Package 'AddiVortes'

September 17, 2026

Title (Bayesian) Additive Voronoi Tessellations

Version 1.0.1

Description Implements the Bayesian Additive Voronoi Tessellation model for non-parametric regression, classification and machine learning as introduced in Stone and Gosling (2025) <doi:10.1080/10618600.2024.2414104>. This package provides a flexible alternative to BART (Bayesian Additive Regression Trees) using Voronoi tessellations instead of trees. Users can fit Bayesian regression and probit classification models, estimate the associated posterior distributions and make predictions. It is particularly useful for spatial data analysis, machine learning, complex function approximation and Bayesian modelling where the underlying structure is unknown.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.0.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr, xml2

SystemRequirements C++20

Config/testthat/edition 3

URL https://johnpaulgosling.github.io/AddiVortes/

VignetteBuilder knitr

BugReports https://github.com/johnpaulgosling/AddiVortes/issues

LazyData true

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Adam Stone [aut] (ORCID: <https://orcid.org/0009-0004-0058-6117>), John Paul Gosling [aut, cre] (ORCID: <https://orcid.org/0000-0002-4072-3022>), Andrew Iskauskas [aut] (ORCID: <https://orcid.org/0000-0003-2825-3651>), Leo Thomson [aut]

Maintainer John Paul Gosling <john-paul.gosling@durham.ac.uk>

Repository CRAN

Date/Publication 2026-09-17 10:30:15 UTC

Contents

`AddiVortes-package` *AddiVortes: Bayesian Additive Voronoi Tessellations for Machine Learning*

Description

AddiVortes implements Bayesian Additive Voronoi Tessellation models for machine learning regression, classification and non-parametric statistical modelling. This package provides a flexible alternative to BART (Bayesian Additive Regression Trees), using Voronoi tessellations instead of trees for spatial partitioning. The method is particularly effective for spatial data analysis, complex function approximation, and Bayesian regression and classification.

Details

Key features include:

- Machine learning regression and classification with Bayesian inference
- Binary and multinomial probit models with latent-variable data augmentation
- Alternative to BART using Voronoi tessellations
- Spatial data analysis and modelling
- Non-parametric regression capabilities
- Complex function approximation
- Uncertainty quantification through posterior inference

Author(s)

Maintainer: John Paul Gosling <john-paul.gosling@durham.ac.uk> (ORCID)

Authors:

- John Paul Gosling <john-paul.gosling@durham.ac.uk> (ORCID)

- Adam Stone <adam.stone2@durham.ac.uk> (ORCID)
- Andrew Iskauskas <andrew.iskauskas@durham.ac.uk> (ORCID)
- Leo Thomson <leo@feasibly.co.uk>

References

Stone, A. and Gosling, J.P. (2025). AddiVortes: (Bayesian) additive Voronoi tessellations. Journal of Computational and Graphical Statistics.

Stone, A.J., Ogundimu, E. and Gosling, J.P. (2026). Binary AddiVortes: (Bayesian) Additive Voronoi Tessellations for Binary Classification with an application to Predicting Home Mortgage Application Outcomes.

Albert, J.H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. Journal of the American Statistical Association.

Kindo, B.P., Wang, H. and Peña, E.A. (2016). Multinomial probit Bayesian additive regression trees. Stat.

See Also

https://johnpaulgosling.github.io/AddiVortes/

`AddiVortes` *Fit an AddiVortes regression or classification model*

Description

The AddiVortes model is a Bayesian nonparametric model that uses additive Voronoi tessellations to relate covariates to a response. For a numeric response the model is Gaussian regression. For a classification response it uses a probit link with Albert-Chib latent variables (binary) or independent multinomial probit latents (three or more classes). The task is chosen automatically from `y`.

The function can handle multiple types of covariates, including continuous, spherical and categorical. Categorical covariates are automatically detected. By default (`cat.onehot = TRUE`) they are one-hot encoded, with the first level of each categorical variable used as the reference category; the `catScaling` parameter then controls the weight of categorical differences in distance calculations. Setting `cat.onehot = FALSE` instead keeps each categorical covariate as a single integer-coded column and uses Eskin distance (Eskin et al., 2002). For spherical covariates, the function assumes that the final spherical dimension corresponds to the polar angle, which has a range of 0 to 2*pi. The `metric` parameter can be used to specify the type of each covariate (Euclidean, Spherical, or Categorical), and the `members` parameter can indicate membership of covariates into different subspaces when using multiple spheres in covariate space.

Usage

```
AddiVortes(
  y,
  x,
  m = 200,
  totalMCMCIter = 2000,
  mcmcBurnIn = 500,
  nu = 6,
  q = 0.85,
  k = 3,
  sd = 0.8,
  Omega = min(3, ncol(x)),
  LambdaRate = 5,
  InitialSigma = "Linear",
  thinning = 1,
  metric = "E",
  members = NULL,
  catScaling = 1,
  cat.onehot = TRUE,
  showProgress = interactive()
)
```

Arguments

y	A vector of response values. Numeric y is treated as regression, except when it has exactly two unique values in {0, 1}, which is binary classification. Factor, character and logical vectors are treated as classification: two levels give a binary probit model and three or more levels give a multinomial probit model. The first factor level (or 0 for numeric 0/1 responses) is the reference class. Missing values are not allowed.
x	A matrix or data frame of the covariates. Character and factor columns are treated as categorical variables and automatically converted to d-1 binary indicator variables via one-hot encoding (with the first level as reference).
m	The number of tessellations. For multinomial classification this is the number of tessellations **per latent dimension** (there are $K-1$ latents for K classes).
totalMCMCIter	The number of MCMC iterations. Default 2000.
mcmcBurnIn	The number of burn-in iterations. Default 500.
nu	The degrees of freedom for the inverse-gamma prior on the residual variance. Ignored for classification, where the latent residual variance is fixed at 1.
q	The quantile used to set the inverse-gamma prior on the residual variance. Ignored for classification.
k	Prior scale for tessellation output values. For regression, $\sigma_\mu = 0.5/(k\sqrt{m})$ on the scaled response. For classification, $\sigma_\mu = 3/(k\sqrt{m})$ on the latent probit scale.
sd	The standard deviation used in centre proposals.
Omega	Omega/(number of covariates) is the prior probability of adding a dimension.

LambdaRate	The rate of the Poisson distribution for the number of centres.
InitialSigma	The method used to calculate the initial residual variance for regression ("Linear" or "Naive"). Ignored for classification.
thinning	The thinning rate.
metric	Either "E" (Euclidean, default), "S" (Spherical), or "C" (Categorical).
members	If needed, indicates membership of covariates into different subspaces (needed if using multiple spheres in covariate space). Default NULL.
catScaling	Numeric scalar controlling the scale of binary indicator variables created from categorical covariates. Each binary indicator takes values 0 (reference level) or catScaling (non-reference level). The default value of 1 matches the range of continuous covariates, which are normalised to [-0.5, 0.5] (range = 1) during fitting, so categorical differences receive comparable weight to continuous differences in the distance calculations. Increase above 1 to give categorical differences more weight; decrease below 1 to give them less weight. Binary indicator columns are named <colname>_<level> (e.g. a column grp with levels "A", "B", "C" produces columns grp_B and grp_C, with "A" as the reference level).
cat.onehot	Should categorical covariates be one-hot encoded? Default TRUE. When TRUE, each categorical covariate with *d* levels is expanded to *d* - 1 binary indicators and distances are Euclidean (weighted by catScaling). When FALSE, categories are kept as a single integer-coded column and mismatches use Eskin distance (Eskin et al., 2002), with squared cost $2/d^2$ when levels differ and 0 when they match; catScaling is then ignored. See the categorical covariates vignette for a comparison and guidance on which to use.
showProgress	Logical; if TRUE, a progress bar is shown during fitting.

Value

An AddiVortes object containing the posterior samples of the tessellations, dimensions and predictions, plus per-iteration trace statistics used by traceplots(). Classification fits also store task, classLevels, nLatents and in-sample accuracy.

References

Stone, A. and Gosling, J.P. (2025). AddiVortes: (Bayesian) additive Voronoi tessellations. *Journal of Computational and Graphical Statistics*.

Stone, A.J., Ogundimu, E. and Gosling, J.P. (2026). Binary AddiVortes: (Bayesian) Additive Voronoi Tessellations for Binary Classification with an application to Predicting Home Mortgage Application Outcomes.

Albert, J.H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422), 669–679.

Kindo, B.P., Wang, H. and Peña, E.A. (2016). Multinomial probit Bayesian additive regression trees. *Stat*, 5(1), 171–181.

Examples

```
# Simple example with simulated data
set.seed(123)
x <- matrix(rnorm(50), 10, 5)
y <- rnorm(10)
# Fit model with reduced iterations for quick example
fit <- AddiVortes(y, x, m = 5, totalMCMCIter = 50, mcmcBurnIn = 10)

# Larger example with categorical covariates (d=2 and d=3) and a test set
set.seed(456)
n_train <- 200
n_test <- 50
x_train <- data.frame(
  x1   = rnorm(n_train),
  x2   = runif(n_train),
  grp2 = sample(c("A", "B"), n_train, replace = TRUE),
  grp3 = sample(c("low", "mid", "high"), n_train, replace = TRUE)
)
y_train <- x_train$x1 + ifelse(x_train$grp2 == "B", 1, 0) + rnorm(n_train, sd = 0.5)

fit2 <- AddiVortes(y_train, x_train,
  m = 10, totalMCMCIter = 200, mcmcBurnIn = 50,
  catScaling = 1, showProgress = FALSE
)

x_test <- data.frame(
  x1   = rnorm(n_test),
  x2   = runif(n_test),
  grp2 = sample(c("A", "B"), n_test, replace = TRUE),
  grp3 = sample(c("low", "mid", "high"), n_test, replace = TRUE)
)
y_test <- x_test$x1 + ifelse(x_test$grp2 == "B", 1, 0) + rnorm(n_test, sd = 0.5)

preds <- predict(fit2, x_test, showProgress = FALSE)
test_rmse <- sqrt(mean((y_test - preds)^2))

# Binary classification is selected automatically from a 0/1 or factor y
set.seed(789)
x_clf <- matrix(runif(80), 40, 2)
y_clf <- factor(ifelse(x_clf[, 1] + x_clf[, 2] > 1, "yes", "no"))
fit_clf <- AddiVortes(y_clf, x_clf, m = 8, totalMCMCIter = 80,
                      mcmcBurnIn = 20, showProgress = FALSE)
p_clf <- predict(fit_clf, x_clf, type = "response", showProgress = FALSE)
cls_clf <- predict(fit_clf, x_clf, type = "class", showProgress = FALSE)
```

`Boston` *Boston Dataset*

Description

The Boston Housing Dataset, derived from information from the U.S. Census Service.

Usage

```
Boston
```

Format

`Boston`**:**

A data.frame with 506 rows and 14 columns:

x1-x13 The covariates in the data

y The response variable

Source

Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol. 5, 81-102, 1978

`Cylinder` *Cylindrical Dataset*

Description

Synthetic data generated for cylindrical parameters. The response is given by the function f(z, theta) = sin(z)cos(theta) + z*sin(theta)^2, with added noise.

Usage

```
Cylinder
```

Format

`Cylinder`**:**

A data.frame with 400 rows and 3 columns:

z Height

theta Polar angle

Y Response

`new_AddiVortes` *Create an AddiVortes Object*

Description

A constructor for the AddiVortes class.

Usage

```
new_AddiVortes(
  posteriorTess,
  posteriorDim,
  posteriorSigma,
  posteriorPred,
  xCentres,
  xRanges,
  yCentre,
  yRange,
  inSampleRmse,
  metric = "E",
  members = rep(1, length(xCentres)),
  metric_aug = "E",
  member_aug = rep(1, length(xCentres)),
  catEncoding = NULL,
  traceStats = NULL,
  task = "regression",
  classLevels = NULL,
  nLatents = 1L,
  mPerLatent = NA_integer_,
  inSampleAccuracy = NA_real_,
  inSampleBrier = NA_real_
)
```

Arguments

`posteriorTess`	A list of the posterior samples of the tessellations.
`posteriorDim`	A list of the posterior samples of the dimensions.
`posteriorSigma`	A list of the posterior samples of the error variance.
`posteriorPred`	A list of the posterior samples of the predictions.
`xCentres`	The centres of the covariates.
`xRanges`	The ranges of the covariates.
`yCentre`	The centre of the output values.
`yRange`	The range of the output values.
`inSampleRmse`	The in-sample RMSE.

metric	The metric used for scaling covariates (default "E" for Euclidean).
members	The membership vector for the covariates
metric_aug	The augmented metric after categorical variables are converted to one-hot
member_aug	The membership vector corresponding to metric_aug
catEncoding	Optional list of categorical encoding metadata returned by encodeCategories_internal, or NULL if no categorical covariates were present.
traceStats	Optional data frame of per-iteration MCMC trace statistics.
task	The modelling task: "regression", "binary" or "multinomial".
classLevels	Character vector of class labels for classification fits, or NULL for regression.
nLatents	Number of latent probit dimensions. 1 for regression and binary classification; $K-1$ for K-class multinomial models.
mPerLatent	Number of tessellations per latent ensemble.
inSampleAccuracy	In-sample classification accuracy, or NA for regression.
inSampleBrier	In-sample Brier score for binary classification, or NA otherwise.

Value

An object of class AddiVortes.

`plot.AddiVortes` *Plot Method for AddiVortes*

Description

Generates comprehensive diagnostic plots for a fitted `AddiVortes` object. This function creates multiple diagnostic plots including residuals, MCMC traces for sigma, and tessellation complexity over iterations.

Usage

```
## S3 method for class 'AddiVortes'
plot(
  x,
  x_train,
  y_train,
  sigma_trace = NULL,
  which = c(1, 2, 3),
  ask = FALSE,
  ...
)
```

Arguments

x	An object of class AddiVortes, typically the result of a call to AddiVortes().
x_train	A matrix or data frame of the original training covariates.
y_train	The original training response. Numeric for regression; factor, character, logical or 0/1 numeric for classification.
sigma_trace	An optional numeric vector of sigma values from MCMC samples. If not provided, the method will attempt to extract the posterior error standard deviation from the model object.
which	A numeric vector specifying which plots to generate: 1 = Residuals plot, 2 = Sigma trace, 3 = Tessellation complexity trace, 4 = Predicted vs Observed. Default is c(1, 2, 3).
ask	Logical; if TRUE, the user is asked to press Enter before each plot.
...	Additional arguments passed to plotting functions.

Details

The function generates up to four diagnostic plots:

1. **Residuals Plot**: Residuals vs fitted values with smoothed trend line
2. **Sigma Trace**: MCMC trace plot for the error standard deviation
3. **Tessellation Complexity**: Trace of average tessellation size over iterations
4. **Predicted vs Observed**: Scatter plot with credible intervals

Value

This function is called for its side effect of creating plots and returns NULL invisibly.

Examples

```
## Not run:
# Assuming 'fit' is a trained AddiVortes object
plot(fit, x_train = x_train_data, y_train = y_train_data)

# Show only specific plots
plot(fit, x_train = x_train_data, y_train = y_train_data, which = c(1, 3))

# With custom sigma trace
plot(fit,
  x_train = x_train_data, y_train = y_train_data,
  sigma_trace = my_sigma_samples
)

## End(Not run)
```

`predict.AddiVortes` *Predict Method for AddiVortes*

Description

Predicts outcomes for new data using a fitted `AddiVortes` model object. Regression fits return means or quantiles of the response. Classification fits return class probabilities, class labels, latent-scale values, or quantiles of the class probabilities.

Usage

```
## S3 method for class 'AddiVortes'
predict(
  object,
  newdata,
  type = c("response", "quantile", "class", "link"),
  quantiles = c(0.025, 0.975),
  interval = c("credible", "prediction"),
  showProgress = interactive(),
  ...
)
```

Arguments

`object`	An object of class `AddiVortes`, typically the result of a call to `AddiVortes()`.
`newdata`	A matrix of covariates for the new test set. The number of columns must match the original training data.
`type`	The type of prediction required. The default `"response"` gives the mean prediction (class probabilities for classification). `"quantile"` returns the quantiles specified by `quantiles`. `"class"` returns predicted class labels (classification only). `"link"` returns the latent sum of tessellations $G(x)$ on the model scale before any response unscaling.
`quantiles`	A numeric vector of probabilities to compute for the predictions when `type = "quantile"`.
`interval`	The type of interval calculation. The default `"credible"` accounts only for uncertainty in the mean (similar to `lm`'s confidence interval). The alternative `"prediction"` also includes the model's error variance, producing wider intervals (similar to `lm`'s prediction interval). Not used for classification models.
`showProgress`	Logical; if TRUE, a progress bar is shown during prediction.
`...`	Further arguments passed to or from other methods (currently unused).

Details

This function relies on the internal helper function `applyScaling_internal` being available in the environment, which is used by the main `AddiVortes` function.

Predictions traverse all retained draws and tessellations in a single C++ call, avoiding repeated R/C++ boundary crossings per tessellation.

When `interval = "prediction"` and `type = "quantile"`, the function samples additional Gaussian noise with variance equal to the sampled sigma squared from the posterior. This accounts for the inherent variability in individual predictions, not just uncertainty in the mean function. The noise is added in the scaled space before unscaling predictions. Classification uses a probit link with residual variance fixed at 1, so prediction intervals are not defined; use `type = "quantile"` for credible intervals on probabilities.

For regression, `"response"` unscales predictions back to the original response units, while `"link"` returns the posterior mean of the latent scaled function $G(x)$.

For binary classification, `"response"` is the posterior mean of $\Phi(G^{(s)}(x))$. For multinomial classification, class probabilities are estimated from independent $N(G, I)$ latents, with the first class as the reference.

Value

If `type = "response"`, a numeric vector of mean predictions for regression or binary classification, or an $n \times K$ probability matrix for multinomial classification. If `type = "quantile"`, a matrix of quantiles (binary/regression) or a named list of such matrices (multinomial). If `type = "class"`, a factor of predicted labels. If `type = "link"`, the latent function $G(x)$ on the model scale before response unscaling.

Examples

```
# Fit a model
set.seed(123)
X <- matrix(rnorm(100), 20, 5)
Y <- rnorm(20)
fit <- AddiVortes(Y, X, m = 5, totalMCMCIter = 50, mcmcBurnIn = 10)

# New data for prediction
X_new <- matrix(rnorm(25), 5, 5)

# Mean predictions
pred_mean <- predict(fit, X_new, type = "response")

# Credible intervals (uncertainty in mean only)
pred_conf <- predict(fit, X_new,
  type = "quantile",
  interval = "credible",
  quantiles = c(0.025, 0.975)
)

# Prediction intervals (includes error variance)
pred_pred <- predict(fit, X_new,
  type = "quantile",
  interval = "prediction",
  quantiles = c(0.025, 0.975)
)
```

```
# Prediction intervals are wider than credible intervals
mean(pred_pred[, 2] - pred_pred[, 1]) > mean(pred_conf[, 2] - pred_conf[, 1])
```

`print.AddiVortes` *Print Method for AddiVortes*

Description

Prints a summary of a fitted `AddiVortes` object, providing information about the model structure, dimensions, and fit quality similar to the output of a linear model summary.

Usage

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

Arguments

`x`	An object of class `AddiVortes`, typically the result of a call to `AddiVortes()`.
`...`	Further arguments passed to or from other methods (currently unused).

Details

The print method displays:

- The model formula representation
- Number of covariates and posterior samples
- Number of tessellations used
- In-sample RMSE
- Covariate scaling information

Value

The function is called for its side effect of printing model information and returns the input object `x` invisibly.

summary.AddiVortes *Summary Method for AddiVortes*

Description

Provides a detailed summary of a fitted `AddiVortes` object, including more comprehensive information than the print method.

Usage

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

Arguments

`object`	An object of class `AddiVortes`, typically the result of a call to `AddiVortes()`.
`...`	Further arguments passed to or from other methods (currently unused).

Value

The function is called for its side effect of printing detailed model information and returns the input object `object` invisibly.

traceplots.AddiVortes *Trace Plot Method for AddiVortes*

Description

Displays four MCMC trace plots for a fitted `AddiVortes` object: the average number of centres per tessellation, the standard deviation of the number of centres per tessellation, the average number of dimensions used per tessellation, and the retained-state log-likelihood component whose differences form the likelihood part of the acceptance ratio.

Usage

```
## S3 method for class 'AddiVortes'
traceplots(x, ask = FALSE, ...)
```

Arguments

`x`	An object of class `AddiVortes`, typically the result of a call to `AddiVortes()`.
`ask`	Logical; if TRUE, the user is asked to press Enter before each plot.
`...`	Additional arguments passed to plotting functions.

Details

The four trace plots are:

1. **Average Centres**: Average number of centres per tessellation.
2. **Centre Count Standard Deviation**: Standard deviation of the number of centres per tessellation.
3. **Average Dimensions**: Average number of active dimensions used per tessellation.
4. **Log Likelihood**: Average retained-state log-likelihood component at the end of each MCMC iteration.

Value

This function is called for its side effect of creating plots and returns `NULL` invisibly.

Examples

```
## Not run:
# Assuming 'fit' is a trained AddiVortes object
traceplots(fit)

## End(Not run)
```

Weather *Weather Dataset*

Description

A subset of data collected as part of the GES DISC datasets for calibrated brightness temperatures.

Usage

```
Weather
```

Format

`Weather`**:**

A data.frame with 2000 rows and 3 columns:

theta Polar angle

phi Azimuthal angle

Y Response variable

Source

https://disc.gsfc.nasa.gov/datasets/GPM_1CGPMGMI_07/summary?keywords=10.5067%2FGPM%2FGMI%2FGPM%2F1

Index