Package ‘StempCens’

February 8, 2019

Type Package

Title Spatio-Temporal Estimation and Prediction for Censored/Missing Responses

Version 0.1.0

Author Katherine A. L. Valeriano, Victor H. Lachos and Larissa Avila Matos

Maintainer Larissa Avila Matos <larissa.amatos@gmail.com>

Description It estimates the parameters of a censored or missing data in spatio-temporal models using the SAEM algorithm (Delyon et al., 1999 <doi:10.1214/aos/1018031103>). This algorithm is a stochastic approximation of the widely used EM algorithm and an important tool for models in which the E-step does not have an analytic form. Besides the expressions obtained to estimate the parameters to the proposed model, we include the calculations for the observed information matrix using the method developed by Louis (1982) <https://www.jstor.org/stable/2345828>. To examine the performance of the fitted model, case-deletion measure are provided.

License GPL (>= 2)

RoxygenNote 6.1.1

Imports ssym, optimx, Matrix, sp, spTimer, mvtnorm, tmvtnorm, MCMCglmm, ggplot2, grid, distances, gridExtra

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-08 17:43:18 UTC

R topics documented:

CovarianceM *Covariance matrix for spatio-temporal model*

Description

It computes the spatio-temporal covariance matrix. For the spatial function we have 5 differents correlation matrix: exponential, gaussian, matern, spherical and power exponential correlation matrix; and for the temporal function is a correlation matrix of an autorregressive model AR(1).

Usage

```
CovarianceM(phi, rho, tau2, sigma2, distSpa, disTemp, kappa, type.S)
```

Arguments

`phi`	value of the spatial scaling parameter.
`rho`	value of the time scaling parameter.
`tau2`	value of the the nugget effect parameter.
`sigma2`	value of the the model variance.
`distSpa`	Spatial distance matrix without considering repetitions.
`disTemp`	Temporal distance matrix without considering repetitions.
`kappa`	parameter for all spatial covariance functions. In the case of exponential, gaussian and spherical function κ is equal to zero.
`type.S`	type of spatial covariance function: '`exponential`' for exponential, '`gaussian`' for gaussian, '`matern`' for matern, '`pow.exp`' for power exponential and '`spherical`' for spherical function, respectively. Default is `exponential` function.

Value

The function returns the spatio-temporal covariance matrix.

Author(s)

Katherine A. L. Valeriano, Victor H. Lachos and Larissa A. Matos

Examples

```
# Initial parameter values
beta <- c(-1,1.50); phi <- 5; rho <- 0.45; tau2 <- 0.80; sigma2 <- 2
# Simulating data
n1 <- 10   # Number of spatial locations
n2 <- 5    # Number of temporal index
set.seed(1000)
x.coord <- round(runif(n1,0,10),9)    # X coordinate
y.coord <- round(runif(n1,0,10),9)    # Y coordinate
coordenadas <- cbind(x.coord,y.coord) # Cartesian coordinates without repetitions
```

```
time <- as.matrix(seq(1,n2,1))        # Time index without repetitions
# Covariance matrix
H <- as.matrix(dist(coordenadas)) # Spatial distances
Mt <- as.matrix(dist(time))       # Temporal distances
Cov <- CovarianceM(phi,rho,tau2,sigma2,distSpa=H,disTemp=Mt,kappa=0,type.S="exponential")
```

`CrossStempCens` *Cross-Validation in spatio-temporal model with censored/missing responses*

Description

This functions performs cross-validation in spatio-temporal model with censored/missing responses, which measure the performance of the predictive model on new test data sets. The cross-validation method for assessing the model performance is Validation set approach (or data split).

Usage

```
CrossStempCens(Pred.StempCens, yObs.pre)
```

Arguments

`Pred.StempCens`	an object of class `Pred.StempCens` given as output by the `PredStempCens` function, predict the outcome of new unseen observations, the training data set.
`yObs.pre`	a vector of the observed responses, the test data.

Value

`Bias`	bias prediction error.
`Mspe`	mean squared prediction error.
`Rmspe`	root mean squared prediction error.
`Mae`	mean absolute error.

Author(s)

Katherine A. L. Valeriano, Victor H. Lachos and Larissa A. Matos

See Also

`EstStempCens`, `PredStempCens`

Examples

```
# Initial parameter values
beta <- c(-1,1.50); phi <- 5; rho <- 0.6; tau2 <- 0.80; sigma2 <- 2
# Simulating data
n1 <- 7    # Number of spatial locations
n2 <- 5    # Number of temporal index
set.seed(400)
x.coord <- round(runif(n1,0,10),9)   # X coordinate
y.coord <- round(runif(n1,0,10),9)   # Y coordinate
coordenadas <- cbind(x.coord,y.coord) # Cartesian coordinates without repetitions
coord2 <- cbind(rep(x.coord,each=n2),rep(y.coord,each=n2)) # Cartesian coordinates with repetitions
time <- as.matrix(seq(1,n2,1))   # Time index without repetitions
time2 <- as.matrix(rep(time,n1)) # Time index with repetitions
x1 <- rexp(n1*n2,2)
x2 <- rnorm(n1*n2,2,1)
x <- cbind(x1,x2)
media <- x%*%beta
# Covariance matrix
H <- as.matrix(dist(coordenadas)) # Spatial distances
Mt <- as.matrix(dist(time))       # Temporal distances
Cov <- CovarianceM(phi,rho,tau2,sigma2,distSpa=H,disTemp=Mt,kappa=0,type.S="gaussian")
# Data
require(mvtnorm)
y <- as.vector(rmvnorm(1,mean=as.vector(media),sigma=Cov))
data <- as.data.frame(cbind(coord2,time2,y,x))
names(data) <- c("x.coord","y.coord","time","yObs","x1","x2")
# Splitting the dataset
local.est <- coordenadas[c(1,2,4,5,6),]
data.est <- data[data$x.coord%in%local.est[,1]&data$y.coord%in%local.est[,2],]
data.valid <- data[data$x.coord%in%coordenadas[c(3,7),1]&data$y.coord%in%coordenadas[c(3,7),2],]
# Censored
perc <- 0.2
y <- data.est$yObs
aa=sort(y);  bb=aa[1:(perc*nrow(data.est))];  cutof<-bb[perc*nrow(data.est)]
cc=matrix(1,nrow(data.est),1)*(y<=cutof)
y[cc==1] <- cutof
data.est <- cbind(data.est[,-c(4,5,6)],y,cc,data.est[,c(5,6)])
names(data.est) <- c("x.coord","y.coord","time","yObs","censored","x1","x2")
# Estimation
y <- data.est$yObs
x <- cbind(data.est$x1,data.est$x2)
cc <- data.est$censored
time2 <- as.data.frame(data.est$time)
coord2 <- data.est[,1:2]
est_teste <- EstStempCens(y, x, cc, time2, coord2, inits.phi=3.5, inits.rho=0.5, inits.tau2=1,
                       type.Data="balanced", cens.type="left", method="nlminb", kappa=0,
                          type.S="gaussian",
                     IMatrix=TRUE, lower.lim=c(0.01,-0.99,0.01), upper.lim=c(30,0.99,10), M=20,
                          perc=0.25, MaxIter=50, pc=0.2, error = 10^-6)
# Prediction
locPre <- data.valid[,1:2]
timePre <- as.data.frame(data.valid$time)
```

```
xPre <- cbind(data.valid$x1,data.valid$x2)
pre_teste <- PredStempCens(est_teste, locPre, timePre, xPre)
class(pre_teste)
# Cross-validation
cross_teste <- CrossStempCens(pre_teste,data.valid$yObs)
cross_teste$Mspe # MSPE
```

`DiagStempCens` *Diagnostic in spatio-temporal model with censored/missing responses*

Description

Return measures and graphics for diagnostic analysis in spatio-temporal model with censored/missing responses.

Usage

```
DiagStempCens(Est.StempCens, type.diag = "individual",
  diag.plot = TRUE, ck = 3)
```

Arguments

`Est.StempCens`	an object of class `Est.StempCens` given as output by the `EstStempCens` function. In the `EstStempCens`function, `IMatrix` must be `TRUE`.
`type.diag`	type of diagnostic: '`individual`' is related when one observation is deleted, '`time`' is related when an entire time is deleted, '`location`' is related when an entire location is deleted and '`all`' the three cases ('`individual`', '`time`' and '`location`'). By default `type.diag` is `location`.
`diag.plot`	`TRUE` or `FALSE`. It indicates if diagnostic plots must be showed. By default = `TRUE`
`ck`	The value for ck considered in the benchmark value for the index plot: $mean(GD) + ck * sd(GD)$, where GD is the vector with all values of the diagnostic measure. By default ck=3.

Details

This function uses the case deletion approach to study the impact of deleting one or more observations from the dataset on the parameters estimates, using the idea of Cook (1977). The measure is defined by

$GD_i(\theta*) = (\theta * - \theta * [i])'[-Q * *(\theta|\theta*)](\theta * - \theta * [i]), i = 1, ...m,$

where $\theta*$ is the estimate of θ using the complete data, $\theta * [i]$ are the estimates obtained after deletion of the i-th observation (or group of observations) and $Q * *(\theta|\theta*)$ is the Hessian matrix.

We can eliminate an observation, an entire location or an entire time index.

Value

The function returns a list with the diagnostic measures.

If `type.diag == individual | time | location`**:** `GD` is a data.frame with the index value of the observation and the GD measure.

If `type.diag == all`**:** `GDind` is a data.frame with the index value of the observation and the GD measure for individual.

`GDtime` is a data.frame with the time index value and the GD measure for time.

`GDloc` is a data.frame with the side index value and the GD measure for location.

Author(s)

Katherine A. L. Valeriano, Victor H. Lachos and Larissa A. Matos

See Also

`EstStempCens`

Examples

```
# Initial parameter values
beta <- c(-1,1.5); phi <- 3; rho <- 0.40; tau2 <- 1; sigma2 <- 2
# Simulating data
n1 <- 8    # Number of spatial locations
n2 <- 4    # Number of temporal index
set.seed(700)
x.coord <- round(runif(n1,0,10),9)   # X coordinate
y.coord <- round(runif(n1,0,10),9)   # Y coordinate
coordenadas <- cbind(x.coord,y.coord) # Cartesian coordinates without repetitions
coord2 <- cbind(rep(x.coord,each=n2),rep(y.coord,each=n2)) # Cartesian coordinates with repetitions
time <- as.matrix(seq(1,n2,1))   # Time index without repetitions
time2 <- as.matrix(rep(time,n1)) # Time index with repetitions
x1 <- rexp(n1*n2,2)
x2 <- rnorm(n1*n2,2,1)
x <- cbind(x1,x2)
media <- x%*%beta
# Covariance matrix
H <- as.matrix(dist(coordenadas)) # Spatial distances
Mt <- as.matrix(dist(time))       # Temporal distances
Cov <- CovarianceM(phi,rho,tau2,sigma2,distSpa=H,disTemp=Mt,kappa=0,type.S="gaussian")
# Data
require(mvtnorm)
y <- as.vector(rmvnorm(1,mean=as.vector(media),sigma=Cov))
perc <- 0.1
aa=sort(y);  bb=aa[1:(perc*n1*n2)];  cutof<-bb[perc*n1*n2]
cc=matrix(1,(n1*n2),1)*(y<=cutof)
y[cc==1] <- cutof
y[17] <- abs(y[17])+2*sd(y)
# Estimation
est <- EstStempCens(y, x, cc, time2, coord2, inits.phi=2.5, inits.rho=0.5, inits.tau2=0.8,
                    type.Data="balanced", cens.type="left", method="nlminb", kappa=0,
                     type.S="gaussian",
                IMatrix=TRUE, lower.lim=c(0.01,-0.99,0.01), upper.lim=c(30,0.99,20), M=20,
                    perc=0.25, MaxIter=10, pc=0.2, error = 10^-6)
# Diagnostic
diag <- DiagStempCens(est, type.diag="time", diag.plot = TRUE, ck=1)
```

`EstStempCens` *ML estimation in spatio-temporal model with censored/missing responses*

Description

Return the maximum likelihood estimates of the unknown parameters of spatio-temporal model with censored/missing responses. The estimates are obtained using SAEM algorithm. Also, the function computes the observed information matrix using the method developed by Thomas (1982). The types of censoring considered are left, right or missing values.

Usage

```
EstStempCens(y, x, cc, tempo, coord, inits.phi, inits.rho, inits.tau2,
  tau2.fixo = FALSE, type.Data = "balanced", cens.type = "left",
  method = "nlminb", kappa = 0, type.S = "exponential",
  IMatrix = TRUE, lower.lim = c(0.01, -0.99, 0.01), upper.lim = c(30,
  0.99, 20), M = 20, perc = 0.25, MaxIter = 300, pc = 0.2,
  error = 10^-6)
```

Arguments

`y`	a vector of responses.
`x`	a matrix or vector of covariates.
`cc`	a vector of censoring indicators. For each observation: 1 if censored/missing and 0 if non-censored/non-missing.
`tempo`	a vector of time.
`coord`	a matrix of coordinates of the spatial locations.
`inits.phi`	initial value of the spatial scaling parameter.
`inits.rho`	initial value of the time scaling parameter.
`inits.tau2`	initial value of the the nugget effect parameter.
`tau2.fixo`	`TRUE` or `FALSE`. Indicate if the nugget effect (τ^2) parameter must be fixed. By default = `FALSE`.
`type.Data`	type of the data: '`balanced`' for balanced data and '`unbalanced`' for unbalanced data. By default = `balanced`.
`cens.type`	type of censoring: '`left`' for left censoring, '`right`' for rigth censoring and '`missing`' for missing response. By default = `left`.

`method`	optimization method used to estimate (ϕ, ρ and τ^2): '`optim`' for the function `optim` and '`nlminb`' for the function `nlminb`. By default = `nlminb`.
`kappa`	parameter for all spatial covariance functions. In the case of exponential, gaussian and spherical function κ is equal to zero.
`type.S`	type of spatial covariance function: '`exponential`' for exponential, '`gaussian`' for gaussian, '`matern`' for matern, '`pow.exp`' for power exponential and '`spherical`' for spherical function, respectively. Default is `exponential` function.
`IMatrix`	`TRUE` or `FALSE`. Indicate if the observed information matrix will be computed. By default = `TRUE`.
`lower.lim, upper.lim`	vectors of lower and upper bounds for the optimization method. If unspecified, the default is `c(0.01,-0.99,0.01)` for the lower bound and `c(30,0.99,20)` for the upper bound.
`M`	number of Monte Carlo samples for stochastic aproximation. By default = 20.
`perc`	percentage of burn-in on the Monte Carlo sample. By default = 0.25.
`MaxIter`	the maximum number of iterations of the SAEM algorithm. By default = 300.
`pc`	percentage of iterations of the SAEM algorithm with no memory. By default = 0.2
`error`	the convergence maximum error. By default = 10^-6.

Details

The spatio-temporal Gaussian model is giving by:

$Y(s_i,t_j) = \mu(s_i,t_j) + Z(s_i,t_j) + \epsilon(s_i,t_j),$

where, the deterministic term $\mu(s_i,t_j)$ and the stochastic terms $Z(s_i,t_j)$, $\epsilon(s_i,t_j)$ can depend on the observed spatio-temporal index for $Y(s_i,t_j)$. We assume Z is normally distributed with zero-mean and covariance matrix $\Sigma_z = \sigma^2\Omega_{\phi\rho}(s,t)$ where σ^2 is the model variance, ϕ and ρ are the spatial and time scaling parameters; $\epsilon(s_i,t_j)$ is an independent and identically distributed measurement error with $E[\epsilon(s_i,t_j)] = 0$, variance function $Var[\epsilon(s_i,t_j)] = \tau^2$ (the nugget effect) and $Cov[\epsilon(s_i,t_j),\epsilon(s_k,t_l)] = 0 \, foralls_i =!s_k$ and $t_j =!t_l$.

In particular, we define $\mu(s_i,t_j)$, the mean of the stochastic process as

$\mu(s_i,t_j) = \sum_{k=1}^{p} x_k(s_i,t_j)\beta_k,$

where $x_1(s_i,t_j),...,x_p(s_i,t_j)$ are known functions of (s_i,t_j), and $\beta_1,...,\beta_p$ are unknown parameters to be estimated. Equivalently, in matrix notation, we have our spatio-temporal linear model as follows:

$Y = X\beta + Z + \epsilon,$

$Z\, N(0,\sigma^2\Omega_{\phi\rho}),$

$\epsilon\, N(0,\tau^2 I).$

Therefore the spatio-temporal process, Y, has normal distribution with mean $E[Y] = X\beta$ and variance $\Sigma = \sigma^2\Omega_{\phi\rho} + \tau^2 I_m$. We assume that Σ is non-singular, and X has full rank. Using standard geostatistical terms, τ^2 is the nugget effect (or the measurement error variance), σ^2 the sill, and $\Omega_{\phi\rho} = [r_{ij}]$ is the mxm spatio-temporal correlation matrix with diagonal elements $r_{ii} = 1$, for $i = 1,...,m$.

The estimation process was computed via SAEM algorithm initially proposed by Deylon et. al.(1999).

Value

The function returns an object of class `Est.StempCens` which is a list given by:

data.model Returns a list with all data components given in input.

results.model A list given by:

`theta`	final estimation of $\theta = (\beta, \sigma^2, \tau^2, \phi, \rho)$.
`Theta`	estimated parameters in all iterations, $\theta = (\beta, \sigma^2, \tau^2, \phi, \rho)$.
`beta`	estimated β.
`sigma2`	estimated σ^2.
`tau2`	estimated τ^2.
`phi`	estimated ϕ.
`rho`	estimated ρ.
`PsiInv`	estimated Ψ^-1.
`Cov`	estimated Σ.
`SAEMy`	stochastic approximation of the first moment for the truncated normal distribution.
`SAEMyy`	stochastic approximation of the second moment for the truncated normal distribution.
`Hessian`	Hessian matrix, the negative of the conditional expected second derivative matrix given the observed values.
`Louis`	the observed information matrix using the Louis' method.
`loglik`	log likelihood for SAEM method.
`AIC`	Akaike information criteria.
`BIC`	Bayesian information criteria.
`AICcorr`	corrected AIC by the number of parameters.
`iteration`	number of iterations needed to convergence.

Author(s)

Katherine A. L. Valeriano, Victor H. Lachos and Larissa A. Matos

Examples

```
# Initial parameter values
beta <- c(-1,1.50); phi <- 5; rho <- 0.45; tau2 <- 0.80; sigma2 <- 1.5
# Simulating data
n1 <- 9    # Number of spatial locations
n2 <- 4    # Number of temporal index
set.seed(700)
x.coord <- round(runif(n1,0,10),9)   # X coordinate
y.coord <- round(runif(n1,0,10),9)   # Y coordinate
coordenadas <- cbind(x.coord,y.coord) # Cartesian coordinates without repetitions
coord2 <- cbind(rep(x.coord,each=n2),rep(y.coord,each=n2)) # Cartesian coordinates with repetitions
```

```
time <- as.matrix(seq(1,n2,1))      # Time index without repetitions
time2 <- as.matrix(rep(time,n1))    # Time index with repetitions
x1 <- rexp(n1*n2,2)
x2 <- rnorm(n1*n2,2,1)
x <- cbind(x1,x2)
media <- x%*%beta
# Covariance matrix
H <- as.matrix(dist(coordenadas)) # Spatial distances
Mt <- as.matrix(dist(time))       # Temporal distances
Cov <- CovarianceM(phi,rho,tau2,sigma2,distSpa=H,disTemp=Mt,kappa=0,type.S="exponential")
# Data
require(mvtnorm)
y <- as.vector(rmvnorm(1,mean=as.vector(media),sigma=Cov))
perc <- 0.2
aa=sort(y);  bb=aa[1:(perc*n1*n2)];  cutof<-bb[perc*n1*n2]
cc=matrix(1,(n1*n2),1)*(y<=cutof)
y[cc==1] <- cutof
# Estimation
est_teste <- EstStempCens(y, x, cc, time2, coord2, inits.phi=3.5, inits.rho=0.5, inits.tau2=0.7,
                        type.Data="balanced", cens.type="left", method="nlminb", kappa=0,
                           type.S="exponential",
                     IMatrix=TRUE, lower.lim=c(0.01,-0.99,0.01), upper.lim=c(30,0.99,20), M=20,
                           perc=0.25, MaxIter=25, pc=0.2, error = 10^-6)

## Not run:
# New York data
library(spTimer)
library(sp)
library(rgdal)
# Transform coordinates
station <- as.data.frame(NYdata[,2:3])
names(station) <- c("lon","lat")
coordinates(station) <- ~lon+lat
proj4string(station) <- CRS("+init=epsg:32116 +proj=longlat +datum=NAD83 +no_defs
                             +ellps=GRS80 +towgs84=0,0,0")
station2 <- spTransform(station,CRS("+proj=utm +zone=18 +north +datum=NAD83"))
station <- as.data.frame(station2)
names(station) <- c("x.Coord","y.Coord")
NYdata <- cbind(NYdata,station)
coord <- unique(station)
EstEstimation <- c(1,2,3,5,6,7,8,9,11,12,13,14,15,16,17,18,19,20,22,23,24,26,27,28)
EstEstimation <- coord[EstEstimation,]
dataEstimation <- NYdata[(NYdata[,11]%in%EstEstimation[,1])&(NYdata[,12]%in%EstEstimation[,2]),]
cc <- vector("numeric",length=nrow(dataEstimation))
cc[is.na(dataEstimation$o8hrmax)==1] <- 1
time <- rep(seq(1,62),nrow(EstEstimation))
dados <- cbind(dataEstimation,cc,time)
names(dados) <- c("s.index","Longitude","Latitude","Year","Month","Day","o8hrmax","cMAXTMP",
                  "WDSP","RH","x.Coord","y.Coord","censure","time")
# Data
y     <- dados$o8hrmax
cc    <- dados$censure
x     <- as.matrix(cbind(dados$cMAXTMP,dados$WDSP,dados$RH))
```

```
tempo <- as.data.frame(dados$time)
coord <- as.matrix(cbind(dados$x.Coord/1000,dados$y.Coord/1000)) # Coordinates in Km
# Power exponential model for the spatial covariance structure
est <- EstStempCens(y, x, cc, tempo, coord, inits.phi = 50, inits.rho = 0.30, inits.tau2 = 25,
                 type.Data="balanced", cens.type="missing", method="nlminb", kappa=0.50,
                    type.S="pow.exp",
                IMatrix=TRUE, lower.lim=c(0.01,-0.80,0.01), upper.lim=c(500,0.80,150), M=20,
                    perc=0.25, MaxIter=500, pc=0.2, error = 10^-5)
## End(Not run)
```

`PredStempCens` *Prediction in spatio-temporal model with censored/missing responses*

Description

This function performs spatio-temporal prediction in a set of new S spatial locations for fixed time points.

Usage

```
PredStempCens(Est.StempCens, locPre, timePre, xPre)
```

Arguments

`Est.StempCens`	an object of class `Est.StempCens` given as output by the `EstStempCens` function.
`locPre`	a matrix of coordinates for which the spatial prediction is performed.
`timePre`	the time point between 1 and n for which the spatial prediction is performed.
`xPre`	a matrix of covariates for which the spatial prediction is performed.

Value

The function returns an object of class `Pred.StempCens` which is a list given by:

`predValues`	predicted values.
`VarPred`	predicted covariance matrix.

Author(s)

Katherine A. L. Valeriano, Victor H. Lachos and Larissa A. Matos

See Also

`EstStempCens`

Examples

```
# Initial parameter values
beta <- c(-1,1.50); phi <- 5; rho <- 0.6; tau2 <- 0.80; sigma2 <- 2
# Simulating data
n1 <- 7   # Number of spatial locations
n2 <- 5    # Number of temporal index
set.seed(400)
x.coord <- round(runif(n1,0,10),9)   # X coordinate
y.coord <- round(runif(n1,0,10),9)   # Y coordinate
coordenadas <- cbind(x.coord,y.coord) # Cartesian coordinates without repetitions
coord2 <- cbind(rep(x.coord,each=n2),rep(y.coord,each=n2)) # Cartesian coordinates with repetitions
time <- as.matrix(seq(1,n2,1))   # Time index without repetitions
time2 <- as.matrix(rep(time,n1)) # Time index with repetitions
x1 <- rexp(n1*n2,2)
x2 <- rnorm(n1*n2,2,1)
x <- cbind(x1,x2)
media <- x%*%beta
# Covariance matrix
H <- as.matrix(dist(coordenadas)) # Spatial distances
Mt <- as.matrix(dist(time))       # Temporal distances
Cov <- CovarianceM(phi,rho,tau2,sigma2,distSpa=H,disTemp=Mt,kappa=0,type.S="gaussian")
# Data
require(mvtnorm)
y <- as.vector(rmvnorm(1,mean=as.vector(media),sigma=Cov))
data <- as.data.frame(cbind(coord2,time2,y,x))
names(data) <- c("x.coord","y.coord","time","yObs","x1","x2")
# Splitting the dataset
local.est <- coordenadas[c(1,2,4,5,6),]
data.est <- data[data$x.coord%in%local.est[,1]&data$y.coord%in%local.est[,2],]
data.valid <- data[data$x.coord%in%coordenadas[c(3,7),1]&data$y.coord%in%coordenadas[c(3,7),2],]
# Censored
perc <- 0.2
y <- data.est$yObs
aa=sort(y);  bb=aa[1:(perc*nrow(data.est))];  cutof<-bb[perc*nrow(data.est)]
cc=matrix(1,nrow(data.est),1)*(y<=cutof)
y[cc==1] <- cutof
data.est <- cbind(data.est[,-c(4,5,6)],y,cc,data.est[,c(5,6)])
names(data.est) <- c("x.coord","y.coord","time","yObs","censored","x1","x2")
# Estimation
y <- data.est$yObs
x <- cbind(data.est$x1,data.est$x2)
cc <- data.est$censored
time2 <- as.data.frame(data.est$time)
coord2 <- data.est[,1:2]
est_teste <- EstStempCens(y, x, cc, time2, coord2, inits.phi=3.5, inits.rho=0.5, inits.tau2=1,
                       type.Data="balanced", cens.type="left", method="nlminb", kappa=0,
                          type.S="gaussian",
                     IMatrix=TRUE, lower.lim=c(0.01,-0.99,0.01), upper.lim=c(30,0.99,10), M=20,
                          perc=0.25, MaxIter=50, pc=0.2, error = 10^-6)
class(est_teste)
# Prediction
locPre <- data.valid[,1:2]
```

```
timePre <- as.data.frame(data.valid$time)
xPre <- cbind(data.valid$x1,data.valid$x2)
pre_teste <- PredStempCens(est_teste, locPre, timePre, xPre)
library(ggplot2)
Model <- rep(c("y Observed","y Predicted"),each=10)
xcoord1 <- rep(seq(1:5),4)
ycoord1 <- c(data.valid$yObs,pre_teste$predValues)
data2 <- data.frame(Model,xcoord1,ycoord1)
# Station 1
fig1 <- ggplot(data=data2[c(1:5,11:15),], aes(x=xcoord1, y=ycoord1)) +
           geom_line(aes(color=Model),show.legend=FALSE) +
           labs(x="",y="",title="Station 1")
# Station 2
fig2 <- ggplot(data=data2[c(6:10,16:20),], aes(x=xcoord1, y=ycoord1)) +
           geom_line(aes(color=Model),show.legend=TRUE) +
           theme(legend.position="bottom") +
           labs(x="",y="",title="Station 2")
library(gridExtra)
grid.arrange(fig1,fig2)
```

Index