Package 'SpatialML'

July 6, 2026

Type Package

Title Spatial Machine Learning

Version 1.8.2

Date 2026-07-06

Description Implements a spatial extension of the random forest algorithm (Georganos et al. (2019) <doi:10.1080/10106049.2019.1595177>). Provides a Geographically Weighted Random Forest regression and a routine to find the optimal bandwidth (Georganos and Kalogirou (2022) <doi:10.3390/ijgi11090471>). A lightweight cross-validation helper for tuning the 'mtry' parameter of a random forest and a generator of synthetic spatial test data are also included. The package depends on 'ranger' as its single random-forest back-end.

License GPL (>= 2)

Encoding UTF-8

Language en-GB

LazyData true

Depends R (>= 4.1.0)

Imports ranger (>= 0.15.1), stats, graphics, utils

Suggests knitr, markdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://stamatisgeoai.eu/

NeedsCompilation no

Author Stamatis Kalogirou [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-1949-6248>), Stefanos Georganos [aut]

Maintainer Stamatis Kalogirou <stamatis.science@gmail.com>

Repository CRAN

Date/Publication 2026-07-06 09:40:09 UTC

Contents

`SpatialML-package` *Spatial Machine Learning: Geographically Weighted Random Forest*

Description

SpatialML implements a spatial extension of the Random Forest algorithm. The main functions are `grf` for fitting a Geographically Weighted Random Forest (GRF), `grf.bw` for selecting an optimal bandwidth, `predict.grf` (the S3 method dispatched by `predict()`) for generating spatial predictions, and `rf.mtry.optim` for tuning the global `mtry` parameter through cross-validation. The package also ships a small synthetic-data generator `random.test.data` and a real data set `Income`.

Details

GRF fits one Random Forest per observation in space, using a local neighbourhood defined by an *adaptive* kernel (the k nearest neighbours) or a *fixed* kernel (a Euclidean radius). Observations within the neighbourhood are optionally weighted by the bi-square kernel $w_{ij} = (1 - (d_{ij}/h)^2)^2$, where d_{ij} is the distance between observations i and j and h is the local bandwidth. The approach combines the flexibility and high predictive accuracy of Random Forests with a treatment of spatial non-stationarity inspired by Geographically Weighted Regression.

The package uses `ranger` as its random-forest back-end. Undefined local OOB predictions are handled with a quiet leave-one-out fallback. See `vignette("SpatialML")` for a complete walk-through of a `mtry` tuning -> bandwidth search -> GRF fit -> prediction workflow.

For tutorials, related publications and contact information visit the maintainer's website at `https://stamatisgeoai.eu/`.

Author(s)

Stamatis Kalogirou <stamatis.science@gmail.com> (maintainer, https://stamatisgeoai.eu/), Stefanos Georganos <stefanos.georganos@kau.se>.

References

Georganos, S., Grippa, T., Niang Gadiaga, A., Linard, C., Lennert, M., Vanhuysse, S., Mboga, N., Wolff, E., Kalogirou, S. (2019) Geographical Random Forests: A Spatial Extension of the Random Forest Algorithm to Address Spatial Heterogeneity in Remote Sensing and Population Modelling. *Geocarto International*, doi:10.1080/10106049.2019.1595177.

Georganos, S. and Kalogirou, S. (2022) A Forest of Forests: A Spatially Weighted and Computationally Efficient Formulation of Geographical Random Forests. *ISPRS International Journal of Geo-Information*, 11(9), 471. doi:10.3390/ijgi11090471.

See Also

grf, grf.bw, predict.grf, rf.mtry.optim, random.test.data, Income, ranger.

---

grf *Geographically Weighted Random Forest Model*

Description

Fits a local version of the Random Forest algorithm. The function fits one sub-model for every observation in the data set, where each local model is trained on the geographically nearest neighbours of the focal observation (adaptive kernel) or on every observation within a fixed distance (fixed kernel). Local observations may further be weighted with a bi-square kernel to reproduce a Geographically Weighted Random Forest.

Usage

```
grf(formula, dframe, bw, kernel = c("adaptive", "fixed"), coords,
    ntree = 500, mtry = NULL, importance = "impurity", nthreads = NULL,
    forests = TRUE, geo.weighted = TRUE, print.results = TRUE,
    progress = print.results, oob.fallback = c("loo", "inbag"), ...)
```

Arguments

formula	a formula (or character string coercible to one) specifying the local model to be fitted, using the syntax of ranger.
dframe	a data frame containing the dependent and independent variables that appear in formula.
bw	a positive number representing either the number of nearest neighbours (when kernel = "adaptive") or the bandwidth in the units of coords (when kernel = "fixed").
kernel	the type of kernel used to define the neighbourhood of each focal observation. One of "adaptive" (default) or "fixed".
coords	a numeric matrix or data frame with two columns giving the X and Y coordinates of the observations in dframe. The number of rows must match nrow(dframe).
ntree	integer; the number of trees to grow for each (global and local) random forest. Default is 500.
mtry	the number of variables randomly sampled as candidates at each split. The default is max(floor(p / 3), 1) where p is the number of predictors in formula.
importance	variable importance measure. Default is "impurity" (see ranger).

nthreads	number of threads used by ranger and by predict. The default (NULL) lets **ranger** pick a sensible value.
forests	logical; if TRUE (default) all local forests are stored in the returned object. Required if you intend to call predict.grf on new data.
geo.weighted	logical; if TRUE (default) the local random forests are fitted using bi-square geographical case weights. If FALSE, local forests are fitted on the same neighbourhood but without weighting (i.e. local but unweighted).
print.results	logical; if TRUE (default) the function prints global and local model summaries to the console.
progress	logical; if TRUE a text progress bar is shown while the local random forests are fitted (only in interactive sessions). Defaults to print.results.
oob.fallback	character; fallback used when **ranger** cannot produce an out-of-bag prediction for the focal observation. The default "loo" fits one additional local forest after removing the focal observation and predicts it out-of-sample. "inbag" uses the in-sample local prediction.
...	additional arguments passed to ranger.

Details

Geographically Weighted Random Forest (GRF) is a spatial analysis method that fits a local version of the Random Forest algorithm to investigate spatial non-stationarity in the relationship between a dependent variable and a set of predictors. A sub-model is fitted for every observation in space, considering only its neighbouring observations. The technique is inspired by Geographically Weighted Regression (Kalogirou, 2003) but replaces the linear local model by a non-linear ensemble that is robust in high dimensions and less prone to over-fitting thanks to its bootstrapping nature, which relaxes the distributional assumptions of classical Gaussian statistics.

For each focal observation i the function:

1. identifies the local neighbourhood of i according to kernel and bw;
2. computes bi-square geographical weights $w_{ij} = (1 - (d_{ij}/h)^2)^2$ where d_{ij} is the distance between observations i and j and h is the local bandwidth;
3. fits a **ranger** random forest on the local sub-sample (with case weights when geo.weighted = TRUE);
4. stores the local variable importance, the OOB and in-sample predictions for i, and (optionally) the entire local forest.

For very small neighbourhoods or highly concentrated case weights, **ranger** can occasionally have no out-of-bag trees for the focal observation. In that case the OOB prediction is undefined. The default oob.fallback = "loo" avoids noisy retry warnings by fitting one extra local forest with the focal row removed and using that out-of-sample prediction for the focal row.

The procedure is described in Georganos et al. (2019) and Georganos and Kalogirou (2022).

Value

A list of class "grf" with the following components:

Global.Model	A ranger object containing the global random forest fitted on dframe.

`Locations` The coords object passed to the function.

`Local.Variable.Importance`
A data frame with one row per observation and one column per predictor giving the local feature importance.

`LGofFit` A data frame with the observed values and local goodness-of-fit statistics: OOB and in-sample predicted values, residuals, mean squared error, R-squared and the number of attempts needed to obtain the OOB or fallback prediction (`LPerm`).

`Forests` When `forests = TRUE`, a list of length `nrow(dframe)` containing all local `ranger` forests.

`LocalModelSummary`
A list with the average local variable importance, mean squared error (OOB and predicted), R-squared (OOB and predicted), and AIC / AICc statistics.

Warning

Large data sets may take a long time to calibrate. A high number of observations combined with `forests = TRUE` can produce a very large output object.

Note

For tutorials, publications and contact information, see the package web site.

Author(s)

Stamatis Kalogirou <stamatis.science@gmail.com>, Stefanos Georganos <stefanos.georganos@kau.se>

References

Georganos, S., Grippa, T., Niang Gadiaga, A., Linard, C., Lennert, M., Vanhuysse, S., Mboga, N., Wolff, E., Kalogirou, S. (2019) Geographical Random Forests: A Spatial Extension of the Random Forest Algorithm to Address Spatial Heterogeneity in Remote Sensing and Population Modelling. *Geocarto International*, doi:10.1080/10106049.2019.1595177.

Georganos, S. and Kalogirou, S. (2022) A Forest of Forests: A Spatially Weighted and Computationally Efficient Formulation of Geographical Random Forests. *ISPRS International Journal of Geo-Information*, 11(9), 471. doi:10.3390/ijgi11090471.

Kalogirou, S. (2003) The Statistical Analysis and Modelling of Internal Migration Flows within England and Wales. PhD Thesis, School of Geography, Politics and Sociology, University of Newcastle upon Tyne.

See Also

`predict.grf`, `grf.bw`, `ranger`

Examples

```
set.seed(1)
RDF <- random.test.data(8, 8, 3)
Coords <- RDF[, 4:5]
m <- grf(dep ~ X1 + X2, dframe = RDF, bw = 12,
```

```
            kernel = "adaptive", coords = Coords,
            ntree = 100, mtry = 1, nthreads = 1,
            print.results = FALSE, progress = FALSE)

  ## Not run:
    ## Real-world demonstration on the Greek Income dataset.
    ## Not run by R CMD check because a full GRF on 325 municipalities
    ## with 500 trees per local fit takes longer than a CRAN example
    ## should.
    data(Income)
    Coords <- Income[, 1:2]
    m <- grf(Income01 ~ UnemrT01 + PrSect01, dframe = Income, bw = 60,
            kernel = "adaptive", coords = Coords)

## End(Not run)
```

`grf.bw` *Optimal Bandwidth Selection for a Geographically Weighted Random Forest*

Description

Searches an exhaustive grid of bandwidths and returns the one that maximises the out-of-bag (OOB) R-squared of the local model produced by `grf`.

Usage

```
grf.bw(formula, dataset, kernel = "adaptive", coords, bw.min = NULL,
       bw.max = NULL, step = 1, trees = 500, mtry = NULL,
       importance = "impurity", nthreads = 1, forests = FALSE,
       geo.weighted = TRUE, verbose = TRUE, ...)
```

Arguments

`formula`	a formula (or character string coercible to one) of the local model, as accepted by `ranger`.
`dataset`	a data frame containing the dependent and independent variables that appear in `formula`.
`kernel`	the kernel used in the regression. Either `"adaptive"` (default) or `"fixed"`.
`coords`	a numeric matrix or data frame with the X and Y coordinates of the observations in `dataset`.
`bw.min`	numeric; lower bound of the bandwidth search.
`bw.max`	numeric; upper bound of the bandwidth search.
`step`	numeric; the step size of the search. Default is `1`.
`trees`	integer; the number of trees to grow for each local random forest.

mtry	the number of variables randomly sampled as candidates at each split. Default is max(floor(p / 3), 1) where p is the number of predictors in formula. Values above p are capped at p.
importance	variable importance measure passed to ranger. Default is "impurity".
nthreads	number of threads passed to ranger and predict.
forests	logical; if TRUE all local forests are kept in memory. Default is FALSE (recommended for bandwidth search).
geo.weighted	logical; if TRUE fits a Geographically Weighted Random Forest using ranger's case-weight option.
verbose	logical; if TRUE (default) prints one progress message per evaluated bandwidth and the selected bandwidth.
...	additional arguments passed to grf and ranger.

Details

The function evaluates a sequence of candidate bandwidths from bw.min to bw.max in increments of step. For each bandwidth a Geographically Weighted Random Forest is fitted via grf and three goodness-of-fit values are recorded:

Local R-squared of the local model alone.

Mixed R-squared of the equally weighted average of local and global predictions.

Low.Local R-squared of 0.25 * local + 0.75 * global.

The optimal bandwidth is selected as the one that maximises the *Local* R-squared.

If you require fully reproducible bandwidth selection, call set.seed() before grf.bw().

Value

A list with two components:

tested.bandwidths	A data frame with one row per evaluated bandwidth and four columns: Bandwidth, Local, Mixed, Low.Local.
Best.BW	The bandwidth that yields the highest Local R-squared.

Warning

Bandwidth selection on large data sets is computationally expensive because a full grf is fitted at every bandwidth in the grid.

Note

Future versions may include heuristic search strategies (e.g. optim).

Author(s)

Stamatis Kalogirou <stamatis.science@gmail.com>, Stefanos Georganos <stefanos.georganos@kau.se>

References

Georganos, S., Grippa, T., Niang Gadiaga, A., Linard, C., Lennert, M., Vanhuysse, S., Mboga, N., Wolff, E., Kalogirou, S. (2019) Geographical Random Forests: A Spatial Extension of the Random Forest Algorithm to Address Spatial Heterogeneity in Remote Sensing and Population Modelling. *Geocarto International*, doi:10.1080/10106049.2019.1595177.

Georganos, S. and Kalogirou, S. (2022) A Forest of Forests: A Spatially Weighted and Computationally Efficient Formulation of Geographical Random Forests. *ISPRS International Journal of Geo-Information*, 11(9), 471. doi:10.3390/ijgi11090471.

See Also

grf, ranger

Examples

```
  set.seed(1)
  RDF <- random.test.data(8, 8, 3)
  Coords <- RDF[, 4:5]
  bw.test <- grf.bw(dep ~ X1 + X2, dataset = RDF, kernel = "adaptive",
                    coords = Coords,
                    bw.min = 12, bw.max = 16, step = 2,
                    trees = 100, mtry = 1, nthreads = 1L,
                    forests = FALSE, geo.weighted = TRUE,
                    verbose = FALSE)
  bw.test$Best.BW

## Not run:
  ## Real-world demonstration on the Greek Income dataset.
  ## Not run by R CMD check because an exhaustive bandwidth search on
  ## 325 municipalities is intrinsically slow.
  data(Income)
  Coords <- Income[, 1:2]

  bwe <- grf.bw(Income01 ~ UnemrT01 + PrSect01, dataset = Income,
                kernel = "adaptive", coords = Coords,
                bw.min = 30, bw.max = 80, step = 5,
                forests = FALSE, geo.weighted = TRUE)

  m <- grf(Income01 ~ UnemrT01 + PrSect01, dframe = Income,
           bw = bwe$Best.BW, kernel = "adaptive", coords = Coords)

## End(Not run)
```

Income *Mean Household Income at the Local Authorities of Greece in 2011*

Description

Municipality centroids and socio-economic variables aggregated to the new local authority geography in Greece (Programme *Kallikratis*).

Usage

```
data("Income")
```

Format

A data frame with 325 observations on the following 6 variables:

`X` numeric; X coordinate of the municipality centroid.

`Y` numeric; Y coordinate of the municipality centroid.

`UnemrT01` numeric; total unemployment rate in 2001 (Census).

`PrSect01` numeric; proportion of the economically active population working in the primary sector (mainly agriculture, fishery and forestry) in 2001 (Census).

`Foreig01` numeric; proportion of inhabitants without Greek citizenship in 2001 (Census).

`Income01` numeric; mean household income, in Euros, earned in 2001 and declared in 2002 tax forms.

Details

The X, Y coordinates refer to the geometric centroids of the 325 municipalities in Greece established by the *Kallikratis* reform in 2011.

Source

The original shapefile of the corresponding polygons was published by the Hellenic Statistical Authority (EL.STAT.) at https://www.statistics.gr/. The population, employment, citizenship and employment-sector data were also published by EL.STAT.; the income data were obtained from the General Secretariat of Information Systems in Greece at the postcode level. All data were aggregated to the new municipalities by the author.

References

Kalogirou, S. and Hatzichristos, T. (2007) A spatial modelling framework for income estimation. *Spatial Economic Analysis*, 2(3), 297-316. doi:10.1080/17421770701576921.

Kalogirou, S. (2010) Spatial inequalities in income and post-graduate educational attainment in Greece. *Journal of Maps*, 6(1), 393-400. doi:10.4113/jom.2010.1095.

Examples

```
data(Income)
boxplot(Income$Income01)
hist(Income$PrSect01)
```

predict.grf *Predict Method for a Geographically Weighted Random Forest*

Description

Generates predictions for new observations using a fitted grf object. For each new observation the local random forest fitted at the geographically nearest training location is used. The user can blend local and global predictions through the local.w and global.w weights.

Usage

```
## S3 method for class 'grf'
predict(object, new.data, x.var.name, y.var.name,
                      local.w = 1, global.w = 0, ...)
```

Arguments

object	an object of class "grf" created by grf with forests = TRUE.
new.data	a data frame containing the predictors required by the model formula and the X and Y coordinates of the new observations.
x.var.name	the name of the column in new.data that contains the X coordinate.
y.var.name	the name of the column in new.data that contains the Y coordinate.
local.w	numeric weight applied to the local model prediction. Default is 1.
global.w	numeric weight applied to the global model prediction. Default is 0.
...	further arguments passed to the ranger predict method (e.g. num.threads).

Details

For every row i in new.data the function computes the Euclidean distance between (new.data[i, x.var.name], new.data[i, y.var.name]) and every training location in object$Locations, picks the local random forest fitted at the nearest training location, and combines its prediction with the prediction of the global random forest using the weights local.w and global.w.

Value

A numeric vector of predictions of length nrow(new.data).

Note

For tutorials, publications and contact information, see the package web site.

Author(s)

Stamatis Kalogirou <stamatis.science@gmail.com>, Stefanos Georganos <stefanos.georganos@kau.se>

References

Georganos, S., Grippa, T., Niang Gadiaga, A., Linard, C., Lennert, M., Vanhuysse, S., Mboga, N., Wolff, E., Kalogirou, S. (2019) Geographical Random Forests: A Spatial Extension of the Random Forest Algorithm to Address Spatial Heterogeneity in Remote Sensing and Population Modelling. *Geocarto International*, doi:10.1080/10106049.2019.1595177.

See Also

grf

Examples

```
set.seed(1)
RDF <- random.test.data(8, 8, 3)
Coords <- RDF[, 4:5]
m <- grf(dep ~ X1 + X2, dframe = RDF, bw = 12,
         kernel = "adaptive", coords = Coords,
         ntree = 100, mtry = 1, nthreads = 1,
         print.results = FALSE, progress = FALSE)

set.seed(2)
RDF.Test <- random.test.data(2, 2, 3)
predict(m, RDF.Test, x.var.name = "X", y.var.name = "Y",
        local.w = 1, global.w = 0)

## Not run:
  ## Real-world demonstration on the Greek Income dataset.
  ## Not run by R CMD check because fitting a full GRF on 325
  ## municipalities exceeds the few-second example budget.
  data(Income)
  Coords <- Income[, 1:2]

  m <- grf(Income01 ~ UnemrT01 + PrSect01, dframe = Income, bw = 60,
           kernel = "adaptive", coords = Coords)

  set.seed(123)
  x <- runif(20, min = 142498, max = 1001578)
  y <- runif(20, min = 3855768, max = 4606754)
  u <- runif(20, min = 5,      max = 50)
  p <- runif(20, min = 0,      max = 100)
  f <- runif(20, min = 2,      max = 30)
  df2 <- data.frame(X = x, Y = y, UnemrT01 = u, PrSect01 = p, Foreig01 = f)

  predict(m, df2, x.var.name = "X", y.var.name = "Y",
          local.w = 1, global.w = 0)

## End(Not run)
```

`random.test.data` *Random Data Generator*

Description

Generates a synthetic data set composed of one dependent variable, a number of independent variables drawn from `Uniform(0, 1)`, and optionally the X and Y coordinates of a regular `nrows` x `ncols` grid. Useful for examples, unit testing and small simulation studies.

Usage

```
random.test.data(nrows = 10, ncols = 10, vars.no = 3,
                 dep.var.dis = c("normal", "poisson"),
                 xycoords = TRUE)
```

Arguments

`nrows`	integer; number of rows of the regular grid. Default `10`.
`ncols`	integer; number of columns of the regular grid. Default `10`.
`vars.no`	integer; total number of model variables (1 dependent plus `vars.no - 1` independent). Must be at least `2`.
`dep.var.dis`	distribution of the dependent variable. One of `"normal"` (default, drawn from `rnorm()`) or `"poisson"` (drawn from `rpois()` with `lambda = 7`).
`xycoords`	logical; if `TRUE` (default) two columns `X` and `Y` containing the grid coordinates are appended.

Details

The function is mostly intended to provide reproducible toy data for the package examples. The independent variables are named `X1, X2, ..., X(vars.no - 1)`.

For reproducible results call `set.seed()` before `random.test.data()`.

Value

A data frame with `nrows * ncols` rows containing one column `dep`, `vars.no - 1` columns named `X1, X2, ...` and, when `xycoords = TRUE`, two extra columns `X` and `Y`.

Author(s)

Stamatis Kalogirou <stamatis.science@gmail.com>

Examples

```
set.seed(1)
RDF <- random.test.data(12, 12, 3)
head(RDF)
```

`rf.mtry.optim` *Optimal mtry for a Random Forest via OOB or Cross-Validation*

Description

Searches for the value of `mtry` that minimises the predictive error of a Random Forest on a user-supplied grid. Three evaluation strategies are available:

- `"oob"` (default): one `ranger` fit per `mtry`; OOB error is the criterion (fast).
- `"cv"`: a single `cv.folds`-fold cross-validation is performed for each `mtry`.
- `"repeatedcv"`: `cv.repeats` repeats of `cv.folds`-fold CV are performed for each `mtry`.

The selected value can then be passed to the `mtry` argument of `grf`.

Usage

```
rf.mtry.optim(formula, dataset, min.mtry = NULL, max.mtry = NULL,
              mtry.step = 1, num.trees = 500,
              cv.method = c("oob", "repeatedcv", "cv"),
              cv.folds = 10, cv.repeats = 5, num.threads = NULL,
              plot.it = TRUE, verbose = TRUE, ...)
```

Arguments

`formula`	a model formula (or a character string coercible to one).
`dataset`	a data frame containing the variables of `formula`.
`min.mtry`	integer; lower bound of the `mtry` grid. Default `1`.
`max.mtry`	integer; upper bound of the `mtry` grid. Default is the number of predictors in `formula`.
`mtry.step`	integer; step of the `mtry` grid. Default `1`.
`num.trees`	integer; number of trees in each random forest fit. Default `500`.
`cv.method`	character; evaluation strategy. One of `"oob"` (default), `"repeatedcv"` or `"cv"`.
`cv.folds`	integer; number of folds when `cv.method != "oob"`. Default `10`.
`cv.repeats`	integer; number of repeats when `cv.method == "repeatedcv"`. Default `5`.
`num.threads`	number of threads passed to `ranger`. The default (`NULL`) lets **ranger** pick a sensible value.
`plot.it`	logical; if `TRUE` (default) the `mtry` vs. RMSE curve is plotted.
`verbose`	logical; if `TRUE` (default) progress messages are printed for every evaluated `mtry`.
`...`	additional arguments forwarded to `ranger` (e.g. `importance = "impurity"`).

Details

The criterion to minimise is the average RMSE across folds (or the OOB RMSE in the `"oob"` case). The average squared correlation between predictions and observations is also recorded as the `Rsquared` column.

For reproducible results call `set.seed()` before `rf.mtry.optim()`.

Compatibility note: the function no longer returns a `caret` `train` object. The dependency on **caret** (and indirectly on **randomForest**) has been removed and the cross-validation is now implemented directly on top of `ranger`. Code that read `$bestTune$mtry` from the previous return value should now read `$best.mtry`.

Value

A list with the following components:

`best.mtry`	The `mtry` value with the lowest average RMSE.
`results`	A data frame with one row per evaluated `mtry` and the columns `mtry`, `RMSE`, `Rsquared`, `SDRMSE` and `SDRsq`. Standard deviations are `NA` for `cv.method == "oob"`.
`cv.method`	The evaluation strategy used.
`num.trees`	Number of trees used in each fit.
`call`	The matched call.

Author(s)

Stamatis Kalogirou <stamatis.science@gmail.com>, Stefanos Georganos <stefanos.georganos@kau.se>

References

Wright, M. N. and Ziegler, A. (2017) ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77(1), 1-17. doi:10.18637/jss.v077.i01.

Georganos, S. and Kalogirou, S. (2022) A Forest of Forests: A Spatially Weighted and Computationally Efficient Formulation of Geographical Random Forests. *ISPRS International Journal of Geo-Information*, 11(9), 471. doi:10.3390/ijgi11090471.

See Also

`grf`, `ranger`

Examples

```
data(Income)
set.seed(123)
res <- rf.mtry.optim(Income01 ~ UnemrT01 + PrSect01, dataset = Income,
                     num.trees = 200, num.threads = 1L,
                     cv.method = "oob", plot.it = FALSE,
                     verbose = FALSE)
res$best.mtry
```

```
res$results
```

Index