## convergence has been reached
## when plotting chains remember that alpha is a circular variable
par(mfrow = c(3,2))
coda::traceplot(check$mcmc)
par(mfrow = c(1,1))


############## Prediction on the validation set
Krig <- WrapKrigSpTi(
  WrapSpTi_out = mod,
  coords_obs =  coords[-val,],
  coords_nobs =  coords[val,],
  times_obs =  coordsT[-val],
  times_nobs =  coordsT[val],
  x_obs = theta[-val]
)
### checking the prediction
Wrap_Ape <- APEcirc(theta[val], Krig$Prev_out)
```

`april` *April waves.*

Description

Four days of waves data on the Adriatic sea in the month of April 2010.

Usage

```
april
```

Format

Date Date, format: yyyy-mm-dd

hour Factor w/ 24 levels corresponding to the 24h, format: 00:00

Lon, Lat decimal longitude and latitude

Hm0 Significant wave heights in meters

Dm Direction of waves in degrees (North=0)

state Factor w/ 3 levels "calm","transition", "storm"

Details

Wave directions and heights are obtained as outputs from a deterministic computer model implemented by Istituto Superiore per la Protezione e la Ricerca Ambientale (ISPRA). The computer model starts from a wind forecast model predicting the surface wind over the entire Mediterranean. The hourly evolution of sea wave spectra is obtained by solving energy transport equations using the wind forecast as input. Wave spectra are locally modified using a source function describing the wind energy, the energy redistribution due to nonlinear wave interactions, and energy dissipation due to wave fracture. The model produces estimates every hour on a grid with 10 x 10 km cells (Inghilesi et al. 2016). The ISPRA dataset has forecasts for a total of 4941 grid points over the Italian Mediterranean. Over the Adriatic Sea area, there are 1494 points.

A list containing 4 data frames each of 35856 rows and 7 columns.

Source

R. Inghilesi, A. Orasi & F. Catini (2016) The ISPRA Mediterranean Coastal Wave Forecasting system: evaluation and perspectives Journal of Operational Oceanography Vol. 9 , Iss. sup1 http://www.tandfonline.com/doi/full/10.1080/1755876X.2015.1115635

CircSpaceTime *CircSpaceTime: implementation of Bayesian models, for spatial and spatio-temporal interpolation of circular data.*

Description

The CircSpaceTime package provides two categories of important functions: Sampling Functions and Posterior (Kriging) Estimation Functions.

CircSpaceTime main functions

`WrapSp` and `ProjSp`, for sampling from a spatial Normal Wrapped and Projected, respectively. `WrapKrigSp` and `ProjKrigSp`, for posterior estimation on spatial Normal Wrapped and Projected, respectively.

`WrapSpTi` and `ProjSpTi`, for sampling from a spatio-temporal Normal Wrapped and Projected, respectively. `WrapKrigSpTi` and `ProjKrigSpTi`, for posterior estimation on spatio-temporal Normal Wrapped and Projected, respectively.

ConvCheck *Testing Convergence of mcmc using package coda*

Description

`ConvCheck` returns an mcmc.list (mcmc) to be used with the `coda` package and the Potential scale reduction factors (Rhat) of the model parameters computed using the `gelman.diag` function in the `coda` package

Usage

```
ConvCheck(mod, startit = 15000, thin = 10)
```

Arguments

`mod`	is a list with $m \geq 1$ elements, one for each chain generated using `WrapSp`, `ProjSp`, `WrapSpTi` or `ProjSpTi`
`startit`	is an integer, the iteration at which the chains start (required to build the mcmc.list)
`thin`	is an integer, the thinning applied to chains

Value

a list of two elements

`mcmc` an `mcmc.list` (mcmc) to be used with the `coda` package

`Rhat` the Potential scale reduction factors of the model parameters computed using the `gelman.diag` function in the `coda` package

See Also

ProjKrigSp and WrapKrigSp for posterior spatial estimations, and ProjKrigSpTi and WrapKrigSpTi for posterior spatio-temporal estimations

Examples

```
library(CircSpaceTime)
## auxiliary function
rmnorm<-function(n = 1, mean = rep(0, d), varcov){
  d <- if (is.matrix(varcov))
    ncol(varcov)
  else 1
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}

####
# Simulation with exponential spatial covariance function
####
set.seed(1)
n <- 20
coords <- cbind(runif(n,0,100), runif(n,0,100))
Dist <- as.matrix(dist(coords))

rho     <- 0.05
sigma2  <- 0.3
alpha   <- c(0.5)
SIGMA   <- sigma2*exp(-rho*Dist)

Y <- rmnorm(1,rep(alpha,times=n), SIGMA)
theta <- c()
for(i in 1:n) {
  theta[i] <- Y[i]%%(2*pi)
}
rose_diag(theta)

#validation set
val <- sample(1:n,round(n*0.1))

set.seed(12345)
mod <- WrapSp(
  x       = theta[-val],
  coords    = coords[-val,],
  start   = list("alpha"      = c(.36,0.38),
                 "rho"     = c(0.041,0.052),
                 "sigma2"    = c(0.24,0.32),
                 "k"       = rep(0,(n - length(val)))),
  priors  = list("rho"     = c(0.04,0.08), #few observations require to be more informative
                  "sigma2"    = c(2,1),
                  "alpha" =  c(0,10)
```

```
  ),
  sd_prop   = list( "sigma2" = 0.1,  "rho" = 0.1),
  iter    = 1000,
  BurninThin    = c(burnin = 500, thin = 5),
  accept_ratio = 0.234,
  adapt_param = c(start = 40000, end = 45000, exp = 0.5),
  corr_fun = "exponential",
  kappa_matern = .5,
  parallel = FALSE,
  #With doParallel, bigger iter (normally around 1e6) and n_cores>=2 it is a lot faster
  n_chains = 2 ,
  n_cores = 1
)
check <- ConvCheck(mod)
check$Rhat ## close to 1 means convergence has been reached
```

CRPScirc *The Continuous Ranked Probability Score for Circular Variables.*

Description

CRPScirc function computes the The Continuous Ranked Probability Score for Circular Variables

Usage

```
CRPScirc(obs, sim, bycol = FALSE)
```

Arguments

obs,	a vector of the values of the process at the test locations
sim,	a matrix with nrow the test locations and ncol the number of posterior samples from the posterior distributions
bycol,	logical. It is TRUE if the columns of sim represent the observations and the rows the posterior samples, the default value is FALSE

Value

a list of 2 elements

CRPSvec a vector of CRPS, one element for each test point

CRPS the overall mean

References

Grimit, Eric P., Tilmann Gneiting, Veronica J. Berrocal, Nicholas Alexander Johnson. "The Continuous Ranked Probability Score for Circular Variables and its Application to Mesoscale Forecast Ensemble Verification", Q.J.R. Meteorol. Soc. 132 (2005), 2925-2942.

See Also

ProjKrigSp and WrapKrigSp for posterior spatial interpolation, and ProjKrigSpTi and WrapKrigSpTi for posterior spatio-temporal interpolation

Other model performance indices: APEcirc

Examples

```
#' library(CircSpaceTime)
## auxiliary function
rmnorm<-function(n = 1, mean = rep(0, d), varcov){
  d <- if (is.matrix(varcov))
    ncol(varcov)
  else 1
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}

####
# Simulation with exponential spatial covariance function
####
set.seed(1)
n <- 20
coords <- cbind(runif(n,0,100), runif(n,0,100))
Dist <- as.matrix(dist(coords))

rho     <- 0.05
sigma2  <- 0.3
alpha   <- c(0.5)
SIGMA   <- sigma2*exp(-rho*Dist)

Y <- rmnorm(1,rep(alpha,times=n), SIGMA)
theta <- c()
for(i in 1:n) {
  theta[i] <- Y[i]%%(2*pi)
}
rose_diag(theta)

#validation set
val <- sample(1:n,round(n*0.1))

set.seed(12345)
mod <- WrapSp(
  x       = theta[-val],
  coords    = coords[-val,],
  start   = list("alpha"      = c(.36,0.38),
                 "rho"     = c(0.041,0.052),
                 "sigma2"    = c(0.24,0.32),
                 "k"       = rep(0,(n - length(val)))),
  priors  = list("rho"     = c(0.04,0.08), #few observations require to be more informative
                  "sigma2"    = c(2,1),
                  "alpha" =  c(0,10)
```

```
  ),
  sd_prop   = list( "sigma2" = 0.1,  "rho" = 0.1),
  iter    = 1000,
  BurninThin    = c(burnin = 500, thin = 5),
  accept_ratio = 0.234,
  adapt_param = c(start = 40000, end = 45000, exp = 0.5),
  corr_fun = "exponential",
  kappa_matern = .5,
  parallel = FALSE,
  #With doParallel, bigger iter (normally around 1e6) and n_cores>=2 it is a lot faster
  n_chains = 2 ,
  n_cores = 1
)
check <- ConvCheck(mod)
check$Rhat ## close to 1 means convergence has been reached
## graphical check
par(mfrow = c(3,1))
coda::traceplot(check$mcmc)
par(mfrow = c(1,1))
##### We move to the spatial interpolation

Krig <- WrapKrigSp(
  WrapSp_out = mod,
  coords_obs =  coords[-val,],
  coords_nobs =  coords[val,],
  x_obs = theta[-val]
)

#### check the quality of the prediction using APE and CRPS
ApeCheck <- APEcirc(theta[val],Krig$Prev_out)
CrpsCheck <- CRPScirc(theta[val],Krig$Prev_out)
```

`may` *May waves.*

Description

Four time slices of waves data on the Adriatic sea in the month of May 2010.

Usage

```
may
```

Format

object each element of the list is one hour of data on the entire area

Date Date, format: yyyy-mm-dd

hour Factor w/ 24 levels corresponding to the 24h, format: 00:00

Lon, Lat decimal longitude and latitude

Hm0 Significant wave heights in meters

Dm Direction of waves in degrees (North=0)

state Factor w/ 3 levels "calm","transition", "storm"

Details

Wave directions and heights are obtained as outputs from a deterministic computer model implemented by Istituto Superiore per la Protezione e la Ricerca Ambientale (ISPRA). The computer model starts from a wind forecast model predicting the surface wind over the entire Mediterranean. The hourly evolution of sea wave spectra is obtained by solving energy transport equations using the wind forecast as input. Wave spectra are locally modified using a source function describing the wind energy, the energy redistribution due to nonlinear wave interactions, and energy dissipation due to wave fracture. The model produces estimates every hour on a grid with 10 x 10 km cells (Inghilesi et al. 2016). The ISPRA dataset has forecasts for a total of 4941 grid points over the Italian Mediterranean. Over the Adriatic Sea area, there are 1494 points.

A list containing 4 data frames each of 1494 rows and 7 columns.

Source

R. Inghilesi, A. Orasi & F. Catini (2016) The ISPRA Mediterranean Coastal Wave Forecasting system: evaluation and perspectives Journal of Operational Oceanography Vol. 9 , Iss. sup1 http://www.tandfonline.com/doi/full/10.1080/1755876X.2015.1115635

`ProjKrigSp` *Kriging using projected normal model.*

Description

`ProjKrigSp` function computes the spatial prediction for circular spatial data using samples from the posterior distribution of the spatial projected normal

Usage

```
ProjKrigSp(ProjSp_out, coords_obs, coords_nobs, x_obs)
```

Arguments

`ProjSp_out`	the function takes the output of `ProjSp` function
`coords_obs`	coordinates of observed locations (in UTM)
`coords_nobs`	coordinates of unobserved locations (in UTM)
`x_obs`	observed values in $[0, 2\pi)$. If they are not in $[0, 2\pi)$, the function will transform the data in the right interval

Value

a list of 3 elements

`M_out` the mean of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by ProjSp

`V_out` the variance of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by ProjSp

`Prev_out` the posterior predicted values at the unobserved locations.

References

F. Wang, A. E. Gelfand, "Modeling space and space-time directional data using projected Gaussian processes", Journal of the American Statistical Association,109 (2014), 1565-1580

G. Mastrantonio, G. Jona Lasinio, A. E. Gelfand, "Spatio-temporal circular models with non-separable covariance structure", TEST 25 (2016), 331-350 https://doi.org/10.1007/s11749-015-0458-y

See Also

`ProjSp` for spatial sampling from Projected Normal , `WrapSp` for spatial sampling from Wrapped Normal and `WrapKrigSp` for spatial interpolation under the wrapped model

Other spatial interpolations: `WrapKrigSp`

Examples

```
library(CircSpaceTime)
## auxiliary function
rmnorm <- function(n = 1, mean = rep(0, d), varcov){
 d <- if (is.matrix(varcov))
   ncol(varcov)
 else 1
 z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
 y <- t(mean + t(z))
 return(y)
}

####
# Simulation using exponential  spatial covariance function
####
set.seed(1)
n <- 20
coords <- cbind(runif(n,0,100), runif(n,0,100))
Dist <- as.matrix(dist(coords))

rho     <- 0.05
tau     <- 0.2
sigma2  <- 1
alpha   <- c(0.5,0.5)
SIGMA   <- sigma2*exp(-rho*Dist)
```

```
Y <- rmnorm(1,rep(alpha,times=n),
        kronecker(SIGMA, matrix(c( sigma2,sqrt(sigma2)*tau,sqrt(sigma2)*tau,1 ) ,nrow=2 )))
theta <- c()
for(i in 1:n) {
 theta[i] <- atan2(Y[(i-1)*2+2],Y[(i-1)*2+1])
}
theta <- theta %% (2*pi) #to be sure to have values in (0,2pi)

hist(theta)
rose_diag(theta)

val <- sample(1:n,round(n*0.1))

################some useful quantities
rho.min <- 3/max(Dist)
rho.max <- rho.min+0.5

set.seed(100)

mod <- ProjSp(
 x       = theta[-val],
 coords    = coords[-val,],
 start   = list("alpha"      = c(0.92, 0.18, 0.56, -0.35),
                "rho"     = c(0.51,0.15),
                "tau"     = c(0.46, 0.66),
                "sigma2"    = c(0.27, 0.3),
                "r"       = abs(rnorm(  length(theta))  )),
 priors   = list("rho"      = c(rho.min,rho.max),
                 "tau"      = c(-1,1),
                 "sigma2"    = c(10,3),
                 "alpha_mu" = c(0, 0),
                 "alpha_sigma" = diag(10,2)
 )  ,
 sd_prop   = list("sigma2" = 0.1, "tau" = 0.1, "rho" = 0.1,
                  "sdr" = sample(.05,length(theta), replace = TRUE)),
 iter    = 10000,
 BurninThin    = c(burnin = 7000, thin = 10),
 accept_ratio = 0.234,
 adapt_param = c(start = 130000, end = 120000, exp = 0.5),#no adaptation
 corr_fun = "exponential",
 kappa_matern = .5,
 n_chains = 2 ,
 parallel = TRUE ,
 n_cores = 2
)
# If you don't want to install/use DoParallel
# please set parallel = FALSE. Keep in mind that it can be substantially slower
# How much it takes?

check <-  ConvCheck(mod)
check$Rhat #close to 1 we have convergence

#### graphical check
```

```
par(mfrow=c(3,2))
coda::traceplot(check$mcmc)

par(mfrow=c(1,1))

# move to prediction once convergence is achieved
Krig <- ProjKrigSp(
 ProjSp_out = mod,
 coords_obs =  coords[-val,],
 coords_nobs =  coords[val,],
 x_obs = theta[-val]
)

# The quality of prediction can be checked using APEcirc and CRPScirc
ape  <- APEcirc(theta[val],Krig$Prev_out)
crps <- CRPScirc(theta[val],Krig$Prev_out)
```

`ProjKrigSpTi` *#' Spatio temporal interpolation using projected spatial temporal normal model.*

Description

`ProjKrigSpTi` function computes the spatio-temporal prediction for circular space-time data using samples from the posterior distribution of the space-time projected normal model.

Usage

```
ProjKrigSpTi(ProjSpTi_out, coords_obs, coords_nobs, times_obs, times_nobs,
  x_obs)
```

Arguments

`ProjSpTi_out`	the functions takes the output of `ProjSpTi` function
`coords_obs`	coordinates of observed locations (in UTM)
`coords_nobs`	coordinates of unobserved locations (in UTM)
`times_obs`	numeric vector of observed time coordinates
`times_nobs`	numeric vector of unobserved time coordinates
`x_obs`	observed values in $[0, 2\pi)$ If they are not in $[0, 2\pi)$, the function will tranform the data in the right interval

Value

a list of 3 elements

`M_out` the mean of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by ProjSpTi

`V_out` the variance of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by ProjSpTi

`Prev_out` are the posterior predicted values at the unobserved locations.

References

G. Mastrantonio, G.Jona Lasinio, A. E. Gelfand, "Spatio-temporal circular models with non-separable covariance structure", TEST 25 (2016), 331–350.

F. Wang, A. E. Gelfand, "Modeling space and space-time directional data using projected Gaussian processes", Journal of the American Statistical Association,109 (2014), 1565-1580

T. Gneiting, "Nonseparable, Stationary Covariance Functions for Space-Time Data", JASA 97 (2002), 590-600

See Also

`ProjSpTi` to sample the posterior distribution of the spatio-temporal Projected Normal model, `WrapSpTi` to sample the posterior distribution of the spatio-temporal Wrapped Normal model and `WrapKrigSpTi` for spatio-temporal interpolation under the same model

Examples

```
library(CircSpaceTime)
#### simulated example
## auxiliary functions
rmnorm <- function(n = 1, mean = rep(0, d), varcov) {
  d <- if (is.matrix(varcov)) {
    ncol(varcov)
  } else {
    1
  }
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}
####
# Simulation using a gneiting covariance function
####
set.seed(1)
n <- 20

coords <- cbind(runif(n, 0, 100), runif(n, 0, 100))
coordsT <- cbind(runif(n, 0, 100))
Dist <- as.matrix(dist(coords))
DistT <- as.matrix(dist(coordsT))

rho <- 0.05
rhoT <- 0.01
sep_par <- 0.1
sigma2 <- 1
alpha <- c(0.5)
SIGMA <- sigma2 * (rhoT * DistT^2 + 1)^(-1) * exp(-rho * Dist / (rhoT * DistT^2 + 1)^(sep_par / 2))
```

```
tau <- 0.2

Y <- rmnorm(
  1, rep(alpha, times = n),
  kronecker(SIGMA, matrix(c(sigma2, sqrt(sigma2) * tau, sqrt(sigma2) * tau, 1), nrow = 2))
)
theta <- c()
for (i in 1:n) {
  theta[i] <- atan2(Y[(i - 1) * 2 + 2], Y[(i - 1) * 2 + 1])
}
theta <- theta %% (2 * pi) ## to be sure we have values in (0,2pi)
rose_diag(theta)
################ some useful quantities
rho_sp.min <- 3 / max(Dist)
rho_sp.max <- rho_sp.min + 0.5
rho_t.min <- 3 / max(DistT)
rho_t.max <- rho_t.min + 0.5
### validation set 20% of the data
val <- sample(1:n, round(n * 0.2))

set.seed(200)

mod <- ProjSpTi(
  x = theta[-val],
  coords = coords[-val, ],
  times = coordsT[-val],
  start = list(
    "alpha" = c(0.66, 0.38, 0.27, 0.13),
    "rho_sp" = c(0.29, 0.33),
    "rho_t" = c(0.25, 0.13),
    "sep_par" = c(0.56, 0.31),
    "tau" = c(0.71, 0.65),
    "sigma2" = c(0.47, 0.53),
    "r" = abs(rnorm(length(theta[-val])))
  ),
  priors = list(
    "rho_sp" = c(rho_sp.min, rho_sp.max), # Uniform prior in this interval
    "rho_t" = c(rho_t.min, rho_t.max), # Uniform prior in this interval
    "sep_par" = c(1, 1), # Beta distribution
    "tau" = c(-1, 1), ## Uniform prior in this interval
    "sigma2" = c(10, 3), # inverse gamma
    "alpha_mu" = c(0, 0), ## a vector of 2 elements,
    ## the means of the bivariate Gaussian distribution
    "alpha_sigma" = diag(10, 2) # a 2x2 matrix, the covariance matrix of the
    # bivariate Gaussian distribution,
  ),
  sd_prop = list(
    "sep_par" = 0.1, "sigma2" = 0.1, "tau" = 0.1, "rho_sp" = 0.1, "rho_t" = 0.1,
    "sdr" = sample(.05, length(theta), replace = TRUE)
  ),
  iter = 4000,
  BurninThin = c(burnin = 2000, thin = 2),
  accept_ratio = 0.234,
```

```
  adapt_param = c(start = 155000, end = 156000, exp = 0.5),
  n_chains = 2,
  parallel = TRUE,
)
check <- ConvCheck(mod)
check$Rhat ### convergence has been reached when the values are close to 1
#### graphical checking
#### recall that it is made of as many lists as the number of chains and it has elements named
#### as the model's parameters
par(mfrow = c(3, 3))
coda::traceplot(check$mcmc)
par(mfrow = c(1, 1))
# once convergence is reached we run the interpolation on the validation set
Krig <- ProjKrigSpTi(
  ProjSpTi_out = mod,
  coords_obs = coords[-val, ],
  coords_nobs = coords[val, ],
  times_obs = coordsT[-val],
  times_nobs = coordsT[val],
  x_obs = theta[-val]
)

#### checking the prediction

Proj_ape <- APEcirc(theta[val], Krig$Prev_out)
Proj_crps <- CRPScirc(theta[val],Krig$Prev_out)
```

ProjSp *Samples from the Projected Normal spatial model*

Description

ProjSp produces samples from the posterior distribtion of the spatial projected normal model.

Usage

```
ProjSp(x = x, coords = coords, start = list(alpha = c(1, 1, 0.5,
  0.5), tau = c(0.1, 0.5), rho = c(0.1, 0.5), sigma2 = c(0.1, 0.5), r =
  rep(1, times = length(x))), priors = list(tau = c(8, 14), rho = c(8,
  14), sigma2 = c(), alpha_mu = c(1, 1), alpha_sigma = c()),
  sd_prop = list(sigma2 = 0.5, tau = 0.5, rho = 0.5, sdr = sample(0.05,
  length(x), replace = TRUE)), iter = 1000, BurninThin = c(burnin = 20,
  thin = 10), accept_ratio = 0.234, adapt_param = c(start = 1, end =
  1e+07, exp = 0.9, sdr_update_iter = 50), corr_fun = "exponential",
  kappa_matern = 0.5, n_chains = 2, parallel = FALSE, n_cores = 1)
```

Arguments

x	a vector of n circular data in $[0, 2\pi)$. If they are not in $[0, 2\pi)$, the function will transform the data in the right interval
coords	an nx2 matrix with the sites coordinates
start	a list of 4 elements giving initial values for the model parameters. Each elements is a vector with n_chains elements • alpha the 2-d vector of the means of the Gaussian bi-variate distribution, • tau the correlation of the two components of the linear representation, • rho the spatial decay parameter, • sigma2 the process variance, • r the vector of length(x), latent variable
priors	a list of 4 elements to define priors for the model parameters: **alpha_mu** a vector of 2 elements, the means of the bivariate Gaussian distribution, **alpha_sigma** a 2x2 matrix, the covariance matrix of the bivariate Gaussian distribution, **rho** vector of 2 elements defining the minimum and maximum of a uniform distribution, **tau** vector of 2 elements defining the minimum and maximum of a uniform distribution, with the constraints min(tau) >= -1 and max(tau) <= 1 **sigma2** a vector of 2 elements defining the shape and rate of an inverse-gamma distribution,
sd_prop	list of 4 elements. To run the MCMC for the rho, tau and sigma2 parameters and r vector we use an adaptive metropolis and in sd_prop we build a list of initial guesses for these three parameters and the r vector
iter	number of iterations
BurninThin	a vector of 2 elements with the burnin and the chain thinning
accept_ratio	it is the desired acceptance ratio in the adaptive metropolis
adapt_param	a vector of 4 elements giving the iteration number at which the adaptation must start and end. The third element (exp) must be a number in (0,1) is a parameter ruling the speed of changes in the adaptation algorithm, it is recommended to set it close to 1, if it is too small non positive definite matrices may be generated and the program crashes. The last element (sdr_update_iter) must be a positive integer defining every how many iterations there is the update of the sd (vector) of (vector) r.
corr_fun	characters, the name of the correlation function; currently implemented functions are c("exponential", "matern","gaussian")
kappa_matern	numeric, the smoothness parameter of the Matern correlation function, default is kappa_matern = 0.5 (the exponential function)
n_chains	integer, the number of chains to be launched (default is 1, but we recommend to use at least 2 for model diagnostic)
parallel	logical, if the multiple chains must be lunched in parallel (you should install doParallel package). Default is FALSE
n_cores	integer, required if parallel=TRUE, the number of cores to be used in the implementation. Default value is 1.

Value

it returns a list of `n_chains` lists each with elements

`rho`,`tau`, `sigma2` vectors with the thinned chains

`alpha` a matrix with `nrow=2` and `ncol=` the length of thinned chains,

`r` a matrix with `nrow=length(x)` and `ncol=` the length of thinned chains

`corr_fun` characters with the type of spatial correlation chosen

`distribution` characters, always "ProjSp"

References

G. Mastrantonio , G. Jona Lasinio, A. E. Gelfand, "Spatio-temporal circular models with non-separable covariance structure", TEST 25 (2016), 331–350.

F. Wang, A. E. Gelfand, "Modeling space and space-time directional data using projected Gaussian processes", Journal of the American Statistical Association,109 (2014), 1565-1580

See Also

`ProjKrigSp` for spatial interpolation under the projected normal model, `WrapSp` for spatial sampling from Wrapped Normal and `WrapKrigSp` for Kriging estimation

Examples

```
library(CircSpaceTime)
## auxiliary function
rmnorm <- function(n = 1, mean = rep(0, d), varcov){
 d <- if (is.matrix(varcov))
   ncol(varcov)
 else 1
 z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
 y <- t(mean + t(z))
 return(y)
}

####
# Simulation using exponential  spatial covariance function
####
set.seed(1)
n <- 20
coords <- cbind(runif(n,0,100), runif(n,0,100))
Dist <- as.matrix(dist(coords))

rho     <- 0.05
tau     <- 0.2
sigma2  <- 1
alpha   <- c(0.5,0.5)
SIGMA   <- sigma2*exp(-rho*Dist)

Y <- rmnorm(1,rep(alpha,times=n),
```

```
        kronecker(SIGMA, matrix(c( sigma2,sqrt(sigma2)*tau,sqrt(sigma2)*tau,1 ) ,nrow=2 )))
theta <- c()
for(i in 1:n) {
 theta[i] <- atan2(Y[(i-1)*2+2],Y[(i-1)*2+1])
}
theta <- theta %% (2*pi) #to be sure to have values in (0,2pi)

hist(theta)
rose_diag(theta)

val <- sample(1:n,round(n*0.1))

################some useful quantities
rho.min <- 3/max(Dist)
rho.max <- rho.min+0.5

set.seed(100)

mod <- ProjSp(
 x       = theta[-val],
 coords    = coords[-val,],
 start   = list("alpha"      = c(0.92, 0.18, 0.56, -0.35),
                "rho"     = c(0.51,0.15),
                "tau"     = c(0.46, 0.66),
                "sigma2"    = c(0.27, 0.3),
                "r"       = abs(rnorm(  length(theta))  )),
 priors   = list("rho"      = c(rho.min,rho.max),
                 "tau"      = c(-1,1),
                 "sigma2"    = c(10,3),
                 "alpha_mu" = c(0, 0),
                 "alpha_sigma" = diag(10,2)
 )  ,
 sd_prop   = list("sigma2" = 0.1, "tau" = 0.1, "rho" = 0.1,
                  "sdr" = sample(.05,length(theta), replace = TRUE)),
 iter    = 10000,
 BurninThin    = c(burnin = 7000, thin = 10),
 accept_ratio = 0.234,
 adapt_param = c(start = 130000, end = 120000, exp = 0.5),#no adaptation
 corr_fun = "exponential",
 kappa_matern = .5,
 n_chains = 2 ,
 parallel = TRUE ,
 n_cores = 2
)
# If you don't want to install/use DoParallel
# please set parallel = FALSE. Keep in mind that it can be substantially slower
# How much it takes?

check <-  ConvCheck(mod)
check$Rhat #close to 1 we have convergence

#### graphical check
par(mfrow=c(3,2))
```

```
coda::traceplot(check$mcmc)

par(mfrow=c(1,1))
# once convergence is achieved move to prediction using ProjKrigSp
```

`ProjSpTi` *Samples from the posterior distribution of the Projected Normal spatial model*

Description

`ProjSpTi` produces samples from the posterior distribution of the spatial projected normal model.

Usage

```
ProjSpTi(x = x, coords = coords, times = c(), start = list(alpha =
  c(1, 1, 0.5, 0.5), tau = c(0.1, 0.5), rho_sp = c(0.1, 0.5), rho_t =
  c(0.1, 0.5), sep_par = c(0.1, 0.5), sigma2 = c(0.1, 0.5), r = sample(1,
  length(x), replace = T)), priors = list(tau = c(8, 14), rho_sp = c(8,
  14), rho_t = c(8, 14), sep_par = c(8, 14), sigma2 = c(), alpha_mu = c(1,
  1), alpha_sigma = c()), sd_prop = list(sigma2 = 0.5, tau = 0.5, rho_sp
  = 0.5, rho_t = 0.5, sep_par = 0.5, sdr = sample(0.05, length(x), replace
  = T)), iter = 1000, BurninThin = c(burnin = 20, thin = 10),
  accept_ratio = 0.234, adapt_param = c(start = 1, end = 1e+07, exp =
  0.9, sdr_update_iter = 50), n_chains = 2, parallel = FALSE,
  n_cores = 1)
```

Arguments

`x`	a vector of n circular data in $[0, 2\pi)$ If they are not in $[0, 2\pi)$, the function will tranform the data in the right interval
`coords`	an nx2 matrix with the sites coordinates
`times`	an n vector with the times of ....
`start`	a list of 4 elements giving initial values for the model parameters. Each elements is a vector with `n_chains` elements • alpha the 2-d vector of the means of the Gaussian bi-variate distribution, • tau the correlation of the two components of the linear representation, • rho_sp the spatial decay parameter, • rho_t the temporal decay parameter, • sigma2 the process variance, • sep_par the separation parameter, • r the vector of `length(x)`, latent variable
`priors`	a list of 7 elements to define priors for the model parameters:

alpha_mu a vector of 2 elements, the means of the bivariate Gaussian distribution,

alpha_sigma a 2x2 matrix, the covariance matrix of the bivariate Gaussian distribution,

rho_sp a vector of 2 elements defining the minimum and maximum of a uniform distribution,

rho_t a vector of 2 elements defining the minimum and maximum of a uniform distribution,

tau vector of 2 elements defining the minimum and maximum of a uniform distribution with the constraints min(tau) >= -1 and max(tau) <= 1,

sep_par a vector of 2 elements defining the two parameters of a beta distribution,

sigma2 a vector of 2 elements defining the shape and rate of an inverse-gamma distribution,

`sd_prop` =list of 4 elements. To run the MCMC for the rho_sp, tau and sigma2 parameters and r vector we use an adaptive metropolis and in sd_prop we build a list of initial guesses for these three parameters and the r vector

`iter` iter number of iterations

`BurninThin` a vector of 2 elements with the burnin and the chain thinning

`accept_ratio` it is the desired acceptance ratio in the adaptive metropolis

`adapt_param` a vector of 4 elements giving the iteration number at which the adaptation must start and end. The third element (exp) must be a number in (0,1) is a parameter ruling the speed of changes in the adaptation algorithm, it is recommended to set it close to 1, if it is too small non positive definite matrices may be generated and the program crashes. The last element (sdr_update_iter) must be a positive integer defining every how many iterations there is the update of the sd (vector) of (vector) r.

`n_chains` integer, the number of chains to be launched (default is 1, but we recommend to use at least 2 for model diagnostic)

`parallel` logical, if the multiple chains must be lunched in parallel (you should install doParallel package). Default is FALSE

`n_cores` integer, required if parallel=TRUE, the number of cores to be used in the implementation. Default value is 1.

Value

it returns a list of `n_chains` lists each with elements

`tau`, `rho_sp`, `rho_t`, `sigma2` vectors with the thinned chains

`alpha` a matrix with `nrow=2` and `ncol=` the length of thinned chains

`r` a matrix with `nrow=length(x)` and `ncol=` the length of thinned chains

References

G. Mastrantonio, G.Jona Lasinio, A. E. Gelfand, "Spatio-temporal circular models with non-separable covariance structure", TEST 25 (2016), 331–350.

F. Wang, A. E. Gelfand, "Modeling space and space-time directional data using projected Gaussian processes", Journal of the American Statistical Association,109 (2014), 1565-1580

T. Gneiting, "Nonseparable, Stationary Covariance Functions for Space-Time Data", JASA 97 (2002), 590-600

See Also

ProjKrigSpTi for spatio-temporal prediction under the spatio-temporal projected normal model, WrapSpTi to sample from the posterior distribution of the spatio-temporal Wrapped Normal model and WrapKrigSpTi for spatio-temporal prediction under the same model

Other spatio-temporal models: WrapSpTi

Examples

```
library(CircSpaceTime)
#### simulated example
## auxiliary functions
rmnorm <- function(n = 1, mean = rep(0, d), varcov) {
  d <- if (is.matrix(varcov)) {
    ncol(varcov)
  } else {
    1
  }
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}
####
# Simulation using a gneiting covariance function
####
set.seed(1)
n <- 20

coords <- cbind(runif(n, 0, 100), runif(n, 0, 100))
coordsT <- cbind(runif(n, 0, 100))
Dist <- as.matrix(dist(coords))
DistT <- as.matrix(dist(coordsT))

rho <- 0.05
rhoT <- 0.01
sep_par <- 0.1
sigma2 <- 1
alpha <- c(0.5)
SIGMA <- sigma2 * (rhoT * DistT^2 + 1)^(-1) * exp(-rho * Dist / (rhoT * DistT^2 + 1)^(sep_par / 2))
tau <- 0.2

Y <- rmnorm(
```

```
  1, rep(alpha, times = n),
  kronecker(SIGMA, matrix(c(sigma2, sqrt(sigma2) * tau, sqrt(sigma2) * tau, 1), nrow = 2))
)
theta <- c()
for (i in 1:n) {
  theta[i] <- atan2(Y[(i - 1) * 2 + 2], Y[(i - 1) * 2 + 1])
}
theta <- theta %% (2 * pi) ## to be sure we have values in (0,2pi)
rose_diag(theta)
################ some useful quantities
rho_sp.min <- 3 / max(Dist)
rho_sp.max <- rho_sp.min + 0.5
rho_t.min <- 3 / max(DistT)
rho_t.max <- rho_t.min + 0.5
### validation set 20% of the data
val <- sample(1:n, round(n * 0.2))

set.seed(200)

mod <- ProjSpTi(
  x = theta[-val],
  coords = coords[-val, ],
  times = coordsT[-val],
  start = list(
    "alpha" = c(0.66, 0.38, 0.27, 0.13),
    "rho_sp" = c(0.29, 0.33),
    "rho_t" = c(0.25, 0.13),
    "sep_par" = c(0.56, 0.31),
    "tau" = c(0.71, 0.65),
    "sigma2" = c(0.47, 0.53),
    "r" = abs(rnorm(length(theta[-val])))
  ),
  priors = list(
    "rho_sp" = c(rho_sp.min, rho_sp.max), # Uniform prior in this interval
    "rho_t" = c(rho_t.min, rho_t.max), # Uniform prior in this interval
    "sep_par" = c(1, 1), # Beta distribution
    "tau" = c(-1, 1), ## Uniform prior in this interval
    "sigma2" = c(10, 3), # inverse gamma
    "alpha_mu" = c(0, 0), ## a vector of 2 elements,
    ## the means of the bivariate Gaussian distribution
    "alpha_sigma" = diag(10, 2) # a 2x2 matrix, the covariance matrix of the
    # bivariate Gaussian distribution,
  ),
  sd_prop = list(
    "sep_par" = 0.1, "sigma2" = 0.1, "tau" = 0.1, "rho_sp" = 0.1, "rho_t" = 0.1,
    "sdr" = sample(.05, length(theta), replace = TRUE)
  ),
  iter = 4000,
  BurninThin = c(burnin = 2000, thin = 2),
  accept_ratio = 0.234,
  adapt_param = c(start = 155000, end = 156000, exp = 0.5),
  n_chains = 2,
  parallel = TRUE,
```

```
)
check <- ConvCheck(mod)
check$Rhat ### convergence has been reached when the values are close to 1
#### graphical checking
#### recall that it is made of as many lists as the number of chains and it has elements named
#### as the model's parameters
par(mfrow = c(3, 3))
coda::traceplot(check$mcmc)
par(mfrow = c(1, 1))
# move to prediction once convergence is achieved using ProjKrigSpTi
```

---

rose_diag *Rose diagram in ggplot2 inspired from rose.diag in package circular.*

Description

Rose diagram in ggplot2 inspired from rose.diag in package circular.

Usage

```
rose_diag(x, bins = 15, color = "red", alpha = 1, start = 0,
  add = FALSE, template = "rad", direction = NULL)
```

Arguments

x	a vector of circular coordinates in radiants $[0, 2\pi)$.
bins	number of bins
color	color of the line and of the fill
alpha	transparency
start	the starting angle of the 0 (the North)
add	add the rose_diag to an existing ggplot2 plot
template	radiants or wind rose. the values are `c("rad","wind_rose")`.. default is `"rad"`
direction	1, clockwise; -1, anticlockwise. For template = "rad" direction is -1 while for template = "wind_rose" direction is 1.

Value

The plot in ggplot2 format.

Examples

```
library(CircSpaceTime)
x <- circular::rwrappedstable(200, index = 1.5, skewness = .5)
x1 <- circular::rwrappedstable(200, index = 2, skewness = .5)
x2 <- circular::rwrappedstable(200, index = 0.5, skewness = 1)
```

```
rose_diag(x, bins = 15, color = "green")
rose_diag(x1, bins = 15, color = "blue", alpha = .5, add = TRUE)
rose_diag(x2, bins = 15, color = "red", alpha = .5, add = TRUE)
```

`WrapKrigSp` *Spatial interpolation using wrapped normal model.*

Description

`WrapKrigSp` function computes the spatial prediction for circular spatial data using samples from the posterior distribution of the spatial wrapped normal

Usage

```
WrapKrigSp(WrapSp_out, coords_obs, coords_nobs, x_obs)
```

Arguments

`WrapSp_out`	the functions takes the output of `WrapSp` function
`coords_obs`	coordinates of observed locations (in UTM)
`coords_nobs`	coordinates of unobserved locations (in UTM)
`x_obs`	observed values

Value

a list of 3 elements

`M_out` the mean of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by WrapSp

`V_out` the variance of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by WrapSp

`Prev_out` the posterior predicted values at the unobserved locations.

Implementation Tips

To facilitate the estimations, the observations x are centered around pi, and the posterior samples of x and posterior mean are changed back to the original scale

References

G. Jona-Lasinio, A .E. Gelfand, M. Jona-Lasinio, "Spatial analysis of wave direction data using wrapped Gaussian processes", The Annals of Applied Statistics, 6 (2012), 1478-1498

See Also

WrapSp for spatial sampling from Wrapped Normal , ProjSp for spatial sampling from Projected Normal and ProjKrigSp for Kriging estimation

Other spatial interpolations: ProjKrigSp

Examples

```
library(CircSpaceTime)
## auxiliary function
rmnorm<-function(n = 1, mean = rep(0, d), varcov){
  d <- if (is.matrix(varcov))
    ncol(varcov)
  else 1
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}

####
# Simulation with exponential spatial covariance function
####
set.seed(1)
n <- 20
coords <- cbind(runif(n,0,100), runif(n,0,100))
Dist <- as.matrix(dist(coords))

rho     <- 0.05
sigma2  <- 0.3
alpha   <- c(0.5)
SIGMA   <- sigma2*exp(-rho*Dist)

Y <- rmnorm(1,rep(alpha,times=n), SIGMA)
theta <- c()
for(i in 1:n) {
  theta[i] <- Y[i]%%(2*pi)
}
rose_diag(theta)

#validation set
val <- sample(1:n,round(n*0.1))

set.seed(12345)
mod <- WrapSp(
  x       = theta[-val],
  coords    = coords[-val,],
  start   = list("alpha"      = c(.36,0.38),
                 "rho"     = c(0.041,0.052),
                 "sigma2"    = c(0.24,0.32),
                 "k"       = rep(0,(n - length(val)))),
  priors  = list("rho"     = c(0.04,0.08), #few observations require to be more informative
                  "sigma2"    = c(2,1),
                  "alpha" =  c(0,10)
```

```
  ),
  sd_prop   = list( "sigma2" = 0.1,  "rho" = 0.1),
  iter    = 1000,
  BurninThin    = c(burnin = 500, thin = 5),
  accept_ratio = 0.234,
  adapt_param = c(start = 40000, end = 45000, exp = 0.5),
  corr_fun = "exponential",
  kappa_matern = .5,
  parallel = FALSE,
  #With doParallel, bigger iter (normally around 1e6) and n_cores>=2 it is a lot faster
  n_chains = 2 ,
  n_cores = 1
)
check <- ConvCheck(mod)
check$Rhat ## close to 1 means convergence has been reached
## graphical check
par(mfrow = c(3,1))
coda::traceplot(check$mcmc)
par(mfrow = c(1,1))
##### We move to the spatial interpolation

Krig <- WrapKrigSp(
  WrapSp_out = mod,
  coords_obs =  coords[-val,],
  coords_nobs =  coords[val,],
  x_obs = theta[-val]
)

#### check the quality of the prediction using APE and CRPS
ApeCheck <- APEcirc(theta[val],Krig$Prev_out)
CrpsCheck <- CRPScirc(theta[val],Krig$Prev_out)
```

`WrapKrigSpTi` *Prediction using wrapped normal spatio-temporal model.*

Description

`WrapKrigSpTi` function computes the spatio-temporal prediction for circular space-time data using samples from the posterior distribution of the space-time wrapped normal model

Usage

```
WrapKrigSpTi(WrapSpTi_out, coords_obs, coords_nobs, times_obs, times_nobs,
  x_obs)
```

Arguments

`WrapSpTi_out`	the functions takes the output of `WrapSpTi` function
`coords_obs`	coordinates of observed locations (in UTM)

coords_nobs	coordinates of unobserved locations (in UTM)
times_obs	numeric vector of observed time coordinates
times_nobs	numeric vector of unobserved time coordinates
x_obs	observed values

Value

a list of 3 elements

M_out the mean of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by WrapSpTi

V_out the variance of the associated linear process on the prediction locations coords_nobs (rows) over all the posterior samples (columns) returned by WrapSpTi

Prev_out the posterior predicted values at the unobserved locations

Implementation Tips

To facilitate the estimations, the observations x are centered around π. Posterior samples of x at the predictive locations and posterior mean are changed back to the original scale

References

G. Mastrantonio, G. Jona Lasinio, A. E. Gelfand, "Spatio-temporal circular models with non-separable covariance structure", TEST 25 (2016), 331–350

T. Gneiting, "Nonseparable, Stationary Covariance Functions for Space-Time Data", JASA 97 (2002), 590-600

See Also

WrapSpTi spatio-temporal sampling from Wrapped Normal, ProjSpTi for spatio-temporal sampling from Projected Normal and ProjKrigSpTi for Kriging estimation

Examples

```
library(CircSpaceTime)
## functions
rmnorm <- function(n = 1, mean = rep(0, d), varcov){
  d <- if (is.matrix(varcov))
    ncol(varcov)
  else 1
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}

#####################################
## Simulation                      ##
#####################################
set.seed(1)
```

```
n <- 20
### simulate coordinates from a unifrom distribution
coords  <- cbind(runif(n,0,100), runif(n,0,100)) #spatial coordinates
coordsT <- sort(runif(n,0,100)) #time coordinates (ordered)
Dist <- as.matrix(dist(coords))
DistT <- as.matrix(dist(coordsT))

rho     <- 0.05 #spatial decay
rhoT    <- 0.01 #temporal decay
sep_par <- 0.5 #separability parameter
sigma2  <- 0.3 # variance of the process
alpha   <- c(0.5)
#Gneiting covariance
SIGMA <- sigma2 * (rhoT * DistT^2 + 1)^(-1) * exp(-rho * Dist/(rhoT * DistT^2 + 1)^(sep_par/2))

Y <- rmnorm(1,rep(alpha, times = n), SIGMA) #generate the linear variable
theta <- c()
## wrapping step
for(i in 1:n) {
  theta[i] <- Y[i] %% (2*pi)
}
### Add plots of the simulated data

rose_diag(theta)
## use this values as references for the definition of initial values and priors
rho_sp.min <- 3/max(Dist)
rho_sp.max <- rho_sp.min+0.5
rho_t.min  <- 3/max(DistT)
rho_t.max  <- rho_t.min+0.5
val <- sample(1:n,round(n*0.2)) #validation set
set.seed(100)
mod <- WrapSpTi(
  x       = theta[-val],
  coords    = coords[-val,],
  times    = coordsT[-val],
  start   = list("alpha"      = c(.79, .74),
                 "rho_sp"     = c(.33,.52),
                 "rho_t"     = c(.19, .43),
                 "sigma2"    = c(.49, .37),
                 "sep_par"  = c(.47, .56),
                 "k"       = sample(0,length(theta[-val]), replace = TRUE)),
  priors   = list("rho_sp"      = c(0.01,3/4), ### uniform prior on this interval
                  "rho_t"      = c(0.01,3/4), ### uniform prior on this interval
                  "sep_par"  = c(1,1), ### beta prior
                  "sigma2"    = c(5,5),## inverse gamma prior with mode=5/6
                  "alpha" =  c(0,20) ## wrapped gaussian with large variance
  )  ,
  sd_prop   = list( "sigma2" = 0.1,  "rho_sp" = 0.1,  "rho_t" = 0.1,"sep_par"= 0.1),
  iter    = 7000,
  BurninThin    = c(burnin = 3000, thin = 10),
  accept_ratio = 0.234,
  adapt_param = c(start = 1, end = 1000, exp = 0.5),
  n_chains = 2 ,
```

```
  parallel = FALSE,
  n_cores = 1
)
check <- ConvCheck(mod,startit = 1 ,thin = 1)
check$Rhat ## convergence has been reached
## when plotting chains remember that alpha is a circular variable
par(mfrow = c(3,2))
coda::traceplot(check$mcmc)
par(mfrow = c(1,1))


############## Prediction on the validation set
Krig <- WrapKrigSpTi(
  WrapSpTi_out = mod,
  coords_obs =  coords[-val,],
  coords_nobs =  coords[val,],
  times_obs =  coordsT[-val],
  times_nobs =  coordsT[val],
  x_obs = theta[-val]
)
### checking the prediction
Wrap_Ape <- APEcirc(theta[val], Krig$Prev_out)
Wrap_Crps <- CRPScirc(theta[val], Krig$Prev_out)
```

`WrapSp` *Samples from the Wrapped Normal spatial model*

Description

The function `WrapSp` produces samples from the posterior distribution of the wrapped normal spatial model.

Usage

```
WrapSp(x = x, coords = coords, start = list(alpha = c(2, 1), rho =
  c(0.1, 0.5), sigma2 = c(0.1, 0.5), k = sample(0, length(x), replace =
  T)), priors = list(alpha = c(pi, 1, -10, 10), rho = c(8, 14), sigma2 =
  c()), sd_prop = list(sigma2 = 0.5, rho = 0.5), iter = 1000,
  BurninThin = c(burnin = 20, thin = 10), accept_ratio = 0.234,
  adapt_param = c(start = 1, end = 1e+07, exp = 0.9),
  corr_fun = "exponential", kappa_matern = 0.5, n_chains = 1,
  parallel = FALSE, n_cores = 1)
```

Arguments

`x`	a vector of n circular data in $[0, 2\pi)$ If they are not in $[0, 2\pi)$, the function will tranform the data in the right interval
`coords`	an nx2 matrix with the sites coordinates

`start` a list of 4 elements giving initial values for the model parameters. Each elements is a numeric vector with `n_chains` elements

- alpha the mean which value is in $[0, 2\pi)$.
- rho the spatial decay parameter
- sigma2 the process variance
- k the vector of `length(x)` winding numbers

`priors` a list of 3 elements to define priors for the model parameters:

alpha a vector of 2 elements the mean and the variance of a Wrapped Gaussian distribution, default is mean π and variance 1,

rho a vector of 2 elements defining the minimum and maximum of a uniform distribution,

sigma2 a vector of 2 elements defining the shape and rate of an inverse-gamma distribution,

`sd_prop` list of 3 elements. To run the MCMC for the rho and sigma2 parameters we use an adaptive metropolis and in sd.prop we build a list of initial guesses for these two parameters and the beta parameter

`iter` number of iterations

`BurninThin` a vector of 2 elements with the burnin and the chain thinning

`accept_ratio` it is the desired acceptance ratio in the adaptive metropolis

`adapt_param` a vector of 3 elements giving the iteration number at which the adaptation must start and end. The third element (exp) must be a number in (0,1) and it is a parameter ruling the speed of changes in the adaptation algorithm, it is recommended to set it close to 1, if it is too small non positive definite matrices may be generated and the program crashes.

`corr_fun` characters, the name of the correlation function; currently implemented functions are c("exponential", "matern","gaussian")

`kappa_matern` numeric, the smoothness parameter of the Matern correlation function, default is `kappa_matern = 0.5` (the exponential function)

`n_chains` integer, the number of chains to be launched (default is 1, but we recommend to use at least 2 for model diagnostic)

`parallel` logical, if the multiple chains must be lunched in parallel (you should install doParallel package). Default is FALSE

`n_cores` integer, required if parallel=TRUE, the number of cores to be used in the implementation. Default value is 1.

Value

It returns a list of `n_chains` lists each with elements

- `alpha`, `rho`,`sigma2` vectors with the thinned chains,
- `k` a matrix with `nrow = length(x)` and `ncol =` the length of thinned chains
- `corr_fun` characters with the type of spatial correlation chosen.
- `distribution` characters, always "WrapSp"

Implementation Tips

To facilitate the estimations, the observations x are centered around pi, and the prior and starting value of alpha are changed accordingly. After the estimations, posterior samples of alpha are changed back to the original scale

References

G. Jona Lasinio, A. Gelfand, M. Jona-Lasinio, "Spatial analysis of wave direction data using wrapped Gaussian processes", The Annals of Applied Statistics 6 (2013), 1478-1498

See Also

WrapKrigSp for spatial interpolation, ProjSp for posterior sampling from the Projected Normal model and ProjKrigSp for spatial interpolation under the same model

Examples

```
library(CircSpaceTime)
## auxiliary function
rmnorm<-function(n = 1, mean = rep(0, d), varcov){
  d <- if (is.matrix(varcov))
    ncol(varcov)
  else 1
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}

####
# Simulation with exponential spatial covariance function
####
set.seed(1)
n <- 20
coords <- cbind(runif(n,0,100), runif(n,0,100))
Dist <- as.matrix(dist(coords))

rho     <- 0.05
sigma2  <- 0.3
alpha   <- c(0.5)
SIGMA   <- sigma2*exp(-rho*Dist)

Y <- rmnorm(1,rep(alpha,times=n), SIGMA)
theta <- c()
for(i in 1:n) {
  theta[i] <- Y[i]%%(2*pi)
}
rose_diag(theta)

#validation set
val <- sample(1:n,round(n*0.1))
```

```
set.seed(12345)
mod <- WrapSp(
  x       = theta[-val],
  coords    = coords[-val,],
  start   = list("alpha"      = c(.36,0.38),
                 "rho"     = c(0.041,0.052),
                 "sigma2"    = c(0.24,0.32),
                 "k"       = rep(0,(n - length(val)))),
  priors  = list("rho"     = c(0.04,0.08), #few observations require to be more informative
                  "sigma2"    = c(2,1),
                  "alpha" =  c(0,10)
  ),
  sd_prop   = list( "sigma2" = 0.1,  "rho" = 0.1),
  iter    = 1000,
  BurninThin    = c(burnin = 500, thin = 5),
  accept_ratio = 0.234,
  adapt_param = c(start = 40000, end = 45000, exp = 0.5),
  corr_fun = "exponential",
  kappa_matern = .5,
  parallel = FALSE,
  #With doParallel, bigger iter (normally around 1e6) and n_cores>=2 it is a lot faster
  n_chains = 2 ,
  n_cores = 1
)
check <- ConvCheck(mod)
check$Rhat ## close to 1 means convergence has been reached
## graphical check
par(mfrow = c(3,1))
coda::traceplot(check$mcmc)
par(mfrow = c(1,1))
##### We move to the spatial interpolation see WrapKrigSp
```

`WrapSpTi` *Samples from the posterior distribution of the Wrapped Normal spatial temporal model*

Description

The `WrapSpTi` function returns samples from the posterior distribution of the spatio-temporal Wrapped Gaussian Model

Usage

```
WrapSpTi(x = x, coords = coords, times, start = list(alpha = c(2, 1),
  rho_sp = c(0.1, 0.5), rho_t = c(0.1, 1), sep_par = c(0.01, 0.1), k =
  sample(0, length(x), replace = T)), priors = list(alpha = c(pi, 1, -10,
  10), rho_sp = c(8, 14), rho_t = c(1, 2), sep_par = c(0.001, 1), sigma2 =
  c()), sd_prop = list(rho_sp = 0.5, rho_t = 0.5, sep_par = 0.5, sigma2 =
  0.5), iter = 1000, BurninThin = c(burnin = 20, thin = 10),
  accept_ratio = 0.234, adapt_param = c(start = 1, end = 1e+07, exp =
  0.9), n_chains = 1, parallel = FALSE, n_cores = 1)
```

Arguments

x	a vector of n circular data in $[0, 2\pi)$. If they are not in $[0, 2\pi)$, the function will transform the data into the right interval
coords	an nx2 matrix with the sites coordinates
times	an n vector with the times of the observations x
start	a list of 4 elements giving initial values for the model parameters. Each elements is a vector with n_chains elements • alpha the mean which value is in $[0, 2\pi)$ • rho_sp the spatial decay parameter, • rho_t the temporal decay parameter, • sigma2 the process variance, • sep_par the separation parameter, • k the vector of length(x) winding numbers
priors	a list of 5 elements to define priors for the model parameters: **alpha** a vector of 2 elements the mean and the variance of a Gaussian distribution, default is mean π and variance 1, **rho_sp** a vector of 2 elements defining the minimum and maximum of a uniform distribution, **rho_t** a vector of 2 elements defining the minimum and maximum of a uniform distribution, **sep_par** a vector of 2 elements defining the two parameters of a beta distribution, **sigma2** a vector of 2 elements defining the shape and rate of an inverse-gamma distribution,
sd_prop	list of 3 elements. To run the MCMC for the rho_sp and sigma2 parameters we use an adaptive metropolis and in sd_prop we build a list of initial guesses for these two parameters and the beta parameter
iter	iter number of iterations
BurninThin	a vector of 2 elements with the burnin and the chain thinning
accept_ratio	it is the desired acceptance ratio in the adaptive metropolis
adapt_param	a vector of 3 elements giving the iteration number at which the adaptation must start and end. The third element (exp) must be a number in (0,1) and it is a parameter ruling the speed of changes in the adaptation algorithm, it is recommended to set it close to 1, if it is too small non positive definite matrices may be generated and the program crashes.
n_chains	integer, the number of chains to be launched (default is 1, but we recommend to use at least 2 for model diagnostic)
parallel	logical, if the multiple chains must be lunched in parallel (you should install doParallel package). Default is FALSE
n_cores	integer, required if parallel=TRUE, the number of cores to be used in the implementation. Default value is 1.

Value

it returns a list of `n_chains` lists each with elements

`alpha`, `rho_sp`, `rho_t`, `sep_par`, `sigma2` vectors with the thinned chains

`k` a matrix with `nrow = length(x)` and `ncol =` the length of thinned chains

`distribution` characters, always "WrapSpTi"

Implementation Tips

To facilitate the estimations, the observations x are centered around pi, and the prior and starting value of alpha are changed accordingly. After the estimations, posterior samples of alpha are changed back to the original scale

References

G. Mastrantonio, G. Jona Lasinio, A. E. Gelfand, "Spatio-temporal circular models with non-separable covariance structure", TEST 25 (2016), 331–350.

T. Gneiting, "Nonseparable, Stationary Covariance Functions for Space-Time Data", JASA 97 (2002), 590-600

See Also

`WrapKrigSpTi` for spatio-temporal prediction, `ProjSpTi` to sample from the posterior distribution of the spatio-temporal Projected Normal model and `ProjKrigSpTi` for spatio-temporal prediction under the same model

Other spatio-temporal models: `ProjSpTi`

Examples

```
library(CircSpaceTime)
## functions
rmnorm <- function(n = 1, mean = rep(0, d), varcov){
  d <- if (is.matrix(varcov))
    ncol(varcov)
  else 1
  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
  y <- t(mean + t(z))
  return(y)
}

#####################################
## Simulation                      ##
#####################################
set.seed(1)
n <- 20
### simulate coordinates from a unifrom distribution
coords  <- cbind(runif(n,0,100), runif(n,0,100)) #spatial coordinates
coordsT <- sort(runif(n,0,100)) #time coordinates (ordered)
Dist <- as.matrix(dist(coords))
```

```
DistT <- as.matrix(dist(coordsT))

rho     <- 0.05 #spatial decay
rhoT    <- 0.01 #temporal decay
sep_par <- 0.5 #separability parameter
sigma2  <- 0.3 # variance of the process
alpha   <- c(0.5)
#Gneiting covariance
SIGMA <- sigma2 * (rhoT * DistT^2 + 1)^(-1) * exp(-rho * Dist/(rhoT * DistT^2 + 1)^(sep_par/2))

Y <- rmnorm(1,rep(alpha, times = n), SIGMA) #generate the linear variable
theta <- c()
## wrapping step
for(i in 1:n) {
  theta[i] <- Y[i] %% (2*pi)
}
### Add plots of the simulated data

rose_diag(theta)
## use this values as references for the definition of initial values and priors
rho_sp.min <- 3/max(Dist)
rho_sp.max <- rho_sp.min+0.5
rho_t.min  <- 3/max(DistT)
rho_t.max  <- rho_t.min+0.5
val <- sample(1:n,round(n*0.2)) #validation set
set.seed(100)
mod <- WrapSpTi(
  x       = theta[-val],
  coords    = coords[-val,],
  times    = coordsT[-val],
  start   = list("alpha"      = c(.79, .74),
                 "rho_sp"     = c(.33,.52),
                 "rho_t"     = c(.19, .43),
                 "sigma2"    = c(.49, .37),
                 "sep_par"  = c(.47, .56),
                 "k"       = sample(0,length(theta[-val]), replace = TRUE)),
  priors   = list("rho_sp"      = c(0.01,3/4), ### uniform prior on this interval
                  "rho_t"      = c(0.01,3/4), ### uniform prior on this interval
                  "sep_par"  = c(1,1), ### beta prior
                  "sigma2"    = c(5,5),## inverse gamma prior with mode=5/6
                  "alpha" =  c(0,20) ## wrapped gaussian with large variance
  )  ,
  sd_prop   = list( "sigma2" = 0.1,  "rho_sp" = 0.1,  "rho_t" = 0.1,"sep_par"= 0.1),
  iter    = 7000,
  BurninThin    = c(burnin = 3000, thin = 10),
  accept_ratio = 0.234,
  adapt_param = c(start = 1, end = 1000, exp = 0.5),
  n_chains = 2 ,
  parallel = FALSE,
  n_cores = 1
)
check <- ConvCheck(mod,startit = 1 ,thin = 1)
check$Rhat ## convergence has been reached
```

```
## when plotting chains remember that alpha is a circular variable
par(mfrow = c(3,2))
coda::traceplot(check$mcmc)
par(mfrow = c(1,1))

#### move to the prediction step with WrapKrigSpTi
```

Index