Package: GWPR.light (via r-universe)

May 29, 2026

Type Package

Title Geographically Weighted Panel Regression

Version 1.0.0

Description A modern, first implementation of Geographically Weighted Panel Regression (GWPR) for spatial panel data. The package provides a unified public API supporting Gaussian and binomial family models, within/pooling/random panel effects, three bandwidth search strategies (grid, Stochastic Gradient Descent, random), five kernel functions, and optional parallel execution via the 'future' framework. Diagnostic tools include spatial Moran's I, local F-test, Hausman test, and Lagrange Multiplier test.

License AGPL (>= 3)

Encoding UTF-8

LazyData true

Imports fixest, glmmTMB, lmtest, plm, sf, stats, utils

Depends R (>= 3.5.0)

Suggests future, future.apply, rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

URL https://github.com/MichaelChaoLi-cpu/GWPR.light

BugReports https://github.com/MichaelChaoLi-cpu/GWPR.light/issues

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Chao Li [aut, cre] (ORCID: <https://orcid.org/0000-0002-6854-4456>), Shunsuke Managi [aut] (ORCID: <https://orcid.org/0000-0001-7883-1427>)

Maintainer Chao Li <chaoli0394@gmail.com>

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

Date/Publication 2026-05-29 12:29:55 UTC

RemoteUrl https://github.com/cran/GWPR.light

RemoteRef HEAD

RemoteSha e1be29714d8cfed19dfcf3e9e668ba4c3439d349

Contents

`api` *Public API for GWPR.light 1.0.0*

Description

High-level user-facing functions for Geographically Weighted Panel Regression. These four functions form the complete public interface; all internal complexity is hidden behind them.

- `gwpr` – full pipeline (bandwidth search + fitting + optional diagnostics).
- `select_bandwidth` – standalone bandwidth search.
- `fit_gwpr` – fit with a known bandwidth.
- `diagnose_gwpr` – run diagnostics on a fitted model.

California *California (sf)*

Description

The counties' boundary in California

Usage

```
data(California)
```

Format

An sf object with 58 rows (one per county) and two columns:

GEOID a numeric vector, fips IDs of the counties

geometry sfc_MULTIPOLYGON, county boundary polygons (CRS: NAD83 longlat)

Author(s)

Chao Li <chaoli0394@gmail.com> Shunsuke Managi <managi.s@gmail.com>

Examples

```
data(California)
class(California)
```

diagnose_gwpr *Run diagnostic tests on a fitted GWPR model*

Description

Top-level interface that dispatches to individual diagnostic sub-functions. Returns a 'gwpr_diagnostics' object containing all requested test results.

Usage

```
diagnose_gwpr(
  object,
  diagnostics = c("moran", "f_test", "hausman", "lm_test"),
  spatial_weights = NULL,
  panel_index = NULL,
  ...
)
```

Arguments

object	A ‘gwpr_fit‘ object returned by ‘fit_gwpr()‘ or similar.
diagnostics	Character vector naming the tests to run. Any subset of ‘c("moran", "f_test", "hausman", "lm_test")‘. Default is all four.
spatial_weights	Required when ‘"moran"‘ is in ‘diagnostics‘. A row-standardised n x n spatial weights matrix.
panel_index	Required when ‘"moran"‘ is in ‘diagnostics‘. A data.frame with columns ‘id‘ and ‘time‘ identifying each element of ‘object$residuals‘.
...	Additional arguments passed to individual diagnostic functions.

Details

Tests that are not applicable to the fitted model (e.g., Hausman test on a pooling model) return a list with ‘status = "not_applicable"‘ and an explanatory ‘message‘, rather than an error.

Value

A ‘gwpr_diagnostics‘ object (list with class ‘"gwpr_diagnostics"‘) whose ‘diagnostics‘ slot contains the result of each requested test.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
  y    = rnorm(20),
  x1   = rnorm(20)
)
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
diag_result <- diagnose_gwpr(fit, diagnostics = c("f_test", "hausman"))
print(diag_result)
```

---

diagnose_hausman *Local Hausman test diagnostic on a gwpr_fit object*

Description

Performs a local Hausman test (within vs. random) for each spatial unit using test statistics pre-computed during model fitting or stored in ‘local_results‘.

Usage

```
diagnose_hausman(object, ...)
```

Arguments

`object`	A ‘gwpr_fit‘ object.
`...`	Currently ignored.

Details

Applicable models: gaussian with ‘model = "random"‘. For pooling models the Hausman test is not meaningful; the function returns a ‘status = "not_applicable"‘ result. For logistic models the function also returns ‘status = "not_applicable"‘.

Panel balance requirement: No constraint at the unit level.

Failure conditions: Returns ‘status = "missing_hausman_data"‘ for any unit where the required statistics are absent from ‘local_results‘.

Logistic interpretation limit: Not applicable.

Value

A named list with elements:

‘local_hausman‘ Data frame with columns ‘unit_id‘, ‘statistic‘, ‘p_value‘, ‘df‘, ‘status‘.

‘n_tested‘ Number of units tested.

‘n_failed‘ Number of units where the test could not be computed.

‘status‘ Overall status: ‘"ok"‘, ‘"not_applicable"‘, or ‘"no_local_results"‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
  y    = rnorm(20),
  x1   = rnorm(20)
)
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
diagnose_hausman(fit)
```

`diagnose_lm` *Local Breusch-Pagan LM test diagnostic on a gwpr_fit object*

Description

Performs a local Breusch-Pagan Lagrange Multiplier test for random effects for each spatial unit using test statistics stored in ‘local_results‘.

Usage

```
diagnose_lm(object, ...)
```

Arguments

`object`	A ‘gwpr_fit‘ object.
`...`	Currently ignored.

Details

Applicable models: gaussian with ‘model = "pooling"‘ or ‘model = "random"‘. For ‘within‘ models the test is not directly applicable (it tests for random effects vs. OLS); the function returns ‘status = "not_applicable"‘. For logistic models also not applicable.

Panel balance requirement: No constraint at the unit level.

Failure conditions: Returns ‘status = "missing_lm_data"‘ for units missing the required statistics.

Logistic interpretation limit: Not applicable.

Value

A named list with elements:

‘local_lm‘ Data frame with columns ‘unit_id‘, ‘statistic‘, ‘p_value‘, ‘df‘, ‘status‘.

‘n_tested‘ Number of units tested.

‘n_failed‘ Number of units where the test could not be computed.

‘status‘ Overall status.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
```

```
  y    = rnorm(20),
  x1   = rnorm(20)
)
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
diagnose_lm(fit)
```

`diagnose_local_f` *Local F test diagnostic on a gwpr_fit object*

Description

Performs a local F test (fixed effects vs. pooling) using per-unit local residuals stored in the fitted model object.

Usage

```
diagnose_local_f(object, ...)
```

Arguments

`object`	A ‘gwpr_fit‘ object.
`...`	Currently ignored.

Details

Applicable models: gaussian (linear). Not applicable to logistic models; returns a ‘status = "not_applicable"‘ result when ‘family = "binomial"‘.

Panel balance requirement: No constraint; the test uses per-unit local residuals already computed during fitting.

Failure conditions: If ‘local_results‘ is empty or missing, all units are reported as failed. If a unit’s local result does not contain the information needed (within and pooling residuals), that unit is reported as failed with an informative ‘status‘.

Logistic interpretation limit: Not applicable; see above.

Value

A named list with elements:

‘local_f‘ Data frame with columns ‘unit_id‘, ‘statistic‘, ‘p_value‘, ‘df1‘, ‘df2‘, ‘status‘.

‘n_tested‘ Number of units tested.

‘n_failed‘ Number of units where the test could not be computed.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
  y    = rnorm(20),
  x1   = rnorm(20)
)
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
diagnose_local_f(fit)
```

`diagnose_moran` *Run Moran's I diagnostic on a gwpr_fit object*

Description

Extracts residuals from a fitted GWPR model (Pearson residuals for logistic models, raw residuals for linear models) and computes the panel Moran's I statistic.

Usage

```
diagnose_moran(object, spatial_weights, panel_index, ...)
```

Arguments

`object`	A 'gwpr_fit' object returned by 'fit_gwpr()' or similar.
`spatial_weights`	A row-standardised n x n spatial weights matrix. 'n' must equal the number of spatial individuals in the fitted model.
`panel_index`	A data.frame or list with columns/elements 'id' and 'time' that identify each element of 'object$residuals'.
`...`	Currently ignored.

Details

Applicable models: gaussian (linear residuals) and binomial (Pearson residuals).

Panel balance: See 'compute_panel_moran()'.

Failure conditions: Fails if 'object' is not a 'gwpr_fit', if 'object$residuals' is 'NULL', or if 'spatial_weights' dimensions do not match the number of individuals.

Logistic interpretation limit: Moran's I computed on Pearson residuals is exploratory; the asymptotic distribution differs from the linear case.

Value

A named list compatible with the ‘diagnostics‘ slot of a ‘gwpr_diagnostics‘ object. Contains the elements returned by ‘compute_panel_moran()‘ plus ‘residual_type‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
  y    = rnorm(20),
  x1   = rnorm(20)
)
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
W <- matrix(1/3, nrow = 4, ncol = 4); diag(W) <- 0
idx <- dat[, c("id", "time")]
diagnose_moran(fit, W, idx)
```

`diagnostics` *Diagnostics Module for GWPR.light 1.0.0*

Description

Unified diagnostic interface for Geographically Weighted Panel Regression models. Provides Moran's I (spatial autocorrelation), local F test (fixed vs. pooling), local Hausman test (fixed vs. random), and local Breusch-Pagan LM test.

Details

Model applicability

| Diagnostic | Linear | Logistic | Notes | |————|——–|———-|————————————| | moran | yes | yes | Logistic uses Pearson residual | | f_test | yes | no | Requires within and pooling models | | hausman | yes | no | Only meaningful for random-effect models | | lm_test | yes | no | Pooling or random-effect models |

Panel balance

‘compute_panel_moran()‘ is fully supported for balanced panels. For unbalanced panels a ‘warning()‘ is issued and the function attempts computation using only the time periods present in every individual; results may be unreliable.

Logistic interpretation

Moran's I computed from Pearson residuals of a Logistic model does not follow the same asymptotic distribution as for linear models. Treat the test result as an exploratory heuristic, not a formal test.

`fit_gwpr` *Fit GWPR with a given bandwidth*

Description

Validates inputs, prepares data, builds spatial weights, and fits the Geographically Weighted Panel Regression for the specified bandwidth. Returns a `gwpr_fit` object.

Usage

```
fit_gwpr(
  formula,
  data,
  spatial,
  id,
  time,
  bandwidth,
  family = c("gaussian", "binomial"),
  model = c("within", "pooling", "random"),
  effect = c("individual", "time", "two-way", "nested"),
  kernel = c("bisquare", "gaussian", "exponential", "tricube", "boxcar"),
  adaptive = FALSE,
  threshold = 0.5,
  workers = 1L,
  seed = NULL,
  ...
)
```

Arguments

`formula`	A `formula` object.
`data`	A `data.frame` with panel data.
`spatial`	An `sf` object.
`id`	Character scalar; unit ID column name.
`time`	Character scalar; time column name.
`bandwidth`	Numeric scalar. The bandwidth to use (fixed distance or number of neighbours when `adaptive = TRUE`).
`family`	`"gaussian"` (default) or `"binomial"`.
`model`	`"within"` (default), `"pooling"`, or `"random"`.
`effect`	`"individual"` (default), `"time"`, `"two-way"`, or `"nested"`.

kernel	Kernel function name (default "bisquare").
adaptive	Logical; FALSE (default) for fixed bandwidth.
threshold	Numeric; classification threshold (binomial only, default 0.5).
workers	Positive integer; number of parallel workers (default 1).
seed	Integer random seed, or NULL.
...	Currently unused.

Value

A gwpr_fit object.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
  y    = rnorm(20),
  x1   = rnorm(20)
)
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
print(fit)
```

---

gwpr *Fit a Geographically Weighted Panel Regression (main entry point)*

Description

Orchestrates the complete GWPR pipeline: input validation, data preparation, optional memory estimation, optional bandwidth search, model fitting, and optional diagnostics.

Usage

```
gwpr(
  formula,
  data,
  spatial,
  id,
  time,
  family = c("gaussian", "binomial"),
```

```
  model = c("within", "pooling", "random"),
  effect = c("individual", "time", "two-way", "nested"),
  bandwidth = NULL,
  bandwidth_method = c("sgd", "grid", "random"),
  bandwidth_control = list(),
  kernel = c("bisquare", "gaussian", "exponential", "tricube", "boxcar"),
  adaptive = FALSE,
  threshold = 0.5,
  workers = 1L,
  seed = NULL,
  diagnostics = TRUE,
  ...
)
```

Arguments

formula	A formula object specifying the model (e.g. y ~ x1 + x2).
data	A data.frame containing the panel data. Must include the columns referenced by id and time.
spatial	An sf object with one row per spatial unit. Must include the column referenced by id.
id	Character scalar. Name of the unit (individual) ID column shared by data and spatial.
time	Character scalar. Name of the time-period column in data.
family	Character scalar. Model family: "gaussian" (default, linear GWPR) or "binomial" (binary panel logistic GWPR).
model	Character scalar. Panel model type: "within" (default), "pooling", or "random".
effect	Character scalar. Panel effect: "individual" (default), "time", "two-way", or "nested".
bandwidth	Numeric scalar or NULL (default). When NULL the bandwidth is selected automatically via select_bandwidth().
bandwidth_method	Character scalar. Method for automatic bandwidth search: "sgd" (default), "grid", or "random". Ignored when bandwidth is supplied.
bandwidth_control	Named list of control parameters passed to the bandwidth search function. For "grid": lower, upper, step. For "sgd": lower, upper, learning rate etc. For "random": lower, upper, n_samples.
kernel	Character scalar. Kernel function: "bisquare" (default), "gaussian", "exponential", "tricube", or "boxcar".
adaptive	Logical scalar. FALSE (default) uses a fixed distance bandwidth; TRUE uses an adaptive (k-nearest-neighbour) bandwidth.
threshold	Numeric scalar. Classification threshold for family = "binomial" (default 0.5).
workers	Positive integer. Number of parallel workers. 1 (default) uses serial execution; values > 1 enable explicit parallelism.

seed	Integer or NULL. Random seed for reproducibility of any stochastic steps (bandwidth search, parallel RNG).
diagnostics	Logical scalar. When TRUE (default), diagnose_gwpr() is called after fitting and its results are stored in the returned object.
...	Additional arguments passed to the bandwidth search or fitting functions.

Value

A gwpr_fit object. Key fields:

local_results Per-unit local model results.

predictions In-sample predicted values / probabilities.

residuals Residuals or Pearson residuals.

metrics Overall goodness-of-fit metrics.

spatial_results Data frame of per-unit coefficients.

search Bandwidth search result (gwpr_bandwidth), or NULL when bandwidth was supplied directly.

diagnostics A gwpr_diagnostics object, or NULL.

Examples

```
# Minimal linear GWPR with a fixed bandwidth
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
  y    = rnorm(20),
  x1   = rnorm(20)
)
fit <- gwpr(y ~ x1, data = dat, spatial = pts, id = "id", time = "time",
            bandwidth = 2, diagnostics = FALSE, workers = 1)
print(fit)
```

---

metrics *Metrics Module*

Description

Functions for computing evaluation metrics for linear and logistic panel regression models.

parallel_module *Parallel Execution Module*

Description

Unified parallel execution interface for bandwidth search, local model fitting, and diagnostics. Shields backend differences and ensures CRAN-friendly behaviour.

print.gwpr_bandwidth *Print a gwpr_bandwidth object*

Description

Displays the search method, best bandwidth, criterion score, and number of iterations explored.

Usage

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

Arguments

x	A ‘gwpr_bandwidth‘ object.
...	Currently ignored.

Value

Invisibly returns ‘x‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(id = rep(1:4, each = 5), time = rep(1:5, 4),
                  y = rnorm(20), x1 = rnorm(20))
bw <- select_bandwidth(y ~ x1, data = dat, spatial = pts, id = "id",
  time = "time", method = "grid",
  control = list(lower = 1, upper = 3, step = 1), workers = 1)
print(bw)
```

print.gwpr_diagnostics *Print a gwpr_diagnostics object*

Description

Displays each diagnostic test name and, where available, its statistic and p-value.

Usage

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

Arguments

x	A ‘gwpr_diagnostics‘ object.
...	Currently ignored.

Value

Invisibly returns ‘x‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(id = rep(1:4, each = 5), time = rep(1:5, 4),
                  y = rnorm(20), x1 = rnorm(20))
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
diag_obj <- diagnose_gwpr(fit, diagnostics = c("f_test", "hausman"))
print(diag_obj)
```

print.gwpr_fit *Print a gwpr_fit object*

Description

Displays a concise summary of a fitted GWPR model: family, panel model type, effect, bandwidth, and top-level goodness-of-fit metrics.

Usage

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

Arguments

x	A ‘gwpr_fit‘ object.
...	Currently ignored.

Value

Invisibly returns ‘x‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(id = rep(1:4, each = 5), time = rep(1:5, 4),
                  y = rnorm(20), x1 = rnorm(20))
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
print(fit)
```

---

result *Result Object Module for GWPR.light 1.0.0*

Description

S3 classes and constructor functions for the three result objects returned by the GWPR.light public API: ‘gwpr_fit‘, ‘gwpr_bandwidth‘, and ‘gwpr_diagnostics‘. Also provides ‘build_spatial_results()‘ for assembling a data.frame that can be aligned with an ‘sf‘ geometry column.

`select_bandwidth` *Select an optimal bandwidth for GWPR*

Description

Validates inputs, prepares data, and dispatches to the appropriate bandwidth search algorithm: grid search, stochastic gradient descent (sgd), or random search, depending on the `method` argument.

Usage

```
select_bandwidth(
  formula,
  data,
  spatial,
  id,
  time,
  family = c("gaussian", "binomial"),
  model = c("within", "pooling", "random"),
  effect = c("individual", "time", "two-way", "nested"),
  method = c("sgd", "grid", "random"),
  control = list(),
  kernel = c("bisquare", "gaussian", "exponential", "tricube", "boxcar"),
  adaptive = FALSE,
  threshold = 0.5,
  workers = 1L,
  seed = NULL,
  ...
)
```

Arguments

`formula`	A `formula` object.
`data`	A `data.frame` with panel data.
`spatial`	An `sf` object.
`id`	Character scalar; unit ID column name.
`time`	Character scalar; time column name.
`family`	`"gaussian"` or `"binomial"`.
`model`	`"within"`, `"pooling"`, or `"random"`.
`effect`	`"individual"`, `"time"`, `"two-way"`, or `"nested"`.
`method`	Bandwidth search method: `"sgd"` (default), `"grid"`, or `"random"`.
`control`	Named list of search control parameters.
`kernel`	Kernel function name.
`adaptive`	Logical; `TRUE` for adaptive (k-NN) bandwidth.

threshold	Numeric; classification threshold (binomial only).
workers	Positive integer; number of parallel workers.
seed	Integer random seed, or NULL.
...	Additional arguments (currently unused).

Value

A gwpr_bandwidth object with fields:

best_bandwidth The selected bandwidth value.

best_score The criterion value at the best bandwidth.

method The search method used.

history Search history data frame.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(
  id   = rep(1:4, each = 5),
  time = rep(1:5, 4),
  y    = rnorm(20),
  x1   = rnorm(20)
)
bw <- select_bandwidth(
  y ~ x1, data = dat, spatial = pts, id = "id", time = "time",
  method  = "grid",
  control = list(lower = 0.5, upper = 2, step = 0.5),
  workers = 1
)
bw$best_bandwidth
```

---

summary.gwpr_bandwidth

Summarise a gwpr_bandwidth object

Description

Prints the search method, best bandwidth, and a brief history overview.

Usage

```
## S3 method for class 'gwpr_bandwidth'
summary(object, ...)
```

Arguments

object	A ‘gwpr_bandwidth‘ object.
...	Currently ignored.

Value

Invisibly returns ‘object‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(id = rep(1:4, each = 5), time = rep(1:5, 4),
                  y = rnorm(20), x1 = rnorm(20))
bw <- select_bandwidth(y ~ x1, data = dat, spatial = pts, id = "id",
  time = "time", method = "grid",
  control = list(lower = 1, upper = 3, step = 1), workers = 1)
summary(bw)
```

---

summary.gwpr_diagnostics *Summarise a gwpr_diagnostics object*

Description

Prints each diagnostic test result with statistic and p-value where available.

Usage

```
## S3 method for class 'gwpr_diagnostics'
summary(object, ...)
```

Arguments

object	A ‘gwpr_diagnostics‘ object.
...	Currently ignored.

Value

Invisibly returns ‘object‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(id = rep(1:4, each = 5), time = rep(1:5, 4),
                  y = rnorm(20), x1 = rnorm(20))
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
diag_obj <- diagnose_gwpr(fit, diagnostics = c("f_test", "hausman"))
summary(diag_obj)
```

`summary.gwpr_fit` *Summarise a gwpr_fit object*

Description

Prints the global model overview, quantile summary of local coefficients (when available), and goodness-of-fit metrics.

Usage

```
## S3 method for class 'gwpr_fit'
summary(object, ...)
```

Arguments

`object`	A ‘gwpr_fit‘ object.
`...`	Currently ignored.

Value

Invisibly returns ‘object‘.

Examples

```
library(sf)
pts <- sf::st_as_sf(
  data.frame(id = 1:4, X = c(0,1,0,1), Y = c(0,0,1,1)),
  coords = c("X", "Y"), crs = NA_integer_
)
dat <- data.frame(id = rep(1:4, each = 5), time = rep(1:5, 4),
                  y = rnorm(20), x1 = rnorm(20))
fit <- fit_gwpr(y ~ x1, data = dat, spatial = pts, id = "id",
                time = "time", bandwidth = 2, workers = 1)
summary(fit)
```

`TransAirPolCalif` *Panel Dataset for Testing GWPR*

Description

Panel dataset to estimate the relationship between county-level PM2.5 concentration and on-road transporation in California.

Usage

```
data(TransAirPolCalif)
```

Format

A `data.frame` with 23 variables, and 928 observations, which are:

GEOID a numeric vector, fips IDs of the counties

year a numeric vector, year

pm25 a numeric vector, annually average PM2.5 concentration in the counties

co2_mean a numeric vector, geographically average CO2 emission from on-road transportation in each year, million tons/km2

Developed_Open_Space_perc a numeric vector, percentage of developed open space of total area in each county

Developed_Low_Intensity_perc a numeric vector, percentage of low-intensity developed area of total area in each county

Developed_Medium_Intensity_perc a numeric vector, percentage of medium-intensity developed area of total area in each county

Developed_High_Intensity_perc a numeric vector, percentage of high-intensity develope area of total area in each county

Open_Water_perc a numeric vector, percentage of open water of total area in each county

Woody_Wetlands_perc a numeric vector, percentage of woody wetland of total area in each county

Emergent_Herbaceous_Wetlands_perc a numeric vector, percentage of emergent herbaceous wetland of total area in each county

Deciduous_Forest_perc a numeric vector, percentage of deciduous forest of total area in each county

Evergreen_Forest_perc a numeric vector, percentage of evergreen forest of total area in each county

Mixed_Forest_perc a numeric vector, percentage of mixed forest of total area in each county

Shrub_perc a numeric vector, percentage of shrub of total area in each county

Grassland_perc a numeric vector, percentage of grassland of total area in each county

Pasture_perc a numeric vector, percentage of pasture of total area in each county

Cultivated_Crops_perc a numeric vector, percentage of cultivated crops of total area in each county

pop_density a numeric vector, average population density in each county

summer_tmmx a numeric vector, average temperature in summer

winter_tmmx a numeric vector, average temperature in winter

summer_rmax a numeric vector, average humidity in summer

winter_rmax a numeric vector, average humidity in winter

Author(s)

Chao Li <chaoli0394@gmail.com> Shunsuke Managi <managi.s@gmail.com>

Examples

```
data(TransAirPolCalif)
head(TransAirPolCalif)
```

`with_reproducible_seed` *Execute an expression with a reproducible seed*

Description

Sets `set.seed(seed)` before evaluating `expr`, then restores the prior RNG state so the caller's random stream is unaffected.

Usage

```
with_reproducible_seed(seed, expr)
```

Arguments

`seed`	Integer scalar. Seed value passed to `set.seed`.
`expr`	An R expression to evaluate.

Value

The value of `expr`.

Examples

```
r1 <- with_reproducible_seed(42, runif(3))
r2 <- with_reproducible_seed(42, runif(3))
stopifnot(identical(r1, r2))
```

Index