Package: geopytorch (via r-universe)

October 21, 2026

Type Package

Title R Interface to 'GeoPyTorch'

Version 0.1.0

Description Fits ordinary kriging and spatial linear models through 'reticulate' with the 'GeoPyTorch' 'Python' package, which is built on 'GPyTorch' and 'PyTorch'. Covariance parameters can be fixed or estimated by maximum likelihood or restricted maximum likelihood, and predictions include their variances. 'Python' and the required packages are installed on first use by 'reticulate'.

License MIT + file LICENSE

URL https://github.com/dbtnc/geopytorch

BugReports https://github.com/dbtnc/geopytorch/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports reticulate (>= 1.41.0), stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

SystemRequirements Python (>= 3.12)

NeedsCompilation no

Author Daniel Tinoco [aut, cre]

Maintainer Daniel Tinoco <dbtnc.ps2@gmail.com>

Config/pak/sysreqs python3

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

Date/Publication 2026-10-21 15:10:08 UTC

RemoteUrl https://github.com/cran/geopytorch

RemoteRef HEAD

RemoteSha f66ec09080be625289500629d73361dae4d4f378

Contents

geopytorch-package 2
geopytorch_config 2
geopytorch_devices 3
geopytorch_module 4
geopytorch_requirement 4
geopytorch_threads 5
kriging_weights 5
ordinary_kriging 6
predict.geopytorch_ordinary_kriging 8
predict.geopytorch_spatial_linear_model 9
print.geopytorch_config 10
print.geopytorch_ordinary_kriging 11
print.geopytorch_prediction 11
spatial_linear_model 12

Index **14**

`geopytorch-package` *R Interface to GeoPyTorch*

Description

The **geopytorch** package exposes the GeoPyTorch Python package through **reticulate**. Python is imported lazily. A set of helper functions is included for convenience.

Details

Set the `GEOPYTORCH_PYTHON_REQUIREMENT` environment variable to a pip-style local requirement such as `geopytorch@/absolute/path/to/geopytorch`. When the variable is unset, the package declares `geopytorch>=0.1.0`.

`geopytorch_config` *Report the Active GeoPyTorch Runtime*

Description

Initializes Python, imports the selected GeoPyTorch module, and reports the selected requirement, module source, and dependency versions.

Usage

```
geopytorch_config()
```

Value

An object of class `geopytorch_config`.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
geopytorch_config()

## End(Not run)
```

---

geopytorch_devices *Inspect and Select PyTorch Devices*

Description

Initializes Python and inspects or validates devices in the selected PyTorch runtime. CPU is always available. Automatic selection prefers CUDA, then MPS for single precision, then CPU.

Usage

```
geopytorch_devices()

geopytorch_device(device = "cpu", dtype = c("float64", "float32"))
```

Arguments

`device`	A device string: `"cpu"`, `"auto"`, `"mps"`, `"cuda"`, or `"cuda:N"`.
`dtype`	Either `"float64"` or `"float32"`.

Value

`geopytorch_devices()` returns available device strings. `geopytorch_device()` returns one validated device string.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
geopytorch_devices()
geopytorch_device("cpu")

## End(Not run)
```

geopytorch_module *Access the Delayed GeoPyTorch Python Module*

Description

Returns the module proxy created during package loading. Retrieving the proxy does not initialize Python; accessing one of its attributes does.

Usage

```
geopytorch_module()
```

Value

A delayed Python module proxy.

Examples

```
# Getting the module does not start Python.
module <- geopytorch_module()

## Not run:
# Using one of its attributes does.
module$COVARIANCE_MODELS

## End(Not run)
```

geopytorch_requirement *Selected GeoPyTorch Python Requirement*

Description

Reports the Python requirement selected when the package is loaded.

Usage

```
geopytorch_requirement()
```

Details

Set `GEOPYTORCH_PYTHON_REQUIREMENT` before loading the package to use a local checkout, for example `geopytorch@/absolute/path/to/geopytorch`.

Value

A length-one character vector.

Examples

```
geopytorch_requirement()
```

---

geopytorch_threads *Get or Set PyTorch CPU Thread Counts*

Description

Initializes Python and optionally configures PyTorch CPU parallelism. PyTorch permits changing the inter-op count only before parallel work starts.

Usage

```
geopytorch_threads(threads = NULL, interop_threads = NULL)
```

Arguments

`threads`	Optional positive intra-op thread count.
`interop_threads`	Optional positive inter-op thread count.

Value

A list containing effective intra-op and inter-op thread counts.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
geopytorch_threads()
geopytorch_threads(threads = 2)

## End(Not run)
```

---

kriging_weights *Extract Kriging Weights*

Description

Computes ordinary- or universal-kriging weights. With a regression design matrix, the weights reproduce the prediction design from the training design.

Usage

```
kriging_weights(object, newdata, newcoords = NULL)
```

Arguments

object	A fitted object returned by `ordinary_kriging()` or `spatial_linear_model()`.
newdata	Prediction coordinates for ordinary kriging, or a prediction data frame for spatial linear models.
newcoords	Optional prediction coordinate matrix for spatial linear models.

Value

A numeric matrix with prediction locations in rows and observations in columns.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
train_x <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), ncol = 2, byrow = TRUE)
train_y <- c(1, 2, 4, 3)
target_x <- matrix(c(0.25, 0.4, 0.8, 0.6), ncol = 2, byrow = TRUE)
fit <- ordinary_kriging(
  train_x,
  train_y,
  covariance_model = "exponential",
  partial_sill = 1.2,
  phi = 0.7,
  nugget = 1e-3,
  training_iterations = 0
)
weights <- kriging_weights(fit, target_x)
rowSums(weights)

## End(Not run)
```

---

ordinary_kriging *Fit an Ordinary-Kriging Model with GeoPyTorch*

Description

Converts R coordinates and values to tensors on the requested device and fits GeoPyTorch's exact Ordinary-Kriging model.

Usage

```
ordinary_kriging(
  train_x,
  train_y,
  covariance_model = c("matern", "exponential", "gaussian", "spherical", "wave"),
  matern_nu = 1.5,
  ard = FALSE,
  partial_sill = NULL,
```

```
  phi = NULL,
  nugget = NULL,
  nugget_lower_bound = 0,
  training_iterations = 100L,
  learning_rate = 0.1,
  likelihood_method = c("ml", "reml"),
  optimizer = c("adam", "lbfgs"),
  device = "cpu",
  dtype = c("float64", "float32")
)
```

Arguments

`train_x`	Numeric matrix or data frame of observation coordinates.
`train_y`	Numeric vector of observed values.
`covariance_model`	Covariance model name.
`matern_nu`	Matérn smoothness: 0.5, 1.5, or 2.5.
`ard`	Whether to use a range per coordinate.
`partial_sill`	Optional fixed or initial covariance partial sill.
`phi`	Optional fixed or initial covariance range.
`nugget`	Optional fixed or initial Gaussian noise variance.
`nugget_lower_bound`	Strict lower bound used when a nugget is supplied.
`training_iterations`	Optimizer iterations; zero retains initialized parameters.
`learning_rate`	Positive optimizer learning rate.
`likelihood_method`	Maximum likelihood (`"ml"`) or restricted maximum likelihood (`"reml"`).
`optimizer`	`"adam"` or `"lbfgs"`.
`device`	PyTorch device. CPU is the default; use `"cuda"` for the first CUDA GPU.
`dtype`	Either `"float64"` or `"float32"`.

Value

An object of class `geopytorch_ordinary_kriging`.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
train_x <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), ncol = 2, byrow = TRUE)
train_y <- c(1, 2, 4, 3)
target_x <- matrix(c(0.25, 0.4, 0.8, 0.6), ncol = 2, byrow = TRUE)
fit <- ordinary_kriging(
  train_x,
```

```
  train_y,
  covariance_model = "exponential",
  partial_sill = 1.2,
  phi = 0.7,
  nugget = 1e-3,
  training_iterations = 0
)
fit

## End(Not run)
```

`predict.geopytorch_ordinary_kriging`
Predict with a GeoPyTorch Ordinary-Kriging Model

Description

Performs exact Ordinary-Kriging prediction on the device used for fitting and materializes requested results as R numeric objects.

Usage

```
## S3 method for class 'geopytorch_ordinary_kriging'
predict(
  object,
  newdata,
  observation_noise = FALSE,
  full_covariance = FALSE,
  ...
)
```

Arguments

`object`	A fitted object returned by `ordinary_kriging()`.
`newdata`	Numeric matrix or data frame of prediction coordinates.
`observation_noise`	Include the fitted nugget in prediction variance.
`full_covariance`	Transfer the full joint covariance matrix to R.
`...`	Unused.

Value

An object of class `geopytorch_prediction` containing mean, marginal variance, standard deviation, and optionally the joint covariance matrix.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
train_x <- matrix(c(0, 0, 1, 0, 0, 1, 1, 1), ncol = 2, byrow = TRUE)
train_y <- c(1, 2, 4, 3)
target_x <- matrix(c(0.25, 0.4, 0.8, 0.6), ncol = 2, byrow = TRUE)
fit <- ordinary_kriging(
  train_x,
  train_y,
  covariance_model = "exponential",
  partial_sill = 1.2,
  phi = 0.7,
  nugget = 1e-3,
  training_iterations = 0
)
prediction <- predict(fit, target_x, observation_noise = TRUE)
prediction$mean
prediction$variance

## End(Not run)
```

---

predict.geopytorch_spatial_linear_model

Predict from a spatial linear model

Description

Predict from a spatial linear model

Usage

```
## S3 method for class 'geopytorch_spatial_linear_model'
predict(
  object,
  newdata,
  newcoords = NULL,
  observation_noise = FALSE,
  full_covariance = FALSE,
  ...
)
```

Arguments

`object`	An object returned by spatial_linear_model().
`newdata`	A data frame containing prediction covariates and coordinate columns.
`newcoords`	Optional coordinate matrix; required if coords was a matrix during fitting.
`observation_noise`	Include independent future nugget variance.

`full_covariance`
: Return the joint prediction covariance in R.

`...`
: Unused.

Value

A geopytorch_prediction object. Uncertainty includes estimation of regression coefficients but not estimation of covariance parameters.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
set.seed(1)
data <- data.frame(east = runif(30), north = runif(30), z = rnorm(30))
data$y <- 1 + 0.5 * data$z + rnorm(30)
fit <- spatial_linear_model(
  y ~ z,
  data = data,
  coords = c("east", "north"),
  covariance_model = "exponential"
)
newdata <- data.frame(east = c(0.2, 0.8), north = c(0.3, 0.7), z = c(0, 1))
prediction <- predict(fit, newdata)
prediction$mean
prediction$variance

## End(Not run)
```

`print.geopytorch_config`

Print a GeoPyTorch Runtime Configuration

Description

Prints the selected R, Python, GeoPyTorch, and numerical dependency versions.

Usage

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

Arguments

`x`
: An object returned by `geopytorch_config()`.

`...`
: Unused.

Value

`x`, invisibly.

print.geopytorch_ordinary_kriging

Print a GeoPyTorch Ordinary-Kriging Model

Description

Prints the training size, covariance model, device, and dtype.

Usage

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

Arguments

x	A fitted Ordinary-Kriging model.
...	Unused.

Value

x, invisibly.

print.geopytorch_prediction

Print a GeoPyTorch Prediction

Description

Prints the prediction size, device, and dtype.

Usage

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

Arguments

x	An object returned by prediction from a GeoPyTorch model.
...	Unused.

Value

x, invisibly.

spatial_linear_model *Fit a Gaussian spatial linear model*

Description

Fits a linear mean with spatially correlated Gaussian errors. Regression coefficients are profiled by generalized least squares. The covariance parameters are fitted by ML or REML using GPyTorch and torch.optim.LBFGS.

Usage

```
spatial_linear_model(
  formula,
  data,
  coords,
  covariance_model = c("exponential", "matern", "gaussian", "spherical", "wave"),
  matern_nu = 1.5,
  initial = NULL,
  likelihood_method = c("reml", "ml"),
  training_iterations = 600L,
  starts = NULL,
  parameter_bounds = NULL,
  device = "cpu",
  dtype = c("float64", "float32")
)
```

Arguments

`formula` A regression formula with a numeric response. Offsets are not supported.

`data` A data frame containing the response and covariates.

`coords` Coordinate column names in data, or a numeric coordinate matrix.

`covariance_model` Spatial covariance model.

`matern_nu` Fixed Matérn smoothness: 0.5, 1.5, or 2.5.

`initial` Initial or fixed c(partial_sill, phi, nugget). NULL derives values from the data.

`likelihood_method` reml (default) or ml.

`training_iterations` Maximum optimizer iterations. Zero keeps initial parameters.

`starts` Optional list of positive initial parameter triples for multiple starts.

`parameter_bounds` Optional list of three c(lower, upper) pairs in physical units. Optimization starts must be strictly inside the bounds. GPyTorch constraints enforce bounds by transforming the raw parameters; boundary optima are approached from the interior. Fixed parameters may lie on bounds. Zero lower bounds allow parameters to approach zero. Python's default is unbounded.

device	PyTorch device, default cpu.
dtype	float64 (default) or float32.

Details

The range phi gives exponential correlation exp(-h/phi), Gaussian correlation exp(-(h/phi)^2), and Matérn Bessel argument h/phi at distance h. Prediction defaults to the latent process and includes regression-coefficient uncertainty, conditional on estimated covariance parameters. Computation is exact and dense. Missing values and rank-deficient designs are rejected. A zero nugget is supported when covariance parameters are fixed. logLik() uses the fitted likelihood method; REML omits the design-only term 0.5 * log(det(t(X) %*% X)).

Value

A geopytorch_spatial_linear_model object with coefficients, their covariance, covariance parameters, likelihoods, optimizer diagnostics, and the retained Python model. Factor levels and contrasts are retained for prediction.

Examples

```
## Not run:
# Needs 'Python'; on first use 'reticulate' installs GeoPyTorch and PyTorch.
set.seed(1)
data <- data.frame(east = runif(30), north = runif(30), z = rnorm(30))
data$y <- 1 + 0.5 * data$z + rnorm(30)
fit <- spatial_linear_model(
  y ~ z,
  data = data,
  coords = c("east", "north"),
  covariance_model = "exponential"
)
coef(fit)
fit$covariance_parameters

## End(Not run)
```

Index

geopytorch *(geopytorch-package)*, 2
geopytorch-package, 2
geopytorch_config, 2
geopytorch_device *(geopytorch_devices)*, 3
geopytorch_devices, 3
geopytorch_module, 4
geopytorch_requirement, 4
geopytorch_threads, 5

kriging_weights, 5

ordinary_kriging, 6

predict.geopytorch_ordinary_kriging, 8
predict.geopytorch_spatial_linear_model, 9
print.geopytorch_config, 10
print.geopytorch_ordinary_kriging, 11
print.geopytorch_prediction, 11

spatial_linear_model, 12