Package ‘autocart’

October 12, 2022

Type Package

Title Autocorrelation Regression Trees

Version 1.4.5

Maintainer Ethan Ancell <ethanancell@gmail.com>

Description A modified version of the classification and regression tree (CART) algorithm for modelling spatial data that features coordinate information. Coordinate information can be used to evaluate measures of spatial autocorrelation and spatial compactness during the splitting phase of the tree, leading to better predictions and more physically realistic predictions on these types of datasets. These methods are described in Ancell and Bean (2021) <arXiv:2101.08258>.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Depends fields, mgcv

Imports Rcpp, RcppParallel

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.1.1

VignetteBuilder knitr

SystemRequirements GNU make

NeedsCompilation yes

Author Ethan Ancell [aut, cre]

Repository CRAN

Date/Publication 2021-05-27 10:50:09 UTC

R topics documented:

autocart *Create an autocart model*

Description

Create an autocart model

Usage

```
autocart(response, data, locations, alpha, beta, control = NULL)
```

Arguments

`response`	A vector of numeric response values with no NA entries.
`data`	A dataframe for the predictor variables used in the autocart tree.
`locations`	A two-column matrix with coordinates for the observations the predictor dataframe.
`alpha`	A scalar value between 0 and 1 to weight autocorrelation against reduction in variance in the tree splitting. A value of 1 indicates full weighting on measures of autocorrelation.
`beta`	A scalar value between 0 and 1 to weight the shape of the region in the splitting
`control`	An object of type "autocartControl" returned by the `autocartControl` function to control the splitting in the autocart tree.

Value

An S3 object of class "autocart".

Examples

```
# Load some data for an autocart example
snow <- na.omit(read.csv(system.file("extdata", "ut2017_snow.csv", package = "autocart")))
y <- snow$yr50[1:40]
X <- data.frame(snow$ELEVATION, snow$MCMT, snow$PPTWT, snow$HUC)[1:40, ]
locations <- as.matrix(cbind(snow$LONGITUDE, snow$LATITUDE))[1:40, ]

# Create an autocart model with 50 trees
snow_model <- autocart(y, X, locations, 0.30, 0)
```

`autocartControl` *Create the object used for the controlling of the splits in the autocart model*

Description

Create the object used for the controlling of the splits in the autocart model

Usage

```
autocartControl(
  minsplit = 20,
  minbucket = round(minsplit/3),
  maxdepth = 30,
  maxobsMtxCalc = NULL,
  distpower = 1,
  islonglat = TRUE,
  givePredAsFactor = TRUE,
  retainCoords = TRUE,
  useGearyC = FALSE,
  runParallel = TRUE,
  spatialWeightsType = "default",
  customSpatialWeights = NULL,
  spatialBandwidthProportion = 1,
  spatialBandwidth = NULL,
  asForest = FALSE
)
```

Arguments

`minsplit`	The minimum observations in a node before a split is attempted
`minbucket`	The minimum number of observations in a terminal node.
`maxdepth`	Set the maximum depth in the final tree. A root node is counted as a height of 0.
`maxobsMtxCalc`	Optional maximum number of observations in a node where computationally intensive matrix calculations like autocorrelation and compactness are performed.
`distpower`	The power of inverse distance to use when calculating spatial weights matrix.
`islonglat`	Are the coordinates longitude and latitude coordinates? If TRUE, then use great circle distance calculations
`givePredAsFactor`	In the returned autocart model, should the prediction vector also be returned as a factor?
`retainCoords`	After creating the autocart model, should the coordinates for each of the predictions be kept in the returned model?
`useGearyC`	Should autocart use Geary's C instead of Moran's I in the splitting function?
`runParallel`	Logical value indicating whether autocart should run using parallel processing.

`spatialWeightsType`
: What type of spatial weighting should be used when calculating spatial autocorrelation? Options are "default" or "gaussian".

`customSpatialWeights`
: Use this parameter to pass in an optional spatial weights matrix for use in autocorrelation calculations. Must have nrow and ncol equal to rows in training dataframe.

`spatialBandwidthProportion`
: What percentage of the maximum pairwise distances should be considered the maximum distance for spatial influence? Cannot be simultaneously set with `spatialBandwidth`

`spatialBandwidth`
: What is the maximum distance where spatial influence can be assumed? Cannot be simultaneously set with `spatialBandwidthProportion`.

`asForest`
: A logical indicating if the tree should be created as a forest component with random subsetting of predictors at each node. Set this to true if you are using this tree inside an ensemble.

Value

An object passed in to the `autocart` function that controls the splitting.

Examples

```
# Load some data for an autocartControl example
snow <- na.omit(read.csv(system.file("extdata", "ut2017_snow.csv", package = "autocart")))
y <- snow$yr50[1:40]
X <- data.frame(snow$ELEVATION, snow$MCMT, snow$PPTWT)[1:40, ]
locations <- as.matrix(cbind(snow$LONGITUDE, snow$LATITUDE))[1:40, ]

# Create a control object that disallows the tree from having a depth more
# than 10 and give spatial weights only to observations that are a third of the
# distance of the longest distance between any two points in the dataset.
snow_control <- autocartControl(maxdepth = 10, spatialBandwidthProportion = 0.33)

# Pass the created control object to an autocart model
snow_model <- autocart(y, X, locations, 0.30, 0, snow_control)
```

`autoforest` *Create a forest of autocart trees..*

Description

Create a forest of autocart trees..

Usage

```
autoforest(
  response,
  data,
  locations,
  alpha,
  beta,
  control,
  numtrees,
  mtry = NULL
)
```

Arguments

`response`	The response vector that goes along with the dataframe of predictors.
`data`	The dataframe of predictors.
`locations`	A matrix of the locations of the dataframe of predictors.
`alpha`	The percentage of weighting on spatial autocorrelation in the splitting function.
`beta`	The percentage of weighting on spatial compactness in the splitting function.
`control`	A control object from the `autocartControl` function that will be used for each tree in the forest.
`numtrees`	The number of autocart trees to create in the forest.
`mtry`	The number of variables to subset at each node of the splitting in the trees. By default, this will be 1/3 of the features.

Value

An object of type "autoforest", which is a list of the autocart trees.

Examples

```
# Load some data for an autoforest example
snow <- na.omit(read.csv(system.file("extdata", "ut2017_snow.csv", package = "autocart")))
y <- snow$yr50[1:40]
X <- data.frame(snow$ELEVATION, snow$MCMT, snow$PPTWT, snow$HUC)[1:40, ]
locations <- as.matrix(cbind(snow$LONGITUDE, snow$LATITUDE))[1:40, ]

# Create a control object for the autoforest tree
snow_control <- autocartControl(spatialBandwidthProportion = 1.0)

# Create an autoforest model with 5 trees
snow_model <- autoforest(y, X, locations, 0.30, 0, snow_control, numtrees = 5)
```

autotune — *Find the best alpha, beta, and bandwidth values with k-fold cross-validation*

Description

Find the best alpha, beta, and bandwidth values with k-fold cross-validation

Usage

```
autotune(
  response,
  data,
  locations,
  k = 8,
  control = NULL,
  customGroups = NULL,
  alphaVals = NULL,
  betaVals = NULL,
  bandwidthVals = NULL,
  outputProgress = FALSE,
  useSpatialNodes = FALSE,
  spatialNodesMethod = "idw",
  spatialNodesDistPower = 2,
  spatialNodesDistPowerRange = c(0, 2),
  spatialNodesModelByResidual = FALSE
)
```

Arguments

`response`	The vector of response values to test on.
`data`	The data frame of predictor variables.
`locations`	The n by 2 matrix of coordinate information for the known observations
`k`	The number of folds to create in k-fold cross-validation for tuning
`control`	An optional control function to send to the autocart creation function
`customGroups`	Here, you may supply custom groups for cross-validation. This parameter must be supplied as a factor and labeled from 1:numLevels.
`alphaVals`	Override the alpha values that are expanded in the grid in this function.
`betaVals`	Override the beta values that are expanded in the grid in this function.
`bandwidthVals`	Override the bandwidth values that are expanded in the grid in this function.
`outputProgress`	Print the result of the cross-validations as you are going. This is useful when the cross-validation will be very long and you do not wish to wait.
`useSpatialNodes`	Use an interpolative process at the terminal nodes of the autocart tree for the prediction process

`spatialNodesMethod`
: The type of interpolation to use at the terminal nodes

`spatialNodesDistPower`
: The power parameter to use in inverse distance weighting at terminal nodes

`spatialNodesDistPowerRange`
: The ranged power parameter p1, p2 to use for a varying power parameter

`spatialNodesModelByResidual`
: Do the interpolative process on the residuals of the prediction formed by response average at terminal nodes

Value

A list of the labeled optimal parameters that were chosen for the best predictive accuracy on cross-validation.

Examples

```
# Load some data for an autotune example
# (Note that a low sample size is used here for quick example computation.
#  In a practical application this function can be quite computationally
#  demanding due to the grid-search nature of the function.)
snow <- na.omit(read.csv(system.file("extdata", "ut2017_snow.csv", package = "autocart")))
y <- snow$yr50[1:35]
X <- data.frame(snow$ELEVATION, snow$MCMT, snow$PPTWT)[1:35, ]
locations <- as.matrix(cbind(snow$LONGITUDE, snow$LATITUDE))[1:35, ]

# Find optimal parameters via cross-validation. We'll search through the
# following alpha/beta/bandwidth values:
alphaVec <- c(0.0, 0.5)
betaVec <- c(0.0, 0.2)
bandwidthVec <- c(1.0)

# We'll find the optimal values with 3-fold cross validation:
# (Due to the large number of cross-validations and trainings that occur,
# this can take a few minutes.)
myTune <- autotune(y, X, locations, k = 3, alphaVals = alphaVec,
                   betaVals = betaVec, bandwidthVals = bandwidthVec)
# Inspect the results
myTune
```

---

predictAutocart *Given an autocart model object, predict for new data passed in*

Description

Given an autocart model object, predict for new data passed in

Usage

```
predictAutocart(autocartModel, newdata)
```

Arguments

`autocartModel`	An S3 object of type "autocart" returned from the autocart function
`newdata`	A dataframe with the same amount of columns used to create the autocart model.

Value

A numeric vector containing the predicted response value for each of the rows in the passed in dataframe.

Examples

```
# Load some data for an autocart predict example
snow <- na.omit(read.csv(system.file("extdata", "ut2017_snow.csv", package = "autocart")))
y <- snow$yr50[1:40]
X <- data.frame(snow$ELEVATION, snow$MCMT, snow$PPTWT, snow$HUC)[1:40, ]
locations <- as.matrix(cbind(snow$LONGITUDE, snow$LATITUDE))[1:40, ]

# Create an autocart model with 50 trees
snow_model <- autocart(y, X, locations, 0.30, 0)

# Predict in autocart
new_X <- X[1:10, ]
new_loc <- locations[1:10, ]
autocart_predictions <- predictAutocart(snow_model, new_X)
```

predictAutoforest — *Make a prediction using an autoforest model returned from the* `autoforest` *function.*

Description

Make a prediction using an autoforest model returned from the `autoforest` function.

Usage

```
predictAutoforest(
  autoforestModel,
  newdata,
  newdataCoords = NULL,
  useSpatialNodes = FALSE,
  method = "idw",
  distpower = 2,
  distpowerRange = c(0, 2),
  modelByResidual = TRUE,
  decideByGC = FALSE
)
```

Arguments

`autoforestModel`	An S3 object of type "autoforest" returned from the `autoforest` function.
`newdata`	The dataframe of predictors for use in prediction.
`newdataCoords`	the matrix of locations for all the information in newdata. Required argument if you set "useSpatialNodes" to TRUE.
`useSpatialNodes`	If TRUE, instead of running all the observations through the autocart tree, use the `spatialNodes` function to make predictions.
`method`	If using the spatial nodes type of prediction, then the type of interpolation to use. The options are "idw" and "tps".
`distpower`	If using "idw" for the method, the power on distance. For example, setting this to 2 would mean inverse squared distance squared weighting.
`distpowerRange`	If using "idw" for the interpolation method, the range of distance powers to use on inverse distance weighting matched to terminal node Moran I measurements.
`modelByResidual`	When using interpolation, make a prediction using the region of interest's average and then interpolate the residual.
`decideByGC`	Use Geary's C in deciding to induce a local spatial process rather than Moran's I.

Value

A vector of predictions that correspond to the rows in `newdata`.

Examples

```
# Load some data for an autoforest example
snow <- na.omit(read.csv(system.file("extdata", "ut2017_snow.csv", package = "autocart")))
y <- snow$yr50[1:40]
X <- data.frame(snow$ELEVATION, snow$MCMT, snow$PPTWT, snow$HUC)[1:40, ]
locations <- as.matrix(cbind(snow$LONGITUDE, snow$LATITUDE))[1:40, ]

# Create a control object for the autoforest tree
snow_control <- autocartControl(spatialBandwidthProportion = 1.0)

# Create an autoforest model with 5 trees (low number chosen for computation time)
snow_model <- autoforest(y, X, locations, 0.30, 0, snow_control, numtrees = 5)

# Predict for a subset of the data
new_X <- X[1:10, ]
new_loc <- locations[1:10, ]
predicted_values <- predictAutoforest(snow_model, new_X, new_loc, TRUE)
```

rmae *Relative mean absolute error*

Description

Relative mean absolute error

Usage

```
rmae(pred, obs, na.rm = TRUE)
```

Arguments

pred	The vector of predictions
obs	The actual observed values
na.rm	Should NA values be taken out of the vectors?

Value

The relative mean average error of the two vectors.

Examples

```
# Create two vectors, add some noise, and evaluate the RMAE.
firstVec <- 1:10
secondVec <- 1:10 + rnorm(10)
rmae(firstVec, secondVec)
```

spatialNodes *Using an autocart model, use the terminal nodes to form a spatial process that uses inverse distance weighting to interpolate. The prediction for the new data that is passed in is formed by making a prediction to assign it to a group. Next, the residual for the new prediction is formed by inverse distance weighting the residual for the other points that are a part of that geometry.*

Description

Using an autocart model, use the terminal nodes to form a spatial process that uses inverse distance weighting to interpolate. The prediction for the new data that is passed in is formed by making a prediction to assign it to a group. Next, the residual for the new prediction is formed by inverse distance weighting the residual for the other points that are a part of that geometry.

Usage

```
spatialNodes(
  autocartModel,
  newdata,
  newdataCoords,
  method = "idw",
  distpower = 2,
  distpowerRange = c(0, 2),
  modelByResidual = TRUE,
  decideByGC = FALSE
)
```

Arguments

`autocartModel`	an autocart model returned from the `autocart` function.
`newdata`	a dataframe that contains the same predictors that were used to form the tree.
`newdataCoords`	a matrix of coordinates for all the predictors contained in `newdata`
`method`	The type of interpolation to use. Options are "idw" for inverse distance weighting and "tps" for thin-plate splines.
`distpower`	the power to use if you would like to use something other than straight inverse distance, such as inverse distance squared.
`distpowerRange`	A range of distpower to use. This is an adaptive inverse distance weighting method that linearly matches measures of spatial autocorrelation measured by Moran I to the range mentioned in distpower.
`modelByResidual`	If true, then predict using the average of the "spatial node", and then model the residual using a spatial process. If false, fit a spatial process directly.
`decideByGC`	When determining if a spatial process should be ran at a terminal node, should we use the Geary C statistic instead of Moran I?

Value

a prediction for the observations that are represented by `newdata` and `newdataCoords`

Examples

```
# Load some data for a spatial nodes example
snow <- na.omit(read.csv(system.file("extdata", "ut2017_snow.csv", package = "autocart")))
y <- snow$yr50[1:40]
X <- data.frame(snow$ELEVATION, snow$MCMT, snow$PPTWT, snow$HUC)[1:40, ]
locations <- as.matrix(cbind(snow$LONGITUDE, snow$LATITUDE))[1:40, ]

# Create an autocart model
snow_model <- autocart(y, X, locations, 0.30, 0)

# Predit with the spatial node effect
new_X <- X[1:10, ]
new_loc <- locations[1:10, ]
```

```
spatial_node_predictions <- spatialNodes(snow_model, new_X, new_loc, distpowerRange = c(0, 2))
```

Index