Package: gwrf (via r-universe)

October 25, 2026

Title Geographically Weighted Random Forests

Version 0.1.1.9000

Description Fits geographically weighted random forest models using spatially localized training neighborhoods and 'ranger' as the random forest engine. Supports fixed-distance and adaptive neighborhoods defined by observation rows or unique spatial locations, including repeated observations at the same location. Provides local predictions and permutation-based variable importance for examining spatial variation in predictive relationships. The geographical random forest approach is described by Georganos et al. (2021) <doi:10.1080/10106049.2019.1595177>, and the 'ranger' engine by Wright and Ziegler (2017) <doi:10.18637/jss.v077.i01>.

License MIT + file LICENSE

URL https://github.com/hac-lab/gwrf

BugReports https://github.com/hac-lab/gwrf/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports ranger, tibble, dplyr, pbapply, stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Repository https://hac-lab.r-universe.dev

Date/Publication 2026-08-26 18:10:46 UTC

RemoteUrl https://github.com/hac-lab/gwrf

RemoteRef HEAD

RemoteSha 611d16f5cfbc16350e0585dccabe1f3d74991087

Contents

build_neighbors 2
diagnostics 3
fit_gwrf 4
fit_local_rf 7
kernel_weights 9
predict.gwrf_fit 10
print.gwrf_fit 11

Index **12**

`build_neighbors` *Build a Spatial Neighborhood*

Description

Identifies neighboring observations or unique spatial locations for a focal observation.

Usage

```
build_neighbors(
  coords,
  focal_index,
  bandwidth,
  adaptive = TRUE,
  neighbor_unit = c("row", "location"),
  location_id = NULL
)
```

Arguments

`coords`	Numeric matrix or data frame containing spatial coordinates.
`focal_index`	Integer index identifying the focal observation.
`bandwidth`	Numeric bandwidth. For adaptive neighborhoods, this is the number of neighboring rows or unique locations.
`adaptive`	Logical indicating whether an adaptive neighborhood is used.
`neighbor_unit`	Character string indicating whether neighborhoods are defined using individual data rows (`"row"`) or unique spatial locations (`"location"`).
`location_id`	Optional vector identifying the spatial location associated with each observation. Required when `neighbor_unit = "location"`.

Value

A list describing the spatial neighborhood of the focal observation. The list contains:

neighbor_index Integer vector giving the rows of `coords` included in the local neighborhood.

distances Numeric vector containing the distance from the focal location to each selected neighboring row.

local_bandwidth Numeric value giving the realized local bandwidth. For adaptive neighborhoods this is the distance to the most distant selected neighbor; for fixed neighborhoods it is the supplied bandwidth.

all_distances Numeric vector of distances from the focal location to all candidate rows or unique locations, depending on `neighbor_unit`.

neighbor_unit Character string indicating whether the neighborhood was constructed using rows or unique locations.

neighbor_location_id For location-based neighborhoods, the identifiers of the selected unique locations; `NULL` for row-based neighborhoods.

Examples

```
coords <- matrix(
  c(0, 0,
    1, 0,
    2, 0,
    3, 0,
    4, 0),
  ncol = 2,
  byrow = TRUE
)

build_neighbors(
  coords = coords,
  focal_index = 3,
  bandwidth = 3,
  adaptive = TRUE
)
```

---

diagnostics *Extract diagnostics from a gwrf_fit object*

Description

Extract diagnostics from a gwrf_fit object

Usage

```
diagnostics(object, ...)
```

Arguments

`object`	A gwrf_fit object.
`...`	Unused.

Value

A named list containing overall model diagnostics:

rmse Root mean squared error across fitted focal observations.

mae Mean absolute error across fitted focal observations.

r2 Coefficient of determination calculated from observed and predicted focal values.

n_focal Number of focal observations for which local models were requested.

Examples

```
example_fit <- structure(
  list(
    diagnostics = list(
      rmse = 0.25,
      mae = 0.18,
      r2 = 0.80,
      n_focal = 3
    )
  ),
  class = "gwrf_fit"
)

diagnostics(example_fit)
```

`fit_gwrf` *Fit a Geographically Weighted Random Forest*

Description

Fits a separate random-forest model for each focal observation or spatial location using observations selected from a geographically defined local neighborhood. Neighborhoods may be defined using individual data rows or unique spatial locations, allowing the function to support repeated observations at the same location, including spatial panel data.

Usage

```
fit_gwrf(
  formula,
  data,
  coords,
  bandwidth,
  adaptive = TRUE,
```

```
  kernel = "bisquare",
  neighbor_unit = c("row", "location"),
  location_id = NULL,
  num.trees = 500,
  mtry = NULL,
  min.node.size = 5,
  importance = "permutation",
  use_case_weights = TRUE,
  focal_indices = NULL,
  keep_local_models = FALSE,
  seed = NULL,
  verbose = TRUE,
  n_workers = 1L,
  num.threads = NULL
)
```

Arguments

formula	A model formula specifying the response and predictor variables.
data	A data frame containing the response, predictors, and any location identifiers used in the model.
coords	A numeric matrix or data frame with two columns containing the spatial coordinates associated with the rows of data.
bandwidth	A positive numeric value defining the local neighborhood. When adaptive = TRUE, this is the number of neighboring rows or unique spatial locations included in each local neighborhood. When adaptive = FALSE, this is a fixed distance threshold expressed in the units of coords.
adaptive	Logical. If TRUE, adaptive neighborhoods are defined using the nearest observations or unique locations. If FALSE, fixed-distance neighborhoods are used.
kernel	Character string specifying the spatial weighting kernel. The default is "bisquare".
neighbor_unit	Character string indicating whether neighborhoods are defined using individual data rows ("row") or unique spatial locations ("location").
location_id	Optional vector identifying the spatial location associated with each observation. Required when neighbor_unit = "location". All eligible observations associated with selected neighboring locations are retained for local model fitting.
num.trees	Number of trees grown in each local random forest.
mtry	Number of predictor variables randomly sampled as candidates at each split. If NULL, the value is determined by ranger::ranger().
min.node.size	Minimum terminal-node size used by each local random forest.
importance	Character string specifying the variable-importance method passed to ranger::ranger(). The default is "permutation".
use_case_weights	Logical indicating whether spatial kernel weights are supplied to the local random forest as case weights.

`focal_indices` Optional integer vector identifying the focal observations for which local models should be fitted. If `NULL`, local models are fitted for all eligible focal observations.

`keep_local_models` Logical indicating whether fitted local `ranger` model objects are retained in the returned object.

`seed` Optional integer random seed used for reproducible local random-forest fitting.

`verbose` Logical indicating whether progress messages are displayed during model fitting.

`n_workers` Positive integer giving the number of focal models to fit concurrently. The default is `1`, which fits focal models sequentially. Values greater than 1 use fork-based parallel processing and are currently supported only on Unix-like systems.

`num.threads` Optional positive integer giving the number of threads used by `ranger::ranger()` within each focal model. If `NULL`, the `ranger` default is used for sequential fitting. When `n_workers > 1` and `num.threads = NULL`, this is automatically set to 1 to avoid nested parallelism.

Details

For each focal observation, the function constructs a spatial neighborhood, fits a local random forest using the observations contained in that neighborhood, and returns the focal prediction and predictor-importance values. When `neighbor_unit = "location"`, adaptive bandwidth refers to the number of nearest unique spatial locations rather than the number of individual rows. This prevents repeated observations from the same location from being treated as separate spatial neighbors.

Spatial weights are determined by the selected kernel and the distances between the focal location and neighboring observations or locations. Variable importance describes predictive reliance within each fitted local forest and does not indicate effect direction, statistical significance, or causality.

Focal models can be fitted in parallel by setting `n_workers > 1`. Parallelization occurs across focal locations rather than within individual random forests. To avoid oversubscribing available processors, each local `ranger` fit should generally use a single thread when multiple focal workers are active.

Value

An object of class `"gwrf_fit"`. The object is a named list containing the model call and specification, input data and coordinates, neighborhood and random-forest settings, local model results, optional fitted local models, and model diagnostics.

The `local_results` component is a tibble with one row per fitted focal observation and columns for the focal index, observed response, local prediction, residual, local sample size, realized bandwidth, coordinates, and, when available, local variable-importance values prefixed with `"vi_"`.

The `diagnostics` component is a list containing overall RMSE, MAE, R-squared, and the number of focal models fitted.

See Also

`ranger`

Examples

```
set.seed(1)

n <- 20
dat <- data.frame(
  y = rnorm(n),
  x1 = rnorm(n),
  x2 = runif(n)
)
coords <- cbind(seq_len(n), rep(0, n))

fit <- fit_gwrf(
  y ~ x1 + x2,
  data = dat,
  coords = coords,
  bandwidth = 12,
  adaptive = TRUE,
  num.trees = 10,
  focal_indices = 1:3,
  seed = 1,
  verbose = FALSE
)

fit
fit$local_results
```

---

fit_local_rf *Fit one local random forest*

Description

Fit one local random forest

Usage

```
fit_local_rf(
  data,
  formula,
  coords,
  focal_index,
  bandwidth,
  adaptive = TRUE,
  kernel = "bisquare",
  num.trees = 500,
  mtry = NULL,
  min.node.size = 5,
  importance = "permutation",
```

```
  use_case_weights = TRUE,
  seed = NULL,
  keep_model = FALSE,
  neighbor_unit = "row",
  location_id = NULL,
  num.threads = NULL
)
```

Arguments

data	Data frame containing response and predictors.
formula	Model formula.
coords	Numeric matrix/data frame with 2 columns.
focal_index	Integer index of the focal observation.
bandwidth	Adaptive k or fixed distance threshold.
adaptive	Logical; adaptive or fixed neighborhood.
kernel	Kernel type.
num.trees	Number of trees for ranger.
mtry	Number of variables tried at each split.
min.node.size	Minimum node size for ranger.
importance	Importance type passed to ranger.
use_case_weights	Logical; whether to pass kernel weights to ranger.
seed	Optional random seed.
keep_model	Logical; whether to retain fitted ranger model.
neighbor_unit	Character string indicating whether neighborhoods are defined using individual data rows ("row") or unique spatial locations ("location").
location_id	Optional vector identifying the spatial location associated with each observation. Required when neighbor_unit = "location".
num.threads	Optional positive integer giving the number of threads used by ranger::ranger() for this local random forest. If NULL, the ranger default is used.

Value

A named list containing the results for one focal local random forest:

focal_index Integer index of the focal observation.

observed Observed response value at the focal observation.

prediction Prediction from the local random forest for the focal observation.

residual Observed minus predicted response for the focal observation.

n_local Number of complete observations used to fit the local random forest.

local_bandwidth Realized spatial bandwidth of the local neighborhood.

variable_importance Named numeric vector of local variable importance values, or NA when importance cannot be calculated.

ranger_model The fitted ranger model when keep_model = TRUE; otherwise NULL.

Examples

```
set.seed(1)

n <- 20
dat <- data.frame(
  y = rnorm(n),
  x1 = rnorm(n),
  x2 = runif(n)
)
coords <- cbind(seq_len(n), rep(0, n))

local_fit <- fit_local_rf(
  data = dat,
  formula = y ~ x1 + x2,
  coords = coords,
  focal_index = 10,
  bandwidth = 12,
  adaptive = TRUE,
  num.trees = 10,
  seed = 1
)

local_fit$prediction
```

---

kernel_weights *Compute kernel weights from distances*

Description

Compute kernel weights from distances

Usage

```
kernel_weights(distances, bandwidth, kernel = "bisquare")
```

Arguments

distances	Numeric vector of distances.
bandwidth	Positive numeric bandwidth.
kernel	Kernel type: "bisquare", "gaussian", or "tricube".

Value

A numeric vector with the same length as `distances`. Each element is the spatial kernel weight assigned to the corresponding observation, with larger weights generally assigned to observations closer to the focal location.

Examples

```
distances <- c(0, 1, 2, 3)

kernel_weights(
  distances = distances,
  bandwidth = 3,
  kernel = "bisquare"
)
```

`predict.gwrf_fit` *Return fitted local predictions*

Description

Return fitted local predictions

Usage

```
## S3 method for class 'gwrf_fit'
predict(object, ...)
```

Arguments

`object`	A gwrf_fit object.
`...`	Unused.

Value

A numeric vector containing the fitted local prediction for each focal observation represented in `object$local_results`, in the same order as the rows of that table.

Examples

```
example_fit <- structure(
  list(
    local_results = data.frame(
      prediction = c(1.2, 2.1, 2.8)
    )
  ),
  class = "gwrf_fit"
)

predict(example_fit)
```

print.gwrf_fit *Print a gwrf_fit object*

Description

Print a gwrf_fit object

Usage

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

Arguments

x	A gwrf_fit object.
...	Unused.

Value

Invisibly returns x, the input object of class "gwrf_fit". The function is primarily called for its side effect of printing a concise summary of the fitted model, including the model specification and global diagnostic statistics.

Examples

```
example_fit <- structure(
  list(
    formula = y ~ x,
    kernel = "bisquare",
    adaptive = TRUE,
    bandwidth = 10,
    num.trees = 50,
    diagnostics = list(
      rmse = 0.25,
      mae = 0.18,
      r2 = 0.80,
      n_focal = 3
    )
  ),
  class = "gwrf_fit"
)

print(example_fit)
```

Index

build_neighbors, 2

diagnostics, 3

fit_gwrf, 4
fit_local_rf, 7

kernel_weights, 9

predict.gwrf_fit, 10
print.gwrf_fit, 11

ranger, *6*