Package 'spNNGP'

September 3, 2019

Title Spatial Regression Models for Large Datasets using Nearest Neighbor Gaussian Processes

Version 0.1.2

Date 2019-08-28

Maintainer Andrew Finley <finleya@msu.edu>

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

Depends R (>= 3.5.0), coda, Formula, RANN

Description Fits univariate Bayesian spatial regression models for large datasets using Nearest Neighbor Gaussian Processes (NNGP) detailed in Finley, Datta, Cook, Morton, Andersen, and Banerjee (2019) <doi:10.1080/10618600.2018.1537924> and Datta, Banerjee, Finley, and Gelfand (2016) <doi:10.1080/01621459.2015.1044091>.

License GPL (>= 2)

Encoding UTF-8

URL http://blue.for.msu.edu/software.html

Repository CRAN

NeedsCompilation yes

Date/Publication 2019-09-03 08:50:02 UTC

R topics documented:

BCEF *Forest Canopy Height from NASA Goddard's LiDAR Hyperspectral and Thermal (G-LiHT) over Bonanza Creek Experimental Forest*

Description

Forest canopy height (FCH) estimates from NASA Goddard's LiDAR Hyperspectral and Thermal (G-LiHT; Cook et al. 2013) Airborne Imager and percent tree cover (Hansen et al. 2013) over a subset of Bonanza Creek Experimental Forest, AK, collected in Summer 2014.

The BCEF matrix columns are longitude (x), latitude (y), forest canopy height (FCH) in meters from ground, and Landsat derived percent tree cover (PTC) for 188,717 observations. Longitude and latitude are in Albers Equal Area (proj4string "+proj=aea +lat_1=55 +lat_2=65 +lat_0=50 +lon_0=-154 +x_0=0 +y_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs"). The last column (holdout) identifies a subset of data useful for assessing wall-to-wall predictive performance.

Usage

```
data(BCEF)
```

Format

A matrix containing 188,717 rows and 6 columns named longitude, latitude, FCH, PTC, and holdout.

Source

G-LiHT data were downloaded from https://gliht.gsfc.nasa.gov.

References

Cook, B.D., L.W. Corp, R.F. Nelson, E.M. Middleton, D.C. Morton, J.T. McCorkel, J.G. Masek, K.J. Ranson, and V. Ly. (2013) NASA Goddard's Lidar, Hyperspectral and Thermal (G-LiHT) airborne imager. Remote Sensing 5:4045-4066.

Hansen, M.C., Potapov, P.V., Moore, R., Hancher, M., Turubanova, S.A., Tyukavina, A.,Thau, D., Stehman, S.V., Goetz, S.J., Loveland, T.R., Kommareddy, A., Egorov, A., Chini, L., Justice, C.O., and Townshend, J.R.G. (2013), High-Resolution Global Mapsof 21st-Century Forest Cover Change, Science, 342, 850-853.

See Also

BCEF_PTC.

BCEF_PTC *Percent Tree Cover over Bonanza Creek Experimental Forest*

Description

Percent tree cover (Hansen et al. 2013) over a subset of Bonanza Creek Experimental Forest, AK. Can be used as the set of prediction locations and covariate for models fit using the BCEF data.

The BCEF_PTC matrix columns are longitude (x), latitude (y), and Landsat derived percent tree cover (PTC) for 237,617 observations. Longitude and latitude are in Albers Equal Area (proj4string "+proj=aea +lat_1=55 +lat_2=65 +lat_0=50 +lon_0=-154 +x_0=0 +y_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=km +no_defs").

Usage

```
data(BCEF_PTC)
```

Format

A matrix containing 237,617 rows and 3 columns named longitude, latitude, and PTC.

References

Hansen, M.C., Potapov, P.V., Moore, R., Hancher, M., Turubanova, S.A., Tyukavina, A.,Thau, D., Stehman, S.V., Goetz, S.J., Loveland, T.R., Kommareddy, A., Egorov, A., Chini, L., Justice, C.O., and Townshend, J.R.G. (2013), High-Resolution Global Mapsof 21st-Century Forest Cover Change, Science, 342, 850-853.

See Also

BCEF.

PGLogit *Function for Fitting Logistic Models using Polya-Gamma Latent Variables*

Description

The function PGLogit fits logistic models to binomial data using Polya-Gamma latent variables.

Usage

```
PGLogit(formula, weights = 1, data = parent.frame(), n.samples,
        n.omp.threads = 1, fit.rep = FALSE, sub.sample, verbose = TRUE, ...)
```

Arguments

formula	a symbolic description of the regression model to be fit. See example below.
weights	specifies the number of trials for each observation. The default is 1 trial for each observation. Valid arguments are a scalar value that specifies the number of trials used if all observations have the same number of trials, and a vector of length n that specifies the number of trials for each observation when there are differences in the number of trials.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `PGLogit` is called.
n.samples	the number of posterior samples to collect.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on all systems.
fit.rep	if `TRUE`, regression fitted and replicate data will be returned. The argument `sub.sample` controls which MCMC samples are used to generate the fitted and replicated data.
sub.sample	an optional list that specifies the samples used for `fit.rep`. Valid tags are `start`, `end`, and `thin`. Given the value associated with the tags, the sample subset is selected using `seq(as.integer(start),as.integer(end),by=as.integer(thin))`. The default values are `start=floor(0.5*n.samples)`, `end=n.samples` and `thin=1`.
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 `PGLogit` which is a list comprising:

p.beta.samples	a coda object of posterior samples for the regression coefficients.
y.hat.samples	if `fit.rep=TRUE`, regression fitted values from posterior samples specified using `sub.sample`.
y.hat.quants	if `fit.rep=TRUE`, 0.5, 0.025, and 0.975 quantiles of the `y.hat.samples`.
y.rep.samples	if `fit.rep=TRUE`, replicated outcome from posterior samples specified using `sub.sample`.
y.rep.quants	if `fit.rep=TRUE`, 0.5, 0.025, and 0.975 quantiles of the `y.rep.samples`.
s.indx	if `fit.rep=TRUE`, the subset index specified with `sub.sample`.
run.time	MCMC sampler execution time reported using `proc.time()`.

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

Note

Some of the underlying code used for generating random number from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Abhirup Datta <abhidatta@jhu.edu>,
Sudipto Banerjee <sudipto@ucla.edu>

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. Journal of the American Statistical Association, 108:1339-1349.

Examples

```
##Generate binary data
set.seed(1)
n <- 100

x <- cbind(1, rnorm(n), runif(n,0,1))
beta <- c(0.1,-5, 5)
p <- 1/(1+exp(-(x%*%beta)))

##Assume 5 trials per outcome
weights <- rep(5, n)

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

m <- PGLogit(y~x-1, weights = rep(5, n), n.samples = 1000)

summary(m)
```

---

print.spDiag *Methods for spDiag Object*

Description

Methods for extracting information from spDiag.

Usage

```
## S3 method for class 'spDiag'
print(x, ...)
```

Arguments

x	object of class spDiag.
...	currently not used.

Details

A standard extractor function for printing objects of class spDiag.

---

spConjNNGP *Function for Fitting Univariate Bayesian Conjugate Spatial Regression Models*

Description

The function spConjNNGP fits Gaussian univariate Bayesian conjugate spatial regression models using Nearest Neighbor Gaussian Processes (NNGP).

Usage

```
spConjNNGP(formula, data = parent.frame(), coords, knots, n.neighbors = 15,
           theta.alpha, sigma.sq.IG, cov.model = "exponential",
           k.fold = 5, score.rule = "crps",
           X.0, coords.0, n.omp.threads = 1, search.type = "cb",
           ord, return.neighbor.info = TRUE,
           neighbor.info, fit.rep = FALSE, n.samples, verbose = TRUE, ...)
```

Arguments

formula	a symbolic description of the regression model to be fit. See example below.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which spConjNNGP is called.
coords	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing), or if data is a data frame then coords can be a vector of length two comprising coordinate column names or indices. There can be no duplicate locations.
knots	an $r \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing). Adding the knots argument invokes SLGP, see Shin et al. (2019) below.
n.neighbors	number of neighbors used in the NNGP.
theta.alpha	a vector or matrix of parameter values for phi, nu, and alpha, where $\alpha = \tau^2/\sigma^2$ and nu is only required if cov.model="matern". A vector is passed to run the model using one set of parameters. The vector elements must be named and hold values for phi, nu, and alpha. If a matrix is passed, columns must be named and hold values for phi, nu, and alpha. Each row in the matrix defines a set of parameters for which the model will be run.

`sigma.sq.IG` a vector of length two that holds the hyperparameters, *shape* and *scale* respectively, for the inverse-Gamma prior on σ^2.

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.

`k.fold` specifies the number of *k* folds for cross-validation. If `theta.alpha` is a vector then cross-validation is not performed and `k-fold` and `score.rule` are ignored. In *k*-fold cross-validation, the data specified in `model` is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k*-1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). Root mean squared prediction error (RMSPE) and continuous ranked probability score (CRPS; Gneiting and Raftery, 2007) rules are averaged over the *k* fold prediction results and reported for the parameter sets defined by `theta.alpha`. The parameter set that yields the *best* performance based on the scoring rule defined by `score.rule` is used to fit the final model that uses all the data and make predictions if `X.0` and `coords.0` are supplied. Results from the *k*-fold cross-validation are returned in the `k.fold.scores` matrix.

`score.rule` a quoted keyword `"rmspe"` or `"crps"` that specifies the scoring rule used to select the *best* parameter set, see argument definition for `k.fold` for more details.

`X.0` the design matrix for prediction locations. An intercept should be provided in the first column if one is specified in `model`.

`coords.0` the spatial coordinates corresponding to `X.0`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.

`search.type` a quoted keyword that specifies type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering (see `ord` argument) then `"cb"` and `"brute"` should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then `"cb"` and `"brute"` might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`ord` an index vector of length n used for the nearest neighbor search. Internally, this vector is used to order `coords`, i.e., `coords[ord,]`, and associated data. Nearest neighbor candidates for the *i*-th row in the ordered `coords` are rows `1:(i-1)`, with the `n.neighbors` nearest neighbors being those with the minimum euclidean distance to the location defined by ordered `coords[i,]`. The default used when `ord` is not specified is x-axis ordering, i.e., `order(coords[,1])`. This argument should typically be left blank. This argument will be ignored if the `neighbor.info` argument is used.

return.neighbor.info
: if TRUE, a list called neighbor.info containing several data structures used for fitting the NNGP model is returned. If there is no change in input data or n.neighbors, this list can be passed to subsequent spNNGP calls via the neighbor.info argument to avoid the neighbor search, which can be time consuming if n is large. In addition to the several cryptic data structures in neighbor.info there is a list called n.indx that is of length n. The i-th element in n.indx corresponds to the i-th row in coords[ord,] and holds the vector of that location's nearest neighbor indices. This list can be useful for plotting the neighbor graph if desired.

neighbor.info
: see the return.neighbor.info argument description above.

fit.rep
: if TRUE, regression fitted and replicate data will be returned. The argument n.samples controls the number of fitted and replicated data samples.

n.samples
: gives the number of posterior samples returned. Note, point and associated variance estimates for model parameters are not based on posterior samples. Only specify n.samples if you wish to generate samples from parameters' posteriors (this is an exact sampling algorithm). If fit.rep is TRUE, then n.samples also controls the number of fitted and replicated data samples.

verbose
: if TRUE, model specification and progress is printed to the screen. Otherwise, nothing is printed to the screen.

...
: currently no additional arguments.

Value

An object of class NNGP and conjugate, and, if knots is provided, SLGP. Among other elements, the object contains:

theta.alpha
: the input theta.alpha vector, or *best* (according to the selected scoring rule) set of parameters in the theta.alpha matrix. All subsequent parameter estimates are based on this parameter set.

beta.hat
: a matrix of regression coefficient estimates corresponding to the returned theta.alpha.

beta.var
: beta.hat variance-covariance matrix.

sigma.sq.hat
: estimate of σ^2 corresponding to the returned theta.alpha.

sigma.sq.var
: sigma.sq.hat variance.

k.fold.scores
: results from the k-fold cross-validation if theta.alpha is a matrix.

y.0.hat
: prediction if X.0 and coords.0 are specified.

y.0.var.hat
: y.0.hat variance.

n.neighbors
: number of neighbors used in the NNGP.

neighbor.info
: returned if return.neighbor.info=TRUE see the return.neighbor.info argument description above.

run.time
: execution time for parameter estimation reported using proc.time(). This time does not include nearest neighbor search time for building the neighbor set.

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Abhirup Datta <abhidatta@jhu.edu>,
Sudipto Banerjee <sudipto@ucla.edu>

References

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111:800-812.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, https://doi.org/10.1080/10618600.2018.1537924.

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

Shirota, S., A.O. Finley, B.D. Cook, and S. Banerjee (2019) Conjugate Nearest Neighbor Gaussian Process models for efficient statistical interpolation of large spatial data. https://arxiv.org/abs/1907.10109.

Examples

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

##Make some data
set.seed(1)
n <- 2000
coords <- cbind(runif(n,0,1), runif(n,0,1))

x <- cbind(1, rnorm(n))

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

sigma.sq <- 5
tau.sq <- 1
phi <- 3/0.5

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

ho <- sample(1:n, 1000)
```

```
y.ho <- y[ho]
x.ho <- x[ho,,drop=FALSE]
w.ho <- w[ho]
coords.ho <- coords[ho,]

y <- y[-ho]
x <- x[-ho,,drop=FALSE]
w <- w[-ho,,drop=FALSE]
coords <- coords[-ho,]

##Fit a Conjugate NNGP model and predict for the holdout
sigma.sq.IG <- c(2, sigma.sq)

cov.model <- "exponential"

g <- 10
theta.alpha <- cbind(seq(phi,30,length.out=g), seq(tau.sq/sigma.sq,5,length.out=g))

colnames(theta.alpha) <- c("phi", "alpha")

m.c <- spConjNNGP(y~x-1, coords=coords, n.neighbors = 10,
                  X.0 = x.ho, coords.0 = coords.ho,
                  k.fold = 5, score.rule = "crps",
                  n.omp.threads = 1,
              theta.alpha = theta.alpha, sigma.sq.IG = sigma.sq.IG, cov.model = cov.model)

m.c
```

---

spDiag *Model fit diagnostics*

Description

The function spDiag calculates measurements of model fit for objects of class NNGP and PGLogit.

Usage

```
spDiag(object, sub.sample, ...)
```

Arguments

object	an object of class NNGP or PGLogit.
sub.sample	an optional list that specifies the samples to included in the computations. Valid tags are start, end, and thin. Given the value associated with the tags, the sample subset is selected using seq(as.integer(start),as.integer(end),by=as.integer(thin)). The default values are start=floor(0.5*n.samples), end=n.samples and thin=1. If sub.samples is not specified, then it is taken from object, or, if

not aviable in `object` the default values of `start`, `end`, and `thin` are used. Note, if the `object` is a `NNGP response` model and `n` is large, then computing the replicated data needed for `GPD` and `GRS` can take a long time.

`...` currently no additional arguments.

Value

A list with the following tags:

`DIC` a data frame holding Deviance information criterion (DIC) and associated values. Values in `DIC` include `DIC` the criterion (lower is better), `D` a goodness of fit, and `pD` the effective number of parameters, see Spiegelhalter et al. (2002) for details.

`GPD` a data frame holding D=G+P and associated values. Values in `GPD` include `G` a goodness of fit, `P` a penalty term, and `D` the criterion (lower is better), see Gelfand and Ghosh (1998) for details.

`GRS` a scoring rule, see Equation 27 in Gneiting and Raftery (2007) for details.

`WAIC` a data frame hold Watanabe-Akaike information criteria (WAIC) and associated values. Values in `WAIC` include `LPPD` log pointwise predictive density, `P.1` penalty term defined in unnumbered equation above Equation (11) in Gelman et al. (2014), `P.2` an alternative penalty term defined in Equation (11), and the criteria `WAIC.1` and `WAIC.2` (lower is better) computed using `P.1` and `P.2`, respectively.

`y.rep.samples` if `y.rep.samples` in `object` were not used (or not available), then the newly computed `y.rep.samples` is returned.

`y.fit.samples` if `y.fit.samples` in `object` were not used (or not available), then the newly computed `y.fit.samples` is returned.

`s.indx` the index of samples used for the computations.

Author(s)

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

References

Spiegelhalter, D.J., Best, N.G., Carlin, B.P., van der Linde, A. (2002). Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society, Series B.*, 64:583-639.

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

Gelman, A., Hwang, J., and Vehtari, A. (2014). Understanding predictive information criteria for Bayesian models. *Statistics and Computing*, 24:997-1016.

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

spNNGP *Function for Fitting Univariate Bayesian Spatial Regression Models*

Description

The function spNNGP fits Gaussian and non-Gaussian univariate Bayesian spatial regression models using Nearest Neighbor Gaussian Processes (NNGP).

Usage

```
spNNGP(formula, data = parent.frame(), coords, method = "response",
      family="gaussian", weights, n.neighbors = 15,
      starting, tuning, priors, cov.model = "exponential",
      n.samples, n.omp.threads = 1, search.type = "cb", ord,
      return.neighbor.info = FALSE, neighbor.info,
      fit.rep = FALSE, sub.sample, verbose = TRUE, n.report = 100, ...)
```

Arguments

formula	a symbolic description of the regression model to be fit. See example below.
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which spNNGP is called.
coords	an $n \times 2$ matrix of the observation coordinates in R^2 (e.g., easting and northing), or if data is a data frame then coords can be a vector of length two comprising coordinate column names or indices. There can be no duplicate locations.
method	a quoted keyword that specifies the NNGP sampling algorithm. Supported method keywords are: "response" and "sequential". When n is large, the "response" algorithm should be faster and provide finer control over Metropolis acceptance rate for covariance parameters. In general, unless estimates of spatial random effects are needed, the "response" algorithm should be used.
family	a quoted keyword that specifies the data likelihood. Choices are "gaussian" for continuous outcome and "binomial" for discrete outcome which assumes a logistic link modeled using Polya-Gamma latent variables.
weights	specifies the number of trials for each observation when family="binomial". The default is 1 trial for each observation. Valid arguments are a scalar value that specifies the number of trials used if all observations have the same number of trials, and a vector of length n that specifies the number of trials for each observation used there are differences in the number of trials.
n.neighbors	number of neighbors used in the NNGP.
starting	a list with each tag corresponding to a parameter name. Valid tags are beta, sigma.sq, tau.sq, phi, and nu. nu is only specified if cov.model="matern". The value portion of each tag is the parameter's startingvalue.

`tuning` a list with each tag corresponding to a parameter name. Valid tags are `sigma.sq`, `tau.sq`, `phi`, and `nu`. If `method="sequential"` then only `phi` and `nu` need to be specified. The value portion of each tag defines the variance of the Metropolis sampler Normal proposal distribution.

`priors` a list with each tag corresponding to a parameter name. Valid tags are `sigma.sq.ig`, `tau.sq.ig`, `phi.unif`, and `nu.unif`. Variance parameters, `simga.sq` and `tau.sq`, are assumed to follow an inverse-Gamma distribution, whereas the spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma are passed as a vector of length two, with the first and second elements corresponding to the *shape* and *scale*, respectively. The hyperparameters of the Uniform are also passed as a vector of length two with the first and second elements corresponding to the lower and upper support, respectively.

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`. See below for details.

`n.samples` the number of posterior samples to collect.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.

`fit.rep` if `TRUE`, regression fitted and replicate data will be returned. The argument `sub.sample` controls which MCMC samples are used to generate the fitted and replicated data.

`sub.sample` an optional list that specifies the samples used for `fit.rep`. Valid tags are `start`, `end`, and `thin`. Given the value associated with the tags, the sample subset is selected using `seq(as.integer(start),as.integer(end),by=as.integer(thin))`. The default values are `start=floor(0.5*n.samples)`, `end=n.samples` and `thin=1`.

`verbose` if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.

`search.type` a quoted keyword that specifies type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering (see `ord` argument) then `"cb"` and `"brute"` should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then `"cb"` and `"brute"` might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`ord` an index vector of length n used for the nearest neighbor search. Internally, this vector is used to order `coords`, i.e., `coords[ord,]`, and associated data. Nearest neighbor candidates for the i-th row in the ordered `coords` are rows `1:(i-1)`, with the `n.neighbors` nearest neighbors being those with the minimum euclidean distance to the location defined by ordered `coords[i,]`. The default

used when ord is not specified is x-axis ordering, i.e., order(coords[,1]). This argument should typically be left blank. This argument will be ignored if the neighbor.info argument is used.

return.neighbor.info
: if TRUE, a list called neighbor.info containing several data structures used for fitting the NNGP model is returned. If there is no change in input data or n.neighbors, this list can be passed to subsequent spNNGP calls via the neighbor.info argument to avoid the neighbor search, which can be time consuming if n is large. In addition to the several cryptic data structures in neighbor.info there is a list called n.indx that is of length n. The i-th element in n.indx corresponds to the i-th row in coords[ord,] and holds the vector of that location's nearest neighbor indices. This list can be useful for plotting the neighbor graph if desired.

neighbor.info
: see the return.neighbor.info argument description above.

n.report
: the interval to report Metropolis sampler acceptance and MCMC progress.

...
: currently no additional arguments.

Details

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

The *no nugget* model is specified by setting tau.sq to zero in the starting and tuning lists.

Value

An object of class NNGP with additional class designations for method and family. The return object is a list comprising:

p.beta.samples
: a coda object of posterior samples for the regression coefficients.

p.theta.samples
: a coda object of posterior samples for covariance parameters.

p.w.samples
: is a matrix of posterior samples for the spatial random effects, where rows correspond to locations in coords and columns hold the n.samples posterior samples. This is only returned if method="sequential".

y.hat.samples
: if fit.rep=TRUE, regression fitted values from posterior samples specified using sub.sample. See additional details below.

y.hat.quants
: if fit.rep=TRUE, 0.5, 0.025, and 0.975 quantiles of the y.hat.samples.

y.rep.samples
: if fit.rep=TRUE, replicated outcome from posterior samples specified using sub.sample. See additional details below.

y.rep.quants
: if fit.rep=TRUE, 0.5, 0.025, and 0.975 quantiles of the y.rep.samples.

s.indx
: if fit.rep=TRUE, the subset index specified with sub.sample.

neighbor.info
: returned if return.neighbor.info=TRUE see the return.neighbor.info argument description above.

run.time
: execution time for parameter estimation reported using proc.time(). This time does not include nearest neighbor search time for building the neighbor set.

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

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Abhirup Datta <abhidatta@jhu.edu>,
Sudipto Banerjee <sudipto@ucla.edu>

References

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111:800-812.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, https://doi.org/10.1080/10618600.2018.1537924.

Examples

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

##Make some data
set.seed(1)
n <- 100
coords <- cbind(runif(n,0,1), runif(n,0,1))

x <- cbind(1, rnorm(n))

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

sigma.sq <- 5
tau.sq <- 1
phi <- 3/0.5

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

##Fit a Response and Sequential NNGP model
n.samples <- 500

starting <- list("phi"=phi, "sigma.sq"=5, "tau.sq"=1)

tuning <- list("phi"=0.5, "sigma.sq"=0.5, "tau.sq"=0.5)

priors <- list("phi.Unif"=c(3/1, 3/0.01), "sigma.sq.IG"=c(2, 5), "tau.sq.IG"=c(2, 1))
```

```
cov.model <- "exponential"

m.s <- spNNGP(y~x-1, coords=coords, starting=starting, method="sequential", n.neighbors=10,
              tuning=tuning, priors=priors, cov.model=cov.model,
              n.samples=n.samples, n.omp.threads=1)

summary(m.s)
plot(apply(m.s$p.w.samples, 1, median), w)

m.r <- spNNGP(y~x-1, coords=coords, starting=starting, method="response", n.neighbors=10,
              tuning=tuning, priors=priors, cov.model=cov.model,
              n.samples=n.samples, n.omp.threads=1)

summary(m.r)

##Fit with some user defined neighbor ordering

##ord <- order(coords[,2]) ##y-axis
ord <- order(coords[,1]+coords[,2]) ##x+y-axis
##ord <- sample(1:n, n) ##random

m.r.xy <- spNNGP(y~x-1, coords=coords, starting=starting, method="response", n.neighbors=10,
              tuning=tuning, priors=priors, cov.model=cov.model,
              ord=ord, return.neighbor.info=TRUE,
              n.samples=n.samples, n.omp.threads=1)

summary(m.r.xy)

## Not run:
##Visualize the neighbor sets and ordering constraint
n.indx <- m.r.xy$neighbor.info$n.indx
ord <- m.r.xy$neighbor.info$ord

##This is how the data are ordered internally for model fitting
coords.ord <- coords[ord,]

for(i in 1:n){

    plot(coords.ord, cex=1, xlab="Easting", ylab="Northing")
    points(coords.ord[i,,drop=FALSE], col="blue", pch=19, cex=1)
    points(coords.ord[n.indx[[i]],,drop=FALSE], col="red", pch=19, cex=1)

    readline(prompt = "Pause. Press <Enter> to continue...")
}

## End(Not run)
```

---

spPredict *Function for prediction at new locations using* NNGP *models.*

Description

The function spPredict collects posterior predictive samples for a set of new locations given an object of class NNGP.

Usage

```
spPredict(sp.obj, X.0, coords.0, sub.sample, n.omp.threads = 1,
          verbose=TRUE, n.report=100, ...)
```

Arguments

sp.obj	an object of class NNGP.
X.0	the design matrix for prediction locations. An intercept should be provided in the first column if one is specified in sp.obj model.
coords.0	the spatial coordinates corresponding to X.0.
sub.sample	an optional list that specifies the samples to included in the composition sampling a non-Conjugate model. Valid tags are start, end, and thin. Given the value associated with the tags, the sample subset is selected using seq(as.integer(start),as.integer The default values are start=floor(0.5*n.samples), end=n.samples and thin=1.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
n.report	the interval to report sampling progress.
...	currently no additional arguments.

Value

An object of class spPredict which is a list comprising:

p.y.0	a matrix that holds the response variable posterior predictive samples where rows are locations corresponding to coords.0 and columns are samples.
p.w.0	a matrix that holds the random effect posterior predictive samples where rows are locations corresponding to coords.0 and columns are samples. This is only returned if the input class has method = "sequential".
run.time	execution time reported using proc.time().

Author(s)

Andrew O. Finley <finleya@msu.edu>,
Abhirup Datta <abhidatta@jhu.edu>,
Sudipto Banerjee <sudipto@ucla.edu>

References

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111:800-812.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Jurnal of Computational and Graphical Statistics*, https://doi.org/10.1080/10618600.2018.1537924.

Examples

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

##Make some data
set.seed(1)
n <- 100
coords <- cbind(runif(n,0,1), runif(n,0,1))

x <- cbind(1, rnorm(n))

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

sigma.sq <- 5
tau.sq <- 1
phi <- 3/0.5

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

ho <- sample(1:n, 50)

y.ho <- y[ho]
x.ho <- x[ho,,drop=FALSE]
w.ho <- w[ho]
coords.ho <- coords[ho,]

y <- y[-ho]
x <- x[-ho,,drop=FALSE]
w <- w[-ho,,drop=FALSE]
coords <- coords[-ho,]

##Fit a response, sequential, and conjugate NNGP model
n.samples <- 500
```

```
starting <- list("phi"=phi, "sigma.sq"=5, "tau.sq"=1)

tuning <- list("phi"=0.5, "sigma.sq"=0.5, "tau.sq"=0.5)

priors <- list("phi.Unif"=c(3/1, 3/0.01), "sigma.sq.IG"=c(2, 5), "tau.sq.IG"=c(2, 1))

cov.model <- "exponential"

n.report <- 500

##Sequential
m.s <- spNNGP(y~x-1, coords=coords, starting=starting, method="sequential", n.neighbors=10,
              tuning=tuning, priors=priors, cov.model=cov.model,
              n.samples=n.samples, n.omp.threads=1, n.report=n.report)

p.s <- spPredict(m.s, X.0 = x.ho, coords.0 = coords.ho, n.omp.threads=1)

plot(apply(p.s$p.w.0, 1, mean), w.ho)
plot(apply(p.s$p.y.0, 1, mean), y.ho)

##Response
m.r <- spNNGP(y~x-1, coords=coords, starting=starting, method="response", n.neighbors=10,
              tuning=tuning, priors=priors, cov.model=cov.model,
              n.samples=n.samples, n.omp.threads=1, n.report=n.report)

p.r <- spPredict(m.r, X.0 = x.ho, coords.0 = coords.ho, n.omp.threads=1)

points(apply(p.r$p.y.0, 1, mean), y.ho, pch=19, col="blue")

##Conjugate
theta.alpha <- c(phi, tau.sq/sigma.sq)
names(theta.alpha) <- c("phi", "alpha")

m.c <- spConjNNGP(y~x-1, coords=coords, n.neighbors=10,
                  theta.alpha=theta.alpha, sigma.sq.IG=c(2, sigma.sq),
                  cov.model=cov.model, n.omp.threads=1)

p.c <- spPredict(m.c, X.0 = x.ho, coords.0 = coords.ho, n.omp.threads=1)

points(p.c$y.0.hat, y.ho, pch=19, col="orange")
```

---

summary.NNGP *Methods for NNGP and Derived Objects*

Description

Methods for extracting information from fitted NNGP model of class NNGP and subsequent prediction objects of class spPredict.

Usage

```
## S3 method for class 'NNGP'
summary(object, sub.sample, quantiles = c(0.025, 0.25,
0.5, 0.75, 0.975), digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'NNGP'
print(x, ...)
## S3 method for class 'spPredict'
print(x, ...)
```

Arguments

object, x	object of class NNGP or spPredict.
sub.sample	an optional list that specifies the samples to included in the summary or composition sampling. Valid tags are start, end, and thin. Given the value associated with the tags, the sample subset is selected using seq(as.integer(start),as.integer(end),by=as.i The default values are start=floor(0.5*n.samples), end=n.samples and thin=1.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report in summary.
...	currently not used.

Details

A set of standard extractor functions for fitted model objects of class NNGP and prediction object of class spPredict, including methods to the generic functions print and summary.

---

summary.PGLogit *Methods for PGLogit Object*

Description

Methods for extracting information from fitted PGLogit model.

Usage

```
## S3 method for class 'PGLogit'
summary(object, sub.sample, quantiles
=c(0.025, 0.25, 0.5, 0.75, 0.975), digits = max(3L, getOption("digits")
- 3L), ...)
## S3 method for class 'PGLogit'
print(x, ...)
```

Arguments

object, x	object of class PGLogit.
sub.sample	an optional list that specifies the samples to included in the summary or composition sampling. Valid tags are start, end, and thin. Given the value associated with the tags, the sample subset is selected using seq(as.integer(start),as.integer(end),by=as.i The default values are start=floor(0.5*n.samples), end=n.samples and thin=1.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
...	currently not used.

Details

A set of standard extractor functions for fitted model objects of class PGLogit, including methods to the generic functions print and summary.

Index