Package: gopher (via r-universe)

June 1, 2026

Title tidymodels-Compatible Gaussian Process Model Zoo

Version 0.5.1

Description A gateway to various Gaussian Process (GP) packages available on CRAN, including gstat, fields, GPvecchia, spNNGP, and PrestoGP. Provides a unified tidymodels-native interface through parsnip for model parameter adjustment and hyperparameter tuning. Supports spatial-only and spatiotemporal Kriging, sf inputs, prediction capabilities, and covariates for residual Gaussian Process (Universal Kriging) models. Parameters are automatically matched between the unified Kriging-centered argument names in gstat and the formulaic arguments used in other packages for the same construct.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports cli, dplyr, parsnip (>= 1.0.0), rlang (>= 1.0.0), sf, stats, tibble

Suggests dials (>= 1.0.0), fields, GPvecchia, gstat, knitr, rmarkdown, sdmTMB, sp, spacetime, spNNGP, PrestoGP, testthat (>= 3.0.0), workflows, withr

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/sigmafelix/gopher

BugReports https://github.com/sigmafelix/gopher/issues

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://sigmafelix.r-universe.dev

Date/Publication 2026-05-02 14:55:04 UTC

RemoteUrl https://github.com/sigmafelix/gopher

RemoteRef HEAD

RemoteSha 2c5ab32f64c9cc73e07a346b0b7952e5a0857e74

Contents

`covariance_function` *Covariance function type (qualitative tuning parameter)*

Description

Covariance function type (qualitative tuning parameter)

Usage

```
covariance_function(
  values = c("exponential", "spherical", "gaussian", "matern")
)
```

Arguments

`values`	Character vector of allowable covariance function names. Defaults to the four most common choices.

Value

A `dials` qualitative parameter object.

Examples

```
covariance_function()
```

`fields_gp_fit` *Fit a Gaussian Process model using the fields engine*

Description

This function is called internally by parsnip. End users should call `parsnip::fit.model_spec()` on a `gaussian_process_spatial` specification with `set_engine("fields")`.

Usage

```
fields_gp_fit(
  formula,
  data,
  covariance_function = "exponential",
  range = NULL,
  nugget = NULL,
  sill = NULL,
  coord_cols = NULL,
  use_mKrig = FALSE,
  ...
)
```

Arguments

`formula`	A two-sided formula. Use `y ~ 1` for ordinary kriging; covariates (e.g. `y ~ x1 + x2`) are passed to `fields::Krig()` as the `Z` matrix (fixed effects / trend surface).
`data`	An `sf` object or a `data.frame` with coordinate columns.
`covariance_function`	Canonical covariance name. Defaults to `"exponential"`. Note: `"spherical"` and `"gaussian"` are approximated via `fields::Exp.cov`. You can also pass a covariance function directly (e.g. `fields::Matern`), including through a one-sided formula used by `parsnip` (e.g. `~fields::Matern`).
`range`	Range (aRange) parameter for the covariance. `NULL` = estimated by `fields::Krig()`.
`nugget`	Nugget variance. Used to compute `lambda = nugget / sigma2`.
`sill`	Partial sill (sigma2). `NULL` = estimated by `Krig`.
`coord_cols`	Character(2) coordinate column names (non-sf path).
`use_mKrig`	Logical. Use `fields::mKrig()` instead of `fields::Krig()` for large datasets. Default `FALSE`.
`...`	Additional arguments forwarded to `fields::Krig()` / `fields::mKrig()`.

Value

A list of class "gopher_fields_fit".

Examples

```
if (requireNamespace("spacetime", quietly = TRUE) &&
    requireNamespace("fields", quietly = TRUE)) {
  data("air", package = "spacetime")

  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  fit <- fields_gp_fit(pm10 ~ coords.x1 + coords.x2, data = air_sf)
  fit
}
```

---

fields_gp_predict *Predict from a fields-fitted Gaussian Process model*

Description

Predict from a fields-fitted Gaussian Process model

Usage

```
fields_gp_predict(
  object,
  new_data,
  type = "numeric",
  level = 0.95,
  coord_cols = NULL,
  ...
)
```

Arguments

object	A "gopher_fields_fit" object returned by fields_gp_fit().
new_data	An sf object or data.frame with coordinates.
type	"numeric" (default) or "pred_int".
level	Confidence level for prediction intervals (default 0.95).
coord_cols	Character(2) coord column names (non-sf path).
...	Forwarded to predict.Krig() / predictSE.Krig().

Value

A tibble::tibble() with prediction columns.

Examples

```
if (requireNamespace("spacetime", quietly = TRUE) &&
    requireNamespace("fields", quietly = TRUE)) {
  data("air", package = "spacetime")

  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  n_train <- floor(0.8 * nrow(air_sf))
  train_sf <- air_sf[seq_len(n_train), ]
  test_sf <- air_sf[seq.int(n_train + 1L, nrow(air_sf)), ]

  fit <- fields_gp_fit(pm10 ~ coords.x1 + coords.x2, data = train_sf)
  fields_gp_predict(fit, new_data = test_sf, type = "pred_int")
}
```

---

gaussian_process_spatial

Gaussian Process Model for Spatial and Spatiotemporal Data

Description

`gaussian_process_spatial()` defines a Gaussian Process model for spatial and spatiotemporal data. This model supports multiple backends ("engines"), including **gstat**, **fields**, **GPvecchia**, **spNNGP**, and **PrestoGP**.

Usage

```
gaussian_process_spatial(
  mode = "regression",
  covariance_function = NULL,
  range = NULL,
  nugget = NULL,
  sill = NULL
)

## S3 method for class 'gaussian_process_spatial'
update(
  object,
  parameters = NULL,
  covariance_function = NULL,
  range = NULL,
  nugget = NULL,
  sill = NULL,
  fresh = FALSE,
  ...
)
```

Arguments

`mode`	A single character string for the prediction outcome mode. The only possible value for this model is `"regression"`.
`covariance_function`	The type of covariance function (variogram model) to use. One of `"exponential"`, `"spherical"`, `"gaussian"`, `"matern"`, or `"stein_matern"`. Defaults to `NULL` (uses engine default, typically `"exponential"`).
`range`	The range (or scale) parameter of the covariance function. Controls the distance at which spatial correlation effectively vanishes. Defaults to `NULL` (estimated from data).
`nugget`	The nugget variance representing micro-scale variation and measurement error. Defaults to `NULL` (estimated from data).
`sill`	The partial sill, i.e., the spatially structured variance component. Defaults to `NULL` (estimated from data).

Details

What does this model do?:

Gaussian Process (GP) models — commonly known as Kriging in geostatistics — predict values at unobserved locations by leveraging spatial autocorrelation. The model assumes observations

are a realisation of a GP with a specified covariance structure. The covariance structure is characterised by a variogram model parameterised by `range`, `nugget`, and `sill`.

Engines:

The following engines are available:

- `"gstat"` — Uses the **gstat** package for variogram-based kriging (ordinary, universal, and simple kriging). Supports spatiotemporal kriging with a `time_col` engine argument.
- `"fields"` — Uses the **fields** package (`Krig`/`mKrig`) for spatial kriging. Supports large datasets via `mKrig`.
- `"GPvecchia"` — Uses the **GPvecchia** package for Vecchia-approximated GP inference, suitable for large spatial and spatiotemporal datasets via `time_col`.
- `"spNNGP"` — Uses the **spNNGP** package for Nearest Neighbor Gaussian Process models, scalable for large spatial datasets.
- `"PrestoGP"` — Uses the **PrestoGP** package for scalable penalized spatiotemporal Gaussian process models with built-in missing-value imputation and limit-of-detection handling.
- `"sdmTMB"` — Uses the **sdmTMB** package for spatial and spatiotemporal models via Template Model Builder (TMB) with an INLA-style SPDE mesh. Supports spatiotemporal modelling via `time_col` and `spatiotemporal` engine arguments.

Parameter Mapping:

Parameters are automatically mapped between the unified gopher interface and engine-specific argument names:

gopher	gstat (vgm)	fields (Krig)	GPvecchia	spNNGP	PrestoGP	sdmTMB
`covariance_function`	`model`	`Covariance`	`covFun`	`cov.model`	Matérn-only (mapped)	Matérn via SPD
`range`	`range`	`aRange`	`range`	`phi`	estimated internally	estimated intern
`nugget`	`nugget`	`sigma2`	`nugget`	`tau.sq`	estimated internally	estimated intern
`sill`	`psill`	`sigma2`	`sigma2`	`sigma.sq`	estimated internally	estimated intern

Spatial Inputs:

Input data can be provided as:

- An `sf` object — geometry column is used for coordinates automatically.
- A `data.frame` with columns `x` and `y` (or `lon` and `lat`).

Covariates (Universal / Residual Kriging):

When covariates are included in the formula (e.g., `y ~ x1 + x2`), the model performs **Universal Kriging** — fitting a linear trend model with spatial correlation of residuals. Use `y ~ 1` for **Ordinary Kriging** (no covariates, constant mean).

Spatiotemporal Kriging:

Spatiotemporal Gaussian process modelling is supported through the `"gstat"`, `"GPvecchia"`, `"PrestoGP"`, and `"sdmTMB"` engines by passing `time_col` as an engine argument via `set_engine()`. The column specified must contain date/time values (or numeric time indices). For `"sdmTMB"`, also pass `spatiotemporal` (one of `"iid"`, `"ar1"`, `"rw"`) to enable the spatiotemporal random field.

Value

A gaussian_process_spatial model specification of class c("gaussian_process_spatial", "model_spec").

See Also

parsnip::set_engine(), parsnip::fit.model_spec(), parsnip::predict.model_fit()

Examples

```
if (requireNamespace("spacetime", quietly = TRUE) &&
    requireNamespace("gstat", quietly = TRUE)) {
  data("air", package = "spacetime")

  # Convert legacy ST components from `air` to a single-day `sf` table.
  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  n_train <- floor(0.8 * nrow(air_sf))
  train_sf <- air_sf[seq_len(n_train), ]
  test_sf <- air_sf[seq.int(n_train + 1L, nrow(air_sf)), ]

  gp_spec <- gaussian_process_spatial(
    covariance_function = "exponential"
  ) |>
    parsnip::set_engine("gstat")

  gp_fit <- parsnip::fit(
    gp_spec,
    pm10 ~ coords.x1 + coords.x2,
    data = train_sf
  )

  predict(gp_fit, new_data = test_sf, type = "pred_int")
}
```

gp_nugget *GP nugget variance (quantitative tuning parameter)*

Description

GP nugget variance (quantitative tuning parameter)

Usage

```
gp_nugget(range = c(0, 5), trans = NULL)
```

Arguments

range	A numeric vector of length 2 giving lower and upper bounds.
trans	A scales transformation, or NULL.

Value

A dials quantitative parameter object.

Examples

```
gp_nugget()
```

gp_range *GP range parameter (quantitative tuning parameter)*

Description

GP range parameter (quantitative tuning parameter)

Usage

```
gp_range(range = c(0.001, 1000), trans = NULL)
```

Arguments

range	A numeric vector of length 2 giving the lower and upper bounds of the range parameter search space. Values are on the original scale.
trans	A scales transformation object, or NULL (default).

Value

A dials quantitative parameter object.

Examples

```
gp_range()
```

gp_sill *GP partial sill variance (quantitative tuning parameter)*

Description

GP partial sill variance (quantitative tuning parameter)

Usage

```
gp_sill(range = c(1e-06, 1000), trans = NULL)
```

Arguments

range	A numeric vector of length 2 giving lower and upper bounds.
trans	A scales transformation, or NULL (default).

Value

A dials quantitative parameter object.

Examples

```
gp_sill()
```

GPvecchia_gp_fit *Fit a Gaussian Process model using the GPvecchia engine*

Description

This function is called internally by parsnip. End users should call parsnip::fit.model_spec() on a gaussian_process_spatial specification with set_engine("GPvecchia").

Usage

```
GPvecchia_gp_fit(
  formula,
  data,
  covariance_function = "exponential",
  range = NULL,
  nugget = NULL,
  sill = NULL,
  m = 15L,
  coord_cols = NULL,
  time_col = NULL,
  time_scale = 1,
  ...
)
```

Arguments

formula	A two-sided formula. Covariates are used as fixed-effect trend (Universal Kriging / residual GP).
data	An sf object or data.frame with coordinate columns.
covariance_function	Canonical covariance name. Defaults to "exponential".
range	Range parameter (scale). NULL = estimated via MLE.
nugget	Nugget (noise) variance. NULL = estimated.
sill	Signal variance (sigma^2). NULL = estimated.
m	Number of nearest neighbours for Vecchia approximation. Default 15.
coord_cols	Character(2) coordinate column names (non-sf path).
time_col	Optional character scalar specifying a time column for spatiotemporal modelling.
time_scale	Numeric scalar used to rescale time when time_col is provided. Default 1.
...	Additional arguments forwarded to GPvecchia::vecchia_specify().

Value

A list of class "gopher_GPvecchia_fit".

Examples

```
if (requireNamespace("spacetime", quietly = TRUE) &&
    requireNamespace("GPvecchia", quietly = TRUE)) {
  data("air", package = "spacetime")

  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  fit <- GPvecchia_gp_fit(pm10 ~ coords.x1 + coords.x2, data = air_sf, m = 10)
  fit
}
```

GPvecchia_gp_predict *Predict from a GPvecchia-fitted Gaussian Process model*

Description

Predict from a GPvecchia-fitted Gaussian Process model

Usage

```
GPvecchia_gp_predict(
  object,
  new_data,
  type = "numeric",
  level = 0.95,
  m_pred = NULL,
  coord_cols = NULL,
  time_col = NULL,
  time_scale = 1,
  ...
)
```

Arguments

object	A "gopher_GPvecchia_fit" returned by GPvecchia_gp_fit().
new_data	An sf object or data.frame with coordinates.
type	"numeric" (default) or "pred_int".
level	Confidence level for prediction intervals (default 0.95).
m_pred	Number of nearest neighbours for prediction approximation. Defaults to the training m.
coord_cols	Character(2) coord column names (non-sf path).
time_col	Optional character scalar specifying a time column for spatiotemporal modelling.
time_scale	Numeric scalar used to rescale time when time_col is provided. Default 1.
...	Additional arguments forwarded to GPvecchia::vecchia_specify() for prediction graph construction.

Value

A tibble::tibble() with prediction columns.

Examples

```
if (requireNamespace("spacetime", quietly = TRUE) &&
    requireNamespace("GPvecchia", quietly = TRUE)) {
  data("air", package = "spacetime")

  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  n_train <- floor(0.8 * nrow(air_sf))
  train_sf <- air_sf[seq_len(n_train), ]
  test_sf <- air_sf[seq.int(n_train + 1L, nrow(air_sf)), ]

  fit <- GPvecchia_gp_fit(pm10 ~ coords.x1 + coords.x2, data = train_sf, m = 10)
  GPvecchia_gp_predict(fit, new_data = test_sf, type = "pred_int")
}
```

`gstat_gp_fit` *Fit a Gaussian Process model using the gstat engine*

Description

This function is called internally by parsnip. End users should call `parsnip::fit.model_spec()` on a `gaussian_process_spatial` specification with `set_engine("gstat")`.

Usage

```
gstat_gp_fit(
  formula,
  data,
  covariance_function = "exponential",
  range = NULL,
  nugget = NULL,
  sill = NULL,
  coord_cols = NULL,
  time_col = NULL,
  fit_variogram = TRUE,
```

```
  cutoff = NULL,
  width = NULL,
  ...
)
```

Arguments

formula	A two-sided formula. Use y ~ 1 for ordinary kriging and y ~ x1 + x2 for universal kriging.
data	An sf object or a data.frame with coordinate columns.
covariance_function	Canonical covariance name (see gaussian_process_spatial()). Defaults to "exponential".
range	Initial/fixed range parameter. NULL = auto-estimate.
nugget	Initial/fixed nugget. NULL = auto-estimate.
sill	Initial/fixed partial sill. NULL = auto-estimate.
coord_cols	Character(2) column names for coordinates when data is not sf. Auto-detected when NULL.
time_col	Character name of the date/time column for spatiotemporal kriging. NULL (default) = purely spatial.
fit_variogram	Logical. When TRUE (default) the empirical variogram is computed and the model is fitted even when range, nugget, and sill are all provided (so initial values are used as starting values for optimisation). Set to FALSE to use the supplied parameters without fitting.
cutoff	Passed to gstat::variogram(): maximum distance for the empirical variogram. NULL = gstat default (one-third of bounding box diagonal).
width	Passed to gstat::variogram(): lag width. NULL = gstat default.
...	Additional arguments forwarded to gstat::fit.variogram().

Value

A list of class "gopher_gstat_fit" containing the fitted variogram model, the original training data, and the formula.

Examples

```
if (requireNamespace("spacetime", quietly = TRUE)) {
  data("air", package = "spacetime")

  # `air` is provided as legacy ST components (matrix + SpatialPoints + Date).
  # Convert one day to an `sf` object for spatial GP fitting.
  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
```

```
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  fit <- gstat_gp_fit(pm10 ~ 1, data = air_sf)
  fit
}
```

`gstat_gp_predict` *Predict from a gstat-fitted Gaussian Process model*

Description

Predict from a gstat-fitted Gaussian Process model

Usage

```
gstat_gp_predict(
  object,
  new_data,
  type = "numeric",
  level = 0.95,
  coord_cols = NULL,
  ...
)
```

Arguments

`object`	A `"gopher_gstat_fit"` object returned by `gstat_gp_fit()`.
`new_data`	An `sf` object or `data.frame` with the same coordinate structure as the training data. For spatiotemporal kriging it must also contain the time column.
`type`	One of `"numeric"` (default, returns `.pred`) or `"pred_int"` (returns `.pred`, `.pred_lower`, `.pred_upper`).
`level`	Confidence level for prediction intervals (default `0.95`).
`coord_cols`	Character(2) coord column names (non-sf path).
`...`	Forwarded to `gstat::krige()` / `gstat::krigeST()`.

Value

A `tibble::tibble()` with prediction columns.

Examples

```
if (requireNamespace("spacetime", quietly = TRUE)) {
  data("air", package = "spacetime")

  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  n_train <- floor(0.8 * nrow(air_sf))
  train_sf <- air_sf[seq_len(n_train), ]
  test_sf <- air_sf[seq.int(n_train + 1L, nrow(air_sf)), ]

  fit <- gstat_gp_fit(pm10 ~ coords.x1 + coords.x2, data = train_sf)
  gstat_gp_predict(fit, new_data = test_sf, type = "pred_int")
}
```

`PrestoGP_gp_fit` *Fit a Gaussian Process model using the PrestoGP engine*

Description

This function is called internally by parsnip. End users should call `parsnip::fit.model_spec()` on a `gaussian_process_spatial` specification with `set_engine("PrestoGP")`.

Usage

```
PrestoGP_gp_fit(
  formula,
  data,
  covariance_function = "matern",
  range = NULL,
  nugget = NULL,
  sill = NULL,
  n_neighbors = 15L,
  model_type = c("vecchia", "full"),
  coord_cols = NULL,
  time_col = NULL,
```

```
  time_scale = 1,
  scaling = NULL,
  common_scale = NULL,
  impute_y = NULL,
  lod_upper = NULL,
  lod_lower = NULL,
  n_impute = 10L,
  eps_impute = 0.01,
  maxit_impute = 0L,
  penalty = c("lasso", "relaxed", "MCP", "SCAD"),
  alpha = 1,
  family = c("gaussian", "binomial"),
  quiet = TRUE,
  ...
)
```

Arguments

formula	A two-sided formula.
data	An sf object or data.frame with coordinate columns.
covariance_function	Canonical covariance name. PrestoGP currently uses Matérn covariance internally; non-Matérn values are accepted but mapped to Matérn.
range	Unused by PrestoGP in the current adapter (kept for unified interface compatibility).
nugget	Unused by PrestoGP in the current adapter (kept for unified interface compatibility).
sill	Unused by PrestoGP in the current adapter (kept for unified interface compatibility).
n_neighbors	Number of neighbors for Vecchia approximation.
model_type	"vecchia" (default) or "full".
coord_cols	Character(2) coordinate column names for non-sf input.
time_col	Optional character scalar specifying a time column for spatiotemporal modelling.
time_scale	Numeric scalar used to rescale time when time_col is provided. Default 1.
scaling	Optional integer vector passed to PrestoGP::prestogp_fit() to group location dimensions by scale parameter.
common_scale	Optional logical passed to PrestoGP::prestogp_fit().
impute_y	Logical or NULL. If NULL, it is automatically enabled when missing outcomes or LOD bounds are present.
lod_upper	Upper LOD bound(s) passed to lod.upper.
lod_lower	Lower LOD bound(s) passed to lod.lower.
n_impute	Number of multiple imputations for LOD-aware missingness.
eps_impute	Convergence tolerance for multiple imputation.

maxit_impute	Maximum iterations for multiple imputation.
penalty	Penalty type passed to PrestoGP ("lasso", "relaxed", "MCP", "SCAD").
alpha	Elastic-net mixing parameter passed to PrestoGP.
family	GLM family passed to PrestoGP ("gaussian" or "binomial").
quiet	Logical; suppress iterative fit messages if TRUE.
...	Additional arguments forwarded to PrestoGP::prestogp_fit().

Value

A list of class "gopher_PrestoGP_fit".

PrestoGP_gp_predict *Predict from a PrestoGP-fitted Gaussian Process model*

Description

Predict from a PrestoGP-fitted Gaussian Process model

Usage

```
PrestoGP_gp_predict(
  object,
  new_data,
  type = "numeric",
  level = 0.95,
  m_pred = NULL,
  coord_cols = NULL,
  time_col = NULL,
  time_scale = 1,
  ...
)
```

Arguments

object	A "gopher_PrestoGP_fit" returned by PrestoGP_gp_fit().
new_data	An sf object or data.frame with coordinates.
type	"numeric" (default) or "pred_int".
level	Confidence level for prediction intervals (default 0.95).
m_pred	Optional number of neighbors for prediction.
coord_cols	Character(2) coordinate column names for non-sf input.
time_col	Optional character scalar specifying a time column for spatiotemporal modelling.
time_scale	Numeric scalar used to rescale time when time_col is provided. Default 1.
...	Forwarded to PrestoGP::prestogp_predict().

Value

A `tibble::tibble()` with prediction columns.

`sdmTMB_gp_fit` *Fit a Gaussian Process model using the sdmTMB engine*

Description

This function is called internally by parsnip. End users should call `parsnip::fit.model_spec()` on a `gaussian_process_spatial` specification with `set_engine("sdmTMB")`.

Usage

```
sdmTMB_gp_fit(
  formula,
  data,
  covariance_function = "matern",
  range = NULL,
  nugget = NULL,
  sill = NULL,
  coord_cols = NULL,
  time_col = NULL,
  spatial = "on",
  spatiotemporal = "off",
  mesh_cutoff = NULL,
  n_knots = NULL,
  family = stats::gaussian(),
  share_range = FALSE,
  extra_time = NULL,
  silent = TRUE,
  ...
)
```

Arguments

`formula`	A two-sided formula. Use `y ~ 1` for a spatial-only model without fixed effects (analogous to ordinary kriging), or include covariates (e.g. `y ~ x1 + x2`) for a model with a fixed-effects trend (analogous to universal kriging).
`data`	An `sf` object or a `data.frame` with coordinate columns.
`covariance_function`	Canonical covariance name (see `gaussian_process_spatial()`). sdmTMB uses a Matérn covariance via the SPDE mesh; any value is accepted but mapped to `"matern"`. A warning is issued for non-Matérn names.
`range`	Unused by sdmTMB in the current adapter (estimated internally via maximum likelihood). Kept for unified interface compatibility.

nugget	Unused by sdmTMB in the current adapter. Kept for unified interface compatibility.
sill	Unused by sdmTMB in the current adapter. Kept for unified interface compatibility.
coord_cols	Character vector of length 2 giving the coordinate column names for non-sf input (x/longitude first, y/latitude second). NULL triggers auto-detection.
time_col	Character scalar naming the time column for spatiotemporal models. NULL (default) = purely spatial.
spatial	One of "on" (default) or "off". Controls whether a spatial random field is included.
spatiotemporal	One of "off" (default), "iid", "ar1", or "rw". Controls the temporal structure of the spatiotemporal random field. Ignored when time_col is NULL.
mesh_cutoff	Numeric scalar passed to sdmTMB::make_mesh() as the cutoff argument (minimum triangle edge length in coordinate units). NULL = automatic (1/10 of the bounding box diagonal).
n_knots	Integer. Number of mesh knots for the k-means mesh type. If provided, mesh_cutoff is ignored and type = "kmeans" is used.
family	A family object (default stats::gaussian()). Passed directly to sdmTMB::sdmTMB().
share_range	Logical. Whether to share the spatial range between the spatial and spatiotemporal random fields. Default FALSE.
extra_time	Integer or numeric vector of additional time values to include in the model that are not present in data. Required when prediction newdata contains time values unseen during training (e.g. future time steps). Passed directly to sdmTMB::sdmTMB(). NULL (default) = include only the time values present in data.
silent	Logical. Suppress model-fitting messages. Default TRUE.
...	Additional arguments forwarded to sdmTMB::sdmTMB().

Value

A list of class "gopher_sdmTMB_fit" containing:

- sdmtmb_fit — the fitted sdmTMB model object.
- mesh — the sdmTMBmesh used for fitting.
- formula — the model formula.
- xy_cols — internal coordinate column names used in the data.
- coord_cols — user-supplied coord_cols (may be NULL).
- time_col — the time column name (may be NULL).
- extra_time — additional time values declared at fit time (may be NULL).
- family — the family object.

Examples

```
if (requireNamespace("sdmTMB", quietly = TRUE)) {
  train <- data.frame(
    x = runif(50, 0, 10),
    y = runif(50, 0, 10)
  )
  train$z <- sin(train$x) + cos(train$y) + rnorm(50, 0, 0.3)
  train_sf <- sf::st_as_sf(train, coords = c("x", "y"), crs = 4326)

  fit <- sdmTMB_gp_fit(z ~ 1, data = train_sf)
  fit
}
```

`sdmTMB_gp_predict` *Predict from an sdmTMB-fitted Gaussian Process model*

Description

Predict from an sdmTMB-fitted Gaussian Process model

Usage

```
sdmTMB_gp_predict(
  object,
  new_data,
  type = "numeric",
  level = 0.95,
  coord_cols = NULL,
  ...
)
```

Arguments

`object`	A `"gopher_sdmTMB_fit"` object returned by `sdmTMB_gp_fit()`.
`new_data`	An `sf` object or `data.frame` with the same coordinate structure (and covariate columns, if any) as the training data.
`type`	One of `"numeric"` (default, returns `.pred`) or `"pred_int"` (returns `.pred`, `.pred_lower`, `.pred_upper`).
`level`	Confidence level for prediction intervals (default `0.95`).
`coord_cols`	Character vector of length 2 giving coordinate column names for non-`sf` new data. `NULL` uses the coord_cols from the fit.
`...`	Additional arguments forwarded to `predict.sdmTMB()`.

Value

A `tibble::tibble()` with prediction columns.

Examples

```
if (requireNamespace("sdmTMB", quietly = TRUE)) {
  set.seed(1)
  train <- data.frame(x = runif(50, 0, 10), y = runif(50, 0, 10))
  train$z <- sin(train$x) + cos(train$y) + rnorm(50, 0, 0.3)
  train_sf <- sf::st_as_sf(train, coords = c("x", "y"), crs = 4326)

  newdat <- sf::st_as_sf(
    expand.grid(x = seq(1, 9, l = 4), y = seq(1, 9, l = 4)),
    coords = c("x", "y"), crs = 4326
  )

  fit   <- sdmTMB_gp_fit(z ~ 1, data = train_sf)
  sdmTMB_gp_predict(fit, new_data = newdat)
}
```

---

spNNGP_gp_fit *Fit a Gaussian Process model using the spNNGP engine*

Description

This function is called internally by parsnip. End users should call `parsnip::fit.model_spec()` on a `gaussian_process_spatial` specification with `set_engine("spNNGP")`.

Usage

```
spNNGP_gp_fit(
  formula,
  data,
  covariance_function = "exponential",
  range = NULL,
  nugget = NULL,
  sill = NULL,
  n_neighbors = 15L,
  n_samples = 1000L,
  n_burnin = 500L,
  method = "response",
  coord_cols = NULL,
  time_col = NULL,
  time_scale = 1,
  ...
)
```

Arguments

`formula`	A two-sided formula. Covariates are included as fixed effects (Universal Kriging).

data	An sf object or data.frame with coordinate columns.
covariance_function	Canonical covariance name. Defaults to "exponential".
range	Starting value for the range parameter (phi). NULL = uses a data-driven initial value.
nugget	Starting value for the nugget (tau.sq). NULL = 10 % of response variance.
sill	Starting value for the partial sill (sigma.sq). NULL = 90 % of response variance.
n_neighbors	Number of nearest neighbours. Default 15.
n_samples	Number of MCMC samples. Default 1000.
n_burnin	MCMC burn-in. Default 500.
method	spNNGP method: "response" (default) or "latent".
coord_cols	Character(2) coordinate column names (non-sf path).
time_col	Optional character scalar specifying a time column for spatiotemporal modelling.
time_scale	Numeric scalar used to rescale time when time_col is provided. Default 1.
...	Additional arguments forwarded to spNNGP::spNNGP().

Value

A list of class "gopher_spNNGP_fit".

Examples

```
if (requireNamespace("spacetime", quietly = TRUE) &&
    requireNamespace("spNNGP", quietly = TRUE)) {
  data("air", package = "spacetime")

  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  fit <- spNNGP_gp_fit(
    pm10 ~ coords.x1 + coords.x2,
    data = air_sf,
    n_neighbors = 8,
    n_samples = 80,
```

```
    n_burnin = 40
  )
  fit
}
```

---

spNNGP_gp_predict *Predict from an spNNGP-fitted Gaussian Process model*

Description

Predict from an spNNGP-fitted Gaussian Process model

Usage

```
spNNGP_gp_predict(
  object,
  new_data,
  type = "numeric",
  level = 0.95,
  coord_cols = NULL,
  time_col = NULL,
  time_scale = 1,
  ...
)
```

Arguments

object	A "gopher_spNNGP_fit" returned by spNNGP_gp_fit().
new_data	An sf object or data.frame with coordinates.
type	"numeric" (default) or "pred_int".
level	Confidence level for prediction intervals (default 0.95).
coord_cols	Character(2) coord column names (non-sf path).
time_col	Optional character scalar specifying a time column for spatiotemporal modelling.
time_scale	Numeric scalar used to rescale time when time_col is provided. Default 1.
...	Forwarded to predict.spNNGP().

Value

A tibble::tibble() with prediction columns.

Examples

```
if (requireNamespace("spacetime", quietly = TRUE) &&
    requireNamespace("spNNGP", quietly = TRUE)) {
  data("air", package = "spacetime")

  day_id <- which.max(colSums(!is.na(air)))
  air_day <- data.frame(
    station = rownames(air),
    pm10 = air[, day_id],
    day = dates[day_id],
    sp::coordinates(stations)
  )
  air_day <- air_day[stats::complete.cases(air_day$pm10), ]
  air_sf <- sf::st_as_sf(
    air_day,
    coords = c("coords.x1", "coords.x2"),
    crs = 4326,
    remove = FALSE
  )

  n_train <- floor(0.8 * nrow(air_sf))
  train_sf <- air_sf[seq_len(n_train), ]
  test_sf <- air_sf[seq.int(n_train + 1L, nrow(air_sf)), ]

  fit <- spNNGP_gp_fit(
    pm10 ~ coords.x1 + coords.x2,
    data = train_sf,
    n_neighbors = 8,
    n_samples = 80,
    n_burnin = 40
  )
  spNNGP_gp_predict(fit, new_data = test_sf, type = "pred_int")
}
```

Index