Package ‘spBayes’

December 11, 2023

Version 0.4-7

Date 2023-12-11

Title Univariate and Multivariate Spatial-Temporal Modeling

Maintainer Andrew Finley <finleya@msu.edu>

Author Andrew Finley [aut, cre],
Sudipto Banerjee [aut]

Depends R (>= 1.8.0)

Imports coda, sp, magic, Formula, Matrix

Suggests MBA

Description Fits univariate and multivariate spatio-temporal random effects models for point-referenced data using Markov chain Monte Carlo (MCMC). Details are given in Finley, Banerjee, and Gelfand (2015) <doi:10.18637/jss.v063.i13> and Finley and Banerjee <doi:10.1016/j.envsoft.2019.104608>.

License GPL (>= 2)

URL https://www.finley-lab.com

Repository CRAN

NeedsCompilation yes

Date/Publication 2023-12-11 22:50:12 UTC

R topics documented:

`adaptMetropGibbs` *Adaptive Metropolis within Gibbs algorithm*

Description

Markov chain Monte Carlo for continuous random vector using an adaptive Metropolis within Gibbs algorithm.

Usage

```
adaptMetropGibbs(ltd, starting, tuning=1, accept.rate=0.44,
                 batch = 1, batch.length=25, report=100,
                 verbose=TRUE, ...)
```

Arguments

`ltd`	an R function that evaluates the log target density of the desired equilibrium distribution of the Markov chain. First argument is the starting value vector of the Markov chain. Pass variables used in the `ltd` via the ... argument of `aMetropGibbs`.
`starting`	a real vector of parameter starting values.
`tuning`	a scalar or vector of initial Metropolis tuning values. The vector must be of `length(starting)`. If a scalar is passed then it is expanded to `length(starting)`.
`accept.rate`	a scalar or vector of target Metropolis acceptance rates. The vector must be of `length(starting)`. If a scalar is passed then it is expanded to `length(starting)`.

`batch`	the number of batches.
`batch.length`	the number of sampler iterations in each batch.
`report`	the number of batches between acceptance rate reports.
`verbose`	if `TRUE`, progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`...`	currently no additional arguments.

Value

A list with the following tags:

`p.theta.samples`	a coda object of posterior samples for the parameters.
`acceptance`	the Metropolis acceptance rate at the end of each batch.
`ltd`	`ltd`
`accept.rate`	`accept.rate`
`batch`	`batch`
`batch.length`	`batch.length`
`proc.time`	the elapsed CPU and wall time (in seconds).

Note

This function is a rework of Rosenthal (2007) with some added niceties.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`sudiptob@biostat.umn.edu`>

References

Roberts G.O. and Rosenthal J.S. (2006). Examples of Adaptive MCMC. `http://probability.ca/jeff/ftpdir/adaptex.pdf` Preprint.

Rosenthal J.S. (2007). AMCMC: An R interface for adaptive MCMC. *Computational Statistics and Data Analysis*. 51:5467-5470.

Examples

```
## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}
```

```
###########################
##Fit a spatial regression
###########################
set.seed(1)
n <- 50
x <- runif(n, 0, 100)
y <- runif(n, 0, 100)

D <- as.matrix(dist(cbind(x, y)))

phi <- 3/50
sigmasq <- 50
tausq <- 20
mu <- 150

s <- (sigmasq*exp(-phi*D))
w <-  rmvn(1, rep(0, n), s)
Y <- rmvn(1, rep(mu, n) + w, tausq*diag(n))
X <- as.matrix(rep(1, length(Y)))

##Priors
##IG sigma^2 and tau^2
a.sig <- 2
b.sig <- 100
a.tau <- 2
b.tau <- 100

##Unif phi
a.phi <- 3/100
b.phi <- 3/1

##Functions used to transform phi to continuous support.
logit <- function(theta, a, b){log((theta-a)/(b-theta))}
logit.inv <- function(z, a, b){b-(b-a)/(1+exp(z))}

##Metrop. target
target <- function(theta){

  mu.cand <- theta[1]
  sigmasq.cand <- exp(theta[2])
  tausq.cand <- exp(theta[3])
  phi.cand <- logit.inv(theta[4], a.phi, b.phi)

  Sigma <- sigmasq.cand*exp(-phi.cand*D)+tausq.cand*diag(n)
  SigmaInv <- chol2inv(chol(Sigma))
  logDetSigma <- determinant(Sigma, log=TRUE)$modulus[1]

  out <- (
          ##Priors
          -(a.sig+1)*log(sigmasq.cand) - b.sig/sigmasq.cand
          -(a.tau+1)*log(tausq.cand) - b.tau/tausq.cand
          ##Jacobians
          +log(sigmasq.cand) + log(tausq.cand)
          +log(phi.cand - a.phi) + log(b.phi -phi.cand)
          ##Likelihood
          -0.5*logDetSigma-0.5*(t(Y-X%*%mu.cand)%*%SigmaInv%*%(Y-X%*%mu.cand))
          )

  return(out)
}


##Run a couple chains
n.batch <- 500
batch.length <- 25

inits <- c(0, log(1), log(1), logit(3/10, a.phi, b.phi))
chain.1 <- adaptMetropGibbs(ltd=target, starting=inits,
                            batch=n.batch, batch.length=batch.length, report=100)

inits <- c(500, log(100), log(100), logit(3/90, a.phi, b.phi))
chain.2 <- adaptMetropGibbs(ltd=target, starting=inits,
                            batch=n.batch, batch.length=batch.length, report=100)

##Check out acceptance rate just for fun
plot(mcmc.list(mcmc(chain.1$acceptance), mcmc(chain.2$acceptance)))

##Back transform
chain.1$p.theta.samples[,2] <- exp(chain.1$p.theta.samples[,2])
chain.1$p.theta.samples[,3] <- exp(chain.1$p.theta.samples[,3])
chain.1$p.theta.samples[,4] <- 3/logit.inv(chain.1$p.theta.samples[,4], a.phi, b.phi)

chain.2$p.theta.samples[,2] <- exp(chain.2$p.theta.samples[,2])
chain.2$p.theta.samples[,3] <- exp(chain.2$p.theta.samples[,3])
chain.2$p.theta.samples[,4] <- 3/logit.inv(chain.2$p.theta.samples[,4], a.phi, b.phi)

par.names <- c("mu", "sigma.sq", "tau.sq", "effective range (-log(0.05)/phi)")
colnames(chain.1$p.theta.samples) <- par.names
colnames(chain.2$p.theta.samples) <- par.names

##Discard burn.in and plot and do some convergence diagnostics
chains <- mcmc.list(mcmc(chain.1$p.theta.samples), mcmc(chain.2$p.theta.samples))
plot(window(chains, start=as.integer(0.5*n.batch*batch.length)))

gelman.diag(chains)

##########################
##Example of fitting a
##a non-linear model
##########################
##Example of fitting a non-linear model
set.seed(1)

########################################################
##Simulate some data.
########################################################
```

```
a <- 0.1 #-Inf < a < Inf
b <- 0.1 #b > 0
c <- 0.2 #c > 0
tau.sq <- 0.1 #tau.sq > 0

fn <- function(a,b,c,x){
  a+b*exp(x/c)
}

n <- 200
x <- seq(0,1,0.01)
y <- rnorm(length(x), fn(a,b,c,x), sqrt(tau.sq))

##check out your data
plot(x, y)

#########################################################
##The log target density
#########################################################
##Define the log target density used in the Metrop.
ltd <- function(theta){

  ##extract and transform as needed
  a <- theta[1]
  b <- exp(theta[2])
  c <- exp(theta[3])
  tau.sq <- exp(theta[4])

  y.hat <- fn(a, b, c, x)

  ##likelihood
  logl <- sum(dnorm(y, y.hat, sqrt(tau.sq), log=TRUE))

  ##priors IG on tau.sq and normal on a and transformed b, c, d
  logl <- (logl
           -(IG.a+1)*log(tau.sq)-IG.b/tau.sq
           +sum(dnorm(theta[1:3], N.mu, N.v, log=TRUE))
           )

  ##Jacobian adjustment for tau.sq
  logl <- logl+log(tau.sq)

  return(logl)
}

#########################################################
##The rest
#########################################################

##Priors
IG.a <- 2
IG.b <- 0.01
```

```
N.mu <- 0
N.v <- 10

theta.tuning <- c(0.01, 0.01, 0.005, 0.01)

##Run three chains with different starting values
n.batch <- 1000
batch.length <- 25

theta.starting <- c(0, log(0.01), log(0.6), log(0.01))
run.1 <- adaptMetropGibbs(ltd=ltd, starting=theta.starting, tuning=theta.tuning,
                          batch=n.batch, batch.length=batch.length, report=100)

theta.starting <- c(1.5, log(0.05), log(0.5), log(0.05))
run.2 <- adaptMetropGibbs(ltd=ltd, starting=theta.starting, tuning=theta.tuning,
                          batch=n.batch, batch.length=batch.length, report=100)

theta.starting <- c(-1.5, log(0.1), log(0.4), log(0.1))
run.3 <- adaptMetropGibbs(ltd=ltd, starting=theta.starting, tuning=theta.tuning,
                          batch=n.batch, batch.length=batch.length, report=100)

##Back transform
samples.1 <- cbind(run.1$p.theta.samples[,1], exp(run.1$p.theta.samples[,2:4]))
samples.2 <- cbind(run.2$p.theta.samples[,1], exp(run.2$p.theta.samples[,2:4]))
samples.3 <- cbind(run.3$p.theta.samples[,1], exp(run.3$p.theta.samples[,2:4]))

samples <- mcmc.list(mcmc(samples.1), mcmc(samples.2), mcmc(samples.3))

##Summary
plot(samples, density=FALSE)
gelman.plot(samples)

burn.in <- 5000

fn.pred <- function(theta,x){
  a <- theta[1]
  b <- theta[2]
  c <- theta[3]
  tau.sq <- theta[4]

  rnorm(length(x), fn(a,b,c,x), sqrt(tau.sq))
}

post.curves <- t(apply(samples.1[burn.in:nrow(samples.1),], 1, fn.pred, x))

post.curves.quants <- summary(mcmc(post.curves))$quantiles

plot(x, y, pch=19, xlab="x", ylab="f(x)")
lines(x, post.curves.quants[,1], lty="dashed", col="blue")
lines(x, post.curves.quants[,3])
lines(x, post.curves.quants[,5], lty="dashed", col="blue")
```

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

---

bayesGeostatExact *Simple Bayesian spatial linear model with fixed semivariogram parameters*

Description

Given a observation coordinates and fixed semivariogram parameters the `bayesGeostatExact` function fits a simple Bayesian spatial linear model.

Usage

```
bayesGeostatExact(formula, data = parent.frame(), n.samples,
                  beta.prior.mean, beta.prior.precision,
                  coords, cov.model="exponential", phi, nu, alpha,
                  sigma.sq.prior.shape, sigma.sq.prior.rate,
                  sp.effects=TRUE, verbose=TRUE, ...)
```

Arguments

`formula`	for a univariate model, this is a symbolic description of the regression model to be fit. See example below.
`data`	an optional data frame containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `spLM` is called.
`n.samples`	the number of posterior samples to collect.
`beta.prior.mean`	β multivariate normal mean vector hyperprior.
`beta.prior.precision`	β multivariate normal precision matrix hyperprior.
`coords`	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing).
`cov.model`	a quoted key word that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`phi`	the fixed value of the spatial decay.
`nu`	if `cov.model` is `"matern"` then the fixed value of the spatial process smoothness must be specified.
`alpha`	the fixed value of the ratio between the nugget τ^2 and partial-sill σ^2 parameters from the specified `cov.model`.
`sigma.sq.prior.shape`	σ^2 (i.e., partial-sill) inverse-Gamma shape hyperprior.

`sigma.sq.prior.rate`	σ^2 (i.e., partial-sill) inverse-Gamma 1/scale hyperprior.
`sp.effects`	a logical value indicating if spatial random effects should be recovered.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`...`	currently no additional arguments.

Value

An object of class `bayesGeostatExact`, which is a list with the following tags:

`p.samples`	a `coda` object of posterior samples for the defined parameters.
`sp.effects`	a matrix that holds samples from the posterior distribution of the spatial random effects. The rows of this matrix correspond to the n point observations and the columns are the posterior samples.
`args`	a list with the initial function arguments.

Author(s)

Sudipto Banerjee <`sudiptob@biostat.umn.edu`>,
Andrew O. Finley <`finleya@msu.edu`>

Examples

```
## Not run:

data(FBC07.dat)
Y <- FBC07.dat[1:150,"Y.2"]
coords <- as.matrix(FBC07.dat[1:150,c("coord.X", "coord.Y")])

n.samples <- 500
n = length(Y)
p = 1

phi <- 0.15
nu <- 0.5

beta.prior.mean <- as.matrix(rep(0, times=p))
beta.prior.precision <- matrix(0, nrow=p, ncol=p)

alpha <- 5/5

sigma.sq.prior.shape <- 2.0
sigma.sq.prior.rate <- 5.0

##############################
##Simple linear model with
##the default exponential
##spatial decay function
##############################
```

```
set.seed(1)
m.1 <- bayesGeostatExact(Y~1, n.samples=n.samples,
                          beta.prior.mean=beta.prior.mean,
                          beta.prior.precision=beta.prior.precision,
                          coords=coords, phi=phi, alpha=alpha,
                          sigma.sq.prior.shape=sigma.sq.prior.shape,
                          sigma.sq.prior.rate=sigma.sq.prior.rate)



print(summary(m.1$p.samples))

##Requires MBA package to
##make surfaces
library(MBA)
par(mfrow=c(1,2))
obs.surf <-
  mba.surf(cbind(coords, Y), no.X=100, no.Y=100, extend=T)$xyz.est
image(obs.surf, xaxs = "r", yaxs = "r", main="Observed response")
points(coords)
contour(obs.surf, add=T)

w.hat <- rowMeans(m.1$sp.effects)
w.surf <-
  mba.surf(cbind(coords, w.hat), no.X=100, no.Y=100, extend=T)$xyz.est
image(w.surf, xaxs = "r", yaxs = "r", main="Estimated random effects")
points(coords)
contour(w.surf, add=T)


##############################
##Simple linear model with
##the matern spatial decay
##function. Note, nu=0.5 so
##should produce the same
##estimates as m.1
##############################
set.seed(1)
m.2 <- bayesGeostatExact(Y~1, n.samples=n.samples,
                          beta.prior.mean=beta.prior.mean,
                          beta.prior.precision=beta.prior.precision,
                          coords=coords, cov.model="matern",
                          phi=phi, nu=nu, alpha=alpha,
                          sigma.sq.prior.shape=sigma.sq.prior.shape,
                          sigma.sq.prior.rate=sigma.sq.prior.rate)

print(summary(m.2$p.samples))

##############################
##This time with the
##spherical just for fun
##############################
m.3 <- bayesGeostatExact(Y~1, n.samples=n.samples,
                          beta.prior.mean=beta.prior.mean,
                          beta.prior.precision=beta.prior.precision,
                          coords=coords, cov.model="spherical",
                          phi=phi, alpha=alpha,
                          sigma.sq.prior.shape=sigma.sq.prior.shape,
                          sigma.sq.prior.rate=sigma.sq.prior.rate)

print(summary(m.3$p.samples))

##############################
##Another example but this
##time with covariates
##############################
data(FORMGMT.dat)

n = nrow(FORMGMT.dat)
p = 5 ##an intercept an four covariates

n.samples <- 50

phi <- 0.0012

coords <- cbind(FORMGMT.dat$Longi, FORMGMT.dat$Lat)
coords <- coords*(pi/180)*6378

beta.prior.mean <- rep(0, times=p)
beta.prior.precision <- matrix(0, nrow=p, ncol=p)

alpha <- 1/1.5

sigma.sq.prior.shape <- 2.0
sigma.sq.prior.rate <- 10.0

m.4 <-
  bayesGeostatExact(Y~X1+X2+X3+X4, data=FORMGMT.dat, n.samples=n.samples,
                    beta.prior.mean=beta.prior.mean,
                    beta.prior.precision=beta.prior.precision,
                    coords=coords, phi=phi, alpha=alpha,
                    sigma.sq.prior.shape=sigma.sq.prior.shape,
                    sigma.sq.prior.rate=sigma.sq.prior.rate)

print(summary(m.4$p.samples))



##Requires MBA package to
##make surfaces
library(MBA)
par(mfrow=c(1,2))
obs.surf <-
  mba.surf(cbind(coords, resid(lm(Y~X1+X2+X3+X4, data=FORMGMT.dat))),
                no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(obs.surf, xaxs = "r", yaxs = "r", main="Observed response")
```

```
points(coords)
contour(obs.surf, add=T)

w.hat <- rowMeans(m.4$sp.effects)
w.surf <-
  mba.surf(cbind(coords, w.hat), no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(w.surf, xaxs = "r", yaxs = "r", main="Estimated random effects")
contour(w.surf, add=T)
points(coords, pch=1, cex=1)



## End(Not run)
```

`bayesLMConjugate` *Simple Bayesian linear model via the Normal/inverse-Gamma conjugate*

Description

Given an `lm` object, the `bayesLMConjugate` function fits a simple Bayesian linear model with Normal and inverse-Gamma priors.

Usage

```
bayesLMConjugate(formula, data = parent.frame(), n.samples,
                     beta.prior.mean, beta.prior.precision,
                     prior.shape, prior.rate, ...)
```

Arguments

`formula`	for a univariate model, this is a symbolic description of the regression model to be fit. See example below.
`data`	an optional data frame containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `spLM` is called.
`n.samples`	the number of posterior samples to collect.
`beta.prior.mean`	β multivariate normal mean vector hyperprior.
`beta.prior.precision`	β multivariate normal precision matrix hyperprior.
`prior.shape`	τ^2 inverse-Gamma shape hyperprior.
`prior.rate`	τ^2 inverse-Gamma 1/scale hyperprior.
`...`	currently no additional arguments.

Value

An object of class `bayesLMConjugate`, which is a list with at least the following tag:

`p.beta.tauSq.samples`
: a coda object of posterior samples for the defined parameters.

Author(s)

Sudipto Banerjee <`sudiptob@biostat.umn.edu`>,
Andrew O. Finley <`finleya@msu.edu`>

Examples

```
## Not run:

data(FORMGMT.dat)

n <- nrow(FORMGMT.dat)
p <- 7 ##an intercept and six covariates

n.samples <- 500

## Below we demonstrate the conjugate function in the special case
## with improper priors. The results are the same as for the above,
## up to MC error.
beta.prior.mean <- rep(0, times=p)
beta.prior.precision <- matrix(0, nrow=p, ncol=p)

prior.shape <- -p/2
prior.rate <- 0

m.1 <-
  bayesLMConjugate(Y ~ X1+X2+X3+X4+X5+X6, data = FORMGMT.dat,
                   n.samples, beta.prior.mean,
                   beta.prior.precision,
                   prior.shape, prior.rate)

summary(m.1$p.beta.tauSq.samples)

## End(Not run)
```

`bayesLMRef` *Simple Bayesian linear model with non-informative priors*

Description

Given a `lm` object, the `bayesLMRef` function fits a simple Bayesian linear model with reference (non-informative) priors.

Usage

```
bayesLMRef(lm.obj, n.samples, ...)
```

Arguments

`lm.obj`	an object returned by `lm`.
`n.samples`	the number of posterior samples to collect.
`...`	currently no additional arguments.

Details

See page 355 in Gelman et al. (2004).

Value

An object of class `bayesLMRef`, which is a list with at least the following tag:

`p.beta.tauSq.samples`	a coda object of posterior samples for the defined parameters.

Author(s)

Sudipto Banerjee <`sudiptob@biostat.umn.edu`>,
Andrew O. Finley <`finleya@msu.edu`>

References

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (2004). Bayesian Data Analysis. 2nd ed. Boca Raton, FL: Chapman and Hall/CRC Press.

Examples

```
## Not run:
set.seed(1)

n <- 100
X <- as.matrix(cbind(1, rnorm(n)))
B <- as.matrix(c(1,5))
tau.sq <- 0.1
y <- rnorm(n, X%*%B, sqrt(tau.sq))

lm.obj <- lm(y ~ X-1)

summary(lm.obj)

##Now with bayesLMRef
n.samples <- 500

m.1 <- bayesLMRef(lm.obj, n.samples)

summary(m.1$p.beta.tauSq.samples)
```

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

`BEF.dat` *Bartlett Experimental Forest inventory data*

Description

Data generated in long-term research studies on the Bartlett Experimental Forest, Bartlett, NH funded by the U.S. Department of Agriculture, Forest Service, Northeastern Research Station.

This dataset holds 1991 and 2002 forest inventory data for 437 points on the BEF.dat. Variables include species specific basal area and biomass; inventory plot coordinates; slope; elevation; and tasseled cap brightness (TC1), greenness (TC2), and wetness (TC3) components from spring, summer, and fall 2002 Landsat images.

Species specific basal area and biomass are recorded as a fraction of totals.

Usage

```
data(BEF.dat)
```

Format

A data frame containing 437 rows and 208 columns.

Source

BEF.dat inventory data provided by:

Marie-Louise Smith USDA Forest Service Northeastern Research Station <`marielouisesmith@fs.fed.us`>

Additional variables provided by:

Andrew Lister USDA Forest Service Northeastern Research Station <`alister@fs.fed.us`>

`FBC07.dat` *Synthetic multivariate data with spatial and non-spatial variance structures*

Description

The synthetic dataset describes a stationary and isotropic bivariate process. Please refer to the vignette Section 4.2 for specifics.

Usage

```
data(FBC07.dat)
```

Format

A data frame of 250 rows and 4 columns. Columns 1 and 2 are coordinates and columns 3 and 4 are response variables.

Source

Finley A.O., S. Banerjee, and B.P. Carlin (2007) spBayes: R package for Univariate and Multivariate Hierarchical Point-referenced Spatial Models. Journal of Statistical Software.

`FORMGMT.dat` *Data used for illustrations*

Description

Data used for illustrations.

Usage

```
data(FORMGMT.dat)
```

`iDist` *Euclidean distance matrix*

Description

Computes the inter-site Euclidean distance matrix for one or two sets of points.

Usage

```
iDist(coords.1, coords.2, ...)
```

Arguments

`coords.1`	an $n \times p$ matrix with each row corresponding to a point in p dimensional space.
`coords.2`	an $m \times p$ matrix with each row corresponding to a point in p dimensional space. If this is missing then `coords.1` is used.
`...`	currently no additional arguments.

Value

The $n \times n$ or $n \times m$ inter-site Euclidean distance matrix.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`sudiptob@biostat.umn.edu`>,

Examples

```
## Not run:
n <- 10
p1 <- cbind(runif(n),runif(n))

m <- 5
p2 <- cbind(runif(m),runif(m))

D <- iDist(p1, p2)

## End(Not run)
```

`mkMvX` *Make a multivariate design matrix*

Description

Given q univariate design matrices, the function `mkMvX` creates a multivariate design matrix suitable for use in `spPredict`.

Usage

```
mkMvX(X)
```

Arguments

`X`	a list of q univariate design matrices. The matrices must have the same number of rows (i.e., observations) but may have different number of columns (i.e., regressors).

Value

A multivariate design matrix suitable for use in `spPredict`.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`sudiptob@biostat.umn.edu`>.

See Also

`spPredict`

Examples

```
## Not run:
##Define some univariate model design matrices
##with intercepts.
X.1 <- cbind(rep(1, 10), matrix(rnorm(50), nrow=10))
X.2 <- cbind(rep(1, 10), matrix(rnorm(20), nrow=10))
X.3 <- cbind(rep(1, 10), matrix(rnorm(30), nrow=10))

##Make a multivariate design matrix suitable
##for use in spPredict.
X.mv <- mkMvX(list(X.1, X.2, X.3))

## End(Not run)
```

`mkSpCov` *Function for calculating univariate and multivariate covariance matrices*

Description

The function `mkSpCov` calculates a spatial covariance matrix given spatial locations and spatial covariance parameters.

Usage

```
mkSpCov(coords, K, Psi, theta, cov.model)
```

Arguments

`coords`	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing).
`K`	the $q \times q$ spatial cross-covariance matrix. For a univariate model this corresponds to the partial sill, σ^2.
`Psi`	the $q \times q$ non-spatial covariance matrix. For a univariate model this corresponds to the nugget, τ^2.
`theta`	a vector of q spatial decay parameters. If `cov.model` is `"matern"` then `theta` is a vector of length $2 \times q$ with the spatial decay parameters in the first q elements and the spatial smoothness parameters in the last q elements.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.

Details

Covariance functions return the covariance $C(h)$ between a pair locations separated by distance h. The covariance function can be written as a product of a variance parameter σ^2 and a positive definite *correlation function* $\rho(h)$: $C(h) = \sigma^2\rho(h)$, see, e.g., Banerjee et al. (2004) p. 27 for more details. The expressions of the correlations functions available in **spBayes** are given below. More will be added upon request.

For all correlations functions, ϕ is the spatial *decay* parameter. Some of the correlation functions will have an extra parameter ν, the *smoothness* parameter. $K_\nu(x)$ denotes the modified Bessel function of the third kind of order ν. See documentation of the function `besselK` for further details. The following functions are valid for $\phi > 0$ and $\nu > 0$, unless stated otherwise.

gaussian

$$\rho(h) = \exp[-(\phi h)^2]$$

exponential

$$\rho(h) = \exp(-\phi h)$$

matern

$$\rho(h) = \frac{1}{2^{\nu-1}\Gamma(\nu)}(\phi h)^\nu K_\nu(\phi h)$$

spherical

$$\rho(h) = \begin{cases} 1 - 1.5\phi h + 0.5(\phi h)^3 \text{ , if } h < \frac{1}{\phi} \\ 0 \text{ , otherwise} \end{cases}$$

Value

`C`	the $nq \times nq$ spatial covariance matrix.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`baner009@umn.edu`>

Examples

```
## Not run:
##A bivariate spatial covariance matrix

n <- 2 ##number of locations
q <- 2 ##number of responses at each location
nltr <- q*(q+1)/2 ##number of triangular elements in the cross-covariance matrix

coords <- cbind(runif(n,0,1), runif(n,0,1))

##spatial decay parameters
theta <- rep(6,q)
```

```
A <- matrix(0,q,q)
A[lower.tri(A,TRUE)] <- rnorm(nltr, 5, 1)
K <- A%*%t(A)

Psi <- diag(1,q)

C <- mkSpCov(coords, K, Psi, theta, cov.model="exponential")

## End(Not run)
```

NETemp.dat *Monthly weather station temperature data across the Northeastern US*

Description

Monthly temperature data (Celsius) recorded across the Northeastern US starting in January 2000. Station UTM coordinates and elevation are also included.

Usage

```
data(NETemp.dat)
```

Format

A data frame containing 356 rows (weather stations) and 132 columns.

NYOzone.dat *Observations of ozone concentration levels.*

Description

These data and subsequent description are drawn from the **spTimer** package (version 0.7). This data set contains values of daily 8-hour maximum average ozone concentrations (ppb; O3.8HRMAX), maximum temperature (degree Celsius; cMAXTMP), wind speed (knots; WDSP), and relative humidity (RM), obtained from 28 monitoring sites in New York, USA, between July 1 and August 31 in 2006. Each row represents a station and columns hold consecutive daily values.

Usage

```
data(NYOzone.dat)
```

Format

Columns for NYdata:

- 1st col = Longitude
- 2nd col = Latitude
- 3rd col = X coordinates in UTM projection
- 4th col = Y coordinates in UTM projection
- 5th col = Ozone July 1 (O3.8HRMAX.1)
- 6th col = Ozone July 2 (O3.8HRMAX.2)
- ...
- 66th col = Ozone August 31 (O3.8HRMAX.62)
- remaining columns organize cMAXTMP, WDSP, and RH identical to the 62 O3.8HRMAX measurements

References

spTimer Bakar, K.S. and S.K. Sahu. `http://www.southampton.ac.uk/~sks/research/papers/spTimeRpaper.pdf`

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

`PM10.dat` *Observed and modeled PM10 concentrations across Europe*

Description

The `PM10.dat` data frame is a subset of data analyzed in Hamm et al. (2015) and Datta et al. (2016). Data comprise April 6, 2010 square root transformed PM10 measurements across central Europe with corresponding output from the LOTOS-EUROS Schaap et al. (2008) chemistry transport model (CTM). CTM data may differ slightly from that considered in the studies noted above due to LOTOS-EUROS CTM code updates. A `NA` value is given at CTM output locations were PM10 is not observed. Point coordinates are in "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=km +no_defs".

Usage

```
data(PM10.dat)
```

Format

Columns for PM10.dat:

- x.coord = x coordinate (see projection information in the description)
- y.coord = y coordinate (see projection information in the description)
- pm10.obs = square root transformed PM10 measurements at monitoring stations (`NA` means there is not a station at the given location)
- pm10.ctm = square root transformed PM10 from CTM

References

Datta A., S. Banerjee, A.O. Finley, N. Hamm, and M. Schaap (2016). Nonseparable dynamic nearest neighbor Gaussian process models for large spatio-temporal data with an application to particulate matter analysis. *Annals of Applied Statistics*, 10(3), 1286–1316. ISSN 1932-6157. doi:10.1214/16-AOAS931.

Hamm N. A.O. Finley, M. Schaap, A. Stein (2015). A Spatially Varying Coefficient Model for Mapping PM10 Air Quality at the European scale. *Atmospheric Environment*, 102, 393–405.

Schaap M., R.M.A Timmermans, M. Roemer, G.A.C. Boersen, P. Builtjes, F. Sauter, G. Velders, J. Beck (2008). The LOTOS-EUROS Model: Description, Validation and Latest Developments. *International Journal of Environment and Pollution*, 32(2), 270–290.

`PM10.poly` *European countries used in PM10.dat*

Description

European countries corresponding to `PM10.dat` locations and used in Hamm et al. (2015) and Datta et al. (2016). Polygon projection is "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=km +no_defs".

Usage

```
data(PM10.poly)
```

Format

List of polygons. See example below to convert to a `SpatialPolygons` object.

References

Datta A., S. Banerjee, A.O. Finley, N. Hamm, and M. Schaap (2016). Nonseparable dynamic nearest neighbor Gaussian process models for large spatio-temporal data with an application to particulate matter analysis. *Annals of Applied Statistics*, 10(3), 1286–1316. ISSN 1932-6157. doi:10.1214/16-AOAS931.

Hamm N. A.O. Finley, M. Schaap, A. Stein (2015). A Spatially Varying Coefficient Model for Mapping PM10 Air Quality at the European scale. *Atmospheric Environment*, 102, 393–405.

Examples

```
## Not run:

library(sp)

prj <- "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=km +no_defs"

pm10.poly <- SpatialPolygons(PM10.poly, pO = 1:length(PM10.poly), proj4string=CRS(prj))
```

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

`pointsInPoly` *Finds points in a polygon*

Description

Given a polygon and a set of points this function returns the subset of points that are within the polygon.

Usage

```
pointsInPoly(poly, points, ...)
```

Arguments

`poly`	an $n \times 2$ matrix of polygon vertices. Matrix columns correspond to vertices' x and y coordinates, respectively.
`points`	an $m \times 2$ matrix of points. Matrix columns correspond to points' x and y coordinates, respectively.
`...`	currently no additional arguments.

Details

It is assumed that the polygon is to be closed by joining the last vertex to the first vertex.

Value

If points are found with the polygon, then a vector is returned with elements corresponding to the row indices of `points`, otherwise `NA` is returned.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`sudiptob@biostat.umn.edu`>,

Examples

```
## Not run:
##Example 1
points <- cbind(runif(1000, 0, 10),runif(1000, 0, 10))

poly <- cbind(c(1:9,8:1), c(1,2*(5:3),2,-1,17,9,8,2:9))
```

```
point.indx <- pointsInPoly(poly, points)

plot(points, pch=19, cex=0.5, xlab="x", ylab="y", col="red")
points(points[point.indx,], pch=19, cex=0.5, col="blue")
polygon(poly)

##Example 2
##a function to partition the domain
tiles <- function(points, x.cnt, y.cnt, tol = 1.0e-10){

  x.min <- min(points[,1])-tol
  x.max <- max(points[,1])+tol
  y.min <- min(points[,2])-tol
  y.max <- max(points[,2])+tol

  x.cnt <- x.cnt+1
  y.cnt <- y.cnt+1

  x <- seq(x.min, x.max, length.out=x.cnt)
  y <- seq(y.min, y.max, length.out=y.cnt)

  tile.list <- vector("list", (length(y)-1)*(length(x)-1))

  l <- 1
  for(i in 1:(length(y)-1)){
    for(j in 1:(length(x)-1)){
      tile.list[[l]] <- rbind(c(x[j], y[i]),
                              c(x[j+1], y[i]),
                              c(x[j+1], y[i+1]),
                              c(x[j], y[i+1]))
       l <- l+1
    }

  }

  tile.list
}

n <- 1000
points <- cbind(runif(n, 0, 10), runif(n, 0, 10))

grd <- tiles(points, x.cnt=10, y.cnt=10)

plot(points, pch=19, cex=0.5, xlab="x", ylab="y")

sum.points <- 0
for(i in 1:length(grd)){
  polygon(grd[[i]], border="red")

  point.indx <- pointsInPoly(grd[[i]], points)

  if(!is.na(point.indx[1])){
    sum.points <- length(point.indx)+sum.points
    text(mean(grd[[i]][,1]), mean(grd[[i]][,2]), length(point.indx), col="red")
  }
}
sum.points


## End(Not run)
```

`spDiag` *Model fit diagnostics*

Description

The function `spDiag` calculates DIC, GP, GRS, and associated statistics given a `spLM`, `spMvLM`, `spGLM`, `spMvGLM`, `spMvGLM`, or `spSVC` object.

Usage

```
spDiag(sp.obj, start=1, end, thin=1, verbose=TRUE, n.report=100, ...)
```

Arguments

`sp.obj`	an object returned by `spLM`, `spMvLM`, `spGLM`, `spMvGLM`. For `spSVC`, `sp.obj` is an object from `spRecover`.
`start`	specifies the first sample included in the computation. The `start`, `end`, and `thin` arguments only apply to `spGLM` or `spMvGLM` objects. Sub-sampling for `spLM` and `spMvLM` is controlled using `spRecover` which must be called prior to `spDiag`.
`end`	specifies the last sample included in the computation. The default is to use all posterior samples in `sp.obj`. See `start` argument description.
`thin`	a sample thinning factor. The default of 1 considers all samples between `start` and `end`. For example, if `thin = 10` then 1 in 10 samples are considered between `start` and `end`.
`verbose`	if `TRUE` calculation progress is printed to the screen; otherwise, nothing is printed to the screen.
`n.report`	the interval to report progress.
`...`	currently no additional arguments.

Value

A list with some of the following tags:

`DIC`	a matrix holding DIC and associated statistics, see Banerjee et al. (2004) for details.
`GP`	a matrix holding GP and associated statistics, see Gelfand and Ghosh (1998) for details.
`GRS`	a scoring rule, see Equation 27 in Gneiting and Raftery (2007) for details.

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Sudipto Banerjee <sudipto@ucla.edu>

References

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton,Fla.

Finley, A.O. and S. Banerjee (2019) Efficient implementation of spatially-varying coefficients models.

Gelfand A.E. and Ghosh, S.K. (1998). Model choice: a minimum posterior predictive loss approach. *Biometrika*. 85:1-11.

Gneiting, T. and Raftery, A.E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*. 102:359-378.

Examples

```
## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

n <- 100
coords <- cbind(runif(n,0,1), runif(n,0,1))
X <- as.matrix(cbind(1, rnorm(n)))

B <- as.matrix(c(1,5))
p <- length(B)

sigma.sq <- 2
tau.sq <- 0.1
phi <- 3/0.5

D <- as.matrix(dist(coords))
R <- exp(-phi*D)
w <- rmvn(1, rep(0,n), sigma.sq*R)
y <- rnorm(n, X%*%B + w, sqrt(tau.sq))

n.samples <- 1000

starting <- list("phi"=3/0.5, "sigma.sq"=50, "tau.sq"=1)

tuning <- list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)
```

```
##too restrictive of prior on beta
priors.1 <- list("beta.Norm"=list(rep(0,p), diag(1,p)),
                 "phi.Unif"=c(3/1, 3/0.1), "sigma.sq.IG"=c(2, 2),
                 "tau.sq.IG"=c(2, 0.1))

##more reasonable prior for beta
priors.2 <- list("beta.Norm"=list(rep(0,p), diag(1000,p)),
                 "phi.Unif"=c(3/1, 3/0.1), "sigma.sq.IG"=c(2, 2),
                 "tau.sq.IG"=c(2, 0.1))

cov.model <- "exponential"

n.report <- 500
verbose <- TRUE

m.1 <- spLM(y~X-1, coords=coords, starting=starting,
            tuning=tuning, priors=priors.1, cov.model=cov.model,
            n.samples=n.samples, verbose=verbose, n.report=n.report)

m.2 <- spLM(y~X-1, coords=coords, starting=starting,
            tuning=tuning, priors=priors.2, cov.model=cov.model,
            n.samples=n.samples, verbose=verbose, n.report=n.report)

##non-spatial model
m.3 <- spLM(y~X-1, n.samples=n.samples, verbose=verbose, n.report=n.report)

burn.in <- 0.5*n.samples

##recover beta and spatial random effects
m.1 <- spRecover(m.1, start=burn.in, verbose=FALSE)
m.2 <- spRecover(m.2, start=burn.in, verbose=FALSE)

##lower is better for DIC, GPD, and GRS
print(spDiag(m.1))
print(spDiag(m.2))
print(spDiag(m.3))

## End(Not run)
```

spDynLM *Function for fitting univariate Bayesian dynamic space-time regression models*

Description

The function `spDynLM` fits Gaussian univariate Bayesian dynamic space-time regression models for settings where space is viewed as continuous but time is taken to be discrete.

Usage

```
spDynLM(formula, data = parent.frame(), coords, knots,
      starting, tuning, priors, cov.model, get.fitted=FALSE,
      n.samples, verbose=TRUE, n.report=100, ...)
```

Arguments

`formula`	a list of N_t symbolic regression models to be fit. Each model represents a time step. See example below.
`data`	an optional data frame containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `spDynLM` is called.
`coords`	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing).
`starting`	a list with each tag corresponding to a parameter name. Valid tags are `beta`, `sigma.sq`, `tau.sq`, `phi`, `nu`, and `sigma.eta`. The value portion of each tag is the parameter's starting value.
`knots`	either a $m \times 2$ matrix of the *predictive process* knot coordinates in R^2 (e.g., easting and northing) or a vector of length two or three with the first and second elements recording the number of columns and rows in the desired knot grid. The third, optional, element sets the offset of the outermost knots from the extent of the `coords`.
`tuning`	a list with each tag corresponding to a parameter name. Valid tags are `phi` and `nu`. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution.
`priors`	a list with tags `beta.0.norm`, `sigma.sq.ig`, `tau.sq.ig`, `phi.unif`, `nu.unif`, and `sigma.eta.iw`. Variance parameters, `simga.sq` and `tau.sq`, are assumed to follow an inverse-Gamma distribution, whereas the spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The `beta.0.norm` is a multivariate Normal distribution with hyperparameters passed as a list of length two with the first and second elements corresponding to the mean vector and positive definite covariance matrix, respectively. The hyperparameters of the inverse-Wishart, `sigma.eta.iw`, are passed as a list of length two, with the first and second elements corresponding to the *df* and $p \times p$ *scale* matrix, respectively. The inverse-Gamma hyperparameters are passed in a list with two vectors that hold the *shape* and *scale*, respectively. The Uniform hyperparameters are passed in a list with two vectors that hold the lower and upper support values, respectively.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`get.fitted`	if `TRUE`, posterior predicted and fitted values are collected. Note, posterior predicted samples are only collected for those $y_t(s)$ that are `NA`.
`n.samples`	the number of MCMC iterations.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.

`n.report`	the interval to report Metropolis sampler acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

Suppose, $y_t(s)$ denotes the observation at location s and time t. We model $y_t(s)$ through a *measurement equation* that provides a regression specification with a space-time varying intercept and serially and spatially uncorrelated zero-centered Gaussian disturbances as measurement error $\epsilon_t(s)$. Next a *transition equation* introduces a $p \times 1$ coefficient vector, say β_t, which is a purely temporal component (i.e., time-varying regression parameters), and a spatio-temporal component $u_t(s)$. Both these are generated through transition equations, capturing their Markovian dependence in time. While the transition equation of the purely temporal component is akin to usual state-space modeling, the spatio-temporal component is generated using Gaussian spatial processes. The overall model is written as

$$y_t(s) = x_t(s)'\beta_t + u_t(s) + \epsilon_t(s), t = 1, 2, \ldots, N_t$$

$$\epsilon_t(s) \sim N(0, \tau_t^2)$$

$$\beta_t = \beta_{t-1} + \eta_t; \eta_t \sim N(0, \Sigma_\eta)$$

$$u_t(s) = u_{t-1}(s) + w_t(s); w_t(s) \sim GP(0, C_t(\cdot, \theta_t))$$

Here $x_t(s)$ is a $p \times 1$ vector of predictors and β_t is a $p \times 1$ vector of coefficients. In addition to an intercept, $x_t(s)$ can include location specific variables useful for explaining the variability in $y_t(s)$. The $GP(0, C_t(\cdot, \theta_t))$ denotes a spatial Gaussian process with covariance function $C_t(\cdot; \theta_t)$. We specify $C_t(s_1, s_2; \theta_t) = \sigma_t^2 \rho(s_1, s_2; \phi_t)$, where $\theta_t = \{\sigma_t^2, \phi_t, \nu_t\}$ and $\rho(\cdot; \phi)$ is a *correlation function* with ϕ controlling the correlation decay and σ_t^2 represents the spatial variance component. The spatial smoothness parameter, ν, is used if the Matern spatial correlation function is chosen. We further assume $\beta_0 \sim N(m_0, \Sigma_0)$ and $u_0(s) \equiv 0$, which completes the prior specifications leading to a well-identified Bayesian hierarhical model and also yield reasonable dependence structures.

Value

An object of class `spDynLM`, which is a list with the following tags:

`coords`	the $n \times 2$ matrix specified by `coords`.
`p.theta.samples`	a `coda` object of posterior samples for τ_t^2, σ_t^2, ϕ_t, ν_t.
`p.beta.0.samples`	a `coda` object of posterior samples for β at $t = 0$.
`p.beta.samples`	a `coda` object of posterior samples for β_t.
`p.sigma.eta.samples`	a `coda` object of posterior samples for Σ_η.

p.u.samples
: a coda object of posterior samples for spatio-temporal random effects u. Samples are over the columns and time steps increase in blocks of n down the columns, e.g., the first n rows correspond to locations $1, 2, \ldots, n$ in $t = 1$ and the last n rows correspond to locations $1, 2, \ldots, n$ in $t = N_t$.

p.y.samples
: a coda object of posterior samples for y. If $y_t(s)$ is specified as `NA` the `p.y.samples` hold the associated posterior predictive samples. Samples are over the columns and time steps increase in blocks of n down the columns, e.g., the first n rows correspond to locations $1, 2, \ldots, n$ in $t = 1$ and the last n rows correspond to locations $1, 2, \ldots, n$ in $t = N_t$.

The return object might include additional data used for subsequent prediction and/or model fit evaluation.

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Sudipto Banerjee <baner009@umn.edu>

References

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2012) Bayesian dynamic modeling for large space-time datasets using Gaussian predictive processes. *Journal of Geographical Systems*, 14:29–47.

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. https://www.jstatsoft.org/article/view/v063i13.

Gelfand, A.E., S. Banerjee, and D. Gamerman (2005) Spatial Process Modelling for Univariate and Multivariate Dynamic Spatial Data, *Environmetrics*, 16:465–479.

See Also

spLM

Examples

```
## Not run:
data("NETemp.dat")
ne.temp <- NETemp.dat

set.seed(1)

##take a chunk of New England
ne.temp <- ne.temp[ne.temp[,"UTMX"] > 5500000 & ne.temp[,"UTMY"] > 3000000,]

##subset first 2 years (Jan 2000 - Dec. 2002)
y.t <- ne.temp[,4:27]
N.t <- ncol(y.t) ##number of months
n <- nrow(y.t) ##number of observation per months

##add some missing observations to illistrate prediction
miss <- sample(1:N.t, 10)
```

```
holdout.station.id <- 5
y.t.holdout <- y.t[holdout.station.id, miss]
y.t[holdout.station.id, miss] <- NA

##scale to km
coords <- as.matrix(ne.temp[,c("UTMX", "UTMY")]/1000)
max.d <- max(iDist(coords))

##set starting and priors
p <- 2 #number of regression parameters in each month

starting <- list("beta"=rep(0,N.t*p), "phi"=rep(3/(0.5*max.d), N.t),
                 "sigma.sq"=rep(2,N.t), "tau.sq"=rep(1, N.t),
                 "sigma.eta"=diag(rep(0.01, p)))

tuning <- list("phi"=rep(5, N.t))

priors <- list("beta.0.Norm"=list(rep(0,p), diag(1000,p)),
               "phi.Unif"=list(rep(3/(0.9*max.d), N.t), rep(3/(0.05*max.d), N.t)),
               "sigma.sq.IG"=list(rep(2,N.t), rep(10,N.t)),
               "tau.sq.IG"=list(rep(2,N.t), rep(5,N.t)),
               "sigma.eta.IW"=list(2, diag(0.001,p)))

##make symbolic model formula statement for each month
mods <- lapply(paste(colnames(y.t),'elev',sep='~'), as.formula)

n.samples <- 2000

m.1 <- spDynLM(mods, data=cbind(y.t,ne.temp[,"elev",drop=FALSE]), coords=coords,
               starting=starting, tuning=tuning, priors=priors, get.fitted =TRUE,
               cov.model="exponential", n.samples=n.samples, n.report=25)

burn.in <- floor(0.75*n.samples)

quant <- function(x){quantile(x, prob=c(0.5, 0.025, 0.975))}

beta <- apply(m.1$p.beta.samples[burn.in:n.samples,], 2, quant)
beta.0 <- beta[,grep("Intercept", colnames(beta))]
beta.1 <- beta[,grep("elev", colnames(beta))]

plot(m.1$p.beta.0.samples)

par(mfrow=c(2,1))
plot(1:N.t, beta.0[1,], pch=19, cex=0.5, xlab="months", ylab="beta.0", ylim=range(beta.0))
arrows(1:N.t, beta.0[1,], 1:N.t, beta.0[3,], length=0.02, angle=90)
arrows(1:N.t, beta.0[1,], 1:N.t, beta.0[2,], length=0.02, angle=90)

plot(1:N.t, beta.1[1,], pch=19, cex=0.5, xlab="months", ylab="beta.1", ylim=range(beta.1))
arrows(1:N.t, beta.1[1,], 1:N.t, beta.1[3,], length=0.02, angle=90)
arrows(1:N.t, beta.1[1,], 1:N.t, beta.1[2,], length=0.02, angle=90)

theta <- apply(m.1$p.theta.samples[burn.in:n.samples,], 2, quant)
sigma.sq <- theta[,grep("sigma.sq", colnames(theta))]
```

```
tau.sq <- theta[,grep("tau.sq", colnames(theta))]
phi <- theta[,grep("phi", colnames(theta))]

par(mfrow=c(3,1))
plot(1:N.t, sigma.sq[1,], pch=19, cex=0.5, xlab="months", ylab="sigma.sq", ylim=range(sigma.sq))
arrows(1:N.t, sigma.sq[1,], 1:N.t, sigma.sq[3,], length=0.02, angle=90)
arrows(1:N.t, sigma.sq[1,], 1:N.t, sigma.sq[2,], length=0.02, angle=90)

plot(1:N.t, tau.sq[1,], pch=19, cex=0.5, xlab="months", ylab="tau.sq", ylim=range(tau.sq))
arrows(1:N.t, tau.sq[1,], 1:N.t, tau.sq[3,], length=0.02, angle=90)
arrows(1:N.t, tau.sq[1,], 1:N.t, tau.sq[2,], length=0.02, angle=90)

plot(1:N.t, 3/phi[1,], pch=19, cex=0.5, xlab="months", ylab="eff. range (km)", ylim=range(3/phi))
arrows(1:N.t, 3/phi[1,], 1:N.t, 3/phi[3,], length=0.02, angle=90)
arrows(1:N.t, 3/phi[1,], 1:N.t, 3/phi[2,], length=0.02, angle=90)

y.hat <- apply(m.1$p.y.samples[,burn.in:n.samples], 1, quant)
y.hat.med <- matrix(y.hat[1,], ncol=N.t)
y.hat.up <- matrix(y.hat[3,], ncol=N.t)
y.hat.low <- matrix(y.hat[2,], ncol=N.t)

y.obs <- as.vector(as.matrix(y.t[-holdout.station.id, -miss]))
y.obs.hat.med <- as.vector(y.hat.med[-holdout.station.id, -miss])
y.obs.hat.up <- as.vector(y.hat.up[-holdout.station.id, -miss])
y.obs.hat.low <- as.vector(y.hat.low[-holdout.station.id, -miss])

y.ho <- as.matrix(y.t.holdout)
y.ho.hat.med <- as.vector(y.hat.med[holdout.station.id, miss])
y.ho.hat.up <- as.vector(y.hat.up[holdout.station.id, miss])
y.ho.hat.low <- as.vector(y.hat.low[holdout.station.id, miss])

par(mfrow=c(2,1))
plot(y.obs, y.obs.hat.med, pch=19, cex=0.5, xlab="observed",
ylab="fitted", main="Observed vs. fitted")
arrows(y.obs, y.obs.hat.med, y.obs, y.obs.hat.up, length=0.02, angle=90)
arrows(y.obs, y.obs.hat.med, y.obs, y.obs.hat.low, length=0.02, angle=90)
lines(-50:50, -50:50, col="blue")

plot(y.ho, y.ho.hat.med, pch=19, cex=0.5, xlab="observed",
ylab="predicted", main="Observed vs. predicted")
arrows(y.ho, y.ho.hat.med, y.ho, y.ho.hat.up, length=0.02, angle=90)
arrows(y.ho, y.ho.hat.med, y.ho, y.ho.hat.low, length=0.02, angle=90)
lines(-50:50, -50:50, col="blue")

## End(Not run)
```

spGLM *Function for fitting univariate Bayesian generalized linear spatial regression models*

Description

The function spGLM fits univariate Bayesian generalized linear spatial regression models. Given a set of knots, spGLM will also fit a *predictive process* model (see references below).

Usage

```
spGLM(formula, family="binomial", weights, data = parent.frame(),
      coords, knots, starting, tuning, priors, cov.model,
      amcmc, n.samples, verbose=TRUE,
      n.report=100, ...)
```

Arguments

formula	a symbolic description of the regression model to be fit. See example below.
family	currently only supports binomial and poisson data using the logit and log link functions, respectively.
weights	an optional vector of weights to be used in the fitting process. Weights correspond to number of trials and *offset* for each location for the binomial and poisson family, respectively.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which spGLM is called.
coords	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing).
knots	either a $m \times 2$ matrix of the *predictive process* knot coordinates in R^2 (e.g., easting and northing) or a vector of length two or three with the first and second elements recording the number of columns and rows in the desired knot grid. The third, optional, element sets the offset of the outermost knots from the extent of the coords.
starting	a list with each tag corresponding to a parameter name. Valid tags are beta, sigma.sq, phi, nu, and w. The value portion of each tag is the parameter's starting value. If the predictive process is used then w must be of length m; otherwise, it must be of length n. Alternatively, w can be set as a scalar, in which case the value is repeated.
tuning	a list with each tag corresponding to a parameter name. Valid tags are beta, sigma.sq, phi, nu, and w. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution. The tuning value for beta can be a vector of length p (where p is the number of regression coefficients) or, if an adaptive MCMC is not used, i.e., amcmc is not specified, the lower-triangle of the $p \times p$ Cholesky square-root of the desired proposal covariance matrix. If the *predictive process* is used then w must be of length m; otherwise, it must be of length n. Alternatively, w can be set as a scalar, in which case the value is repeated.
priors	a list with each tag corresponding to a parameter name. Valid tags are sigma.sq.ig, phi.unif, nu.unif, beta.norm, and beta.flat. Variance parameter simga.sq is assumed to follow an inverse-Gamma distribution, whereas the spatial decay phi and smoothness nu parameters are assumed to follow Uniform distributions.

	The hyperparameters of the inverse-Gamma are passed as a vector of length two, with the first and second elements corresponding to the *shape* and *scale*, respectively. The hyperparameters of the Uniform are also passed as a vector of length two with the first and second elements corresponding to the lower and upper support, respectively. If the regression coefficients are each assumed to follow a Normal distribution, i.e., `beta.norm`, then mean and variance hyperparameters are passed as the first and second list elements, respectively. If `beta` is assumed flat then no arguments are passed. The default is a flat prior.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`amcmc`	a list with tags `n.batch`, `batch.length`, and `accept.rate`. Specifying this argument invokes an adaptive MCMC sampler, see Roberts and Rosenthal (2007) for an explanation.
`n.samples`	the number of MCMC iterations. This argument is ignored if `amcmc` is specified.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report Metropolis sampler acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

If a `binomial` model is specified the response vector is the number of successful trials at each location and `weights` is the total number of trials at each location.

For a `poisson` specification, the `weights` vector is the count offset, e.g., population, at each location. This differs from the `glm` `offset` argument which is passed as the log of this value.

A non-spatial model is fit when `coords` is not specified. See example below.

Value

An object of class `spGLM`, which is a list with the following tags:

`coords`	the $n \times 2$ matrix specified by `coords`.
`knot.coords`	the $m \times 2$ matrix as specified by `knots`.
`p.beta.theta.samples`	a `coda` object of posterior samples for the defined parameters.
`acceptance`	the Metropolis sampler acceptance rate. If `amcmc` is used then this will be a matrix of each parameter's acceptance rate at the end of each batch. Otherwise, the sampler is a Metropolis with a joint proposal of all parameters.
`acceptance.w`	if this is a non-predictive process model and `amcmc` is used then this will be a matrix of the Metropolis sampler acceptance rate for each location's spatial random effect.
`acceptance.w.knots`	if this is a *predictive process* model and `amcmc` is used then this will be a matrix of the Metropolis sampler acceptance rate for each knot's spatial random effect.

`p.w.knots.samples`
: a matrix that holds samples from the posterior distribution of the knots' spatial random effects. The rows of this matrix correspond to the m knot locations and the columns are the posterior samples. This is only returned if a *predictive process* model is used.

`p.w.samples`
: a matrix that holds samples from the posterior distribution of the locations' spatial random effects. The rows of this matrix correspond to the n point observations and the columns are the posterior samples.

The return object might include additional data used for subsequent prediction and/or model fit evaluation.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`baner009@umn.edu`>

References

Banerjee, S., A.E. Gelfand, A.O. Finley, and H. Sang. (2008) Gaussian Predictive Process Models for Large Spatial Datasets. *Journal of the Royal Statistical Society Series B*, 70:825–848.

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004) Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, Fla.

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. https://www.jstatsoft.org/article/view/v063i13.

Finley, A.O., S. Banerjee, and R.E. McRoberts. (2008) A Bayesian approach to quantifying uncertainty in multi-source forest area estimates. *Environmental and Ecological Statistics*, 15:241–258.

Roberts G.O. and Rosenthal J.S. (2006) Examples of Adaptive MCMC. http://probability.ca/jeff/ftpdir/adaptex.pdf Preprint.

See Also

`spMvGLM`

Examples

```
## Not run:
library(MBA)
library(coda)

set.seed(1)

rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p) %*% D + rep(mu,rep(n,p)))
}

################################
##Spatial binomial
################################

##Generate binary data
coords <- as.matrix(expand.grid(seq(0,100,length.out=8), seq(0,100,length.out=8)))
n <- nrow(coords)

phi <- 3/50
sigma.sq <- 2

R <- sigma.sq*exp(-phi*as.matrix(dist(coords)))
w <- rmvn(1, rep(0,n), R)

x <- as.matrix(rep(1,n))
beta <- 0.1
p <- 1/(1+exp(-(x%*%beta+w)))

weights <- rep(1, n)
weights[coords[,1]>mean(coords[,1])] <- 10

y <- rbinom(n, size=weights, prob=p)

##Collect samples
fit <- glm((y/weights)~x-1, weights=weights, family="binomial")
beta.starting <- coefficients(fit)
beta.tuning <- t(chol(vcov(fit)))

n.batch <- 200
batch.length <- 50
n.samples <- n.batch*batch.length

m.1 <- spGLM(y~1, family="binomial", coords=coords, weights=weights,
             starting=list("beta"=beta.starting, "phi"=0.06,"sigma.sq"=1, "w"=0),
             tuning=list("beta"=beta.tuning, "phi"=0.5, "sigma.sq"=0.5, "w"=0.5),
         priors=list("beta.Normal"=list(0,10), "phi.Unif"=c(0.03, 0.3), "sigma.sq.IG"=c(2, 1)),
          amcmc=list("n.batch"=n.batch, "batch.length"=batch.length, "accept.rate"=0.43),
             cov.model="exponential", verbose=TRUE, n.report=10)

burn.in <- 0.9*n.samples
sub.samps <- burn.in:n.samples

print(summary(window(m.1$p.beta.theta.samples, start=burn.in)))

beta.hat <- m.1$p.beta.theta.samples[sub.samps,"(Intercept)"]
w.hat <- m.1$p.w.samples[,sub.samps]

p.hat <- 1/(1+exp(-(x%*%beta.hat+w.hat)))

y.hat <- apply(p.hat, 2, function(x){rbinom(n, size=weights, prob=p.hat)})
```

```
y.hat.mu <- apply(y.hat, 1, mean)
y.hat.var <- apply(y.hat, 1, var)

##Take a look
par(mfrow=c(1,2))
surf <- mba.surf(cbind(coords,y.hat.mu),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Interpolated mean of posterior rate\n(observed rate)")
contour(surf, add=TRUE)
text(coords, label=paste("(",y,")",sep=""))

surf <- mba.surf(cbind(coords,y.hat.var),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Interpolated variance of posterior rate\n(observed #
of trials)")
contour(surf, add=TRUE)
text(coords, label=paste("(",weights,")",sep=""))

###########################
##Spatial poisson
###########################
##Generate count data
set.seed(1)

n <- 100

coords <- cbind(runif(n,1,100),runif(n,1,100))

phi <- 3/50
sigma.sq <- 2

R <- sigma.sq*exp(-phi*as.matrix(dist(coords)))
w <- rmvn(1, rep(0,n), R)

x <- as.matrix(rep(1,n))
beta <- 0.1
y <- rpois(n, exp(x%*%beta+w))

##Collect samples
beta.starting <- coefficients(glm(y~x-1, family="poisson"))
beta.tuning <- t(chol(vcov(glm(y~x-1, family="poisson"))))

n.batch <- 500
batch.length <- 50
n.samples <- n.batch*batch.length

##Note tuning list is now optional

m.1 <- spGLM(y~1, family="poisson", coords=coords,
             starting=list("beta"=beta.starting, "phi"=0.06,"sigma.sq"=1, "w"=0),
             tuning=list("beta"=0.1, "phi"=0.5, "sigma.sq"=0.5, "w"=0.5),
             priors=list("beta.Flat", "phi.Unif"=c(0.03, 0.3), "sigma.sq.IG"=c(2, 1)),
         amcmc=list("n.batch"=n.batch, "batch.length"=batch.length, "accept.rate"=0.43),
             cov.model="exponential", verbose=TRUE, n.report=10)
```

```
##Just for fun check out the progression of the acceptance
##as it moves to 43% (same can be seen for the random spatial effects).
plot(mcmc(t(m.1$acceptance)), density=FALSE, smooth=FALSE)

##Now parameter summaries, etc.
burn.in <- 0.9*n.samples
sub.samps <- burn.in:n.samples

m.1$p.samples[,"phi"] <- 3/m.1$p.samples[,"phi"]

plot(m.1$p.beta.theta.samples)
print(summary(window(m.1$p.beta.theta.samples, start=burn.in)))

beta.hat <- m.1$p.beta.theta.samples[sub.samps,"(Intercept)"]
w.hat <- m.1$p.w.samples[,sub.samps]

y.hat <- apply(exp(x%*%beta.hat+w.hat), 2, function(x){rpois(n, x)})

y.hat.mu <- apply(y.hat, 1, mean)

##Take a look
par(mfrow=c(1,2))
surf <- mba.surf(cbind(coords,y),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Observed counts")
contour(surf, add=TRUE)
text(coords, labels=y, cex=1)

surf <- mba.surf(cbind(coords,y.hat.mu),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Fitted counts")
contour(surf, add=TRUE)
text(coords, labels=round(y.hat.mu,0), cex=1)

## End(Not run)
```

---

spLM *Function for fitting univariate Bayesian spatial regression models*

Description

The function spLM fits Gaussian univariate Bayesian spatial regression models. Given a set of knots, spLM will also fit a *predictive process* model (see references below).

Usage

```
spLM(formula, data = parent.frame(), coords, knots,
      starting, tuning, priors, cov.model,
      modified.pp = TRUE, amcmc, n.samples,
      verbose=TRUE, n.report=100, ...)
```

Arguments

`formula`	a symbolic description of the regression model to be fit. See example below.
`data`	an optional data frame containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `spLM` is called.
`coords`	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing).
`knots`	either a $m \times 2$ matrix of the *predictive process* knot coordinates in R^2 (e.g., easting and northing) or a vector of length two or three with the first and second elements recording the number of columns and rows in the desired knot grid. The third, optional, element sets the offset of the outermost knots from the extent of the `coords`.
`starting`	a list with each tag corresponding to a parameter name. Valid tags are `beta`, `sigma.sq`, `tau.sq`, `phi`, and `nu`. The value portion of each tag is the parameter's starting value.
`tuning`	a list with each tag corresponding to a parameter name. Valid tags are `sigma.sq`, `tau.sq`, `phi`, and `nu`. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution.
`priors`	a list with each tag corresponding to a parameter name. Valid tags are `sigma.sq.ig`, `tau.sq.ig`, `phi.unif`, `nu.unif`, `beta.norm`, and `beta.flat`. Variance parameters, `simga.sq` and `tau.sq`, are assumed to follow an inverse-Gamma distribution, whereas the spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma are passed as a vector of length two, with the first and second elements corresponding to the *shape* and *scale*, respectively. The hyperparameters of the Uniform are also passed as a vector of length two with the first and second elements corresponding to the lower and upper support, respectively. If the regression coefficients, i.e., `beta` vector, are assumed to follow a multivariate Normal distribution then pass the hyperparameters as a list of length two with the first and second elements corresponding to the mean vector and positive definite covariance matrix, respectively. If `beta` is assumed flat then no arguments are passed. The default is a flat prior.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`modified.pp`	a logical value indicating if the *modified predictive process* should be used (see references below for details). Note, if a predictive process model is not used (i.e., `knots` is not specified) then this argument is ignored.
`amcmc`	a list with tags `n.batch`, `batch.length`, and `accept.rate`. Specifying this argument invokes an adaptive MCMC sampler, see Roberts and Rosenthal (2007) for an explanation.
`n.samples`	the number of MCMC iterations. This argument is ignored if `amcmc` is specified.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report Metropolis sampler acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

Model parameters can be fixed at their `starting` values by setting their `tuning` values to zero.

The *no nugget* model is specified by removing `tau.sq` from the `starting` list.

Value

An object of class `spLM`, which is a list with the following tags:

`coords`	the $n \times 2$ matrix specified by `coords`.
`knot.coords`	the $m \times 2$ matrix as specified by `knots`.
`p.theta.samples`	a `coda` object of posterior samples for the defined parameters.
`acceptance`	the Metropolis sampling acceptance percent. Reported at `batch.length` or `n.report` intervals for `amcmc` specified and non-specified, respectively.

The return object might include additional data used for subsequent prediction and/or model fit evaluation.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`baner009@umn.edu`>

References

Banerjee, S., A.E. Gelfand, A.O. Finley, and H. Sang. (2008) Gaussian Predictive Process Models for Large Spatial Datasets. *Journal of the Royal Statistical Society Series B*, 70:825–848.

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, FL.

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. `https://www.jstatsoft.org/article/view/v063i13`.

Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2009) Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, 53:2873–2884.

Roberts G.O. and Rosenthal J.S. (2006). Examples of Adaptive MCMC. `http://probability.ca/jeff/ftpdir/adaptex.pdf`.

See Also

`spMvLM` `spSVC`

Examples

```
library(coda)

## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

n <- 100
coords <- cbind(runif(n,0,1), runif(n,0,1))
X <- as.matrix(cbind(1, rnorm(n)))

B <- as.matrix(c(1,5))
p <- length(B)

sigma.sq <- 2
tau.sq <- 0.1
phi <- 3/0.5

D <- as.matrix(dist(coords))
R <- exp(-phi*D)
w <- rmvn(1, rep(0,n), sigma.sq*R)
y <- rnorm(n, X%*%B + w, sqrt(tau.sq))

n.samples <- 2000

starting <- list("phi"=3/0.5, "sigma.sq"=50, "tau.sq"=1)

tuning <- list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)

priors.1 <- list("beta.Norm"=list(rep(0,p), diag(1000,p)),
                 "phi.Unif"=c(3/1, 3/0.1), "sigma.sq.IG"=c(2, 2),
                 "tau.sq.IG"=c(2, 0.1))

priors.2 <- list("beta.Flat", "phi.Unif"=c(3/1, 3/0.1),
                 "sigma.sq.IG"=c(2, 2), "tau.sq.IG"=c(2, 0.1))

cov.model <- "exponential"

n.report <- 500
verbose <- TRUE

m.1 <- spLM(y~X-1, coords=coords, starting=starting,
            tuning=tuning, priors=priors.1, cov.model=cov.model,
            n.samples=n.samples, verbose=verbose, n.report=n.report)
```

```
m.2 <- spLM(y~X-1, coords=coords, starting=starting,
            tuning=tuning, priors=priors.2, cov.model=cov.model,
            n.samples=n.samples, verbose=verbose, n.report=n.report)

burn.in <- 0.5*n.samples

##recover beta and spatial random effects
m.1 <- spRecover(m.1, start=burn.in, verbose=FALSE)
m.2 <- spRecover(m.2, start=burn.in, verbose=FALSE)

round(summary(m.1$p.theta.recover.samples)$quantiles[,c(3,1,5)],2)
round(summary(m.2$p.theta.recover.samples)$quantiles[,c(3,1,5)],2)

round(summary(m.1$p.beta.recover.samples)$quantiles[,c(3,1,5)],2)
round(summary(m.2$p.beta.recover.samples)$quantiles[,c(3,1,5)],2)

m.1.w.summary <- summary(mcmc(t(m.1$p.w.recover.samples)))$quantiles[,c(3,1,5)]
m.2.w.summary <- summary(mcmc(t(m.2$p.w.recover.samples)))$quantiles[,c(3,1,5)]

plot(w, m.1.w.summary[,1], xlab="Observed w", ylab="Fitted w",
     xlim=range(w), ylim=range(m.1.w.summary), main="Spatial random effects")
arrows(w, m.1.w.summary[,1], w, m.1.w.summary[,2], length=0.02, angle=90)
arrows(w, m.1.w.summary[,1], w, m.1.w.summary[,3], length=0.02, angle=90)
lines(range(w), range(w))

points(w, m.2.w.summary[,1], col="blue", pch=19, cex=0.5)
arrows(w, m.2.w.summary[,1], w, col="blue", m.2.w.summary[,2], length=0.02, angle=90)
arrows(w, m.2.w.summary[,1], w, col="blue", m.2.w.summary[,3], length=0.02, angle=90)

###########################
##Predictive process model
###########################
m.1 <- spLM(y~X-1, coords=coords, knots=c(6,6,0.1), starting=starting,
            tuning=tuning, priors=priors.1, cov.model=cov.model,
            n.samples=n.samples, verbose=verbose, n.report=n.report)

m.2 <- spLM(y~X-1, coords=coords, knots=c(6,6,0.1), starting=starting,
            tuning=tuning, priors=priors.2, cov.model=cov.model,
            n.samples=n.samples, verbose=verbose, n.report=n.report)

burn.in <- 0.5*n.samples

round(summary(window(m.1$p.beta.samples, start=burn.in))$quantiles[,c(3,1,5)],2)
round(summary(window(m.2$p.beta.samples, start=burn.in))$quantiles[,c(3,1,5)],2)

round(summary(window(m.1$p.theta.samples, start=burn.in))$quantiles[,c(3,1,5)],2)
round(summary(window(m.2$p.theta.samples, start=burn.in))$quantiles[,c(3,1,5)],2)

## End(Not run)
```

spMisalignGLM *Function for fitting multivariate generalized linear Bayesian spatial regression models to misaligned data*

Description

The function `spMisalignGLM` fits Gaussian multivariate Bayesian generalized linear spatial regression models to misaligned data.

Usage

```
spMisalignGLM(formula, family="binomial", weights, data = parent.frame(), coords,
      starting, tuning, priors, cov.model,
      amcmc, n.samples, verbose=TRUE, n.report=100, ...)
```

Arguments

`formula`	a list of q symbolic regression models to be fit. See example below.
`family`	currently only supports `binomial` and `poisson` data using the logit and log link functions, respectively.
`weights`	an optional list of weight vectors associated with each model in the formula list. Weights correspond to number of trials and *offset* for each location for the `binomial` and `poisson` family, respectively.
`data`	an optional data frame containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `spMisalignGLM` is called.
`coords`	a list of q $n_i \times 2$ matrices of the observation coordinates in R^2 (e.g., easting and northing) where $i = (1, 2, \ldots, q)$.
`starting`	a list with tags corresponding to `A`, `phi`, and `nu`. The value portion of each tag is a vector that holds the parameter's starting values. `A` is of length $\frac{q(q+1)}{2}$ and holds the lower-triangle elements in column major ordering of the Cholesky square root of the spatial cross-covariance matrix $K = AA'$. `phi` and `nu` are of length q.
`tuning`	a list with tags `A`, `phi`, and `nu`. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution. `A` is of length $\frac{q(q+1)}{2}$ and `phi` and `nu` are of length q.
`priors`	a list with each tag corresponding to a parameter name. Valid tags are `beta.flat`, `beta.norm`, `K.iw`, `phi.unif`, and `nu.unif`. If the regression coefficients are each assumed to follow a Normal distribution, i.e., `beta.norm`, then mean and variance hyperparameters are passed as the first and second list elements, respectively. If `beta` is assumed flat then no arguments are passed. The default is a flat prior. The spatial cross-covariance matrix $K = AA'$ is assumed to follow an inverse-Wishart distribution, whereas the spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Wishart are passed as a list of length two, with the first and second elements corresponding to the *df* and $q \times q$ *scale* matrix, respectively.

	The hyperparameters of the Uniform are also passed as a list of vectors with the first and second list elements corresponding to the lower and upper support, respectively.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`amcmc`	a list with tags `n.batch`, `batch.length`, and `accept.rate`. Specifying this argument invokes an adaptive MCMC sampler see Roberts and Rosenthal (2007) for an explanation.
`n.samples`	the number of MCMC iterations. This argument is ignored if `amcmc` is specified.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report Metropolis acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

If a `binomial` model is specified the response vector is the number of successful trials at each location and `weights` is the total number of trials at each location.

For a `poisson` specification, the `weights` vector is the count offset, e.g., population, at each location. This differs from the `glm` `offset` argument which is passed as the log of this value.

Value

An object of class `spMisalignGLM`, which is a list with the following tags:

`p.beta.theta.samples`	a `coda` object of posterior samples for the defined parameters.
`acceptance`	the Metropolis sampler acceptance rate. If `amcmc` is used then this will be a matrix of each parameter's acceptance rate at the end of each batch. Otherwise, the sampler is a Metropolis with a joint proposal of all parameters.
`acceptance.w`	if `amcmc` is used then this will be a matrix of the Metropolis sampler acceptance rate for each location's spatial random effect.
`p.w.samples`	a matrix that holds samples from the posterior distribution of the locations' spatial random effects. Posterior samples are organized with the first response variable n_1 locations held in rows $1\ldots,n_1$ rows, then the next response variable samples held in the $(n_1+1),\ldots,(n_1+n_2)$, etc.

The return object might include additional data used for subsequent prediction and/or model fit evaluation.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`baner009@umn.edu`>

References

Banerjee, S., A.E. Gelfand, A.O. Finley, and H. Sang. (2008) Gaussian Predictive Process Models for Large Spatial Datasets. *Journal of the Royal Statistical Society Series B*, 70:825–848.

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, Fla.

Finley, A.O., S. Banerjee, and B.D. Cook. (2014) Bayesian hierarchical models for spatially misaligned data. *Methods in Ecology and Evolution*, 5:514–523.

Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2009) Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, 53:2873–2884.

Finley, A.O., S. Banerjee, A.R. Ek, and R.E. McRoberts. (2008) Bayesian multivariate process modeling for prediction of forest attributes. *Journal of Agricultural, Biological, and Environmental Statistics*, 13:60–83.

See Also

`spMvGLM` `spMisalignLM`

Examples

```
## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p)))){stop("Dimension problem!")}
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

##generate some data
n <- 100 ##number of locations
q <- 3 ##number of outcomes at each location
nltr <- q*(q+1)/2 ##number of triangular elements in the cross-covariance matrix

coords <- cbind(runif(n,0,1), runif(n,0,1))

##parameters for generating a multivariate spatial GP covariance matrix
theta <- rep(3/0.5,q) ##spatial decay

A <- matrix(0,q,q)
A[lower.tri(A,TRUE)] <- c(1,1,-1,1,0.5,0.25)
K <- A%*%t(A)
K ##spatial cross-covariance
cov2cor(K) ##spatial cross-correlation

C <- mkSpCov(coords, K, diag(0,q), theta, cov.model="exponential")

w <- rmvn(1, rep(0,nrow(C)), C) ##spatial random effects
```

```
w.a <- w[seq(1,length(w),q)]
w.b <- w[seq(2,length(w),q)]
w.c <- w[seq(3,length(w),q)]

##covariate portion of the mean
x.a <- cbind(1, rnorm(n))
x.b <- cbind(1, rnorm(n))
x.c <- cbind(1, rnorm(n))
x <- mkMvX(list(x.a, x.b, x.c))

B.1 <- c(1,-1)
B.2 <- c(-1,1)
B.3 <- c(-1,-1)
B <- c(B.1, B.2, B.3)

y <- rpois(nrow(C), exp(x%*%B+w))

y.a <- y[seq(1,length(y),q)]
y.b <- y[seq(2,length(y),q)]
y.c <- y[seq(3,length(y),q)]

##subsample to make spatially misaligned data
sub.1 <- 1:50
y.1 <- y.a[sub.1]
w.1 <- w.a[sub.1]
coords.1 <- coords[sub.1,]
x.1 <- x.a[sub.1,]

sub.2 <- 25:75
y.2 <- y.b[sub.2]
w.2 <- w.b[sub.2]
coords.2 <- coords[sub.2,]
x.2 <- x.b[sub.2,]

sub.3 <- 50:100
y.3 <- y.c[sub.3]
w.3 <- w.c[sub.3]
coords.3 <- coords[sub.3,]
x.3 <- x.c[sub.3,]

##call spMisalignGLM
q <- 3
A.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]

n.batch <- 200
batch.length <- 25
n.samples <- n.batch*batch.length

starting <- list("beta"=rep(0,length(B)), "phi"=rep(3/0.5,q), "A"=A.starting, "w"=0)

tuning <- list("beta"=rep(0.1,length(B)), "phi"=rep(1,q), "A"=rep(0.1,length(A.starting)), "w"=1)

priors <- list("phi.Unif"=list(rep(3/0.75,q), rep(3/0.25,q)),
                "K.IW"=list(q+1, diag(0.1,q)),  rep(0.1,q))

m.1 <- spMisalignGLM(list(y.1~x.1-1, y.2~x.2-1, y.3~x.3-1), family="poisson",
                     coords=list(coords.1, coords.2, coords.3),
                     starting=starting, tuning=tuning, priors=priors,
              amcmc=list("n.batch"=n.batch, "batch.length"=batch.length, "accept.rate"=0.43),
                     cov.model="exponential", n.report=10)

burn.in <- floor(0.75*n.samples)

plot(m.1$p.beta.theta.samples, density=FALSE)

##predict for all locations, i.e., observed and not observed
out <- spPredict(m.1, start=burn.in, thin=10, pred.covars=list(x.a, x.b, x.c),
                 pred.coords=list(coords, coords, coords))

##summary and check
quants <- function(x){quantile(x, prob=c(0.5,0.025,0.975))}

y.hat <- apply(out$p.y.predictive.samples, 1, quants)

##unstack and plot
y.a.hat <- y.hat[,1:n]
y.b.hat <- y.hat[,(n+1):(2*n)]
y.c.hat <- y.hat[,(2*n+1):(3*n)]

par(mfrow=c(1,3))
plot(y.a ,y.a.hat[1,], xlab="Observed y.a", ylab="Fitted & predicted y.a")
plot(y.b, y.b.hat[1,], xlab="Observed y.b", ylab="Fitted & predicted y.b")
plot(y.c, y.c.hat[1,], xlab="Observed y.c", ylab="Fitted & predicted y.c")


## End(Not run)
```

`spMisalignLM` *Function for fitting multivariate Bayesian spatial regression models to misaligned data*

Description

The function `spMisalignLM` fits Gaussian multivariate Bayesian spatial regression models to misaligned data.

Usage

```
spMisalignLM(formula, data = parent.frame(), coords,
      starting, tuning, priors, cov.model,
      amcmc, n.samples, verbose=TRUE, n.report=100, ...)
```

Arguments

`formula`	a list of q symbolic regression models to be fit. See example below.
`data`	an optional data frame containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `spMisalignLM` is called.
`coords`	a list of q $n_i \times 2$ matrices of the observation coordinates in R^2 (e.g., easting and northing) where $i = (1, 2, \ldots, q)$. .
`starting`	a list with tags corresponding to `A`, `phi`, `nu`, and `Psi`. The value portion of each tag is a vector that holds the parameter's starting values. `A` is of length $\frac{q(q+1)}{2}$ and holds the lower-triangle elements in column major ordering of the Cholesky square root of the spatial cross-covariance matrix. `phi` and `nu` are of length q. The vector of residual variances `Psi` is also of length q.
`tuning`	a list with tags `A`, `phi`, `nu`, and `Psi`. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution. `A` is of length $\frac{q(q+1)}{2}$ and `Psi`, `phi`, and `nu` are of length q.
`priors`	a list with tags `beta.flat`, `K.iw`, `Psi.ig`, `phi.unif` and `nu.unif`. The hyperparameters of the inverse-Wishart for the cross-covariance matrix $K = AA'$ are passed as a list of length two, with the first and second elements corresponding to the *df* and $q \times q$ *scale* matrix, respectively. The inverse-Gamma hyperparameters for the non-spatial residual variances are specified as a list `Psi.ig` of length two with the first and second list elements consisting of vectors of the q *shape* and *scale* hyperparameters, respectively. The hyperparameters of the Uniform `phi.unif`, and `nu.unif` are also passed as a list of vectors with the first and second list elements corresponding to the lower and upper support, respectively.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`amcmc`	a list with tags `n.batch`, `batch.length`, and `accept.rate`. Specifying this argument invokes an adaptive MCMC sampler see Roberts and Rosenthal (2007) for an explanation.
`n.samples`	the number of MCMC iterations. This argument is ignored if `amcmc` is specified.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report Metropolis acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

Model parameters can be fixed at their `starting` values by setting their `tuning` values to zero.

Value

An object of class `spMisalignLM`, which is a list with the following tags:

`p.theta.samples`	a `coda` object of posterior samples for the defined parameters.
`acceptance`	the Metropolis sampling acceptance percent. Reported at `batch.length` or `n.report` intervals for `amcmc` specified and non-specified, respectively

The return object might include additional data used for subsequent prediction and/or model fit evaluation.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`baner009@umn.edu`>

References

Banerjee, S., A.E. Gelfand, A.O. Finley, and H. Sang. (2008) Gaussian Predictive Process Models for Large Spatial Datasets. *Journal of the Royal Statistical Society Series B*, 70:825–848.

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, Fla.

Finley, A.O., S. Banerjee, and B.D. Cook. (2014) Bayesian hierarchical models for spatially misaligned data. *Methods in Ecology and Evolution*, 5:514–523.

Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2009) Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, 53:2873–2884.

Finley, A.O., S. Banerjee, A.R. Ek, and R.E. McRoberts. (2008) Bayesian multivariate process modeling for prediction of forest attributes. *Journal of Agricultural, Biological, and Environmental Statistics*, 13:60–83.

See Also

`spMvLMspMisalignGLM`

Examples

```
## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p)))){stop("Dimension problem!")}
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

##generate some data
n <- 100 ##number of locations
```

```
q <- 3 ##number of outcomes at each location
nltr <- q*(q+1)/2 ##number of triangular elements in the cross-covariance matrix

coords <- cbind(runif(n,0,1), runif(n,0,1))

##parameters for generating a multivariate spatial GP covariance matrix
theta <- rep(3/0.5,q) ##spatial decay

A <- matrix(0,q,q)
A[lower.tri(A,TRUE)] <- c(1,1,-1,1,0.5,0.25)
K <- A%*%t(A)
K ##spatial cross-covariance
cov2cor(K) ##spatial cross-correlation

C <- mkSpCov(coords, K, diag(0,q), theta, cov.model="exponential")

w <- rmvn(1, rep(0,nrow(C)), C) ##spatial random effects

w.a <- w[seq(1,length(w),q)]
w.b <- w[seq(2,length(w),q)]
w.c <- w[seq(3,length(w),q)]

##covariate portion of the mean
x.a <- cbind(1, rnorm(n))
x.b <- cbind(1, rnorm(n))
x.c <- cbind(1, rnorm(n))
x <- mkMvX(list(x.a, x.b, x.c))

B.1 <- c(1,-1)
B.2 <- c(-1,1)
B.3 <- c(-1,-1)
B <- c(B.1, B.2, B.3)

Psi <- c(0.1, 0.1, 0.1) ##non-spatial residual variance, i.e., nugget

y <- rnorm(n*q, x%*%B+w, rep(sqrt(Psi),n))

y.a <- y[seq(1,length(y),q)]
y.b <- y[seq(2,length(y),q)]
y.c <- y[seq(3,length(y),q)]

##subsample to make spatially misaligned data
sub.1 <- 1:50
y.1 <- y.a[sub.1]
w.1 <- w.a[sub.1]
coords.1 <- coords[sub.1,]
x.1 <- x.a[sub.1,]

sub.2 <- 25:75
y.2 <- y.b[sub.2]
w.2 <- w.b[sub.2]
coords.2 <- coords[sub.2,]
x.2 <- x.b[sub.2,]
```

```
sub.3 <- 50:100
y.3 <- y.c[sub.3]
w.3 <- w.c[sub.3]
coords.3 <- coords[sub.3,]
x.3 <- x.c[sub.3,]

##call spMisalignLM
q <- 3
A.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]
n.samples <- 5000

starting <- list("phi"=rep(3/0.5,q), "A"=A.starting, "Psi"=rep(1,q))
tuning <- list("phi"=rep(0.5,q), "A"=rep(0.01,length(A.starting)), "Psi"=rep(0.1,q))
priors <- list("phi.Unif"=list(rep(3/0.75,q), rep(3/0.25,q)),
               "K.IW"=list(q+1, diag(0.1,q)), "Psi.ig"=list(rep(2,q), rep(0.1,q)))

m.1 <- spMisalignLM(list(y.1~x.1-1, y.2~x.2-1, y.3~x.3-1),
                    coords=list(coords.1, coords.2, coords.3),
                    starting=starting, tuning=tuning, priors=priors,
                    n.samples=n.samples, cov.model="exponential", n.report=100)

burn.in <- floor(0.75*n.samples)

plot(m.1$p.theta.samples, density=FALSE)

##recover regression coefficients and random effects
m.1 <- spRecover(m.1, start=burn.in)

round(summary(m.1$p.theta.recover.samples)$quantiles[,c(3,1,5)],2)
round(summary(m.1$p.beta.recover.samples)$quantiles[,c(3,1,5)],2)

##predict for all locations, i.e., observed and not observed
out <- spPredict(m.1, start=burn.in, thin=10, pred.covars=list(x.a, x.b,
x.c),
                 pred.coords=list(coords, coords, coords))

##summary and check
quants <- function(x){quantile(x, prob=c(0.5,0.025,0.975))}

y.hat <- apply(out$p.y.predictive.samples, 1, quants)

##unstack and plot
y.a.hat <- y.hat[,1:n]
y.b.hat <- y.hat[,(n+1):(2*n)]
y.c.hat <- y.hat[,(2*n+1):(3*n)]

par(mfrow=c(1,3))
plot(y.a, y.a.hat[1,], xlab="Observed y.a", ylab="Fitted & predicted y.a",
     xlim=range(y), ylim=range(y.hat), main="")
arrows(y.a[-sub.1], y.a.hat[1,-sub.1], y.a[-sub.1], y.a.hat[2,-sub.1], length=0.02, angle=90)
arrows(y.a[-sub.1], y.a.hat[1,-sub.1], y.a[-sub.1], y.a.hat[3,-sub.1], length=0.02, angle=90)
lines(range(y.a), range(y.a))
```

```
plot(y.b, y.b.hat[1,], xlab="Observed y.b", ylab="Fitted & predicted y.b",
     xlim=range(y), ylim=range(y.hat), main="")
arrows(y.b[-sub.2], y.b.hat[1,-sub.2], y.b[-sub.2], y.b.hat[2,-sub.2], length=0.02, angle=90)
arrows(y.b[-sub.2], y.b.hat[1,-sub.2], y.b[-sub.2], y.b.hat[3,-sub.2], length=0.02, angle=90)
lines(range(y.b), range(y.b))

plot(y.c, y.c.hat[1,], xlab="Observed y.c", ylab="Fitted & predicted y.c",
     xlim=range(y), ylim=range(y.hat), main="")
arrows(y.c[-sub.3], y.c.hat[1,-sub.3], y.c[-sub.3], y.c.hat[2,-sub.3], length=0.02, angle=90)
arrows(y.c[-sub.3], y.c.hat[1,-sub.3], y.c[-sub.3], y.c.hat[3,-sub.3], length=0.02, angle=90)
lines(range(y.c), range(y.c))

## End(Not run)
```

spMvGLM — *Function for fitting multivariate Bayesian generalized linear spatial regression models*

Description

The function spMvGLM fits multivariate Bayesian generalized linear spatial regression models. Given a set of knots, spMvGLM will also fit a *predictive process* model (see references below).

Usage

```
spMvGLM(formula, family="binomial", weights, data = parent.frame(), coords, knots,
      starting, tuning, priors, cov.model,
      amcmc, n.samples,
      verbose=TRUE, n.report=100, ...)
```

Arguments

formula	a list of q symbolic regression model descriptions to be fit. See example below.
family	currently only supports binomial and poisson data using the logit and log link functions, respectively.
weights	an optional $n \times q$ matrix of weights to be used in the fitting process. The order of the columns correspond to the univariate models in the formula list. Weights correspond to number of trials and *offset* for each location for the binomial and poisson family, respectively.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which spMvGLM is called.
coords	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing).

`knots`	either a $m \times 2$ matrix of the *predictive process* knot coordinates in R^2 (e.g., easting and northing) or a vector of length two or three with the first and second elements recording the number of columns and rows in the desired knot grid. The third, optional, element sets the offset of the outermost knots from the extent of the `coords`.
`starting`	a list with each tag corresponding to a parameter name. Valid tags are `beta`, `A`, `phi`, `nu`, and `w`. The value portion of each tag is a vector that holds the parameter's starting values and are of length p for `beta` (where p is the total number of regression coefficients in the multivariate model), $\frac{q(q+1)}{2}$ for `A`, and q for `phi`, and `nu`. Here, `A` holds the the lower-triangle elements in column major ordering of the Cholesky square root of the spatial cross-covariance matrix. If the *predictive process* is used then `w` must be of length qm; otherwise, it must be of length qn. Alternatively, `w` can be set as a scalar, in which case the value is repeated.
`tuning`	a list with tags `beta`, `A`, `phi`, `nu`, and `w`. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution. The value portion of these tags is of length p for `beta`, $\frac{q(q+1)}{2}$ for `A`, and q for `phi`, and `nu`. Here, `A` holds the tuning values corresponding to the lower-triangle elements in column major ordering of the Cholesky square root of the spatial cross-covariance matrix. If the *predictive process* is used then `w` must be of length qm; otherwise, it must be of length qn. Alternatively, `w` can be set as a scalar, in which case the value is repeated. The tuning value for `beta` can be a vector of length p or, if an adaptive MCMC is not used, i.e., `amcmc` is not specified, the lower-triangle of the $p \times p$ Cholesky square-root of the desired proposal covariance matrix.
`priors`	a list with each tag corresponding to a parameter name. Valid tags are `beta.flat`, `beta.norm`, `K.iw`, `phi.unif`, and `nu.unif`. If the regression coefficients are each assumed to follow a Normal distribution, i.e., `beta.norm`, then mean and variance hyperparameters are passed as the first and second list elements, respectively. If `beta` is assumed flat then no arguments are passed. The default is a flat prior. The spatial cross-covariance matrix `K` is assumed to follow an inverse-Wishart distribution, whereas the spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Wishart are passed as a list of length two, with the first and second elements corresponding to the *df* and $q \times q$ *scale* matrix, respectively. The hyperparameters of the Uniform are also passed as a list of vectors with the first and second list elements corresponding to the lower and upper support, respectively.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`amcmc`	a list with tags `n.batch`, `batch.length`, and `accept.rate`. Specifying this argument invokes an adaptive MCMC sampler see Roberts and Rosenthal (2007) for an explanation.
`n.samples`	the number of MCMC iterations. This argument is ignored if `amcmc` is specified.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.

`n.report`	the interval to report Metropolis sampler acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

If a `binomial` model is specified the response vector is the number of successful trials at each location and `weights` is the total number of trials at each location.

For a `poisson` specification, the `weights` vector is the count offset, e.g., population, at each location. This differs from the `glm` `offset` argument which is passed as the log of this value.

A non-spatial model is fit when `coords` is not specified. See example below.

Value

An object of class `spMvGLM`, which is a list with the following tags:

`coords`	the $n \times 2$ matrix specified by `coords`.
`knot.coords`	the $m \times 2$ matrix as specified by `knots`.
`p.beta.theta.samples`	a `coda` object of posterior samples for the defined parameters.
`acceptance`	the Metropolis sampler acceptance rate. If `amcmc` is used then this will be a matrix of each parameter's acceptance rate at the end of each batch. Otherwise, the sampler is a Metropolis with a joint proposal of all parameters.
`acceptance.w`	if this is a non-predictive process model and `amcmc` is used then this will be a matrix of the Metropolis sampler acceptance rate for each location's spatial random effect.
`acceptance.w.knots`	if this is a *predictive process* model and `amcmc` is used then this will be a matrix of the Metropolis sampler acceptance rate for each knot's spatial random effect.
`p.w.knots.samples`	a matrix that holds samples from the posterior distribution of the knots' spatial random effects. The rows of this matrix correspond to the $q \times m$ knot locations and the columns are the posterior samples. This is only returned if a *predictive process* model is used.
`p.w.samples`	a matrix that holds samples from the posterior distribution of the locations' spatial random effects. The rows of this matrix correspond to the $q \times n$ point observations and the columns are the posterior samples.

The return object might include additional data used for subsequent prediction and/or model fit evaluation.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`baner009@umn.edu`>

References

Finley, A.O., S. Banerjee, and R.E. McRoberts. (2008) A Bayesian approach to quantifying uncertainty in multi-source forest area estimates. *Environmental and Ecological Statistics*, 15:241–258.

Banerjee, S., A.E. Gelfand, A.O. Finley, and H. Sang. (2008) Gaussian Predictive Process Models for Large Spatial Datasets. *Journal of the Royal Statistical Society Series B*, 70:825–848.

Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2009) Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, 53:2873-2884.

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. `https://www.jstatsoft.org/article/view/v063i13`.

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, Fla.

Roberts G.O. and Rosenthal J.S. (2006) Examples of Adaptive MCMC. `http://probability.ca/jeff/ftpdir/adaptex.pdf` Preprint.

See Also

`spGLM`

Examples

```
## Not run:
library(MBA)

##Some useful functions
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p)))){stop("Dimension problem!")}
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

##Generate some data
n <- 25 ##number of locations
q <- 2 ##number of outcomes at each location
nltr <- q*(q+1)/2 ##number of triangular elements in the cross-covariance matrix

coords <- cbind(runif(n,0,1), runif(n,0,1))

##Parameters for the bivariate spatial random effects
theta <- rep(3/0.5,q)

A <- matrix(0,q,q)
A[lower.tri(A,TRUE)] <- c(1,-1,0.25)
K <- A%*%t(A)
```

```
Psi <- diag(0,q)

C <- mkSpCov(coords, K, Psi, theta, cov.model="exponential")

w <- rmvn(1, rep(0,nrow(C)), C)

w.1 <- w[seq(1,length(w),q)]
w.2 <- w[seq(2,length(w),q)]

##Covariate portion of the mean
x.1 <- cbind(1, rnorm(n))
x.2 <- cbind(1, rnorm(n))
x <- mkMvX(list(x.1, x.2))

B.1 <- c(1,-1)
B.2 <- c(-1,1)
B <- c(B.1, B.2)

weight <- 10 ##i.e., trials
p <- 1/(1+exp(-(x%*%B+w)))
y <- rbinom(n*q, size=rep(weight,n*q), prob=p)

y.1 <- y[seq(1,length(y),q)]
y.2 <- y[seq(2,length(y),q)]

##Call spMvLM
fit <- glm((y/weight)~x-1, weights=rep(weight, n*q), family="binomial")
beta.starting <- coefficients(fit)
beta.tuning <- t(chol(vcov(fit)))

A.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]

n.batch <- 100
batch.length <- 50
n.samples <- n.batch*batch.length

starting <- list("beta"=beta.starting, "phi"=rep(3/0.5,q), "A"=A.starting, "w"=0)
tuning <- list("beta"=beta.tuning, "phi"=rep(1,q), "A"=rep(0.1,length(A.starting)),
               "w"=0.5)
priors <- list("beta.Flat", "phi.Unif"=list(rep(3/0.75,q), rep(3/0.25,q)),
               "K.IW"=list(q+1, diag(0.1,q)))

m.1 <- spMvGLM(list(y.1~x.1-1, y.2~x.2-1),
               coords=coords, weights=matrix(weight,n,q),
               starting=starting, tuning=tuning, priors=priors,
           amcmc=list("n.batch"=n.batch,"batch.length"=batch.length,"accept.rate"=0.43),
               cov.model="exponential", n.report=25)

burn.in <- 0.75*n.samples
sub.samps <- burn.in:n.samples

print(summary(window(m.1$p.beta.theta.samples, start=burn.in))$quantiles[,c(3,1,5)])
```

```
beta.hat <- t(m.1$p.beta.theta.samples[sub.samps,1:length(B)])
w.hat <- m.1$p.w.samples[,sub.samps]

p.hat <- 1/(1+exp(-(x%*%beta.hat+w.hat)))

y.hat <- apply(p.hat, 2, function(x){rbinom(n*q, size=rep(weight, n*q), prob=p)})

y.hat.mu <- apply(y.hat, 1, mean)

##Unstack to get each response variable fitted values
y.hat.mu.1 <- y.hat.mu[seq(1,length(y.hat.mu),q)]
y.hat.mu.2 <- y.hat.mu[seq(2,length(y.hat.mu),q)]

##Take a look
par(mfrow=c(2,2))
surf <- mba.surf(cbind(coords,y.1),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Observed y.1 positive trials")
contour(surf, add=TRUE)
points(coords)
zlim <- range(surf[["z"]], na.rm=TRUE)

surf <- mba.surf(cbind(coords,y.hat.mu.1),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, zlim=zlim, main="Fitted y.1 positive trials")
contour(surf, add=TRUE)
points(coords)

surf <- mba.surf(cbind(coords,y.2),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Observed y.2 positive trials")
contour(surf, add=TRUE)
points(coords)
zlim <- range(surf[["z"]], na.rm=TRUE)

surf <- mba.surf(cbind(coords,y.hat.mu.2),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, zlim=zlim, main="Fitted y.2 positive trials")
contour(surf, add=TRUE)
points(coords)

## End(Not run)
```

---

spMvLM *Function for fitting multivariate Bayesian spatial regression models*

Description

The function spMvLM fits Gaussian multivariate Bayesian spatial regression models. Given a set of knots, spMvLM will also fit a *predictive process* model (see references below).

Usage

```
spMvLM(formula, data = parent.frame(), coords, knots,
        starting, tuning, priors, cov.model,
        modified.pp = TRUE, amcmc, n.samples,
        verbose=TRUE, n.report=100, ...)
```

Arguments

`formula`	a list of q symbolic regression model descriptions to be fit. See example below.
`data`	an optional data frame containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `spMvLM` is called.
`coords`	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing).
`knots`	either a $m \times 2$ matrix of the *predictive process* knot coordinates in R^2 (e.g., easting and northing) or a vector of length two or three with the first and second elements recording the number of columns and rows in the desired knot grid. The third, optional, element sets the offset of the outermost knots from the extent of the `coords`.
`starting`	a list with tags corresponding to `beta`, `A`, `phi`, and `nu`. Depending on the specification of the non-spatial residual, tags are `L` or `Psi` for a block diagonal or diagonal covariance matrix, respectively. The value portion of each tag is a vector that holds the parameter's starting values and are of length p for `beta` (where p is the total number of regression coefficients in the multivariate model), $\frac{q(q+1)}{2}$ for `A` and `L`, and q for `Psi`, `phi`, and `nu`. Here, `A` and `L` hold the lower-triangle elements in column major ordering of the Cholesky square root of the spatial and non-spatial cross-covariance matrices, respectively.
`tuning`	a list with tags `A`, `phi`, and `nu`. Depending on the specification of the non-spatial residual, tags are `L` or `Psi` for a block diagonal or diagonal covariance matrix, respectively. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution. For `A` and `L` the vectors are of length $\frac{q(q+1)}{2}$ and q for `Psi`, `phi`, and `nu`.
`priors`	a list with tags `beta.flat`, `beta.norm`, `K.iw`, `Psi.iw`, `Psi.ig`, `phi.unif`, and `nu.unif`. If the regression coefficients, i.e., `beta` vector, are assumed to follow a multivariate Normal distribution then pass the hyperparameters as a list of length two with the first and second elements corresponding to the mean vector and positive definite covariance matrix, respectively. If `beta` is assumed flat then no arguments are passed. The default is a flat prior. Use `Psi.iw` if the non-spatial residual covariance matrix is assumed block diagonal. Otherwise if the non-spatial residual covariance matrix is assumed diagonal then each of the q diagonal element are assumed to follow an inverse-Gamma in which case use `Psi.ig`. The hyperparameters of the inverse-Wishart, i.e., for cross-covariance matrices AA' `K.iw` and LL' `Psi.iw`, are passed as a list of length two, with the first and second elements corresponding to the *df* and $q \times q$ *scale* matrix, respectively. If `Psi.ig` is specified, the inverse-Gamma hyperparameters of the diagonal variance elements are pass using a list of length two with the first and second list elements consisting of vectors of the q *shape* and *scale* hyperparameters, respectively. The hyperparameters of the Uniform `phi.unif`, and `nu.unif`

	are also passed as a list of vectors with the first and second list elements corresponding to the lower and upper support, respectively.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`modified.pp`	a logical value indicating if the *modified predictive process* should be used (see references below for details). Note, if a predictive process model is not used (i.e., `knots` is not specified) then this argument is ignored.
`amcmc`	a list with tags `n.batch`, `batch.length`, and `accept.rate`. Specifying this argument invokes an adaptive MCMC sampler see Roberts and Rosenthal (2007) for an explanation.
`n.samples`	the number of MCMC iterations. This argument is ignored if `amcmc` is specified.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report Metropolis acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

Model parameters can be fixed at their `starting` values by setting their `tuning` values to zero.

The *no nugget* model is specified by removing `Psi` and `L` from the `starting` list.

Value

An object of class `spMvLM`, which is a list with the following tags:

`coords`	the $n \times 2$ matrix specified by `coords`.
`knot.coords`	the $m \times 2$ matrix as specified by `knots`.
`p.theta.samples`	a `coda` object of posterior samples for the defined parameters.
`acceptance`	the Metropolis sampling acceptance percent. Reported at `batch.length` or `n.report` intervals for `amcmc` specified and non-specified, respectively

The return object might include additional data used for subsequent prediction and/or model fit evaluation.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`baner009@umn.edu`>

References

Banerjee, S., A.E. Gelfand, A.O. Finley, and H. Sang. (2008) Gaussian Predictive Process Models for Large Spatial Datasets. *Journal of the Royal Statistical Society Series B*, 70:825–848.

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, Fla.

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. https://www.jstatsoft.org/article/view/v063i13.

Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2009) Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, 53:2873–2884.

Finley, A.O., S. Banerjee, A.R. Ek, and R.E. McRoberts. (2008) Bayesian multivariate process modeling for prediction of forest attributes. *Journal of Agricultural, Biological, and Environmental Statistics*, 13:60–83.

See Also

spLM

Examples

```
## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p)))){stop("Dimension problem!")}
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

##Generate some data
n <- 25 ##number of locations
q <- 2 ##number of outcomes at each location
nltr <- q*(q+1)/2 ##number of triangular elements in the cross-covariance matrix

coords <- cbind(runif(n,0,1), runif(n,0,1))

##Parameters for the bivariate spatial random effects
theta <- rep(3/0.5,q)

A <- matrix(0,q,q)
A[lower.tri(A,TRUE)] <- c(1,-1,0.25)
K <- A%*%t(A)

Psi <- diag(0,q)

C <- mkSpCov(coords, K, Psi, theta, cov.model="exponential")
```

```
w <- rmvn(1, rep(0,nrow(C)), C)

w.1 <- w[seq(1,length(w),q)]
w.2 <- w[seq(2,length(w),q)]

##Covariate portion of the mean
x.1 <- cbind(1, rnorm(n))
x.2 <- cbind(1, rnorm(n))
x <- mkMvX(list(x.1, x.2))

B.1 <- c(1,-1)
B.2 <- c(-1,1)
B <- c(B.1, B.2)

Psi <- diag(c(0.1, 0.5))

y <- rnorm(n*q, x%*%B+w, diag(n)%x%Psi)

y.1 <- y[seq(1,length(y),q)]
y.2 <- y[seq(2,length(y),q)]

##Call spMvLM
A.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]
n.samples <- 1000

starting <- list("phi"=rep(3/0.5,q), "A"=A.starting, "Psi"=rep(1,q))
tuning <- list("phi"=rep(1,q), "A"=rep(0.01,length(A.starting)), "Psi"=rep(0.01,q))
priors <- list("beta.Flat", "phi.Unif"=list(rep(3/0.75,q), rep(3/0.25,q)),
               "K.IW"=list(q+1, diag(0.1,q)), "Psi.ig"=list(c(2,2), c(0.1,0.1)))

m.1 <- spMvLM(list(y.1~x.1-1, y.2~x.2-1),
               coords=coords, starting=starting, tuning=tuning, priors=priors,
               n.samples=n.samples, cov.model="exponential", n.report=100)

burn.in <- 0.75*n.samples

m.1 <- spRecover(m.1, start=burn.in)

round(summary(m.1$p.theta.recover.samples)$quantiles[,c(3,1,5)],2)
round(summary(m.1$p.beta.recover.samples)$quantiles[,c(3,1,5)],2)

m.1.w.hat <- summary(mcmc(t(m.1$p.w.recover.samples)))$quantiles[,c(3,1,5)]
m.1.w.1.hat <- m.1.w.hat[seq(1, nrow(m.1.w.hat), q),]
m.1.w.2.hat <- m.1.w.hat[seq(2, nrow(m.1.w.hat), q),]

par(mfrow=c(1,2))
plot(w.1, m.1.w.1.hat[,1], xlab="Observed w.1", ylab="Fitted w.1",
     xlim=range(w), ylim=range(m.1.w.hat), main="Spatial random effects w.1")
arrows(w.1, m.1.w.1.hat[,1], w.1, m.1.w.1.hat[,2], length=0.02, angle=90)
arrows(w.1, m.1.w.1.hat[,1], w.1, m.1.w.1.hat[,3], length=0.02, angle=90)
lines(range(w), range(w))

plot(w.2, m.1.w.2.hat[,1], xlab="Observed w.2", ylab="Fitted w.2",
     xlim=range(w), ylim=range(m.1.w.hat), main="Spatial random effects w.2")
arrows(w.2, m.1.w.2.hat[,1], w.2, m.1.w.2.hat[,2], length=0.02, angle=90)
arrows(w.2, m.1.w.2.hat[,1], w.2, m.1.w.2.hat[,3], length=0.02, angle=90)
lines(range(w), range(w))

## End(Not run)
```

---

spPredict *Function for new locations given a model object*

Description

The function spPredict collects posterior predictive samples for a set of new locations given a spLM, spMvLM, spGLM, spMvGLM, spMisalignLM, spMisalignGLM, bayesGeostatExact, bayesLMConjugate bayesLMRef or spSVC object.

Usage

```
spPredict(sp.obj, pred.coords, pred.covars, joint=FALSE, start=1, end, thin=1,
          verbose=TRUE, n.report=100, n.omp.threads=1, ...)
```

Arguments

sp.obj	an object returned by spLM, spMvLM, spGLM, spMvGLM, spMisalignLM, spMisalignGLM, bayesGeostatExact, bayesLMConjugate or bayesLMRef. For spSVC, sp.obj is an object from spRecover.
pred.coords	for spLM, spMvLM, spGLM, spMvGLM, and bayesGeostatExact pred.coords is a $n^* \times 2$ matrix of n^* prediction location coordinates in R^2 (e.g., easting and northing). For spMisalignLM and spMisalignGLM pred.coords is a list of q $n^*_i \times 2$ matrices of prediction location coordinates where $i = (1, 2, \ldots, q)$. For spSVC pred.coords is an $n^* \times m$ matrix of n^* prediction location coordinates in R^m.
pred.covars	for spLM, spMvLM, spGLM, spMvGLM, bayesGeostatExact, bayesLMConjugate, bayesLMRef, and spSVC pred.covars is a $n^* \times p$ design matrix associated with the new locations (including the intercept if one is specified in sp.obj's formula argument). If this is a multivariate prediction defined by q models, i.e., for spMvLM or spMvGLM, the multivariate design matrix can be created by passing a list of the q univariate design matrices to the mkMvX function. For spMisalignLM and spMisalignGLM pred.covars is a list of q $n^*_i \times p_i$ design matrices where $i = (1, 2, \ldots, q)$
joint	specifies whether posterior samples should be drawn from the joint or point-wise predictive distribution. This argument is only implemented for spSVC. Prediction for all other models uses the point-wise posterior predictive distribution.
start	specifies the first sample included in the composition sampling.
end	specifies the last sample included in the composition. The default is to use all posterior samples in sp.obj.

`thin`	a sample thinning factor. The default of 1 considers all samples between `start` and `end`. For example, if `thin = 10` then 1 in 10 samples are considered between `start` and `end`.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report sampling progress.
`n.omp.threads`	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. This argument is only implemented for `spSVC`.
`...`	currently no additional arguments.

Value

`p.y.predictive.samples`

a matrix that holds the response variable(s) posterior predictive samples. For multivariate models `spMvLM` or `spMvGLM` the rows of this matrix correspond to the predicted locations and the columns are the posterior predictive samples. If prediction is for q response variables the `p.y.predictive.samples` matrix has qn^* rows, where n^* is the number of prediction locations. The predictions for locations are held in rows $1:q, (q+1):2q, \ldots, ((n^*-1)q+1):qn^*$ (i.e., the samples for the first location's q response variables are in rows $1:q$, second location in rows $(q+1):2q$, etc.).

For `spMisalignLM` and `spMisalignGLM` the posterior predictive samples are organized differently in `p.y.predictive.samples` with the first response variable n_1^* locations held in rows $1\ldots,n_1^*$ rows, then the next response variable samples held in the $(n_1^*+1),\ldots,(n_1^*+n_2^*)$, etc.

For `spSVC` given the r space-varying coefficients, `p.y.predictive.samples` has rn^* rows and the columns are the posterior predictive samples. The predictions for coefficient are held in rows $1:r, (r+1):2r, \ldots, ((n^*-1)r+1):rn^*$ (i.e., the samples for the first location's r regression coefficients are in rows 1:r, second location in rows $(r+1):2r$, etc.).

For `spGLM` and `spMisalignGLM` the `p.y.predictive.samples` matrix holds posterior predictive samples $\frac{1}{1+\exp(-x(s)'\beta-w(s))}$ and $\exp(x(s)'\beta+w(s))$ for `family` binomial and poisson, respectively. Here s indexes the prediction location, β is the vector of regression coefficients, and w is the associated spatial random spatial effect. These values can be fed directly into `rbinom` or `rpois` to generate the realization from the respective distribution.

`p.w.predictive.samples`

a matrix organized the same as `p.y.predictive.samples`, that holds the spatial random effects posterior predictive samples.

`p.w.predictive.samples.list`

only returned for `spSVC`. This provides `p.w.predictive.samples` in a different (more convenient form). Elements in this list hold samples for each of the r coefficients. List element names indicate either the coefficient index or name specified in `spSVC`'s `svc.cols` argument. The sample matrices are n^* rows and predictive samples along the columns.

`p.tilde.beta.predictive.samples.list`
: only returned for spSVC. Like `p.w.predictive.samples.list` but with the addition of the corresponding β posterior samples (i.e., $\beta + w(s)$).

`center.scale.pred.covars`
: only returned for the spSVC when its `center.scale` argument is `TRUE`. This is the prediction design matrix centered and scaled with respect to column means and variances of the design matrix used to estimate model parameters, i.e., the one defined in spSVC's formula argument.

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Sudipto Banerjee <sudipto@ucla.edu>

References

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, FL.

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. https://www.jstatsoft.org/article/view/v063i13.

Finley, A.O. and S. Banerjee (2019) Bayesian spatially varying coefficient models in the spBayes R package. https://arxiv.org/abs/1903.03028.

Examples

```
## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

n <- 200
coords <- cbind(runif(n,0,1), runif(n,0,1))
X <- as.matrix(cbind(1, rnorm(n)))

B <- as.matrix(c(1,5))
p <- length(B)
sigma.sq <- 10
tau.sq <- 0.01
phi <- 3/0.5

D <- as.matrix(dist(coords))
R <- exp(-phi*D)
w <- rmvn(1, rep(0,n), sigma.sq*R)
y <- rnorm(n, X%*%B + w, sqrt(tau.sq))
```

```
##partition the data for out of sample prediction
mod <- 1:100
y.mod <- y[mod]
X.mod <- X[mod,]
coords.mod <- coords[mod,]

n.samples <- 1000

starting <- list("phi"=3/0.5, "sigma.sq"=50, "tau.sq"=1)
tuning <- list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)
priors <- list("beta.Flat", "phi.Unif"=c(3/1, 3/0.1),
               "sigma.sq.IG"=c(2, 5), "tau.sq.IG"=c(2, 0.01))
cov.model <- "exponential"

m.1 <- spLM(y.mod~X.mod-1, coords=coords.mod, starting=starting, tuning=tuning,
priors=priors, cov.model=cov.model, n.samples=n.samples)

m.1.pred <- spPredict(m.1, pred.covars=X, pred.coords=coords,
start=0.5*n.samples)

y.hat <- apply(m.1.pred$p.y.predictive.samples, 1, mean)

quant <- function(x){quantile(x, prob=c(0.025, 0.5, 0.975))}

y.hat <- apply(m.1.pred$p.y.predictive.samples, 1, quant)

plot(y, y.hat[2,], pch=19, cex=0.5, xlab="observed y", ylab="predicted y")
arrows(y[-mod], y.hat[2,-mod], y[-mod], y.hat[1,-mod], angle=90, length=0.05)
arrows(y[-mod], y.hat[2,-mod], y[-mod], y.hat[3,-mod], angle=90, length=0.05)

## End(Not run)
```

`spRecover` *Function for recovering regression coefficients and spatial random effects for `spLM`, `spMvLM`, `spMisalignLM`, `spSVC` using composition sampling*

Description

Function for recovering regression coefficients and spatial random effects for `spLM`, `spMvLM`, and `spMisalignLM` using composition sampling.

Usage

```
spRecover(sp.obj, get.beta=TRUE, get.w=TRUE, start=1, end, thin=1,
          verbose=TRUE, n.report=100, n.omp.threads=1, ...)
```

Arguments

`sp.obj`	an object returned by `spLM`, `spMvLM`, `spMisalignLM`, or `spSVC`.
`get.beta`	if `TRUE`, regression coefficients will be recovered.
`get.w`	if `TRUE`, spatial random effects will be recovered.
`start`	specifies the first sample included in the composition sampling.
`end`	specifies the last sample included in the composition. The default is to use all posterior samples in `sp.obj`.
`thin`	a sample thinning factor. The default of 1 considers all samples between `start` and `end`. For example, if `thin = 10` then 1 in 10 samples are considered between `start` and `end`.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report sampling progress.
`n.omp.threads`	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. This argument is only implemented for `spSVC`.
`...`	currently no additional arguments.

Value

The input `sp.obj` with posterior samples of regression coefficients and/or spatial random effects appended. tags:

`p.theta.recover.samples`
: those `p.theta.samples` used in the composition sampling.

`p.beta.recover.samples`
: a coda object of regression coefficients posterior samples.

`p.w.recover.samples`
: a coda object of spatial random effects posterior samples. Rows correspond to locations' random effects and columns are posterior samples. Given q responses, the `p.w.recover.samples` matrix for `spMvLM` has qn rows. The recovered random effects for locations are held in rows $1 : q, (q+1) : 2q, \ldots, ((n-1)q+1) : qn$ (i.e., the samples for the first location's q response variables are in rows 1:q, second location in rows $(q + 1) : 2q$, etc.).

  For `spSVC` given the r space-varying coefficients, `p.w.recover.samples` has rn rows. The recovered random effects for locations are held in rows $1 : r, (r + 1) : 2r, \ldots, ((n - 1)r + 1) : rn$ (i.e., the samples for the first location's r regression coefficients are in rows 1:r, second location in rows $(r + 1) : 2r$, etc.).

`p.w.recover.samples.list`
: only returned for `spSVC`. This provides `p.w.recover.samples` in a different (more convenient form). Elements in this list hold samples for each of the r coefficients. List element names indicate either the coefficient index or name specified in `spSVC`'s `svc.cols` argument. The sample matrices are n rows and predictive samples along the columns.

`p.tilde.beta.recover.samples.list`
: only returned for `spSVC`. Like `p.w.recover.samples.list` but with the addition of the corresponding β posterior samples (i.e., $\beta + w(s)$).

`p.y.samples`
: only returned for `spSVC`. These posterior are the fitted values with locations on the rows and samples on the columns. For a given sample the fitted value for the i^{th} location is $N(x(s_i)\beta + z(s_i)w(s_i), \tau^2)$.

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Sudipto Banerjee <sudipto@ucla.edu>

References

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). Hierarchical modeling and analysis for spatial data. Chapman and Hall/CRC Press, Boca Raton, FL.

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. https://www.jstatsoft.org/article/view/v063i13.

Finley, A.O. and S. Banerjee (2019) Bayesian spatially varying coefficient models in the spBayes R package. https://arxiv.org/abs/1903.03028.

Examples

```
## Not run:
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)

n <- 50
coords <- cbind(runif(n,0,1), runif(n,0,1))
X <- as.matrix(cbind(1, rnorm(n)))

B <- as.matrix(c(1,5))
p <- length(B)
sigma.sq <- 10
tau.sq <- 0.01
phi <- 3/0.5

D <- as.matrix(dist(coords))
R <- exp(-phi*D)
w <- rmvn(1, rep(0,n), sigma.sq*R)
y <- rnorm(n, X%*%B + w, sqrt(tau.sq))

n.samples <- 1000
```

```
starting <- list("phi"=3/0.5, "sigma.sq"=50, "tau.sq"=1)
tuning <- list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)
priors <- list("beta.Flat", "phi.Unif"=c(3/1, 3/0.1),
               "sigma.sq.IG"=c(2, 5), "tau.sq.IG"=c(2, 0.01))
cov.model <- "exponential"

m.1 <- spLM(y~X-1, coords=coords, starting=starting, tuning=tuning,
            priors=priors, cov.model=cov.model, n.samples=n.samples)

m.1 <- spRecover(m.1, start=0.5*n.samples, thin=2)

summary(window(m.1$p.beta.recover.samples))

w.hat <- apply(m.1$p.w.recover.samples, 1, mean)
plot(w, w.hat, xlab="Observed w", ylab="Fitted w")

## End(Not run)
```

spSVC *Function for fitting univariate Bayesian spatially-varying coefficient regression models*

Description

The function spSVC fits Gaussian univariate Bayesian spatially-varying coefficient regression models.

Usage

```
spSVC(formula, data = parent.frame(), svc.cols=1, coords,
      priors, starting, tuning, cov.model, center.scale=FALSE,
      amcmc, n.samples, n.omp.threads = 1,
      verbose=TRUE, n.report=100, ...)
```

Arguments

formula	a symbolic description of the regression model to be fit. See example below.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which spSVC is called.
svc.cols	a vector indicating which columns of the regression design matrix X should be space-varying. svc.cols can be an integer vector with values indicating X columns or a character vector with values corresponding to X column names. svc.cols default argument of 1 results in a space-varying intercept model (assuming an intercept is specified in the first column of the design matrix).
coords	an $n \times m$ matrix of the observation coordinates in R^m (e.g., R^2 might be easting and northing).

`priors`	a list with each tag corresponding to a parameter name. Valid tags are `sigma.sq.ig`, `k.iw`, `tau.sq.ig`, `phi.unif`, `nu.unif`, `beta.norm`, and `beta.flat`. Scalar variance parameters `simga.sq` and `tau.sq` are assumed to follow an inverse-Gamma distribution. Cross-covariance matrix parameter `K` is assumed to follow an inverse-Wishart. The spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The regression coefficient priors can be either flat or multivariate Normal. There are two specification for the Gaussian Process (GP) on the `svc.cols` columns: 1) univariate GPs on each column; 2) multivariate GP on the r columns (i.e., where r equals `length(svc.cols)`). If univariate GPs are desired, specify `sigma.sq.ig` as a list of length two with the first and second elements corresponding to the length r *shape* and *scale* hyperparameter vectors, respectively. If a multivariate GP is desired, specify `k.iw` as a list of length two with the first and second elements corresponding to the degrees-of-freedom *df* and $r \times r$ *scale* matrix, respectively. This inverse-Wishart prior is on the $r \times r$ multivariate GP cross-covariance matrix defined as $K = AA'$ where A is the lower-triangle Cholesky square root of K. If the regression coefficients, i.e., `beta` vector, are assumed to follow a multivariate Normal distribution then pass the hyperparameters as a list of length two with the first and second elements corresponding to the mean vector and positive definite covariance matrix, respectively. If `beta` is assumed flat then no arguments are passed. The default is a flat prior. Similarly, `phi` and `nu` are specified as lists of length two with the first and second elements holding vectors of length r lower and upper bounds of the Uniforms' support, respectively.
`starting`	a list with each tag corresponding to a parameter name. Valid tags are `beta`, `sigma.sq`, `A`, `tau.sq`, `phi`, and `nu`. The value portion of each tag is the parameter's starting value(s). Starting values must be set for the r univariate or multivariate GP `phi` and `nu`. For univariate GPs `sigma.sq.ig` is specified as a vector of length r and for a multivariate GP `A` is specified as a vector of $r \times (r+1)/2$ that gives the lower-triangle elements in column major ordering of the Cholesky square root of the cross-covaraince matrix $K = AA'$. `tau.sq` is a single value. See Finley and Banerjee (2019) for more details.
`tuning`	a list with each tag corresponding to a parameter name. Valid tags are `sigma.sq`, `A`, `tau.sq`, `phi`, and `nu`. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution. For `sigma.sq`, `phi`, and `nu` the tuning value vectors are of length r and `A` is of length $r \times (r+1)/2$. `tuning` vector elements correspond to `starting` vector elements. `tau.sq` is a single value.
`cov.model`	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.
`center.scale`	if `TRUE`, non-constant columns of X are centered on zero and scaled to have variance one. If `spPredict` is subsequently called this centering and scaling is applied to `pred.covars`.
`amcmc`	a list with tags `n.batch`, `batch.length`, and `accept.rate`. Specifying this argument invokes an adaptive MCMC sampler, see Roberts and Rosenthal (2007)

	for an explanation.
`n.samples`	the number of MCMC iterations. This argument is ignored if `amcmc` is specified.
`n.omp.threads`	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report Metropolis sampler acceptance and MCMC progress.
`...`	currently no additional arguments.

Details

Model parameters can be fixed at their `starting` values by setting their `tuning` values to zero.

The *no nugget* model is specified by removing `tau.sq` from the `starting` list.

Value

An object of class `spSVC`, which is a list comprising:

`coords`	the $n \times m$ matrix specified by `coords`.
`p.theta.samples`	a coda object of posterior samples for the defined parameters.
`acceptance`	the Metropolis sampling acceptance percent. Reported at `batch.length` or `n.report` intervals for `amcmc` specified and non-specified, respectively.

The return object will include additional objects used for subsequent parameter recovery, prediction, and model fit evaluation using `spRecover`, `spPredict`, `spDiag`, respectively.

Author(s)

Andrew O. Finley <`finleya@msu.edu`>,
Sudipto Banerjee <`sudipto@ucla.edu`>

References

Finley, A.O., S. Banerjee, and A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63:1–28. `https://www.jstatsoft.org/article/view/v063i13`.

Roberts G.O. and Rosenthal J.S. (2006). Examples of Adaptive MCMC. `http://probability.ca/jeff/ftpdir/adaptex.pdf`.

Finley, A.O. and S. Banerjee (2019) Bayesian spatially varying coefficient models in the spBayes R package. `https://arxiv.org/abs/1903.03028`.

See Also

`spRecover`, `spDiag`, `spPredict`

Examples

```
## Not run:

library(Matrix)

rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}


##Assume both columns of X are space-varying and the two GPs don't covary
set.seed(1)
n <- 200
coords <- cbind(runif(n,0,1), runif(n,0,1))

X <- as.matrix(cbind(1, rnorm(n)))
colnames(X) <- c("x.1", "x.2")

Z <- t(bdiag(as.list(as.data.frame(t(X)))))

B <- as.matrix(c(1,5))
p <- length(B)

sigma.sq <- c(1,5)
tau.sq <- 1
phi <- 3/0.5

D <- as.matrix(dist(coords))

C <- exp(-phi*D)%x%diag(sigma.sq)

w <- rmvn(1, rep(0,p*n), C)

mu <- as.vector(X%*%B + Z%*%w)

y <- rnorm(n, mu, sqrt(tau.sq))

##fit a model to the simulated dat
starting <- list("phi"=rep(3/0.5, p), "sigma.sq"=rep(1, p), "tau.sq"=1)

tuning <- list("phi"=rep(0.1, p), "sigma.sq"=rep(0.1, p), "tau.sq"=0.1)

cov.model <- "exponential"

priors <- list("phi.Unif"=list(rep(3/2, p), rep(3/0.0001, p)),
               "sigma.sq.IG"=list(rep(2, p), rep(2, p)),
               "tau.sq.IG"=c(2, 1))
```

```
##fit model
n.samples <- 2000

m.1 <- spSVC(y~X-1, coords=coords, starting=starting, svc.cols=c(1,2),
             tuning=tuning, priors=priors, cov.model=cov.model,
             n.samples=n.samples, n.omp.threads=4)

plot(m.1$p.theta.samples, density=FALSE)

##recover posterior samples
m.1 <- spRecover(m.1, start=floor(0.75*n.samples), thin=2, n.omp.threads=4)

summary(m.1$p.beta.recover.samples)
summary(m.1$p.theta.recover.samples)

##check fitted values
quant <- function(x){quantile(x, prob=c(0.025, 0.5, 0.975))}

##fitted y
y.hat <- apply(m.1$p.y.samples, 1, quant)

rng <- c(-15, 20)
plot(y, y.hat[2,], pch=19, cex=0.5, xlab="Fitted y", ylab="Observed y",
     xlim=rng, ylim=rng)
arrows(y, y.hat[2,], y, y.hat[1,], angle=90, length=0.05)
arrows(y, y.hat[2,], y, y.hat[3,], angle=90, length=0.05)
lines(rng, rng, col="blue")

##recovered w
w.hat <- apply(m.1$p.w.recover.samples, 1, quant)

w.1.indx <- seq(1, p*n, p)
w.2.indx <- seq(2, p*n, p)

par(mfrow=c(1,2))

rng <- c(-5,5)
plot(w[w.1.indx], w.hat[2,w.1.indx], pch=19, cex=0.5, xlab="Fitted w.1", ylab="Observed w.1",
     xlim=rng, ylim=rng)
arrows(w[w.1.indx], w.hat[2,w.1.indx], w[w.1.indx], w.hat[1,w.1.indx], angle=90, length=0.05)
arrows(w[w.1.indx], w.hat[2,w.1.indx], w[w.1.indx], w.hat[3,w.1.indx], angle=90, length=0.05)
lines(rng, rng, col="blue")

rng <- c(-10,10)
plot(w[w.2.indx], w.hat[2,w.2.indx], pch=19, cex=0.5, xlab="Fitted w.2", ylab="Observed w.2",
     xlim=rng, ylim=rng)
arrows(w[w.2.indx], w.hat[2,w.2.indx], w[w.2.indx], w.hat[1,w.2.indx], angle=90, length=0.05)
arrows(w[w.2.indx], w.hat[2,w.2.indx], w[w.2.indx], w.hat[3,w.2.indx], angle=90, length=0.05)
lines(rng, rng, col="blue")

## End(Not run)
```

SVCMvData.dat *Synthetic data from a space-varying coefficients model*

Description

Data simulated from a space-varying coefficients model.

Usage

```
data(SVCMvData.dat)
```

Format

The data frame generated from the code in the example section below.

Examples

```
## Not run:
##The dataset was generated with the code below.

library(Matrix)

rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

set.seed(1)
n <- 200

coords <- cbind(runif(n,0,1), runif(n,0,1))
colnames(coords) <- c("x.coords","y.coords")

X <- as.matrix(cbind(1, rnorm(n), rnorm(n)))
colnames(X) <- c("intercept","a","b")

Z <- t(bdiag(as.list(as.data.frame(t(X)))))

beta <- c(1, 10, -10)
p <- length(beta)

q <- 3
A <- matrix(0, q, q)
A[lower.tri(A, T)] <- c(1, -1, 0, 1, 1, 0.1)
K <- A
K
```

```
cov2cor(K)

phi <- c(3/0.75, 3/0.5, 3/0.5)

Psi <- diag(0,q)
C <- mkSpCov(coords, K, Psi, phi, cov.model="exponential")

tau.sq <- 0.1

w <- rmvn(1, rep(0,q*n), C)

y <- rnorm(n, as.vector(X%*%beta + Z%*%w), sqrt(tau.sq))

w.0 <- w[seq(1, length(w), by=q)]
w.a <- w[seq(2, length(w), by=q)]
w.b <- w[seq(3, length(w), by=q)]

SVCMvData <- data.frame(cbind(coords, y, X[,2:3], w.0, w.a, w.b))


## End(Not run)
```

WEF.dat *Western Experimental Forest inventory data*

Description

Data generated as part of a long-term research study on an experimental forest in central Oregon. This dataset holds the coordinates for all trees in the experimental forest. The typical stem measurements are recorded for each tree. Crown radius was measured at the cardinal directions for a subset of trees. Mean crown radius was calculated for all trees using a simple relationship between DBH, Height, and observed crown dimension.

Usage

```
data(WEF.dat)
```

Format

A data frame containing 2422 rows and 15 columns.

`Zurich.dat` *Zurichberg Forest inventory data*

Description

Inventory data of the Zurichberg Forest, Switzerland (see Mandallaz 2008 for details). These data are provided with the kind authorization of the Forest Service of the Caton of Zurich.

This dataset holds the coordinates for all trees in the Zurichberg Forest. Species (SPP), basal area (BAREA) diameter at breast height (DBH), and volume (VOL) are recorded for each tree. See species codes below.

Usage

```
data(Zurich.dat)
```

Format

A data frame containing 4954 rows and 6 columns.

Examples

```
## Not run:
data(Zurich.dat)

coords <- Zurich.dat[,c("X_TREE", "Y_TREE")]

spp.name <- c("beech","maple","ash","other broadleaves",
              "spruce","silver fir", "larch", "other coniferous")

spp.col <- c("yellow","red","orange","pink",
             "green","dark green","black","gray")

plot(coords, col=spp.col[Zurich.dat$SPP+1],
     pch=19, cex=0.5, ylab="Northing", xlab="Easting")

legend.coords <- c(23,240)

legend(legend.coords, pch=19, legend=spp.name,
       col=spp.col, bty="n")



## End(Not run)
```

Index