Package ‘glmmfields’

May 18, 2019

Type Package

Title Generalized Linear Mixed Models with Robust Random Fields for Spatiotemporal Modeling

Version 0.1.3

Description Implements Bayesian spatial and spatiotemporal models that optionally allow for extreme spatial deviations through time. ’glmmfields’ uses a predictive process approach with random fields implemented through a multivariate-t distribution instead of the usual multivariate normal. Sampling is conducted with ’Stan’. References: Anderson and Ward (2019) <doi:10.1002/ecy.2403>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.4.0), Rcpp (>= 0.12.18), methods

Imports rstan (>= 2.18.2), ggplot2 (>= 2.2.0), rstantools (>= 1.5.1), cluster, dplyr (>= 0.8.0), reshape2, mvtnorm, broom, loo (>= 2.0.0), assertthat, nlme, forcats

LinkingTo StanHeaders (>= 2.18.0), rstan (>= 2.18.2), BH (>= 1.66.0), Rcpp (>= 0.12.8), RcppEigen (>= 0.3.3.3.0)

Suggests testthat, parallel, bayesplot, knitr, rmarkdown, viridis, coda

URL https://github.com/seananderson/glmmfields

BugReports https://github.com/seananderson/glmmfields/issues

RoxygenNote 6.1.1

VignetteBuilder knitr

SystemRequirements GNU make

NeedsCompilation yes

Author Sean C. Anderson [aut, cre],
    Eric J. Ward [aut],
    Trustees of Columbia University [cph]

Maintainer Sean C. Anderson <sean@seananderson.ca>
The `glmmfields` package.

Description

Implements Bayesian spatial and spatiotemporal models that optionally allow for extreme spatial deviations through time. `glmmfields` uses a predictive process approach with random fields implemented through a multivariate-t distribution instead of the usual multivariate normal. Sampling is conducted with 'Stan'.

References


Format data for fitting a glmmfields model

Description

Format data for fitting a glmmfields model
**Usage**

```r
format_data(data, y, X, time, lon = "lon", lat = "lat",
station = NULL, nknots = 25L, covariance = c("squared-exponential",
"exponential", "matern"), fixed_intercept = FALSE, cluster = c("pam",
"kmeans"))
```

**Arguments**

- **data**: A data frame to be formatted
- **y**: A numeric vector of the response
- **X**: A matrix of the predictors
- **time**: A character object giving the name of the time column
- **lon**: A character object giving the name of the longitude column
- **lat**: A character object giving the name of the latitude column
- **station**: A numeric vector giving the integer ID of the station
- **nknots**: The number of knots
- **covariance**: The type of covariance function
- **fixed_intercept**: Should the intercept be fixed?
- **cluster**: The type of clustering algorithm used to determine the not locations. "pam" = pam. kmeans is faster for large datasets.

---

**glmmfields**

*Fit a spatiotemporal random fields GLMM*

**Description**

Fit a spatiotemporal random fields model that optionally uses the MVT distribution instead of a MVN distribution to allow for spatial extremes through time. It is also possible to fit a spatial random fields model without a time component.

**Usage**

```r
glmmfields(formula, data, lon, lat, time = NULL, nknots = 15L,
prior_gp_theta = half_t(3, 0, 5), prior_gp_sigma = half_t(3, 0, 5),
prior_sigma = half_t(3, 0, 5), prior_rw_sigma = half_t(3, 0, 5),
prior_intercept = student_t(3, 0, 10), prior_beta = student_t(3, 0, 3),
prior_phi = student_t(1000, 0, 0.5), fixed_df_value = 1000,
fixed_phi_value = 0, estimate_df = FALSE, estimate_ar = FALSE,
family = gaussian(link = "identity"),
covariance = c("squared-exponential", "exponential", "matern"),
matern_kappa = 0.5, algorithm = c("sampling", "meanfield"),
year_re = FALSE, nb_lower_truncation = 0,
control = list(adapt_delta = 0.9), save_log_lik = FALSE,
df_lower_bound = 2, cluster = c("pam", "kmeans"), ...)
```
Arguments

formula
The model formula.
data
A data frame.
lon
A character object giving the name of the longitude column.
lat
A character object giving the name of the latitude column.
time
A character object giving the name of the time column. Leave as NULL to fit a spatial GLMM without a time element.
knots
The number of knots to use in the predictive process model. Smaller values will be faster but may not adequately represent the shape of the spatial pattern.
prior_gp_theta
The prior on the Gaussian Process scale parameter. Must be declared with half_t(). Here, and throughout, priors that are normal or half-normal can be implemented by setting the first parameter in the half-t or student-t distribution to a large value. E.g. something greater than 100.
prior_gp_sigma
The prior on the Gaussian Process eta parameter. Must be declared with half_t().
prior_sigma
The prior on the observation process scale parameter. Must be declared with half_t(). This acts as a substitute for the scale parameter in whatever observation distribution is being used. E.g. the CV for the Gamma or the dispersion parameter for the negative binomial.
prior_rw_sigma
The prior on the standard deviation parameter of the random walk process (if specified). Must be declared with half_t().
prior_intercept
The prior on the intercept parameter. Must be declared with student_t().
prior_beta
The prior on the slope parameters (if any). Must be declared with student_t().
prior_phi
The prior on the AR parameter. Must be declared with student_t().
fixed_df_value
The fixed value for the student-t degrees of freedom parameter if the degrees of freedom parameter is fixed in the MVT. If the degrees of freedom parameter is estimated then this argument is ignored. Must be 1 or greater. Very large values (e.g. the default value) approximate the normal distribution. If the value is >=1000 then a true MVN distribution will be fit.
fixed_phi_value
The fixed value for temporal autoregressive parameter, between random fields at time(t) and time(t-1). If the phi parameter is estimated then this argument is ignored.
estimate_df
Logical: should the degrees of freedom parameter be estimated?
estimate_ar
Logical: should the AR (autoregressive) parameter be estimated? Here, this refers to an autoregressive process in the evolution of the spatial field through time.
family
Family object describing the observation model. Note that only one link is implemented for each distribution. Gamma, negative binomial (specified via nbinom2() as nbinom2(link = "log"), and Poisson must have a log link. Binomial must have a logit link. Also implemented is the lognormal (specified via lognormal()) as lognormal(link = "log"). Besides the negative binomial and lognormal, other families are specified as shown in family.
covariance The covariance function of the Gaussian Process. One of "squared-exponential", "exponential", or "matern".

matern_kappa Optional parameter for the Matern covariance function. Optional values are 1.5 or 2.5. Values of 0.5 are equivalent to exponential.

algorithm Character object describing whether the model should be fit with full NUTS MCMC or via the variational inference mean-field approach. See rstan::vb(). Note that the variational inference approach should not be trusted for final inference and is much more likely to give incorrect inference than MCMC.

year_re Logical: estimate a random walk for the time variable? If TRUE, then no fixed effects (B coefficients) will be estimated. In this case, prior_intercept will be used as the prior for the initial value in time.

nb_lower_truncation For NB2 only: lower truncation value. E.g. 0 for no truncation, 1 for 1 and all values above. Note that estimation is likely to be considerably slower with lower truncation because the sampling is not vectorized. Also note that the log likelihood values returned for estimating quantities like LOOIC will not be correct if lower truncation is implemented.

control List to pass to rstan::sampling(). For example, increase adapt_delta if there are warnings about divergent transitions: control = list(adapt_delta = 0.99). By default, glmmfields sets adapt_delta = 0.9.

save_log_lik Logical: should the log likelihood for each data point be saved so that information criteria such as LOOIC or WAIC can be calculated? Defaults to FALSE so that the size of model objects is smaller.

df_lower_bound The lower bound on the degrees of freedom parameter. Values that are too low, e.g. below 2 or 3, it might affect chain convergence. Defaults to 2.

cluster The type of clustering algorithm used to determine the knot locations. "pam" = cluster::pam(). The "kmeans" algorithm will be faster on larger datasets.

... Any other arguments to pass to rstan::sampling().

Details

Note that there is no guarantee that the default priors are reasonable for your data. Also, there is no guarantee the default priors will remain the same in future versions. Therefore it is important that you specify any priors that are used in your model, even if they replicate the defaults in the package. It is particularly important that you consider that prior on gp_theta since it depends on the distance between your location points. You may need to scale your coordinate units so they are on a ballpark range of 1-10 by, say, dividing the coordinates (say in UTMs) by several order of magnitude.

Examples

# Spatiotemporal example:
set.seed(1)
s <- sim_glmmfields(n_draws = 12, n_knots = 12, gp_theta = 1.5,
gp_sigma = 0.2, sd_obs = 0.2)
print(s$plot)
# options(mc.cores = parallel::detectCores()) # for parallel processing
lognormal

Description
Lognormal family

Usage
lognormal(link = "log")

Arguments
link The link (must be log)

Examples
lognormal()
Description

Extract the LOOIC (leave-one-out information criterion) using `loo::loo()`.

Usage

```r
## S3 method for class 'glmmfields'
loo(x, ...)
```

Arguments

- `x`: Output from `glmmfields()`. Must be fit with `save_log_lik = TRUE`, which is not the default.
- `...`: Arguments for `loo::relative_eff()` and `loo::loo.array()`.

Examples

```r
set.seed(1)
s <- sim_glmmfields(n_draws = 12, n_knots = 12, gp_theta = 1.5,
gp_sigma = 0.2, sd_obs = 0.2)
# options(mc.cores = parallel::detectCores()) # for parallel processing

# save_log_lik defaults to FALSE to save space but is needed for loo():
m <- glmmfields(y ~ 0, time = "time",
lat = "lat", lon = "lon", data = s$dat,
nknots = 12, iter = 1000, chains = 4, seed = 1,
save_log_lik = TRUE)
loo(m)
```

Description

This is the NB2 parameterization where the variance scales quadratically with the mean.

Usage

```r
nbinom2(link = "log")
```
Arguments

link The link (must be log)

Examples

nbinom2()

---

plot.glmmfields Plot predictions from an glmmfields model

Description

Plot predictions from an glmmfields model

Usage

## S3 method for class 'glmmfields'
plot(x, type = c("prediction", "spatial-residual", "residual-vs-fitted"), link = TRUE, ...)

Arguments

x An object returned by glmmfields
type Type of plot
link Logical: should the plots be made on the link scale or on the natural scale?
... Other arguments passed to predict.glmmfields

Examples

# Spatiotemporal example:
set.seed(1)
s <- sim.glmmfields(n_draws = 12, n_knots = 12, gp_theta = 1.5, gp_sigma = 0.2, sd_obs = 0.1)
# options(mc.cores = parallel::detectCores()) # for parallel processing
m <- glmmfields(y ~ 0, time = "time", lat = "lat", lon = "lon", data = s$dat, nknots = 12, iter = 600, chains = 1)
x <- plot(m, type = "prediction")
x + ggplot2::scale_color_gradient2()
plot(m, type = "spatial-residual")
plot(m, type = "residual-vs-fitted")
predict

Predict from a glmmfields model

Description

These functions extract posterior draws or credible intervals. The helper functions are named to match those in the rstanarm package and call the function predict() with appropriate argument values.

Usage

```r
## S3 method for class 'glmmfields'
predictive_interval(object, ...)

## S3 method for class 'glmmfields'
posterior_linpred(object, ...)

## S3 method for class 'glmmfields'
posterior_predict(object, ...)

## S3 method for class 'glmmfields'
predict(object, newdata = NULL,
estimate_method = c("median", "mean"), conf-level = 0.95,
interval = c("confidence", "prediction"), type = c("link",
"response"), return_mcmc = FALSE, iter = "all", ...)
```

Arguments

- **object**: An object returned by glmmfields().
- **...**: Ignored currently
- **newdata**: Optionally, a data frame to predict on
- **estimate_method**: Method for computing point estimate ("mean" or "median")
- **conf_level**: Probability level for the credible intervals.
- **interval**: Type of interval calculation. Same as for stats::predict.lm().
- **type**: Whether the predictions are returned on "link" scale or "response" scale (Same as for stats::predict.glm().
- **return_mcmc**: Logical. Should the full MCMC draws be returned for the predictions?
- **iter**: Number of MCMC iterations to draw. Defaults to all.

Examples

```r
library(ggplot2)
```
# simulate:
s <- sim_glmmfields(
  n_draws = 12, n_knots = 12, gp_theta = 2.5,
  gp_sigma = 0.2, sd_obs = 0.1
)

# fit:
# options(mc.cores = parallel::detectCores()) # for parallel processing
m <- glmmfields(y ~ 0,
  data = s$dat, time = "time",
  lat = "lat", lon = "lon",
  nknots = 12, iter = 800, chains = 1
)

# Predictions:
# Link scale credible intervals:
p <- predict(m, type = "link", interval = "confidence")
head(p)

# Prediction intervals on new observations (include observation error):
p <- predictive_interval(m)
head(p)

# Posterior prediction draws:
p <- posterior_predict(m, iter = 100)
dim(p) # rows are iterations and columns are data elements

# Draws from the linear predictor (not in link space):
p <- posterior_linpred(m, iter = 100)
dim(p) # rows are iterations and columns are data elements

# Use the `tidy` method to extract parameter estimates as a data frame:
head(tidy(m, conf.int = TRUE, conf.method = "HPDinterval"))

# Make predictions on a fine-scale spatial grid:
pred_grid <- expand.grid(
  lat = seq(min(s$dat$lat), max(s$dat$lat), length.out = 25),
  lon = seq(min(s$dat$lon), max(s$dat$lon), length.out = 25),
  time = unique(s$dat$time)
)
pred_grid$prediction <- predict(m,
  newdata = pred_grid, type = "response", iter = 100,
  estimate_method = "median"
)$estimate

ggplot(pred_grid, aes(lon, lat, fill = prediction)) +
  facet_wrap(~time) +
  geom_raster() +
  scale_fill_gradient2()
Simulate a random field with a MVT distribution

Description

Simulate a random field with a MVT distribution

Usage

sim_glmmfields(n_knots = 15, n_draws = 10, gp_theta = 0.5,
gp_sigma = 0.2, mvt = TRUE, df = 1e+06, seed = NULL,
n_data_points = 100, sd_obs = 0.1,
covariance = c("squared-exponential", "exponential", "matern"),
matern_kappa = 0.5, obs_error = c("normal", "gamma", "poisson",
"nb2", "binomial", "lognormal"), B = c(0), phi = 0, X = rep(1,
n_draws * n_data_points), g = data.frame(lon = runif(n_data_points, 0,
10), lat = runif(n_data_points, 0, 10)))

Arguments

- n_knots: The number of knots
- n_draws: The number of draws (for example, the number of years)
- gp_theta: The Gaussian Process scale parameter
- gp_sigma: The Gaussian Process variance parameter
- mvt: Logical: MVT? (vs. MVN)
- df: The degrees of freedom parameter for the MVT distribution
- seed: The random seed value
- n_data_points: The number of data points per draw
- sd_obs: The observation process scale parameter
- covariance: The covariance function of the Gaussian process ("squared-exponential", "exponential", "matern")
- matern_kappa: The optional matern parameter. Can be 1.5 or 2.5. Values of 0.5 equivalent to exponential model.
- obs_error: The observation error distribution
- B: A vector of parameters. The first element is the intercept
- phi: The auto regressive parameter on the mean of the random field knots
- X: The model matrix
- g: Grid of points

Examples

s <- sim_glmmfields(n_draws = 12, n_knots = 12, gp_theta = 1.5,
gp_sigma = 0.2, sd_obs = 0.2)
names(s)
**stan_pars**

*Return a vector of parameters*

**Description**

Return a vector of parameters

**Usage**

```r
stan_pars(obs_error, estimate_df = TRUE, est_temporalRE = FALSE,
estimate_ar = FALSE, fixed_intercept = FALSE, save_log_lik = FALSE)
```

**Arguments**

- **obs_error**
  - The observation error distribution

- **estimate_df**
  - Logical indicating whether the degrees of freedom parameter should be estimated

- **est_temporalRE**
  - Logical: estimate a random walk for the time variable?

- **estimate_ar**
  - Logical indicating whether the ar parameter should be estimated

- **fixed_intercept**
  - Should the intercept be fixed?

- **save_log_lik**
  - Logical: should the log likelihood for each data point be saved so that information criteria such as LOOIC or WAIC can be calculated? Defaults to FALSE so that the size of model objects is smaller.

**student_t**

*Student-t and half-t priors*

**Description**

Student-t and half-t priors. Note that this can be used to represent an effectively normal distribution prior by setting the first argument (the degrees of freedom parameter) to a large value (roughly 50 or above).

**Usage**

```r
student_t(df = 3, location = 0, scale = 1)

half_t(df = 3, location = 0, scale = 1)
```

**Arguments**

- **df**
  - Degrees of freedom parameter

- **location**
  - Location parameter

- **scale**
  - Scale parameter
tidy

Examples

```r
student_t(3, 0, 1)
half_t(3, 0, 1)
```

Description

Tidy model output

Usage

```r
tidy(x, ...)
```

```r
## S3 method for class 'glmmfields'
tidy(x, ...)
```

Arguments

- `x` : Output from `glmmfields`
- `...` : Other arguments
Index

cluster::pam(), 5
family, 4
format_data, 2
glmmfields, 3, 8
glmmfields(), 7, 9, 13
glmmfields~package, 2
half_t(student_t), 12
half_t(), 4
lognormal, 6
lognormal(), 4
loo(loo.glmmfields), 7
loo.glmmfields, 7
loo::loo(), 7
loo::loo.array(), 7
loo::relative_eff(), 7
nbinom2, 7
nbinom2(), 4
pam, 3
plot.glmmfields, 8
posterior_linpred(predict), 9
posterior_predict(predict), 9
predict, 9
predict.glmmfields, 8
predictive_interval(predict), 9
rstan::sampling(), 5
rstan::vb(), 5
sim_glmmfields, 11
stan_pars, 12
stats::predict.glm(), 9
stats::predict.lm(), 9
student_t, 12
student_t(), 4
tidy, 13