

# Package ‘spBayes’

January 13, 2012

**Version** 0.2-2

**Date** 2011-5-30

**Title** Univariate and Multivariate Spatial Modeling

**Author** Andrew O. Finley <finleya@msu.edu>, Sudipto Banerjee  
<sudiptob@biostat.umn.edu>, Bradley P. Carlin <brad@biostat.umn.edu>

**Maintainer** Andrew O. Finley <finleya@msu.edu>

**Depends** R (>= 1.8.0), coda, magic, lattice, abind

**Suggests** MBA, geoR, MASS

**Description** spBayes fits univariate and multivariate models with  
Markov chain Monte Carlo (MCMC).

**License** GPL (>= 2)

**URL** <http://blue.for.msu.edu/software>

**Repository** CRAN

**Date/Publication** 2012-01-13 21:35:54

## R topics documented:

adaptMetropGibbs	2
bayesGeostatExact	7
bayesLMConjugate	12
bayesLMRef	14
BEF.dat	15
covGivens	16
covInvLogDet	17
FBC07.dat	20
FORMGMT.dat	22
hexGrid	22
iDist	23

mkMvX . . . . .	24
mvCovInvLogDet . . . . .	25
mvLM . . . . .	28
pointsInPoly . . . . .	31
prior . . . . .	33
spDiag . . . . .	34
spGGT . . . . .	36
spGLM . . . . .	45
spLM . . . . .	52
spMvGLM . . . . .	56
spMvLM . . . . .	63
spPredict . . . . .	67
synthetic.dat . . . . .	78
WEF.dat . . . . .	79
Zurich.dat . . . . .	80

<b>Index</b>	<b>81</b>
--------------	-----------

---

adaptMetropGibbs	<i>Adaptive Metropolis within Gibbs algorithm</i>
------------------	---

---

## 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.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/ftpd/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:
#####
##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 <- mvrnorm(1, rep(0, n), s)
Y <- mvrnorm(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$p.samples[,2] <- exp(chain.1$p.samples[,2])
chain.1$p.samples[,3] <- exp(chain.1$p.samples[,3])
chain.1$p.samples[,4] <- 3/logit.inv(chain.1$p.samples[,4], a.phi, b.phi)

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

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

##Discard burn.in and plot and do some convergence diagnostics
chains <- mcmc.list(mcmc(chain.1$p.samples), mcmc(chain.2$p.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$p.samples[,1], exp(run.1$p.samples[,2:4]))
samples.2 <- cbind(run.2$p.samples[,1], exp(run.2$p.samples[,2:4]))
samples.3 <- cbind(run.3$p.samples[,1], exp(run.3$p.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.quantiles <- summary(mcmc(post.curves))$quantiles

plot(x, y, pch=19, xlab="x", ylab="f(x)")
lines(x, post.curves.quantiles[,1], lty="dashed", col="blue")
lines(x, post.curves.quantiles[,3])
lines(x, post.curves.quantiles[,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 <code>environment(formula)</code> , typically the environment from which <code>spLM</code> is called.
n.samples	the number of posterior samples to collect.
beta.prior.mean	$\beta$ multivariate normal mean vector hyperprior.
beta.prior.precision	$\beta$ 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 <code>cov.model</code> is "matern" then the fixed value of the spatial process smoothness must be specified.
alpha	the fixed value of the ratio between the nugget $\tau^2$ and partial-sill $\sigma^2$ parameters from the specified <code>cov.model</code> .
sigma.sq.prior.shape	$\sigma^2$ (i.e., partial-sill) inverse-Gamma shape hyperprior.
sigma.sq.prior.rate	$\sigma^2$ (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)

#####
##Now with synthetic data
#####
data(rf.n1000.dat)

```

```

Y <- rf.n1000.dat$Y
coords <- as.matrix(rf.n1000.dat[,c("x.coords", "y.coords")])
w <- rf.n1000.dat$w
p <- 1
alpha <- 1/10
phi <- 3/6

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

sigma.sq.prior.shape <- 2.0
sigma.sq.prior.rate <- 1/10.0

n.samples <- 50

m.5 <-
  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.5$p.samples))

##Requires MBA package to
##make surfaces
library(MBA)
par(mfrow=c(1,2))
obs.surf <-
  mba.surf(cbind(coords, w),
           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.5$sp.effects)
w.surf <-
  mba.surf(cbind(coords, w.hat), no.X=100, no.Y=100, extend=TRUE)$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	<i>Simple Bayesian linear model via the Normal/inverse-Gamma conjugate</i>
------------------	--

---

## 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

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

## Value

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

<code>p.samples</code>	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.samples)

## End(Not run)
```

---

 bayesLMRef

*Simple Bayesian linear model with non-informative priors*


---

**Description**

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

**Usage**

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

**Arguments**

<code>lm.obj</code>	an object returned by <code>lm</code> .
<code>n.samples</code>	the number of posterior samples to collect.
<code>...</code>	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.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**

```
data(FORMGMT.dat)

lm.obj <- lm(Y ~ X1+X2+X3+X4+X5+X6, data = FORMGMT.dat)

summary(lm.obj)

##Now with bayesLMRef
n.samples <- 500

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

summary(m.1$p.sample)
```

---

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>

---

covGivens	<i>Returns covariance matrix C given components of P and L in C=PLP'</i>
-----------	--

---

**Description**

Given components of  $P$  and  $\Lambda$  in  $C = P\Lambda P^T$ , covGivens returns  $C$ .

**Usage**

```
covGivens(theta, lambda, p, strict=TRUE)
```

**Arguments**

theta	a $p(p-1)/2$ vector of Givens angles that are restricted to lie in the interval $(-\pi/2, \pi/2)$ for uniqueness and positive definiteness of the covariance matrix. These are arranged in column major order in $P$ .
lambda	a vector of length $p$ of positive eigenvalues which form the diagonal matrix $\Lambda$ .
p	refers to the dimension of the desired $p \times p$ covariance matrix.
strict	checks that theta elements lie in the interval $(-\pi/2, \pi/2)$ and the lambda elements are positive.

**Details**

The spectral decomposition of a matrix  $C$  is given by  $P\Lambda P^T$ , where  $P$  is an orthogonal matrix of eigenvectors and  $\Lambda$  is a diagonal matrix of positive eigenvalues. In the case of distinct eigenvalues,  $P$  can be expressed as  $P = G_{12}, G_{13}, G_{23}, \dots, G_{p-1,p}$  where  $G_{ij}$  is a  $p \times p$  matrix with  $\cos(\theta_{il})$  in the  $i$ th and  $l$ th diagonal elements,  $\sin(\theta_{il})$  in the  $il$ th and  $-sin(\theta_{il})$  in the  $li$ th elements, zeros on the rest of the off-diagonal elements and 1's on the rest of the diagonal. The  $p(p-1)/2$  angles are called Givens angles and are restricted to lie in the interval  $(-\pi/2, \pi/2)$  for uniqueness and positive definiteness of the resulting covariance matrix.

**Value**

A list with the following tags:

C                     $p \times p$  covariance matrix.  
 P                     $p \times p$  P matrix.  
 Lambda              diagonal elements of the  $p \times p$   $\Lambda$  matrix.

**Author(s)**

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

**References**

Daniels, M.J and R.E. Kass (1999) Nonconjugate Bayesian estimation of covariance matrices and its use in hierarchical models, J. Amer. Statist. Assoc. 94:1254-1263.

**Examples**

```
## Not run:
#####
##Syntetic covariance matrix IIIR_1 from
##Daniels, M.J. and R.E. Kass (1999)
##JASA 94(448):1254-1263.
#####
p <- 5
theta <- rep(pi/4, p*(p-1)/2)
lambda <- c(1, 0.75, 0.56, 0.06, 0.003)

cov <- covGivens(theta, lambda, p)$C
cov2cor(cov)

## End(Not run)
```

---

covInvLogDet	<i>Utility function for constructing predictive process covariance matrices</i>
--------------	---

---

**Description**

Utility function for constructing predictive process covariance matrices

**Usage**

```
covInvLogDet(coords, knots, cov.model, sigma.sq,
              tau.sq, theta, modified.pp=TRUE, SWM=TRUE, ...)
```

**Arguments**

coords	a $n \times 2$ matrix of the observation coordinates in $R^2$ (e.g., easting and northing).
knots	a $m \times 2$ matrix of knot coordinates.
sigma.sq	partial sill value.
tau.sq	nugget value.
theta	if cov.model is matern then theta is a vector of the spatial decay and smoothness parameters; otherwise, theta is the spatial decay.
modified.pp	a logical value indicating if the <i>modified predictive process</i> should be used.
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.
SWM	a logical value indicating if the Sherman-Woodbury-Morrison equation should be used for computing the inverse (this is ignored if modified.pp is FALSE).
...	currently no additional arguments.

**Value**

Returns the covariance matrix, log-determinant, and inverse.

**Author(s)**

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

**Examples**

```
set.seed(1)

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

m <- 50
knots <- cbind(runif(m), runif(m))

theta <- c(6, 2)
sigma.sq <- 1
tau.sq <- 1
tol <- 1.0e-5

##
##non bias-adjusted predictive process
##
c1 <- covInvLogDet(coords=coords, knots=knots, cov.model="exponential",
  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
  modified.pp=FALSE, SWM=FALSE)
```

```

if(max(abs(chol2inv(chol(c1$C)) - c1$C.inv)) > tol) stop("test-1 failed")
if(max(abs(2*sum(log(diag(chol(c1$C)))) - c1$log.det)) > tol)
stop("test-1 failed")

c2 <- covInvLogDet(coords=coords, knots=knots, cov.model="matern",
                  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
                  modified.pp=FALSE, SWM=FALSE)

if(max(abs(chol2inv(chol(c2$C)) - c2$C.inv)) > tol) stop("test-2 failed")
if(max(abs(2*sum(log(diag(chol(c2$C)))) - c2$log.det)) > tol)
stop("test-2 failed")

##
##bias-adjusted predictive process
##
c3 <- covInvLogDet(coords=coords, knots=knots, cov.model="exponential",
                  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
                  modified.pp=TRUE, SWM=FALSE)

if(max(abs(chol2inv(chol(c3$C)) - c3$C.inv)) > tol) stop("test-3 failed")
if(max(abs(2*sum(log(diag(chol(c3$C)))) - c3$log.det)) > tol)
stop("test-3 failed")

c4 <- covInvLogDet(coords=coords, knots=knots, cov.model="matern",
                  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
                  modified.pp=TRUE, SWM=FALSE)

if(max(abs(chol2inv(chol(c4$C)) - c4$C.inv)) > tol) stop("test-4 failed")
if(max(abs(2*sum(log(diag(chol(c4$C)))) - c4$log.det)) > tol)
stop("test-4 failed")

##
##non bias-adjusted predictive process using SWM
##
c5 <- covInvLogDet(coords=coords, knots=knots, cov.model="exponential",
                  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
                  modified.pp=FALSE, SWM=TRUE)

if(max(abs(chol2inv(chol(c5$C)) - c5$C.inv)) > tol) stop("test-5 failed")
if(max(abs(2*sum(log(diag(chol(c5$C)))) - c5$log.det)) > tol)
stop("test-5 failed")

c6 <- covInvLogDet(coords=coords, knots=knots, cov.model="matern",
                  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
                  modified.pp=FALSE, SWM=TRUE)

if(max(abs(chol2inv(chol(c6$C)) - c6$C.inv)) > tol) stop("test-6 failed")
if(max(abs(2*sum(log(diag(chol(c6$C)))) - c6$log.det)) > tol)
stop("test-6 failed")

##

```

```

##Check SWM
##
##c.str <- sigma.sq*exp(-theta[1]*iDist(knots))
##ct <- sigma.sq*exp(-theta[1]*iDist(coords,knots))
##C.inv.swm <- diag(1/tau.sq,n)-diag(1/tau.sq,n)/%*%/ct/%*%/
## chol2inv(chol(c.str+t(ct)/%*%/diag(1/tau.sq,n)/%*%/ct)/%*%/
## t(ct)/%*%/diag(1/tau.sq,n)
##if(max(abs(C.inv.swm - c5$C.inv)) > tol) stop("test-5a failed")

##
##bias-adjusted predictive process using SWM
##
c7 <- covInvLogDet(coords=coords, knots=knots, cov.model="exponential",
                  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
                  modified.pp=TRUE, SWM=TRUE)

if(max(abs(chol2inv(chol(c7$C)) - c7$C.inv)) > tol) stop("test-7 failed")
if(max(abs(2*sum(log(diag(chol(c7$C)))) - c7$log.det)) > tol)
stop("test-7 failed")

c8 <- covInvLogDet(coords=coords, knots=knots, cov.model="matern",
                  sigma.sq=sigma.sq, tau.sq=tau.sq, theta=theta,
                  modified.pp=TRUE, SWM=TRUE)

if(max(abs(chol2inv(chol(c8$C)) - c8$C.inv)) > tol) stop("test-8 failed")
if(max(abs(2*sum(log(diag(chol(c8$C)))) - c8$log.det)) > tol)
stop("test-8 failed")

```

---

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*.

**Examples**

```
## Not run:

data(FBC07.dat)
library(geoR)

max <- 40
bins <- 20
pts <- 1:150

vario.1 <- variog(coords=FBC07.dat[pts,1:2], data=FBC07.dat[pts,3],
                 uvec=(seq(0, max, length=bins)))

vario.2 <- variog(coords=FBC07.dat[pts,1:2], data=FBC07.dat[pts,4],
                 uvec=(seq(0,max, length=bins)))

vario.fit.1 <-variofit(vario.1, ini.cov.pars=c(5.0, 1.0),
                     cov.model="exponential",
                     minimisation.function="nls",
                     weights="equal")

vario.fit.2 <-variofit(vario.2, ini.cov.pars=c(5.0, 10.0),
                     cov.model="exponential",
                     minimisation.function="nls",
                     weights="equal")

par(mfrow=c(1,2))

plot(vario.1$u, vario.1$v, axes=FALSE, type = "n",
     ylim=c(0,15), xlab="Distance", ylab="Semivariance")
points(vario.1$u, vario.1$v, pch=19, cex=0.5)
axis(1, seq(0,max,10))
axis(2, seq(0,15,5))
abline(h=vario.fit.1$nugget)
abline(h=vario.fit.1$cov.pars[1]+vario.fit.1$nugget)
abline(v=3/(1/vario.fit.1$cov.pars[2]))
lines(vario.fit.1)

plot(vario.2$u, vario.2$v, axes=FALSE, type = "n",
     ylim=c(0,15), xlab="Distance", ylab="")
points(vario.2$u, vario.2$v, pch=19, cex=0.5)
axis(1, seq(0,max,10))
abline(h=vario.fit.2$nugget)
abline(h=vario.fit.2$cov.pars[1]+vario.fit.2$nugget)
abline(v=3/(1/vario.fit.2$cov.pars[2]))
lines(vario.fit.2)

## End(Not run)
```

---

FORMGMT.dat	<i>Data used for illustrations</i>
-------------	------------------------------------

---

**Description**

Data used for illustrations.

**Usage**

data(FORMGMT.dat)

---

hexGrid	<i>Generates a hexagon tessellation within a bounding box</i>
---------	---

---

**Description**

This function generates a hexagon tessellation within a bounding box.

**Usage**

hexGrid(nodes.per.layer, b.box, ...)

**Arguments**

nodes.per.layer	the number of hexagons from min.x to max.x at min.y.
b.box	the domain's bounding box defined with a vector of length four and elements ordered min.x, min.y, max.x, max.y.
...	currently no additional arguments.

**Value**

A list with the following tags:

hx.hy	a vector of length two that holds the height and width, respectively, between hexagon centroids.
layers	the number of hexagon layers within the domain.
nodes.per.layer	the number of hexagons from min.x to max.x at min.y.
hex.centroids	an $n \times 2$ matrix with rows corresponding to the $x$ and $y$ coordinates of the $n$ hexagons within the domain.
hex.polygons	a list with $6 \times 2$ matrices which hold the hexagons' $x$ and $y$ vertices coordinates.

**Author(s)**

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

**Examples**

```
## Not run:

##Define the bounding box and make the call
b.box <- c(0,0,10,10)

out <- hexGrid(20, b.box)

##Plot using lapply
par(mfrow=c(1,2))

plot(out$hex.centroids, pch=19, cex=0.5,
      ylab="Northing", xlab="Easting")
lapply(out$hex.polygons, polygon, col="blue")

##Now color hexagons based on value
my.col.ramp <- function(z){
  zlim <- range(z)
  zlen <- zlim[2]-zlim[1]+1
  colorlut <- heat.colors(as.integer(zlen))
  col <- colorlut[z-zlim[1]+1]
  col
}

n <- nrow(out$hex.centroids)
col <- my.col.ramp(rnorm(n))

plot(out$hex.centroids, typ="n", ylab="", xlab="Easting")
for(i in 1:n){
  polygon(out$hex.polygons[[i]], col=col[i], border=col[i])
}

## End(Not run)
```

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

**Author(s)**

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

**Examples**

```
n <- 10
p1 <- cbind(runif(n),runif(n))

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

D <- iDist(p1, p2)
```

---

mkMvX

*Make a multivariate design matrix*

---

**Description**

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

**Usage**

```
mkMvX(X)
```

**Arguments**

`X` a list of  $m$  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**

```
##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))
```

---

mvCovInvLogDet	<i>Utility function for constructing multivariate predictive process covariance matrices</i>
----------------	--

---

**Description**

Utility function for constructing multivariate predictive process covariance matrices

**Usage**

```
mvCovInvLogDet(coords, knots, cov.model, V,
               Psi, theta, modified.pp=TRUE, SWM=TRUE, ...)
```

**Arguments**

coords	a $n \times 2$ matrix of the observation coordinates in $R^2$ (e.g., easting and northing).
knots	a $m \times 2$ matrix of knot coordinates.
V	spatial cross-covariance matrix.
Psi	residual cross-covariance matrix.
theta	if cov.model is matern then theta is a vector of the multivariate spatial decays followed by the associated smoothness parameters; otherwise, theta is the spatial decays vector.
modified.pp	a logical value indicating if the <i>modified predictive process</i> should be used.

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.
SWM	a logical value indicating if the Sherman-Woodbury-Morrison equation should be used for computing the inverse (this is ignored if modified.pp is FALSE).
...	currently no additional arguments.

**Value**

Returns the covariance matrix, log-determinant, and inverse.

**Author(s)**

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

**Examples**

```
set.seed(1)

n <- 100 ##number of coords
m <- 25  ##number of knots
q <- 5   ##number of response variables

coords <- cbind(runif(n),runif(n))
knots <- cbind(runif(m), runif(m))

theta <- c(rep(6,q),rep(0.5,q)) #phi and nu

A <- matrix(0,q,q)
A[lower.tri(A,TRUE)] <- rnorm(q*(q-1)/2+q)

V <- A*%t(A)

P <- matrix(0,q,q)
P[lower.tri(A,TRUE)] <- rnorm(q*(q-1)/2+q)

Psi <- P*%t(P)

tol <- 1.0e-8

##
##non bias-adjusted predictive process
##
c1 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="exponential",
                    V=V, Psi=Psi, theta=theta,
                    modified.pp=FALSE, SWM=FALSE)

if(max(abs(chol2inv(chol(c1$C)) - c1$C.inv)) > tol) stop("test-1 failed")
```

```

if(max(abs(2*sum(log(diag(chol(c1$C)))) - c1$log.det)) > tol)
stop("test-1 failed")

c2 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="matern",
                    V=V, Psi=Psi, theta=theta,
                    modified.pp=FALSE, SWM=FALSE)

if(max(abs(chol2inv(chol(c2$C)) - c2$C.inv)) > tol) stop("test-2 failed")
if(max(abs(2*sum(log(diag(chol(c2$C)))) - c2$log.det)) > tol)
stop("test-2 failed")

##
##bias-adjusted predictive process
##
c3 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="exponential",
                    V=V, Psi=Psi, theta=theta,
                    modified.pp=TRUE, SWM=FALSE)

if(max(abs(chol2inv(chol(c3$C)) - c3$C.inv)) > tol) stop("test-3 failed")
if(max(abs(2*sum(log(diag(chol(c3$C)))) - c3$log.det)) > tol)
stop("test-3 failed")

c4 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="matern",
                    V=V, Psi=Psi, theta=theta,
                    modified.pp=TRUE, SWM=FALSE)

if(max(abs(chol2inv(chol(c4$C)) - c4$C.inv)) > tol) stop("test-4 failed")
if(max(abs(2*sum(log(diag(chol(c4$C)))) - c4$log.det)) > tol)
stop("test-4 failed")

##
##non bias-adjusted predictive process using SWM
##
c5 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="exponential",
                    V=V, Psi=Psi, theta=theta,
                    modified.pp=FALSE, SWM=TRUE)

if(max(abs(chol2inv(chol(c5$C)) - c5$C.inv)) > tol) stop("test-5 failed")
if(max(abs(2*sum(log(diag(chol(c5$C)))) - c5$log.det)) > tol)
stop("test-5 failed")

c6 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="matern",
                    V=V, Psi=Psi, theta=theta,
                    modified.pp=FALSE, SWM=TRUE)

if(max(abs(chol2inv(chol(c6$C)) - c6$C.inv)) > tol) stop("test-6 failed")
if(max(abs(2*sum(log(diag(chol(c6$C)))) - c6$log.det)) > tol)
stop("test-6 failed")

##
##bias-adjusted predictive process using SWM

```

```
##
c7 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="exponential",
  V=V, Psi=Psi, theta=theta,
  modified.pp=TRUE, SWM=TRUE)

if(max(abs(chol2inv(chol(c7$C)) - c7$C.inv)) > tol) stop("test-7 failed")
if(max(abs(2*sum(log(diag(chol(c7$C)))) - c7$log.det)) > tol)
stop("test-7 failed")

c8 <- mvCovInvLogDet(coords=coords, knots=knots, cov.model="matern",
  V=V, Psi=Psi, theta=theta,
  modified.pp=TRUE, SWM=TRUE)

if(max(abs(chol2inv(chol(c8$C)) - c8$C.inv)) > tol) stop("test-8 failed")
if(max(abs(2*sum(log(diag(chol(c8$C)))) - c8$log.det)) > tol)
stop("test-8 failed")
```

---

mvLM

---

*Function for fitting multivariate Bayesian regression models*


---

## Description

The function mvLM fits a Gaussian multivariate Bayesian regression model.

## Usage

```
mvLM(formula, data = parent.frame(), starting, tuning, priors,
  n.samples, sub.samples, verbose=TRUE, n.report=100, ...)
```

## Arguments

- |          |  |
|----------|--|
| formula  | for a multivariate model with $q$ response variables, this is a list of univariate formulas. See example below.  |
| data     | an optional data frame containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which mvLM is called.  |
| starting | a list with each tag corresponding to a parameter name. Valid list tags are beta and L. The value portion of each tag is a vector of parameter's starting value. For L the vector is of length $\frac{(q^2-q)}{2} + q$ . Here, L holds the the lower-triangle elements in column major ordering of the Cholesky square root of the residual cross-covariance matrix. |
| tuning   | a list with each tag corresponding to a parameter name. The only valid list tag is L. The value portion of each tag defines the step size of the proposal used in the Metropolis sampling. For L the vector is of length $\frac{(q^2-q)}{2} + q$ .   |

priors	a list with each tag corresponding to a parameter name. The only valid list tag is <code>Psi.IW</code> (Beta priors are assumed <i>flat</i> ). <code>Psi</code> is assumed to follow an inverse-Wishart distribution. The hyperparameters of the inverse-Wishart are passed as a list of length two, with the first and second elements corresponding to the <i>df</i> and $q \times q$ <i>scale</i> matrix, respectively.
n.samples	the number of MCMC iterations.
sub.samples	a vector of length 3 that specifies <i>start</i> , <i>end</i> , and <i>thin</i> , respectively, of the MCMC samples. The default is <code>c(1, n.samples, 1)</code> (i.e., all samples).
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.

### Value

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

p.samples	a coda object of posterior samples for the defined parameters.
acceptance	the Metropolis sampling acceptance rate.

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

### See Also

[spMvLM](#), [spGGT](#)

### Examples

```
## Not run:
##Generate some data
set.seed(1)

n <- 100
q <- 3
nltr <- q*(q-1)/2+q

L <- matrix(0,q,q)
L[lower.tri(L,TRUE)] <- rnorm(nltr,1,1)
Psi <- L%*%t(L)

X <- mkMvX(list(matrix(1,n,1), matrix(1,n,1), matrix(1,n,1)))
beta <- c(-1,0,1)

y <- mvrnorm(1, X%*%beta, diag(n)%x%Psi)

y.1 <- y[seq(1,length(y),q)]
y.2 <- y[seq(2,length(y),q)]
y.3 <- y[seq(3,length(y),q)]
```

```

n.samples <- 10000

##starting
L.starting <- diag(q)[lower.tri(diag(q),TRUE)]
L.tuning <- rep(0.03,nltr)

m.1 <- mvLM(list(y.1~1,y.2~1,y.3~1), starting=list("L"=L.starting),
             tuning=list("L"=L.tuning), priors=list("Psi.IW"=list(q+1, diag(0.1,q))),
             n.samples=n.samples, verbose=TRUE, n.report=1000)

L.starting <- diag(5,q)[lower.tri(diag(q),TRUE)]
m.2 <- mvLM(list(y.1~1,y.2~1,y.3~1), starting=list("L"=L.starting),
             tuning=list("L"=L.tuning), priors=list("Psi.IW"=list(q+1, diag(0.1,q))),
             n.samples=n.samples, verbose=TRUE, n.report=1000)

L.starting <- diag(10,q)[lower.tri(diag(q),TRUE)]
m.3 <- mvLM(list(y.1~1,y.2~1,y.3~1), starting=list("L"=L.starting),
             tuning=list("L"=L.tuning), priors=list("Psi.IW"=list(q+1, diag(0.1,q))),
             n.samples=n.samples, verbose=TRUE, n.report=1000)

samps <- mcmc.list(m.1$p.samples, m.2$p.samples, m.3$p.samples)
gelman.plot(samps)

##Note that Psi[2,1], Psi[3,1], and Psi[3,2] are slow to converge.
##Try a bit of hand tuning or run the chains out longer.
n.samples <- 25000

##starting
L.starting <- diag(q)[lower.tri(diag(q),TRUE)]
L.tuning <- rep(0.03,nltr)

m.1 <- mvLM(list(y.1~1,y.2~1,y.3~1), starting=list("L"=L.starting),
             tuning=list("L"=L.tuning),
             priors=list("Psi.IW"=list(q+1, diag(0.1,q))),
             n.samples=n.samples, verbose=TRUE, n.report=1000)

L.starting <- diag(5,q)[lower.tri(diag(q),TRUE)]
m.2 <- mvLM(list(y.1~1,y.2~1,y.3~1), starting=list("L"=L.starting),
             tuning=list("L"=L.tuning),
             priors=list("Psi.IW"=list(q+1, diag(0.1,q))),
             n.samples=n.samples, verbose=TRUE, n.report=1000)

L.starting <- diag(10,q)[lower.tri(diag(q),TRUE)]
m.3 <- mvLM(list(y.1~1,y.2~1,y.3~1), starting=list("L"=L.starting),
             tuning=list("L"=L.tuning),
             priors=list("Psi.IW"=list(q+1, diag(0.1,q))),
             n.samples=n.samples, verbose=TRUE, n.report=1000)

samps <- mcmc.list(m.1$p.samples, m.2$p.samples, m.3$p.samples)
gelman.plot(samps)

##parameters posterior summary
burn.in <- 10000

```

```
plot(window(samps, start=burn.in))  
print(round(summary(window(samps, start=burn.in))$quantiles[,c(3,1,5)],2))  
## End(Not run)
```

---

pointsInPoly	<i>Finds points in a polygon</i>
--------------	----------------------------------

---

### 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)

```

---

prior *Creates prior distribution definitions*

---

## Description

The function prior creates a valid prior definition for use in [spGGT](#).

## Usage

```
prior(dist, ...)
```

## Arguments

dist	a quoted key word that identifies the prior distribution. The choices are inverse-gamma "IG", uniform "UNIF", half-Cauchy "HC", normal "NORMAL", flat "FLAT", inverse-Wishart "IWISH", and fixed scalar or matrix "FIXED".
...	the hyperparameters of the chosen prior distribution, passed as quoted key words with associated values. See details below.

## Details

Up to a proportionality constant the possible priors are:

- Uniform:  $1/(b - a)$ ,  $b > a$ , where  $b > a$ . Use key words "a", and "b".
- Inverse-gamma:  $x^{-(a+1)}exp(-b/x)$ , where  $a$  is shape  $> 0$  and  $b$  is scale  $> 0$ . Use key words "shape", and "scale".
- Half-Cauchy:  $(x + a^2)^{-1}$ . Use key word "a".
- Normal:  $exp(-1/2(\beta - mu)^t V^{-1}(\beta - mu))$ , where  $mu$  is the mean vector of length  $p$  and  $V^{-1}$  is the  $p \times p$  precision matrix. Use key words "mu" and "precision".

- Inverse-Wishart:  $\det(S)^{df/2} \det(W)^{-(df+m+1)/2} \exp(-1/2tr(SW^{-1}))$ , where  $W$  is the  $m \times m$  covariance matrix,  $df$  is the degrees of freedom, and  $S$  is the positive definite  $m \times m$  scale matrix. Use key words "df" and "S".

The normal and flat priors can only be used for the regressors specified in [spGGT](#).

### Value

prior                    a list object of class `ggt.prior` which is used for the value portion of the prior tag in the [spGGT](#) function.

### Author(s)

Andrew O. Finley <finleya@msu.edu>,  
Sudipto Banerjee <sudiptob@biostat.umn.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.  
Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models 1(3):515-533, Bayesian Analysis.

### See Also

[spGGT](#)

---

spDiag

*Model fit diagnostics DIC and GP*

---

### Description

The function `spDiag` calculates DIC, GP, and associated statistics given a [bayesLMRef](#), [bayesLMConjugate](#), [spGGT](#), [spLM](#), [spMvLM](#), [bayesGeostatExact](#), [spGLM](#), or [mvLM](#) object.

### Usage

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

### Arguments

`sp.obj`                an object returned by [bayesLMRef](#), [bayesLMConjugate](#), [spGGT](#), [spLM](#), [spMvLM](#), [mvLM](#), [bayesGeostatExact](#), or [spGLM](#)

`start`                 specifies the first sample included in the calculation. This is useful for those who choose to acknowledge chain burn-in.

`end`                    specifies the last sample included in the prediction calculation. 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 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. This is only available for <a href="#">bayesLMRef</a> , <a href="#">bayesLMConjugate</a> , <a href="#">spLM</a> , <a href="#">spMvLM</a> , <a href="#">bayesGeostatExact</a> , and <a href="#">mvLM</a> objects
sp.effects	if sp.obj specifies a spatial model without pre-calculated spatial effects then spDiag calculates the spatial effects.

**Author(s)**

Andrew O. Finley <finleya@msu.edu>,  
Sudipto Banerjee <sudiptob@biostat.umn.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.
- Gelfand A.E. and Ghosh, S.K. (1998). Model choice: a minimum posterior predictive loss approach. *Biometrika*. 85:1-11.

**See Also**

[bayesLMRef](#), [bayesLMConjugate](#), [spGGT](#), [bayesGeostatExact](#), [spLM](#), [spGLM](#), [spMvLM](#), [mvLM](#)

**Examples**

```
## Not run:
data(rf.n200.dat)

Y <- rf.n200.dat$Y
coords <- as.matrix(rf.n200.dat[,c("x.coords", "y.coords")])
w <- rf.n200.dat$w

n.samples <- 1000

##non-spatial regression
m.1 <- bayesLMRef(lm(Y~1), n.samples=n.samples)
```

```
##spatial regression
m.2 <- spLM(Y~1, coords=coords,
            starting=list("phi"=0.6,"sigma.sq"=1, "tau.sq"=1),
            sp.tuning=list("phi"=0.01, "sigma.sq"=0.05, "tau.sq"=0.05),
            priors=list("phi.Unif"=c(0.3, 3), "sigma.sq.IG"=c(2, 1),
                       "tau.sq.IG"=c(2, 1)),
            cov.model="exponential",
            n.samples=n.samples, verbose=TRUE, n.report=100)

##compare
print(spDiag(m.1))
print(spDiag(m.2, start=500, thin=2))

## End(Not run)
```

---

spGGT

*Function for fitting univariate and multivariate Bayesian spatial regression models*

---

## Description

The function spGGT fits Gaussian univariate and multivariate Bayesian spatial regression models. In most cases, we encourage users to use `spLM` or `splm` over spGGT. These functions offer a simplified interface and additional functionality to handle large data sets.

## Usage

```
spGGT(formula, data = parent.frame(), coords, run.control,
      var.update.control, beta.update.control, cov.model,
      verbose=TRUE, ...)
```

## Arguments

formula	for a univariate model, this is a symbolic description of the regression model to be fit. For a multivariate model, this is a list of univariate formulas. See example below.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which spGGT is called.
coords	an $n \times 2$ matrix of the observation coordinates in $R^2$ (e.g., easting and northing).
run.control	a list with tags: <code>n.samples</code> the value of which is the number of MCMC iterations; <code>sp.effects</code> a logical value indicating if spatial random effects should be recovered; <code>DIC</code> a logical value indicating if marginalized DIC and associated statistics should be computed; <code>DIC.start</code> and <code>DIC.end</code> the MCMC iteration interval over which DIC should be calculated; and <code>linpack</code> a logical value indicating if LINPACK or LAPACK should be used for matrix algebra operations. The default is to use LINPACK.

`var.update.control`

a list with each tag corresponding to a parameter name used to define the variance structure of the model. Valid list tags are `K`, `Psi`, `phi`, and `nu`. The value portion of each of these tags is another list with tags `sample.order`, `starting`, `tuning`, and `prior`. The positive scalar value of the optional tag `sample.order` determines the order in which parameters are updated by sequential Metropolis-Hastings steps. The `sample.order` value is relative to the sample order specified for the other parameters. The value of the `starting` tag defines the starting value of the parameter in the Metropolis-Hastings sampling. If the list element defines a vector or matrix of parameters, then the starting value should be set accordingly. If a scalar starting value is given for a vector or matrix of parameters then the scalar value is recycled. If the model defines `K` or `Psi` as a full cross-covariance matrix, opposed to a diagonal matrix or scalar, then the starting value should be the lower-triangle of the Cholesky square root of the desired starting matrix value. If `K` or `Psi` is a diagonal matrix, then the starting values should be passed as a vector or scalar. The value associated with the `tuning` tag defines the step size of the proposal used in the Metropolis-Hastings sampling. Again the `tuning` value is recycled if needed. If the parameter element is a matrix, i.e., `K` or `Psi`, then the diagonal elements in the tuning matrix correspond to the column major lower-triangle elements in the parameter matrix. The value of the `prior` tag is a valid `prior` object which describes the prior distribution of the parameter. Any parameter in the specified model can be fixed by using the `FIXED prior` with the `starting` value set to the desired fixed value. See example below.

`beta.update.control`

a list with tags `update`, `tuning`, `starting`, `prior`, and `sample.order`. The value of `update` is either "GIBBS" or "MH". With the "MH" option, the value of the optional tag `tuning` is the upper-triangle of the Cholesky decomposed  $p \times p$  tuning matrix, where  $p$  is the number of regression parameters. The tuning matrix controls the proposal step size in the Metropolis-Hastings sampling. The tuning values along the diagonal of this matrix correspond to the order of the regressors in the model formula. Off-diagonal elements are the covariance among the terms. The optional tag `starting` receives a vector of length  $p$ , whose elements define the starting values of the regressors used in the MCMC sampling. The order of the starting values in this vector corresponds to the order of the regressors in the model formula. The optional tag `prior` is a valid `prior` object that defines either a normal or flat distribution. If `prior` is not included in the `beta.update.control` list then the prior distribution on the regressors is assumed flat (i.e., proportional to 1.0). With the "MH" option, the value of the optional tag `sample.order` determines the order in which parameters are updated by sequential Metropolis-Hastings steps, see below for details.

`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.

`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.

**Details**

Please refer to Section 3.1 in the vignette.

**Value**

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

<code>cov.model</code>	the covariance model specified by <code>cov.model</code> .
<code>no.Psi</code>	a logical value indicating if the parameter <code>Psi</code> was used.
<code>coords</code>	the $n \times 2$ matrix specified by <code>coords</code> .
<code>X</code>	the $n \times p$ matrix of regressors specified by <code>formula</code> .
<code>Y</code>	the $n \times 1$ response variable vector specified by <code>formula</code> .
<code>p.samples</code>	a matrix of posterior samples for the defined parameters. If <code>K</code> or <code>Psi</code> is specified as a full cross-covariance matrix then the corresponding samples in <code>p.samples</code> are column major lower-triangle elements of the matrix.
<code>acceptance</code>	a matrix of the Metropolis-Hastings sampling acceptance rate. Row names correspond to the parameters updated with Metropolis-Hastings.
<code>DIC</code>	a matrix that holds marginalized DIC and associated values is returned if <code>DIC</code> is true in the <code>run.control</code> list.
<code>sp.effects</code>	a matrix that holds samples from the posterior distribution of the response specific spatial random effects. The rows of this matrix correspond to the $n$ point observations and the columns are the posterior samples. If a multivariate model is fit to $m$ response variables, the <code>sp.effects</code> matrix has $mn$ rows. The random effects samples for points are held in rows $1:m$ , $(m+1):2m$ , ..., $((i-1)m+1):im$ , ..., $((n-1)m+1):nm$ , where $i = 1 \dots n$ (e.g., the samples for the first point are in rows $1:m$ , second point in rows $(m+1):2m$ , etc.).

**Author(s)**

Andrew O. Finley <finleya@msu.edu>  
 Sudipto Banerjee <sudiptob@biostat.umn.edu>  
 Bradley P. Carlin <brad@biostat.umn.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.

**See Also**

[prior](#), [spPredict](#)

## Examples

```

## Not run:

data(FBC07.dat)
Y.1 <- FBC07.dat[1:100,"Y.1"]
Y.2 <- FBC07.dat[1:100,"Y.2"]
coords <- as.matrix(FBC07.dat[1:100,c("coord.X", "coord.Y")])

#####
## Univariate models
#####

##Fit some competing models.
##Full model with nugget (Psi), partial sill (K),
##and spatial decay (Phi).
K.prior <- prior(dist="IG", shape=2, scale=5)
Psi.prior <- prior(dist="IG", shape=2, scale=5)
phi.prior <- prior(dist="UNIF", a=0.06, b=3)

var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
        "Psi"=list(starting=5, tuning=0.5, prior=Psi.prior),
        "phi"=list(starting=0.1, tuning=0.5, prior=phi.prior)
      )

beta.control <- list(update="GIBBS", prior=prior(dist="FLAT"))

run.control <- list("n.samples"=1000, "sp.effects"=TRUE)

Fit.m1 <- spGGT(formula=Y.2~1, run.control=run.control,
                coords=coords, var.update.control=var.update.control,
                beta.update.control=beta.control,
                cov.model="exponential")

plot(Fit.m1$p.samples)
summary(Fit.m1$p.samples)

##First reduced model with no nugget just partial sill and decay.
var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
        "phi"=list(starting=0.1, tuning=0.5, prior=phi.prior)
      )

Fit.m2 <- spGGT(formula=Y.2~1, run.control=run.control,
                coords=coords, var.update.control=var.update.control,
                beta.update.control=beta.control,
                cov.model="exponential")

plot(Fit.m2$p.samples)

##Second reduced model with no nugget just partial sill and fixed
##decay at the WRONG value, which forces the wrong estimate of K.

```

```

phi.prior <- prior(dist="FIXED")

var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
       "phi"=list(starting=0.1, prior=phi.prior)
       )

Fit.m3 <- spGGT(formula=Y.2~1, run.control=run.control,
                coords=coords, var.update.control=var.update.control,
                beta.update.control=beta.control,
                cov.model="exponential")
plot(Fit.m3$p.samples)

##For this data, DIC cannot distinguish between the first two
##models but does show that the third model is clearly
##wrong. An empirical semivariogram is useful
##for recognizing that the full model is the correct choice
##over the second model.
print(spDiag(Fit.m1))
print(spDiag(Fit.m2))
print(spDiag(Fit.m3))

##Use akima or MBA to produce interpolated surfaces of
##estimated random spatial effects.

m1.surf <- mba.surf(cbind(coords, rowMeans(Fit.m1$sp.effects)),
                  no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(m1.surf, xaxs="r", yaxs="r",
      main="Y.2 random spatial effects")
points(coords, pch=19)

##Now the full model using the Matern spatial correlation function.
##Also, for fun, update the spatial decay parameters
##phi and nu in a separate MH block. This provides finer control
##on the parameters' acceptance rate, but takes a bit longer to
##run.
phi.prior <- prior(dist="UNIF", a=0.06, b=3)
nu.prior <- prior(dist="UNIF", a=0.01, b=2)

var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
       "Psi"=list(starting=5, tuning=0.5, prior=Psi.prior),
       "phi"=list(sample.order=1, starting=0.1,
                  tuning=0.5, prior=phi.prior),
       "nu"=list(sample.order=1, starting=0.1,
                  tuning=0.5, prior=nu.prior)
       )

run.control <- list("n.samples"=1000, "sp.effects"=FALSE)

Fit.m4 <-
  spGGT(formula=Y.2~1, run.control=run.control,
        coords=coords, var.update.control=var.update.control,

```

```

        beta.update.control=beta.control,
        cov.model="matern")

plot(Fit.m4$p.samples)

##Solve the Matern correlation function for the estimated
##95% CI for effective decay (e.g., distance at which the
##correlation drops to 0.05).
phi.hat <- quantile(Fit.m4$p.samples[, "Phi"], c(0.50,0.025,0.975))
nu.hat <- quantile(Fit.m4$p.samples[, "Nu"], c(0.50,0.025,0.975))

matern <- function(cor, eff.d, phi, nu){
  cor - (eff.d*phi)^nu/(2^(nu-1)*gamma(nu))*besselK(x=eff.d*phi, nu=nu)
}

##50
print(uniroot(matern, interval=c(1, 50),
             cor=0.05, phi=phi.hat[1], nu=nu.hat[1])$root)
print(uniroot(matern, interval=c(1, 50),
             cor=0.05, phi=phi.hat[2], nu=nu.hat[2])$root)
print(uniroot(matern, interval=c(1, 50),
             cor=0.05, phi=phi.hat[3], nu=nu.hat[3])$root)

##Finally, the full model again but this time update the Beta
##using MH. Using the default of sample.order=0, this sampling
##scheme is simply a single block MH proposal of all model parameters.
##Also, the first run uses the default tuning value for the Beta
##(i.e., chol(vcov(lm(Y~X))). Both chains use a normal prior on Beta;
##however, the first chain allows for a vague prior, where as the
##second is much too strict.
var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
       "Psi"=list(starting=5, tuning=0.5, prior=Psi.prior),
       "phi"=list(starting=0.1, tuning=0.5, prior=phi.prior)
       )

##Chain 1: vague prior normal variance of 1/0.001 = 1000
beta.prior <- prior(dist="NORMAL", mu=5, precision=0.001)
beta.control <- list(update="MH", prior=beta.prior)

Fit.m5.a <-
  spGGT(formula=Y.2~1, run.control=run.control,
        coords=coords, var.update.control=var.update.control,
        beta.update.control=beta.control,
        cov.model="exponential")

##Chain 2: restrictive prior normal variance of 1/5 = 0.2
##(i.e., crazy strict)
beta.prior <- prior(dist="NORMAL", mu=5, precision=5)
beta.control <- list(update="MH", prior=beta.prior, tuning=0.75)

Fit.m5.b <-
  spGGT(formula=Y.2~1, run.control=run.control,

```

```

        coords=coords, var.update.control=var.update.control,
        beta.update.control=beta.control,
        cov.model="exponential")

plot(mcmc.list(Fit.m5.a$p.samples, Fit.m5.b$p.samples))

##Now just for fun, here is the full model but using the weakly
##informative half-Cauchy for the nugget (Psi) and partial sill (K).

K.prior <- prior(dist="HC", a=50)
Psi.prior <- prior(dist="HC", a=50)
phi.prior <- prior(dist="UNIF", a=0.06, b=3)

var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
       "Psi"=list(starting=5, sample.order=1, tuning=2, prior=Psi.prior),
       "phi"=list(starting=0.1, sample.order=2, tuning=1, prior=phi.prior)
  )

beta.control <- list(update="GIBBS", prior=prior(dist="FLAT"))

run.control <- list("n.samples"=3000, "sp.effects"=TRUE)

Fit.m1 <- spGGT(formula=Y.2~1, run.control=run.control,
               coords=coords, var.update.control=var.update.control,
               beta.update.control=beta.control,
               cov.model="exponential")

plot(Fit.m1$p.samples)
summary(Fit.m1$p.samples)

#####
##  Multivariate models
#####

##Full model with non-spatial cross-covariance
##(Psi) matrix, spatial cross-covariance matrix (K),
##and response specific spatial decay parameter (Phi)
##(i.e., a non-separable model).

##These chains really need to be run for several thousand
##samples (e.g., 10,000). Also note that it is much easier to maintain
##a healthy acceptance rate for each parameter if they
##are updated individually using the sample.order
##directive, as shown in the second example below.
K.prior <- prior(dist="IWISH", df=2, S=diag(c(3, 6)))
Psi.prior <- prior(dist="IWISH", df=2, S=diag(c(7, 5)))
phi.prior <- prior(dist="UNIF", a=0.06, b=3)

##The matrix starting values for K and Psi are actually
##passed as the sqrt of the desired starting values.

```

```

##This is only done if K or Psi are full cross-covariance
##matrices and serve to insure that the true starting value
##matrix is positive definite.
K.starting <- matrix(c(2,-3, 0, 1), 2, 2)
Psi.starting <- diag(c(3, 2))

var.update.control <-
  list("K"=list(starting=K.starting, tuning=diag(c(0.08, 0.3, 0.08)),
    prior=K.prior),
    "Psi"=list(starting=Psi.starting, tuning=diag(c(0.08, 0.3, 0.08)),
    prior=Psi.prior),
    "phi"=list(starting=0.1, tuning=0.5,
    prior=list(phi.prior, phi.prior))
  )

beta.control <- list(update="GIBBS", prior=prior(dist="FLAT"))

run.control <- list("n.samples"=2000, "sp.effects"=FALSE)

Fit.mv.m1 <-
  spGGT(formula=list(Y.1~1, Y.2~1), run.control=run.control,
    coords=coords, var.update.control=var.update.control,
    beta.update.control=beta.control,
    cov.model="exponential")

plot(Fit.mv.m1$p.samples)

##Calculate the mean K correlation matrix. Note that since K is
##specified as a full cross-covariance matrix, the posterior
##samples are returned for the lower triangle of the symmetric K,
##in column major order.
K <- matrix(0, 2, 2)
K[lower.tri(K, diag=TRUE)] <-
  colMeans(Fit.mv.m1$p.samples[,c("K_1", "K_2", "K_3")])
K[upper.tri(K, diag=FALSE)] <- t(K)[upper.tri(K, diag=FALSE)]
print(cov2cor(K))

##Now using sample order. Note this takes ~3 times as long
##to run, but provides much finer control of acceptance rates.
var.update.control <-
  list("K"=list(sample.order=0, starting=K.starting,
    tuning=diag(c(0.3, 0.5, 0.2)), prior=K.prior),
    "Psi"=list(sample.order=1, starting=Psi.starting,
    tuning=diag(c(0.2, 0.5, 0.2)), prior=Psi.prior),
    "phi"=list(sample.order=2, starting=0.1,
    tuning=0.5, prior=list(phi.prior, phi.prior))
  )

run.control <- list("n.samples"=2000, "sp.effects"=TRUE)

Fit.mv.m2 <-
  spGGT(formula=list(Y.1~1, Y.2~1), run.control=run.control,

```

```

        coords=coords, var.update.control=var.update.control,
        beta.update.control=beta.control,
        cov.model="exponential")

plot(Fit.mv.m2$p.samples)

##Now again check out the random spatial effects. Recall,
##these are stored in the sp.effects matrix, which is organized
##as m consecutive rows for each point.
rand.effect.Y.1 <-
  Fit.mv.m2$sp.effects[seq(1, nrow(Fit.mv.m2$sp.effects), 2),]

rand.effect.Y.2 <-
  Fit.mv.m2$sp.effects[seq(2, nrow(Fit.mv.m2$sp.effects), 2),]

rand.effect.Y.1.surf <-
  mba.surf(cbind(coords, rowMeans(rand.effect.Y.1)),
           no.X=100, no.Y=100, extend=TRUE)$xyz.est

rand.effect.Y.2.surf <-
  mba.surf(cbind(coords, rowMeans(rand.effect.Y.2)),
           no.X=100, no.Y=100, extend=TRUE)$xyz.est

##The estimated negative value of off-diagonal element in K is
##clearly seen in the map of random spatial effects.
par(mfrow=c(1,2))
image(rand.effect.Y.1.surf, xaxs="r", yaxs="r",
      main="Y.1 random spatial effects")
contour(rand.effect.Y.1.surf, add=TRUE)
image(rand.effect.Y.2.surf, xaxs="r", yaxs="r",
      main="Y.2 random spatial effects")
contour(rand.effect.Y.2.surf, add=TRUE)

##Now a simpler model. Specifically, the full model shows that
##there is negligible non-spatial cross-covariance, therefore,
##Psi might just be modeled with a diagonal covariance matrix.
##This is actually the model which gave rise to this synthetic
##data set.
K.prior <- prior(dist="IWISH", df=2, S=diag(c(3, 6)))
Psi.prior <- prior(dist="IG", shape=2, scale=5)
phi.prior <- prior(dist="UNIF", a=0.06, b=3)

var.update.control <-
  list("K"=list(sample.order=0, starting=K.starting,
               tuning=diag(c(0.3, 0.5, 0.2)), prior=K.prior),
       "Psi"=list(sample.order=1, starting=5,
                  tuning=0.5, prior=list(Psi.prior, Psi.prior)),
       "phi"=list(sample.order=2, starting=0.1,
                  tuning=0.5, prior=list(phi.prior, phi.prior))
  )

run.control <- list("n.samples"=2000)

```

```

Fit.mv.m3 <-
  spGGT(formula=list(Y.1~1, Y.2~1), run.control=run.control,
        coords=coords, var.update.control=var.update.control,
        beta.update.control=beta.control,
        cov.model="exponential")

plot(Fit.mv.m3$p.samples)
summary(Fit.mv.m3$p.samples)

##Finally, a really simple model which we know makes no sense
##for this data set. Specifically, a single variance parameter
##for the diagonal elements of K and Psi and common spatial decay,
##phi. Also, we use a spherical spatial covariance function, just
##because we can.
K.prior <- prior(dist="IG", shape=2, scale=5)
Psi.prior <- prior(dist="IG", shape=2, scale=5)
phi.prior <- prior(dist="UNIF", a=0.06, b=3)

var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
       "Psi"=list(starting=5, tuning=0.5, prior=Psi.prior),
       "phi"=list(starting=0.1, tuning=0.5, prior=phi.prior)
       )

Fit.mv.m4 <-
  spGGT(formula=list(Y.1~1, Y.2~1), run.control=run.control,
        coords=coords, var.update.control=var.update.control,
        beta.update.control=beta.control,
        cov.model="spherical")

plot(Fit.mv.m4$p.samples)

## 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, amcmc, starting, tuning, priors, cov.model,
      modified.pp = TRUE, n.samples, sub.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 when family is binomial. Unless otherwise specified a vector of ones is assumed i.e., one trial at each location.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , 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 <i>predictive process</i> 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.
amcmc	a list with tags <code>n.batch</code> , <code>batch.length</code> , and <code>accept.rate</code> . Specifying this argument invokes an adaptive MCMC sampler (see Roberts and Rosenthal, 2006 and <code>adaptMetropGibbs</code> function).
starting	a list with each tag corresponding to a parameter name. Valid list tags are <code>beta</code> , <code>sigma.sq</code> , <code>phi</code> , <code>nu</code> , and <code>w</code> . The value portion of each tag is the parameter's starting value. If the predictive process is used then <code>w</code> must be of length $m$ ; otherwise, it must be of length $n$ . Alternatively, <code>w</code> 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 list tags are <code>beta</code> , <code>sigma.sq</code> , <code>phi</code> , <code>nu</code> , <code>w</code> , and <code>e</code> . The value portion of each tag defines the variance of the Metropolis normal proposal distribution. The tuning value for <code>beta</code> can be a vector of length $p$ or the lower-triangle of the $p \times p$ Cholesky square-root of the desired proposal variance matrix. If the predictive process is used then <code>w</code> must be of length $m$ ; otherwise, it must be of length $n$ . Alternatively, <code>w</code> can be set as a scalar, in which the value is repeated. If a modified predictive process is not used then <code>e</code> is ignored.
priors	a list with each tag corresponding to a parameter name. Valid list tags are <code>beta.flat</code> , <code>beta.normal</code> , <code>sigma.sq.ig</code> , <code>phi.unif</code> , and <code>nu.unif</code> . <code>sigma.sq</code> is assumed to follow an inverse-Gamma distribution, whereas the spatial decay <code>phi</code> and smoothness <code>nu</code> parameters are assumed to follow Uniform distributions. If <code>beta.normal</code> then covariate specific mean and variance hyperparameters are passed as the first and second list elements, respectively. The hyperparameters of the inverse-Gamma are passed as a vector of length two, with the first and second elements corresponding to the <i>shape</i> and <i>scale</i> , 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.
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.

modified.pp	a logical value indicating if the <i>modified predictive process</i> 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.
n.samples	the number of MCMC iterations. This is ignored if amcmc is specified.
sub.samples	a vector of length 3 that specifies <i>start</i> , <i>end</i> , and <i>thin</i> , respectively, of the MCMC samples. The default is <code>c(1, n.samples, 1)</code> (i.e., all samples).
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. Note, this is different from the `glm` argument which is passed as a rate.

### Value

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

coords	the $n \times 2$ matrix specified by <code>coords</code> .
knot.coords	the $m \times 2$ matrix as specified by <code>knots</code> .
p.samples	a coda object of posterior samples for the defined parameters.
acceptance	the Metropolis sampling acceptance rate. If <code>amcmc</code> 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, and the acceptance rate is the average over all proposals.
acceptance.w	If this is a non-predictive process model and <code>amcmc</code> is used then this will be a matrix of each random spatial effects acceptance rate at the end of each batch.
acceptance.w.str	If this is a predictive process model and <code>amcmc</code> is used then this will be a matrix of each random spatial effects acceptance rate at the end of each batch.
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.

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. (2008) Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, DOI: 10.1016/j.csda.2008.09.008
- 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

[spGGT](#), [spMvLM](#), [spMvGLM](#)

## Examples

```
## Not run:

library(MASS)
library(MBA)
library(fields)

#####
##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 <- mvrnorm(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, 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 <- 500
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"=0.1, "w"=0.01),
             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",
             n.samples=n.samples, sub.samples=c(1000,n.samples,10),
             verbose=TRUE, n.report=10)

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

beta.hat <- m.1$p.samples[, "(Intercept)"]
w.hat <- m.1$sp.effects

y.hat <- 1/(1+exp(-(x%*%beta.hat+w.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.plot(surf, main="Interpolated mean of posterior rate\n(observed rate)")
text(coords, label=paste("(",y/weights,")",sep=""))

surf <- mba.surf(cbind(coords,y.hat.var),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf, main="Interpolated variance of posterior rate\n(observed # of trials)")
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 <- mvrnorm(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"=0.1, "w"=0.1),
             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",
             n.samples=n.samples, sub.samples=c(1000,n.samples,10),
             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.
m.1$p.samples[,"phi"] <- 3/m.1$p.samples[,"phi"]

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

beta.hat <- mean(m.1$p.samples[,"(Intercept)"])
w.hat <- rowMeans(m.1$sp.effects)

y.hat <-exp(x%*%beta.hat+w.hat)

##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="obs")
contour(surf, drawlabels=FALSE, add=TRUE)

```

```

text(coords, labels=y, cex=1, col="blue")

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

#####
##Spatial logistic
#####

##Generate binary data
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 <- mvrnorm(1, rep(0,n), R)

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

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

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

m.1 <- spGLM(y~1, family="binomial", coords=coords,
            starting=
            list("beta"=beta.starting, "phi"=0.06,"sigma.sq"=1, "w"=0),
            tuning=
            list("beta"=beta.tuning, "phi"=0.5, "sigma.sq"=0.1, "w"=0.01),
            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",
            n.samples=n.samples, sub.samples=c(1000,n.samples,10),
            verbose=TRUE, n.report=10)

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

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

```

```

beta.hat <- mean(m.1$p.samples[, "(Intercept)"])
w.hat <- rowMeans(m.1$sp.effects)

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

##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")
contour(surf, add=TRUE)
points(coords[y==1,], pch=19, cex=1)
points(coords[y==0,], cex=1)

surf <- mba.surf(cbind(coords,y.hat),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Fitted probabilities")
contour(surf, add=TRUE)
points(coords[y==1,], pch=19, cex=1)
points(coords[y==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, sp.tuning, priors, cov.model,
      modified.pp = TRUE, n.samples, sub.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 <code>environment(formula)</code> , 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 <i>predictive process</i> 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 extent.
starting	a list with each tag corresponding to a parameter name. Valid list tags are beta, sigma.sq, tau.sq, phi, and nu. The value portion of each of each tag is the parameter's starting value.
sp.tuning	a list with each tag corresponding to a parameter name. Valid list tags are sigma.sq, tau.sq, phi, and nu. The value portion of each of each tag defines the variance of the Metropolis normal proposal distribution.
modified.pp	a logical value indicating if the <i>modified predictive process</i> 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.
priors	a list with each tag corresponding to a parameter name. Valid list tags are sigma.sq.ig, tau.sq.ig, phi.unif, and nu.unif (Beta priors are assumed <i>flat</i> ). Variance parameters, sigma.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 <i>shape</i> and <i>scale</i> , 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.
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.
n.samples	the number of MCMC iterations.
sub.samples	a vector of length 3 that specifies <i>start</i> , <i>end</i> , and <i>thin</i> , respectively, of the MCMC samples. The default is c(1, n.samples, 1) (i.e., all samples).
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.

### 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.samples	a coda object of posterior samples for the defined parameters.
acceptance	the Metropolis sampling acceptance rate.
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.

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.
- Finley, A.O., S. Banerjee, P. Waldmann, and T. Ericsson. (2008). Hierarchical spatial modeling of additive and dominance genetic variance for large spatial trial datasets. *Biometrics*. DOI: 10.1111/j.1541-0420.2008.01115.x
- Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2008). Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, DOI: 10.1016/j.csda.2008.09.008
- 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.

**See Also**

[spGGT](#), [spMvLM](#)

**Examples**

```
## Not run:
data(rf.n200.dat)

Y <- rf.n200.dat$Y
coords <- as.matrix(rf.n200.dat[,c("x.coords", "y.coords")])
w <- rf.n200.dat$w

#####
##Simple spatial regression
#####
m.1 <- spLM(Y~1, coords=coords,
            starting=list("phi"=0.6, "sigma.sq"=1, "tau.sq"=1),
            sp.tuning=list("phi"=0.01, "sigma.sq"=0.05, "tau.sq"=0.05),
            priors=list("phi.Unif"=c(0.3, 3), "sigma.sq.IG"=c(2, 1),
                       "tau.sq.IG"=c(2, 1)),
            cov.model="exponential",
            n.samples=1000, verbose=TRUE, n.report=100)

print(summary(m.1$p.samples))
plot(m.1$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=TRUE)$xyz.est
```

```

image(obs.surf, xaxs = "r", yaxs = "r", main="Observed response")
points(coords)
contour(obs.surf, add=T)

w.hat <- rowMeans(m.1$sp.effects)
w.surf <-
  mba.surf(cbind(coords, w.hat), no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(w.surf, xaxs = "r", yaxs = "r", main="Estimated random effects")
points(coords)
points(m.1$knot.coords, pch=19, cex=1)
contour(w.surf, add=T)

#####
##Predictive process
#####
##Use some more observations
data(rf.n500.dat)

Y <- rf.n500.dat$Y
coords <- as.matrix(rf.n500.dat[,c("x.coords", "y.coords")])
w <- rf.n500.dat$w

m.2 <- spLM(Y~1, coords=coords, knots=c(6,6,0),
  starting=list("phi"=0.6, "sigma.sq"=1, "tau.sq"=1),
  sp.tuning=list("phi"=0.01, "sigma.sq"=0.01, "tau.sq"=0.01),
  priors=list("phi.Unif"=c(0.3, 3), "sigma.sq.IG"=c(2, 1),
    "tau.sq.IG"=c(2, 1)),
  cov.model="exponential",
  modified.pp=FALSE, n.samples=2000, verbose=TRUE, n.report=100)

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

##Requires MBA package to
##make surfaces
library(MBA)
par(mfrow=c(1,2))
obs.surf <-
  mba.surf(cbind(coords, w), 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.2$sp.effects)
w.surf <-
  mba.surf(cbind(coords, w.hat), no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(w.surf, xaxs = "r", yaxs = "r", main="Estimated random effects")
contour(w.surf, add=T)
points(coords, pch=1, cex=1)
points(m.2$knot.coords, pch=19, cex=1)
legend(1.5,2.5, legend=c("Obs.", "Knots"), pch=c(1,19), bg="white")

#####

```

```

##Modified predictive process
#####
m.3 <- spLM(Y~1, coords=coords, knots=c(6,6,0),
            starting=list("phi"=0.6,"sigma.sq"=1, "tau.sq"=1),
            sp.tuning=list("phi"=0.01, "sigma.sq"=0.01, "tau.sq"=0.01),
            priors=list("phi.Unif"=c(0.3, 3), "sigma.sq.IG"=c(2, 1),
                       "tau.sq.IG"=c(2, 1)),
            cov.model="exponential",
            n.samples=2000, verbose=TRUE, n.report=100)

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

##Requires MBA package to
##make surfaces
library(MBA)
par(mfrow=c(1,2))
obs.surf <-
  mba.surf(cbind(coords, w), 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.3$sp.effects)
w.surf <-
  mba.surf(cbind(coords, w.hat), no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(w.surf, xaxs = "r", yaxs = "r", main="Estimated random effects")
contour(w.surf, add=T)
points(coords, pch=1, cex=1)
points(m.3$knot.coords, pch=19, cex=1)
legend(1.5,2.5, legend=c("Obs.", "Knots"), pch=c(1,19), bg="white")

## 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,
         amcmc, starting, tuning, priors, cov.model, n.samples, sub.samples,
         verbose=TRUE, n.report=100, ...)

```

**Arguments**

formula	for a multivariate model with $q$ response variables, this is a list of univariate formulas. 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 when family is binomial. Here the order of the columns correspond to the univariate models in the formula list. Unless otherwise specified a matrix of ones is assumed i.e., one trial at each location.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , 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 <i>predictive process</i> 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 extent.
amcmc	a list with tags <code>n.batch</code> , <code>batch.length</code> , and <code>accept.rate</code> .
starting	a list with each tag corresponding to a parameter name. Valid list tags are <code>beta</code> , <code>A</code> , <code>phi</code> , and <code>nu</code> . The value portion of each tag is a vector of parameter's starting value. For <code>A</code> the vector is of length $\frac{(q^2-q)}{2} + q$ and <code>phi</code> and <code>nu</code> are of length $q$ . Here, <code>A</code> 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 <code>w</code> must be of length $qm$ ; otherwise, it must be of length $qn$ . Alternatively, <code>w</code> 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 list tags are <code>beta</code> , <code>A</code> , <code>phi</code> , <code>nu</code> , and <code>w</code> . The value portion of each tag defines the variance of the Metropolis normal proposal distribution. The tuning value for <code>beta</code> can be a vector of length $p$ or the lower-triangle of the $p \times p$ Cholesky square-root of the desired proposal variance matrix. For <code>A</code> , the vector is of length $\frac{(q^2-q)}{2} + q$ and <code>phi</code> and <code>nu</code> are of length $q$ . If the predictive process is used then <code>w</code> must be of length $m$ ; otherwise, it must be of length $n$ . Alternatively, <code>w</code> can be set as a scalar, in which the value is repeated.
priors	a list with each tag corresponding to a parameter name. Valid list tags are <code>beta.flat</code> , <code>beta.normal</code> , <code>K.IW</code> , <code>Psi.IW</code> , <code>phi.unif</code> , and <code>nu.unif</code> . If <code>beta.normal</code> then covariate specific mean and variance hyperparameters are passed as the first and second list elements, respectively. <code>K</code> is assumed to follow an inverse-Wishart distribution, whereas the spatial decay <code>phi</code> and smoothness <code>nu</code> 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 <i>df</i> and $q \times q$ <i>scale</i> matrix, respectively. The hyperparameters of the Uniform are also passed as a vector of length $2 \times q$ with consecutive elements representing the first and second elements corresponding to the lower and upper support in the order of the univariate models given in formula.

<code>cov.model</code>	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.
<code>n.samples</code>	the number of MCMC iterations.
<code>sub.samples</code>	a vector of length 3 that specifies <i>start</i> , <i>end</i> , and <i>thin</i> , respectively, of the MCMC samples. The default is <code>c(1, n.samples, 1)</code> (i.e., all samples).
<code>verbose</code>	if TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
<code>n.report</code>	the interval to report Metropolis acceptance and MCMC progress.
<code>...</code>	currently no additional arguments.

### Details

If a binomial model is specified the response vector is the number of successful trials at each location. Note, this is different from the `glm` argument which is passed as a rate.

### Value

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

<code>coords</code>	the $n \times 2$ matrix specified by <code>coords</code> .
<code>knot.coords</code>	the $m \times 2$ matrix as specified by <code>knots</code> .
<code>p.samples</code>	a coda object of posterior samples for the defined parameters.
<code>acceptance</code>	the Metropolis sampling acceptance rate. If <code>amcmc</code> 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, and the acceptance rate is the average over all proposals.
<code>acceptance.w</code>	If this is a non-predictive process model and <code>amcmc</code> is used then this will be a matrix of each random spatial effects acceptance rate at the end of each batch.
<code>acceptance.w.str</code>	If this is a predictive process model and <code>amcmc</code> is used then this will be a matrix of each random spatial effects acceptance rate at the end of each batch.
<code>sp.effects</code>	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.

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., S. Banerjee, P. Waldmann, and T. Ericsson. (2008). Hierarchical spatial modeling of additive and dominance genetic variance for large spatial trial datasets. *Biometrics*. DOI: 10.1111/j.1541-0420.2008.01115.x
- Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2008). Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, DOI: 10.1016/j.csda.2008.09.008
- 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

[spGGT](#), [spMvLM](#), [spMvGLM](#)

## Examples

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

#####
##Spatial multivariate binomial
#####

##Generate some data
n <- 50
q <- 3
nltr <- q*(q-1)/2+q

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

theta <- rep(3/50,q)

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

Psi <- diag(0,q)

c1 <- mvCovInvLogDet(coords=coords, knots=coords,
  cov.model="exponential",
  V=K, Psi=Psi, theta=theta,
```

```

        modified.pp=TRUE, SWM=FALSE)

w <- mvrnorm(1,rep(0,nrow(c1$C)),c1$C)

X <- mkMvX(list(matrix(1,n,1), matrix(1,n,1), matrix(1,n,1)))
beta <- c(-1,0,1)

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

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

##Specify starting values and collect samples
A.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]

fit <- glm((y/weight)~X-1, weights=rep(weight, n*q), 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 <- spMvGLM(list(y.1~1,y.2~1,y.3~1), weights=matrix(weight,n,q),
  family="binomial", coords=coords,
  starting=
  list("beta"=beta.starting, "phi"=rep(0.06,q),"A"=A.starting, "w"=0),
  tuning=
  list("beta"=beta.tuning, "phi"=rep(0.05,q),"A"=rep(0.01,nltr), "w"=0.001),
  priors=
  list("beta.Flat", "phi.Unif"=rep(c(0.03, 0.3),q),"K.IW"=list(q+1, diag(0.1,q))),
  amcmc=list("n.batch"=n.batch,"batch.length"=batch.length,"accept.rate"=0.43),
  cov.model="exponential", sub.sample=c(5000,n.samples,10),
  verbose=TRUE, n.report=100)

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

beta.hat <- colMeans(m.1$p.samples[,1:q])
w.hat <- rowMeans(m.1$sp.effects)

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

p.hat.1 <- p.hat[seq(1,length(p.hat),q)]
p.hat.2 <- p.hat[seq(2,length(p.hat),q)]
p.hat.3 <- p.hat[seq(3,length(p.hat),q)]

##Take a look
par(mfrow=c(3,2))
surf <- mba.surf(cbind(coords,y.1/weight),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf, main="Observed rates")

```

```

surf <- mba.surf(cbind(coords,p.hat.1),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf, main="Fitted rates")

surf <- mba.surf(cbind(coords,y.2/weight),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

surf <- mba.surf(cbind(coords,p.hat.2),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

surf <- mba.surf(cbind(coords,y.3/weight),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

surf <- mba.surf(cbind(coords,p.hat.3),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

##Prediction
m <- 200
pred.coords <- cbind(runif(m,1,100),runif(m,1,100))
pred.covars <- mkMvX(list(matrix(1,m,1), matrix(1,m,1), matrix(1,m,1)))

pred <- spPredict(m.1, pred.coords, pred.covars)

pred.p <- rowMeans(pred$y.pred)
pred.p.1 <- pred.p[seq(1,length(pred.p),q)]
pred.p.2 <- pred.p[seq(2,length(pred.p),q)]
pred.p.3 <- pred.p[seq(3,length(pred.p),q)]

##Take a look
par(mfrow=c(3,2))
surf <- mba.surf(cbind(coords,y.1/weight),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf, main="Observed rates")

surf <- mba.surf(cbind(pred.coords,pred.p.1),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf, main="Fitted rates")

surf <- mba.surf(cbind(coords,y.2/weight),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

surf <- mba.surf(cbind(pred.coords,pred.p.2),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

surf <- mba.surf(cbind(coords,y.3/weight),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

surf <- mba.surf(cbind(pred.coords,pred.p.3),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image.plot(surf)

#####
##Spatial multivariate poisson
#####
##Generate some data
n <- 100

```

```

q <- 3
nltr <- q*(q-1)/2+q

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

theta <- rep(3/50,q)

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

Psi <- diag(0,q)

c1 <- mvCovInvLogDet(coords=coords, knots=coords,
  cov.model="exponential",
  V=K, Psi=Psi, theta=theta,
  modified.pp=TRUE, SWM=FALSE)

w <- mvrnorm(1,rep(0,nrow(c1$C)),c1$C)

X <- mkMvX(list(matrix(1,n,1), matrix(1,n,1), matrix(1,n,1)))
beta <- c(-1,0,1)
y <- rpois(n*q, exp(X%%beta+w))

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

##Specify starting values and collect samples. For
##a true analysis, several longer chains should be
##run.
A.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]

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

n.samples <- 5000

m.1 <- spMvGLM(list(y.1~1,y.2~1,y.3~1), family="poisson", coords=coords,
  starting=
  list("beta"=beta.starting, "phi"=rep(0.06,q),
    "A"=A.starting, "w"=0),
  tuning=
  list("beta"=beta.tuning, "phi"=rep(0.01,q),
    "A"=rep(0.005,nltr), "w"=0.001),
  priors=
  list("beta.Flat", "phi.Unif"=rep(c(0.03, 0.3),q),
    "K.IW"=list(q+1, diag(0.1,q))),
  cov.model="exponential",
  n.samples=n.samples, sub.sample=c(5000,n.samples,10),
  verbose=TRUE, n.report=500)

m.1$p.samples[,paste("phi_",1:q,sep="")] <-

```

```

3/m.1$p.samples[,paste("phi_",1:q,sep="")]

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

beta.hat <- colMeans(m.1$p.samples[,1:q])
w.hat <- rowMeans(m.1$sp.effects)

y.hat <- exp(X%*%beta.hat+w.hat)

y.hat.1 <- y.hat[seq(1,length(y.hat),q)]
y.hat.2 <- y.hat[seq(2,length(y.hat),q)]
y.hat.3 <- y.hat[seq(3,length(y.hat),q)]

##Take a look
par(mfrow=c(3,2))
surf <- mba.surf(cbind(coords,y.1),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Observed counts")
contour(surf, drawlabels=FALSE, add=TRUE)
text(coords, labels=y.1, cex=1, col="blue")

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

surf <- mba.surf(cbind(coords,y.2),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf)
contour(surf, drawlabels=FALSE, add=TRUE)
text(coords, labels=y.2, cex=1, col="blue")

surf <- mba.surf(cbind(coords,y.hat.2),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf)
contour(surf, drawlabels=FALSE, add=TRUE)
text(coords, labels=round(y.hat.2,0), cex=1, col="blue")

surf <- mba.surf(cbind(coords,y.3),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf)
contour(surf, drawlabels=FALSE, add=TRUE)
text(coords, labels=y.3, cex=1, col="blue")

surf <- mba.surf(cbind(coords,y.hat.3),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf)
contour(surf, drawlabels=FALSE, add=TRUE)
text(coords, labels=round(y.hat.3,0), cex=1, col="blue")

## End(Not run)

```

## 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, sp.tuning, priors, cov.model,
        modified.pp = TRUE, n.samples, sub.samples,
        verbose=TRUE, n.report=100, ...)
```

## Arguments

<code>formula</code>	for a multivariate model with $q$ response variables, this is a list of univariate formulas. See example below.
<code>data</code>	an optional data frame containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>spMvLM</code> is called.
<code>coords</code>	an $n \times 2$ matrix of the observation coordinates in $R^2$ (e.g., easting and northing).
<code>knots</code>	either a $m \times 2$ matrix of the <i>predictive process</i> 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 <code>coords</code> extent.
<code>starting</code>	a list with each tag corresponding to a parameter name. Valid list tags are <code>beta</code> , <code>A</code> , <code>L</code> , <code>phi</code> , and <code>nu</code> . The value portion of each tag is a vector of parameter's starting value. For <code>A</code> and <code>L</code> the vectors are of length $\frac{(q^2-q)}{2} + q$ and <code>phi</code> and <code>nu</code> are of length $q$ . Here, <code>A</code> and <code>L</code> hold the the lower-triangle elements in column major ordering of the Cholesky square root of the spatial and non-spatial cross-covariance matrices, respectively.
<code>sp.tuning</code>	a list with each tag corresponding to a parameter name. Valid list tags are <code>A</code> , <code>L</code> , <code>phi</code> , and <code>nu</code> . The value portion of each of each tag defines the step size of the proposal used in the Metropolis sampling. For <code>A</code> and <code>L</code> the vectors are of length $\frac{(q^2-q)}{2} + q$ and <code>phi</code> and <code>nu</code> are of length $q$ .
<code>modified.pp</code>	a logical value indicating if the <i>modified predictive process</i> should be used (see references below for details). Note, if a predictive process model is not used (i.e., <code>knots</code> is not specified) then this argument is ignored.
<code>priors</code>	a list with each tag corresponding to a parameter name. Valid list tags are <code>K.IW</code> , <code>Psi.IW</code> , <code>phi.unif</code> , and <code>nu.unif</code> (Beta priors are assumed <i>flat</i> ). Variance parameters, <code>K</code> and <code>Psi</code> , are assumed to follow an inverse-Wishart distribution, whereas the spatial decay <code>phi</code> and smoothness <code>nu</code> 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 <i>df</i> and $q \times q$ <i>scale</i> matrix, respectively. The hyperparameters of the Uniform are also passed as a vector of length $2 \times q$ with consecutive elements representing the first and second elements corresponding to the lower and upper support in the order of the univariate models given in <code>formula</code> .

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.
n.samples	the number of MCMC iterations.
sub.samples	a vector of length 3 that specifies <i>start</i> , <i>end</i> , and <i>thin</i> , respectively, of the MCMC samples. The default is <code>c(1, n.samples, 1)</code> (i.e., all samples).
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.

### Value

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

coords	the $n \times 2$ matrix specified by <code>coords</code> .
knot.coords	the $m \times 2$ matrix as specified by <code>knots</code> .
p.samples	a coda object of posterior samples for the defined parameters.
acceptance	the Metropolis sampling acceptance rate.
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.

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.
- Finley, A.O., S. Banerjee, P. Waldmann, and T. Ericsson. (2008). Hierarchical spatial modeling of additive and dominance genetic variance for large spatial trial datasets. *Biometrics*. DOI: 10.1111/j.1541-0420.2008.01115.x
- Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2008). Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, DOI: 10.1016/j.csda.2008.09.008
- 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.

**See Also**

[spMvLM](#), [spGGT](#)

**Examples**

```
## Not run:
#####
##Multivariate spatial regression
#####

##Generate some data
n <- 50 ##observed
q <- 3
nltr <- q*(q-1)/2+q

coords <- cbind(runif(n),runif(n))
theta <- rep(3/0.5,q)

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

Psi <- diag(1,q)

c1 <- mvCovInvLogDet(coords=coords, knots=coords,
                     cov.model="exponential",
                     V=K, Psi=Psi, theta=theta,
                     modified.pp=FALSE, SWM=FALSE)

w <- mvrnorm(1,rep(0,nrow(c1$C)),c1$C)

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

##Specify starting values and collect samples
K.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]
Psi.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]

n.samples <- 5000

m.1 <- spMvLM(list(w.1~1,w.2~1,w.3~1), coords=coords,
                 starting=list("beta"=rep(1,q), "phi"=rep(3/0.5,q),
                               "A"=K.starting, "L"=Psi.starting),
                 sp.tuning=list("phi"=rep(0.1,q),
                                "A"=rep(0.1,nltr), "L"=rep(0.1,nltr)),
                 priors=list("phi.Unif"=rep(c(3/1,3/0.1),q),
                             "K.IW"=list(q+1, diag(1,q)), "Psi.IW"=list(q+1, diag(1,q))),
                 modified.pp=TRUE, cov.model="exponential",
                 n.samples=n.samples, sub.samples=c(2500,n.samples,5),
                 verbose=TRUE, n.report=100)
```

```

m.1$p.samples[,paste("phi_",1:q,sep="")] <-
  3/m.1$p.samples[,paste("phi_",1:q,sep="")]

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

w.hat <- rowMeans(m.1$sp.effects)
w.hat.1 <- w.hat[seq(1,length(w.hat),q)]
w.hat.2 <- w.hat[seq(2,length(w.hat),q)]
w.hat.3 <- w.hat[seq(3,length(w.hat),q)]

##Take a look
par(mfrow=c(3,2))
surf <- mba.surf(cbind(coords,w.1),
                no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Observed 1"); contour(surf, add=TRUE)

surf <- mba.surf(cbind(coords,w.hat.1),
                no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Fitted 1"); contour(surf, add=TRUE)
points(m.1$knot.coords, pch=19, cex=1)

surf <- mba.surf(cbind(coords,w.2),
                no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Observed 2"); contour(surf, add=TRUE)

surf <- mba.surf(cbind(coords,w.hat.2),
                no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Fitted 2"); contour(surf, add=TRUE)
points(m.1$knot.coords, pch=19, cex=1)

surf <- mba.surf(cbind(coords,w.3),
                no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Observed 3"); contour(surf, add=TRUE)

surf <- mba.surf(cbind(coords,w.hat.3),
                no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Fitted 3"); contour(surf, add=TRUE)
points(m.1$knot.coords, pch=19, cex=1)

## End(Not run)

```

---

spPredict

*Prediction for new locations given a model object*


---

## Description

The function `spPredict` collects a posterior predictive sample for a set of new locations given a `spGGT`, `spLM`, `spMvLM`, `spGLM`, `spMvGLM`, or `bayesGeostatExact` object.

**Usage**

```
spPredict(sp.obj, pred.coords, pred.covars,
          start=1, end, thin=1, verbose=TRUE, ...)
```

**Arguments**

sp.obj	an object returned by <code>spGGT</code> , <code>bayesGeostatExact</code> , <code>spLM</code> , <code>spMvLM</code> , <code>spGLM</code> , or <code>spMvGLM</code>
pred.coords	an $n \times 2$ matrix of $m$ prediction location coordinates in $R^2$ (e.g., easting and northing). The first column is assumed to be easting coordinates and the second column northing coordinates. Prediction locations cannot coincide with observed locations.
pred.covars	an $n \times p$ design matrix associated with the new locations. If this is a multivariate prediction defined by $m$ models, the multivariate design matrix can be created by passing a list of the $m$ univariate design matrices to the <code>mkMvX</code> function.
start	specifies the first sample included in the prediction calculation. This is useful for those who choose to acknowledge chain burn-in.
end	specifies the last sample included in the prediction calculation. The default is to use all posterior samples in <code>sp.obj</code> .
thin	a sample thinning factor. The default of 1 considers all samples between <code>start</code> and <code>end</code> . For example, if <code>thin = 10</code> then 1 in 10 samples are considered between <code>start</code> and <code>end</code> .
verbose	if TRUE calculation progress is printed to the screen; otherwise, nothing is printed to the screen.
...	currently no additional arguments.

**Value**

y.pred	a matrix that holds samples from the posterior predictive distribution. For multivariate models the rows of this matrix correspond to the predicted locations and the columns are the posterior predictive samples. If prediction is for $m$ response variables the <code>y.pred</code> matrix has $mn$ rows. The predictions for locations are held in rows $1:m$ , $(m+1):2m$ , ..., $((i-1)m+1):im$ , ..., $((n-1)m+1):nm$ , where $i = 1 \dots n$ (e.g., the samples for the first location are in rows $1:m$ , second location in rows $(m+1):2m$ , etc.). For <code>spGLM</code> the <code>y.pred</code> matrix holds posterior predictive samples for $\frac{1}{1+\exp(-x(s)'\beta-w(s))}$ and $\exp(x(s)'\beta + w(s))$ for family binomial and poisson, respectively. Here $s$ indexes the prediction location, $\beta$ are the regression coefficients, and $w$ is the associated random spatial effect. These values can be fed directly into <code>rbinom</code> or <code>rpois</code> to generate the realization from the respective distribution.
w.pred	a matrix that holds samples from the random spatial effects posterior predictive distribution. Again, matrix rows correspond to locations and columns to samples.

**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.
- Finley, A.O., H. Sang, S. Banerjee, and A.E. Gelfand. (2008). Improving the performance of predictive process modeling for large datasets. *Computational Statistics and Data Analysis*, DOI: 10.1016/j.csda.2008.09.008
- 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.
- Banerjee, S., Gelfand, A.E., Finley, A.O., and Sang, H. (In press). Gaussian predictive process models for large spatial datasets. *Journal of the Royal Statistical Society Series B*.

**See Also**

[spGGT](#), [bayesGeostatExact](#), [spLM](#), [spMvLM](#), [spGLM](#), [spMvGLM](#)

**Examples**

```
## Not run:
##Portions of this example requires MBA package to make surfaces
library(MBA)

#####
##Prediction for spGLM
#####
set.seed(1234)

##Generate count data
n <- 300

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

phi <- 3/75
sigma.sq <- 3

R <- sigma.sq*exp(-phi*as.matrix(dist(coords)))
w <- mvrnorm(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.samples <- 15000

m.1 <- spGLM(y~1, family="poisson", coords=coords, knots=c(8,8,0),
  starting=
  list("beta"=beta.starting, "phi"=0.06,
    "sigma.sq"=1, "w"=0),
  tuning=
  list("beta"=beta.tuning, "phi"=0.1,
    "sigma.sq"=0.1, "w"=0.001),
  priors=
  list("beta.Flat", "phi.Unif"=c(0.03, 0.3),
    "sigma.sq.IG"=c(2, 1)),
  cov.model="exponential",
  n.samples=n.samples, sub.samples=c(10000,n.samples,5),
  verbose=TRUE, n.report=500)

##Make prediction grid
pred.coords <- expand.grid(seq(0,100,5), seq(0,100,5))
out <- spPredict(m.1, pred.coords=pred.coords,
  pred.covars=as.matrix(rep(1,nrow(pred.coords))))

##Get realizations using rpois
y.pred <-
  rowMeans(apply(out$y.pred, 2, function(x) rpois(length(x),x)))

##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)

surf <-
  mba.surf(cbind(pred.coords, y.pred),no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(surf, main="Predicted counts")
contour(surf, add=TRUE)
points(pred.coords, pch=19, cex=0.5, col="blue")

#####
##Prediction for spMvLM
#####

##Generate some data
n <- 100 ##observed
m <- 50 ##predict
N <- n+m
q <- 3
nltr <- q*(q-1)/2+q

coords <- cbind(runif(N),runif(N))
theta <- rep(3/0.5,q)

```

```

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

Psi <- diag(1,q)

c1 <- mvCovInvLogDet(coords=coords, knots=coords,
                    cov.model="exponential",
                    V=V, Psi=Psi, theta=theta)

w <- mvrnorm(1,rep(0,nrow(c1$C)),c1$C)
obs.w <- w[1:(n*q)]

w.1 <- obs.w[seq(1,length(obs.w),q)]
w.2 <- obs.w[seq(2,length(obs.w),q)]
w.3 <- obs.w[seq(3,length(obs.w),q)]

##Specify starting values and collect samples
A.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]
L.starting <- diag(1,q)[lower.tri(diag(1,q), TRUE)]

n.samples <- 1000

obs.coords <- coords[1:n,]

m.1 <- spMvLM(list(w.1~1,w.2~1,w.3~1), coords=obs.coords,
                knots=c(8,8,0),
                starting=list("beta"=rep(1,q), "phi"=rep(3/0.5,q),
                              "nu"=rep(1,q), "A"=A.starting, "L"=L.starting),
                sp.tuning=list("phi"=rep(0.1,q), "nu"=rep(0.1,q),
                              "A"=rep(0.1,nltr), "L"=rep(0.1,nltr)),
                priors=list("phi.Unif"=rep(c(3/1,3/0.1),q),
                           "nu.Unif"=rep(c(0.1,2),q),
                           "K.IW"=list(q+1, diag(1,q)), "Psi.IW"=list(q+1, diag(1,q))),
                modified.pp=FALSE, cov.model="matern",
                n.samples=n.samples, verbose=TRUE, n.report=100)

##Predict for holdout set
pred.coords <- coords[(n+1):nrow(coords),]
pred.covars <- mkMvX(list(matrix(1,m,1), matrix(1,m,1), matrix(1,m,1)))

pred <- spPredict(m.1, pred.coords, pred.covars)

ho.w <- w[(n*q+1):length(w)]
ho.w.1 <- ho.w[seq(1,length(ho.w),q)]
ho.w.2 <- ho.w[seq(2,length(ho.w),q)]
ho.w.3 <- ho.w[seq(3,length(ho.w),q)]

burn.in <- 500

pred.w <- rowMeans(pred$y.pred[,burn.in:ncol(pred$y.pred)])
pred.w.1 <- pred.w[seq(1,length(pred.w),q)]
pred.w.2 <- pred.w[seq(2,length(pred.w),q)]

```

```

pred.w.3 <- pred.w[seq(3,length(pred.w),q)]

##Take a look
par(mfrow=c(3,2))
surf <- mba.surf(cbind(obs.coords,w.1),
                 no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Observed"); contour(surf, add=TRUE)

surf <- mba.surf(cbind(pred.coords,pred.w.1),
                 no.X=100, no.Y=100, extend=T)$xyz.est
image(surf, main="Predicted"); contour(surf, add=TRUE)
points(m.1$knot.coords, pch=19, cex=1)

surf <- mba.surf(cbind(obs.coords,w.2),
                 no.X=100, no.Y=100, extend=T)$xyz.est
image(surf); contour(surf, add=TRUE)

surf <- mba.surf(cbind(pred.coords,pred.w.2),
                 no.X=100, no.Y=100, extend=T)$xyz.est
image(surf); contour(surf, add=TRUE)
points(m.1$knot.coords, pch=19, cex=1)

surf <- mba.surf(cbind(obs.coords,w.3),
                 no.X=100, no.Y=100, extend=T)$xyz.est
image(surf); contour(surf, add=TRUE)

surf <- mba.surf(cbind(pred.coords,pred.w.3),
                 no.X=100, no.Y=100, extend=T)$xyz.est
image(surf); contour(surf, add=TRUE)
points(m.1$knot.coords, pch=19, cex=1)

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

n.samples <-1000
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
#####
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,
                        sp.effects=TRUE)

##Now prediction
set.seed(1)
pred.coords <- expand.grid(seq(0,100,length=10),seq(0,100,length=10))
pred.covars <- as.matrix(rep(1,nrow(pred.coords)))

m.1.pred <- spPredict(m.1, pred.coords=pred.coords,
                    pred.covars=pred.covars, thin=5)

par(mfrow=c(2,2))
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, pch=19, cex=1, col="green")
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="Random effects")
points(coords, pch=19, cex=1, col="green")
contour(w.surf, add=T)

y.hat <- rowMeans(m.1.pred)
y.surf <-
  mba.surf(cbind(pred.coords, y.hat), no.X=100, no.Y=100, extend=T)$xyz.est
image(y.surf, xaxs = "r", yaxs = "r", main="Predicted response")
points(pred.coords, pch=19, cex=1, col="black")
rect(0, 0, 50, 50, col=NA, border="green")
contour(y.surf, add=T)

y.var <- apply(m.1.pred, 1, var)
y.surf <-
  mba.surf(cbind(pred.coords, y.var), no.X=100, no.Y=100, extend=T)$xyz.est
image(y.surf, xaxs = "r", yaxs = "r", main="Predicted response\nvariance")
points(coords, pch=19, cex=1, col="green")
points(pred.coords, pch=19, cex=1, col="black")
rect(0, 0, 50, 50, col=NA, border="green")
contour(y.surf, add=T)

#####
```

```

##      Prediction for spLM
#####
data(rf.n200.dat)

Y <- rf.n200.dat$Y
coords <- as.matrix(rf.n200.dat[,c("x.coords", "y.coords")])
w <- rf.n200.dat$w

pred.coords <- expand.grid(seq(1,10,1), seq(1,10,1))
n.pred <- nrow(pred.coords)

#####
##Prediction with a spLM model
#####
m.2 <- spLM(Y~1, coords=coords,
            starting=list("phi"=0.6, "sigma.sq"=1, "tau.sq"=1),
            sp.tuning=list("phi"=0.01, "sigma.sq"=0.05, "tau.sq"=0.05),
            priors=list("phi.Unif"=c(0.3, 3), "sigma.sq.IG"=c(2, 1),
                       "tau.sq.IG"=c(2, 1)),
            cov.model="exponential",
            n.samples=1000, verbose=TRUE, n.report=100)

pred <- spPredict(m.2, pred.coords,
                 pred.covars=as.matrix(rep(1,n.pred)))

par(mfrow=c(1,2))
obs.surf <-
  mba.surf(cbind(coords, Y), 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)

y.hat <- rowMeans(pred$y.pred)
y.pred.surf <-
  mba.surf(cbind(pred.coords, y.hat), no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(y.pred.surf, xaxs = "r", yaxs = "r", main="Predicted response")
points(coords, pch=1, cex=1)
points(pred.coords, pch=19, cex=1)
contour(y.pred.surf, add=T)
legend(1.5,2.5, legend=c("Obs.", "Pred."), pch=c(1,19), bg="white")

#####
##Prediction with a spLM
##predictive process model
#####
m.3 <- spLM(Y~1, coords=coords, knots=c(6,6,0),
            starting=list("phi"=0.6, "sigma.sq"=1, "tau.sq"=1),
            sp.tuning=list("phi"=0.01, "sigma.sq"=0.01, "tau.sq"=0.01),
            priors=list("phi.Unif"=c(0.3, 3), "sigma.sq.IG"=c(2, 1),
                       "tau.sq.IG"=c(2, 1)),
            cov.model="exponential",
            n.samples=2000, verbose=TRUE, n.report=100)

```

```

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

pred <- spPredict(m.3, pred.coords,
                  pred.covars=as.matrix(rep(1,n.pred)))

par(mfrow=c(1,2))
obs.surf <-
  mba.surf(cbind(coords, Y), 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)

y.hat <- rowMeans(pred$y.pred)
y.pred.surf <-
  mba.surf(cbind(pred.coords, y.hat), no.X=100, no.Y=100, extend=TRUE)$xyz.est
image(y.pred.surf, xaxs = "r", yaxs = "r", main="Predicted response")
points(coords, pch=1, cex=1)
points(m.3$knot.coords, pch=3, cex=1)
points(pred.coords, pch=19, cex=1)
contour(y.pred.surf, add=T)
legend(1.5,2.5, legend=c("Obs.", "Knots", "Pred."),
      pch=c(1,3,19), bg="white")

#####
##      Prediction for spGGT
#####
data(FBC07.dat)

##Divide the data into model and prediction sets
Y.1 <- FBC07.dat[1:100,"Y.1"]
Y.2 <- FBC07.dat[1:100,"Y.2"]
model.coords <- as.matrix(FBC07.dat[1:100,c("coord.X", "coord.Y")])
pred.coords <- as.matrix(FBC07.dat[151:200,c("coord.X", "coord.Y")])

#####
##  Univariate model
#####

##Fit some model with spGGT.
K.prior <- prior(dist="IG", shape=2, scale=5)
Psi.prior <- prior(dist="IG", shape=2, scale=5)
phi.prior <- prior(dist="UNIF", a=0.06, b=3)

var.update.control <-
  list("K"=list(starting=5, tuning=0.5, prior=K.prior),
       "Psi"=list(starting=5, tuning=0.5, prior=Psi.prior),
       "phi"=list(starting=0.1, tuning=0.5, prior=phi.prior)
  )

beta.control <- list(update="GIBBS", prior=prior(dist="FLAT"))

run.control <- list("n.samples"=1000)

```

```

Fit <- spGGT(formula=Y.2~1, run.control=run.control,
             coords=model.coords,
             var.update.control=var.update.control,
             beta.update.control=beta.control,
             cov.model="exponential")

##Now make predictions for the holdout set.
##Step 1. make the design matrix for the prediction locations.
pred.covars <- as.matrix(rep(1, nrow(pred.coords)))

##Step 2. call spPredict.
Pred <- spPredict(Fit, pred.covars=pred.covars,
                 pred.coords=pred.coords)

##Step 3. check out the predicted random effects and
##predicted response variable.

Pred.sp.effects.surf <-
  mba.surf(cbind(pred.coords, rowMeans(Pred$pred.sp.effects)),
           no.X=100, no.Y=100, extend=TRUE)$xyz.est

Pred.Y.surf <-
  mba.surf(cbind(pred.coords, rowMeans(Pred$pred.y)),
           no.X=100, no.Y=100, extend=TRUE)$xyz.est

par(mfrow=c(1,2))
image(Pred.sp.effects.surf, xaxs="r", yaxs="r",
      main="Predicted random spatial effects")
contour(Pred.sp.effects.surf, add=TRUE)

image(Pred.Y.surf, xaxs="r", yaxs="r",
      main="Predicted Y.2")
contour(Pred.Y.surf, add=TRUE)

#####
##  Multivariate models
#####

##Fit some model with spGGT.
K.prior <- prior(dist="IWISH", df=2, S=diag(c(3, 6)))
Psi.prior <- prior(dist="IWISH", df=2, S=diag(c(7, 5)))
phi.prior <- prior(dist="UNIF", a=0.06, b=3)

K.starting <- matrix(c(2,-3, 0, 1), 2, 2)
Psi.starting <- diag(c(3, 2))

var.update.control <-
  list("K"=list(starting=K.starting, tuning=diag(c(0.1, 0.5, 0.1)),
               prior=K.prior),
       "Psi"=list(starting=Psi.starting, tuning=diag(c(0.1, 0.5, 0.1)),
                  prior=Psi.prior),
       "phi"=list(starting=0.1, tuning=0.5,

```

```

        prior=list(phi.prior, phi.prior))
    )

beta.control <- list(update="GIBBS", prior=prior(dist="FLAT"))

run.control <- list("n.samples"=1000, "sp.effects"=FALSE)

Fit.mv <-
  spGGT(formula=list(Y.1~1, Y.2~1), run.control=run.control,
        coords=model.coords,
        var.update.control=var.update.control,
        beta.update.control=beta.control,
        cov.model="exponential")

##Now make predictions for the holdout set.
##Step 1. make the design matrix for the prediction locations using
##the mkMvX function.
pred.covars.1 <- as.matrix(rep(1, nrow(pred.coords)))
pred.covars.2 <- as.matrix(rep(1, nrow(pred.coords)))

pred.covars.mv <- mkMvX(list(pred.covars.1, pred.covars.2))

##Step 2. call spPredict.
Pred.mv <- spPredict(Fit.mv, pred.covars=pred.covars.mv,
                    pred.coords=pred.coords)

##Step 3. check out the predicted random effects and
##predicted response variables. Recall, these are
##organized as m consecutive rows for each location.
Pred.sp.effects.1 <-
  Pred.mv$pred.sp.effects[seq(1, nrow(Pred.mv$pred.sp.effects), 2),]

Pred.sp.effects.2 <-
  Pred.mv$pred.sp.effects[seq(2, nrow(Pred.mv$pred.sp.effects), 2),]

Pred.Y.1 <-
  Pred.mv$pred.sp.effects[seq(1, nrow(Pred.mv$pred.y), 2),]

Pred.Y.2 <-
  Pred.mv$pred.sp.effects[seq(2, nrow(Pred.mv$pred.y), 2),]

##Then map.
Pred.sp.effects.1.surf <-
  mba.surf(cbind(pred.coords, rowMeans(Pred.sp.effects.1)),
           no.X=100, no.Y=100, extend=TRUE)$xyz.est

Pred.sp.effects.2.surf <-
  mba.surf(cbind(pred.coords, rowMeans(Pred.sp.effects.2)),
           no.X=100, no.Y=100, extend=TRUE)$xyz.est

Pred.Y.1.surf <-
  mba.surf(cbind(pred.coords, rowMeans(Pred.Y.1)),
           no.X=100, no.Y=100, extend=TRUE)$xyz.est

```

```
Pred.Y.2.surf <-  
  mba.surf(cbind(pred.coords, rowMeans(Pred.Y.2)),  
           no.X=100, no.Y=100, extend=TRUE)$xyz.est  
  
par(mfrow=c(2,2))  
image(Pred.sp.effects.1.surf, xaxs="r", yaxs="r",  
      main="Predicted random spatial effects Y.1")  
contour(Pred.sp.effects.1.surf, add=TRUE)  
  
image(Pred.sp.effects.2.surf, xaxs="r", yaxs="r",  
      main="Predicted random spatial effects Y.2")  
contour(Pred.sp.effects.2.surf, add=TRUE)  
  
image(Pred.Y.1.surf, xaxs="r", yaxs="r",  
      main="Predicted Y.1")  
contour(Pred.Y.1.surf, add=TRUE)  
  
image(Pred.Y.2.surf, xaxs="r", yaxs="r",  
      main="Predicted Y.2")  
contour(Pred.Y.2.surf, add=TRUE)  
  
## End(Not run)
```

---

synthetic.dat

*Synthetic univariate data used for illustrations*

---

## Description

The synthetic datasets were generated from a stationary and isotropic spatial process using an exponential correlation function (code below).

## Usage

```
data(rf.n200.dat)  
data(rf.n500.dat)  
data(rf.n1000.dat)
```

## Format

The rf.n200.dat, rf.n500.dat, rf.n1000.dat files contain n rows (i.e., points) and 4 columns. Column 1, Y, is the response variable. Column 2, w, is the vector of random spatial effects. Columns 4 and 5 are the x and y coordinates.

**Examples**

```
## Not run:

##These synthetic data were generated with the following:

##n <- 200
##n <- 500
n <- 1000

coords <- cbind(runif(n, 1, 10), runif(n, 1, 10))

sigma.sq <- 10
tau.sq <- 1
phi <- 3/5

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

Y <- w+rnorm(n, 0, sqrt(tau.sq))

## 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 Canton 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

## \*Topic **datasets**

BEF.dat, 15  
FBC07.dat, 20  
FORMGMT.dat, 22  
synthetic.dat, 78  
WEF.dat, 79  
Zurich.dat, 80

## \*Topic **misc**

adaptMetropGibbs, 2  
bayesGeostatExact, 7  
bayesLMConjugate, 13  
bayesLMRef, 14  
covGivens, 16  
covInvLogDet, 17  
hexGrid, 22  
iDist, 23  
mkMvX, 24  
mvCovInvLogDet, 25  
mvLM, 28  
pointsInPoly, 31  
prior, 33  
spDiag, 34  
spGGT, 36  
spGLM, 45  
spLM, 52  
spMvGLM, 56  
spMvLM, 63  
spPredict, 67

adaptMetropGibbs, 2

bayesGeostatExact, 7, 34, 35, 67–69  
bayesLMConjugate, 12, 34, 35  
bayesLMRef, 14, 34, 35  
BEF.dat, 15

covGivens, 16  
covInvLogDet, 17

FBC07.dat, 20

FORMGMT.dat, 22

hexGrid, 22

iDist, 23

mkMvX, 24, 68  
mvCovInvLogDet, 25  
mvLM, 28, 34, 35

pointsInPoly, 31  
prior, 33, 37, 38

rf.n1000.dat (synthetic.dat), 78  
rf.n200.dat (synthetic.dat), 78  
rf.n500.dat (synthetic.dat), 78

spDiag, 34  
spGGT, 29, 33–35, 36, 48, 54, 59, 66–69  
spGLM, 34, 35, 45, 67–69  
spLM, 34–36, 52, 67–69  
spMvGLM, 48, 56, 59, 67–69  
spMvLM, 29, 34, 35, 48, 54, 59, 63, 66–69  
spPredict, 24, 25, 38, 67  
synthetic.dat, 78

WEF.dat, 79

Zurich.dat, 80