Package: MRFtools (via r-universe)

May 14, 2026

Version 0.0-6

Title Tools for Constructing and Plotting Markov Random Fields in R for Graphical Data

Maintainer Eric J. Pedersen <eric.pedersen@concordia.ca>

Depends R (>= 4.1.0)

Imports gratia, stats, methods, generics, Matrix, assertthat, sf, phylobase, colorspace, tidygraph, ggraph, ggplot2, tidyr, dplyr, rlang, ape

Suggests testthat, vdiffr, mgcv, gamm4, sp, quarto, withr, ggtree

Description Utility functions for using Markov Random Field smooths in Generalized Additive Models fitted with the 'mgcv' package.

License GPL-3

LazyData true

URL https://github.com/gam-mafia/MRFtools

BugReports https://github.com/gam-mafia/MRFtools/issues

Roxygen list(markdown = TRUE)

Encoding UTF-8

Config/testthat/edition 3

VignetteBuilder quarto

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs libabsl-dev cmake libfontconfig1-dev libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev libicu-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://gam-mafia.r-universe.dev

Date/Publication 2026-05-14 17:34:54 UTC

RemoteUrl https://github.com/gam-mafia/MRFtools

RemoteRef HEAD

RemoteSha 117f7f789f316b74d93da89f1ef0dd5304483200

Contents

`as.matrix.mrf_penalty` *Convert a MRF penalty object to a matrix*

Description

Convert a MRF penalty object to a matrix

Usage

```
## S3 method for class 'mrf_penalty'
as.matrix(x, ...)
```

Arguments

`x`	an object inheriting from class `"mrf_penalty"`
`...`	arguments passed to other methods

Examples

```
p <- mrf_penalty(1:10)
as.matrix(p)
```

`as_mrf_penalty` *Create a* `"mrf_penalty"` *object from a matrix and a config*

Description

Create a `"mrf_penalty"` object from a matrix and a config

Usage

```
as_mrf_penalty(penalty, config)
```

Arguments

`penalty`	matrix
`config`	list

`get_config` *Extract configuration details of an MRF penalty*

Description

Extract configuration details of an MRF penalty

Usage

```
get_config(penalty)
```

Arguments

`penalty`	an object of class `"mrf_penalty"`

Value

An object of class `"mrf_config"`, a list.

get_labels *Extract MRF node labels from an MRF penalty*

Description

Extract MRF node labels from an MRF penalty

Usage

```
get_labels(penalty)
```

Arguments

penalty	an object of class "mrf_penalty"

get_model *Extract the model type and parameters from an MRF penalty*

Description

Extract the model type and parameters from an MRF penalty

Usage

```
get_model(penalty)
```

Arguments

penalty	an object of class "mrf_penalty"

get_mrf *Extract a fitted MRF*

Description

Extract a fitted MRF

Usage

```
get_mrf(object, ...)

## S3 method for class 'bam'
get_mrf(object, ...)

## S3 method for class 'gamm'
get_mrf(object, ...)

## S3 method for class 'gamm4'
get_mrf(object, ...)

## S3 method for class 'gam'
get_mrf(object, term, ...)
```

Arguments

`object`	An object from which to extract the fitted MRF. Currently only for objects of classes `gam`, `bam`, and `gamm`, and GAMMs fitted by `gamm4::gamm4()`.
`...`	Arguments passed to other methods.
`term`	character; the MRF term to extract. Can be a partial match to a term, which is matched against the smooth label.

Value

A object representing the fitted MRF

`get_obj` *Extract the original object used to construct an MRF penalty*

Description

Extract the original object used to construct an MRF penalty

Usage

```
get_obj(object)

## S3 method for class 'mrf_config'
get_obj(object)

## S3 method for class 'mrf_penalty'
get_obj(object)
```

Arguments

`object`	an object of class `"mrf_penalty"` or `"mrf_config"`.

Value

A length 1 character vector containing the type of MRF penalty.

`get_penalty` *Extract a MRF penalty matrix*

Description

Extract a MRF penalty matrix

Usage

```
get_penalty(penalty, ...)

## S3 method for class 'mrf_penalty'
get_penalty(penalty, ...)
```

Arguments

`penalty`	an R object from which to extract the MRF penalty matrix.
`...`	arguments passed to other methods.

Value

A penalty matrix of class `"matrix"`.

`get_type` *Extract the type of MRF from the penalty*

Description

Extract the type of MRF from the penalty

Usage

```
get_type(object)

## S3 method for class 'mrf_penalty'
get_type(object)

## S3 method for class 'mrf_config'
get_type(object)

## S3 method for class 'mrf_penalty'
get_type(object)
```

Arguments

object	an object of class "mrf_penalty" or "mrf_config".

Value

A length 1 character vector containing the type of MRF penalty.

---

mrf_config *MRF penalty configuration data*

Description

MRF penalty configuration data

Usage

```
mrf_config(
  type = NULL,
  model = NULL,
  params = NULL,
  node_labels = NULL,
  delta = NULL,
  obj = NULL
)
```

Arguments

type	character
model	character
params	list?
node_labels	character
delta	numeric
obj	character

mrf_penalty *Markov Random Field Penalty*

Description

Markov Random Field Penalty

Usage

```
mrf_penalty(object, ...)
```

Arguments

object	an R object to create the MRF penalty from.
...	arguments passed to other methods.

mrf_penalty.dendrogram *MRF penalty from a dendrogram*

Description

MRF penalty from a dendrogram

Usage

```
## S3 method for class 'dendrogram'
mrf_penalty(
  object,
  model = c("rw1", "ou", "brownian"),
  alpha = NULL,
  at_tips = NULL,
  internal_nodes = TRUE,
  delta = FALSE,
  ...
)
```

Arguments

object	an R object to create the MRF penalty from.
model	character; one of "full" or "individual" indicating if a fully connected graph ("full") or a random effect (random intercepts; "individual") penalty is created.
alpha	numeric; the autoregressive parameter for an OU stochastic process. Should be >1e-5 (alpha = 0 would correspond to a "rw1" model).

at_tips	character; vector of tip labels to calculate the penalty at. All values in the vector must correpond to tip name labels in the tree.
internal_nodes	logical; should the internal nodes of the tree be included in the penalty (TRUE), or just the tips (terminal nodes; FALSE)
delta	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
...	arguments passed to other methods.

`mrf_penalty.factor` *Fully connected graph and random effect MRF penalties from a factor*

Description

Fully connected graph and random effect MRF penalties from a factor

Usage

```
## S3 method for class 'factor'
mrf_penalty(
  object,
  model = c("full", "individual"),
  node_labels = NULL,
  delta = FALSE,
  ...,
  type
)
```

Arguments

object	an R object to create the MRF penalty from.
model	character; one of "full" or "individual" indicating if a fully connected graph ("full") or a random effect (random intercepts; "individual") penalty is created.
node_labels	character; a vector of alternative labels for the levels of the factor.
delta	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
...	arguments passed to other methods.
type	character; deprecated. Use, model instead.

Examples

```
# a factor
fv <- factor(letters[1:10])

# create the MRF penalty for a fully connected graph
p <- mrf_penalty(fv, model = "full")
p
as.matrix(p)

# create the MRF penalty equivalent of random effects
p <- mrf_penalty(fv, model = "individual")
p
as.matrix(p)
```

---

mrf_penalty.hclust *MRF penalty from a hclust object*

Description

MRF penalty from a hclust object

Usage

```
## S3 method for class 'hclust'
mrf_penalty(
  object,
  model = c("rw1", "ou", "brownian"),
  alpha = NULL,
  at_tips = NULL,
  internal_nodes = TRUE,
  delta = FALSE,
  ...
)
```

Arguments

`object`	an R object to create the MRF penalty from.
`model`	character; one of `"full"` or `"individual"` indicating if a fully connected graph (`"full"`) or a random effect (random intercepts; `"individual"`) penalty is created.
`alpha`	numeric; the autoregressive parameter for an OU stochastic process. Should be >1e-5 (alpha = 0 would correspond to a "rw1" model).
`at_tips`	character; vector of tip labels to calculate the penalty at. All values in the vector must correpond to tip name labels in the tree.
`internal_nodes`	logical; should the internal nodes of the tree be included in the penalty (`TRUE`), or just the tips (terminal nodes; `FALSE`)

delta	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
...	arguments passed to other methods.

`mrf_penalty.numeric` *Continuous-time random walk MRF penalty from a numeric vector*

Description

Models one-dimensional numeric vectors as random-walk models.

Usage

```
## S3 method for class 'numeric'
mrf_penalty(
  object,
  model = c("rw1", "rw2", "rw2_d", "ar1", "ou"),
  cyclic = FALSE,
  alpha = NULL,
  rho = NULL,
  at_nodes = NULL,
  node_labels = NULL,
  end_points = NULL,
  end_dist = NULL,
  delta = FALSE,
  ...,
  type
)
```

Arguments

object	an R object to create the MRF penalty from.
model	character; one of "rw1","ou", "ar1", "rw2", or "rw2_d". "rw1" is a first-order continuous-time random walk model (I.e. Brownian motion). "ou" is the Ornstein-Uhlenbeck (I.e. continuous-time first-order autoregressive model); model = "ou" also requires specifying a value of alpha. "ar1" is a first-order discrete-time autoregressive model, and requires specifying the rho parameter. "rw2" is a second-order random-walk model (the "CRW2" model from Rue and Held 2005); it returns a dense precision matrix for the values of the function evaluated at the chosen points. "rw2_d" is a sparse version of "rw2", that includes both the function itself and the derivatives of the function in its precision matrix. See Description for details on the models
cyclic	logical; If TRUE, the end points are treated as neighbouring each other. See Description for details

alpha	numeric; autoregression parameter for a continuous-time random walk ("ou"). rho must be >0.
rho	numeric; autoregression parameter for a discrete-time random walk ("ar1"). abs(rho) must be < 1 if specified.
at_nodes	numeric; what nodes (I.e. object values) that you want to evaluate the penalty at. Must include all values levels specified in `object`, but can also include additional levels that you want to evaluate the penalty at (see details).
node_labels	character; a vector of alternative labels for the levels of the factor.
end_points	numeric; an optional vector of length 2 providing the end points of the period of cycle.
end_dist	numeric;
delta	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
...	arguments passed to other methods.
type	character; deprecated. Use, `model` instead.

Examples

```
# create some test data
x_cont <- seq(0,5, length = 10)
x_disc <- 1:10

# linear rw1: 1st order continuous-time random walk
p1 <- mrf_penalty(x_cont)
p1

# cyclic rw1:
p2 <- mrf_penalty(x_cont, model = "rw1", cyclic = TRUE)
p2

# cyclic with user-specified end points
p3 <- mrf_penalty(x_cont, model = "rw1", cyclic = TRUE, end_points = c(0,6))
p3

# Continuous-time auto-regressive (I.e. "ou") model:
p4 <- mrf_penalty(x_cont, model = "ou",alpha = 2)
p4

# Discrete-time autoregressive model with negative 1st order autocorrelation
p5 <- mrf_penalty(x_disc, model = "ar1", rho = -0.5)
```

`mrf_penalty.phylo` *MRF penalty from a phylogeny*

Description

MRF penalty from a phylogeny

Usage

```
## S3 method for class 'phylo'
mrf_penalty(
  object,
  model = c("rw1", "ou", "brownian"),
  alpha = NULL,
  at_tips = NULL,
  internal_nodes = TRUE,
  delta = FALSE,
  ...
)
```

Arguments

`object`	an R object to create the MRF penalty from.
`model`	character; one of `"full"` or `"individual"` indicating if a fully connected graph (`"full"`) or a random effect (random intercepts; `"individual"`) penalty is created.
`alpha`	numeric; the autoregressive parameter for an OU stochastic process. Should be >1e-5 (alpha = 0 would correspond to a "rw1" model).
`at_tips`	character; vector of tip labels to calculate the penalty at. All values in the vector must correpond to tip name labels in the tree.
`internal_nodes`	logical; should the internal nodes of the tree be included in the penalty (`TRUE`), or just the tips (terminal nodes; `FALSE`)
`delta`	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
`...`	arguments passed to other methods.

Examples

```
#Example code
```

`mrf_penalty.phylo4` *MRF penalty from a phylogeny from a phylo4 object*

Description

MRF penalty from a phylogeny from a phylo4 object

Usage

```
## S3 method for class 'phylo4'
mrf_penalty(
  object,
  model = c("rw1", "ou", "brownian"),
  alpha = NULL,
  at_tips = NULL,
  internal_nodes = TRUE,
  delta = FALSE,
  ...
)
```

Arguments

`object`	an R object to create the MRF penalty from.
`model`	character; one of `"full"` or `"individual"` indicating if a fully connected graph (`"full"`) or a random effect (random intercepts; `"individual"`) penalty is created.
`alpha`	numeric; the autoregressive parameter for an OU stochastic process. Should be >1e-5 (alpha = 0 would correspond to a "rw1" model).
`at_tips`	character; vector of tip labels to calculate the penalty at. All values in the vector must correpond to tip name labels in the tree.
`internal_nodes`	logical; should the internal nodes of the tree be included in the penalty (`TRUE`), or just the tips (terminal nodes; `FALSE`)
`delta`	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
`...`	arguments passed to other methods.

Examples

```
#loading the geospiza dataset from phylobase
library(phylobase)
data(geospiza)

#Random-walk (rw1) penalty for both tips and nodes:
pen_rw <- mrf_penalty(geospiza, model = "rw1")
```

```
#Same model, but for a reduced number of species
species <- c("fortis", "pauper", "fusca", "olivacea")
pen_rw_subset <- mrf_penalty(geospiza, model = "rw1", at_tips = species)
plot(get_obj(pen_rw_subset))

#Random-walk penalty matrix for just the tips for all geospiza data:
pen_rw_tips <- mrf_penalty(geospiza, model = "rw1", internal_nodes = FALSE)
pen_rw_tips

#Ornstein-Uhlenbeck ("ou") process penalty matrix, specifying alpha parameter (autocorrelation)
pen_ou <- mrf_penalty(geospiza, model = "ou", alpha = 1)
```

`mrf_penalty.sf` *MRF penalty from polygon or multi-polygon simple features*

Description

MRF penalty from polygon or multi-polygon simple features

Usage

```
## S3 method for class 'sf'
mrf_penalty(
  object,
  model = "icar",
  node_labels = NULL,
  buffer = NULL,
  delta = FALSE,
  ...
)
```

Arguments

`object`	an R object to create the MRF penalty from.
`model`	character; one of `"full"` or `"individual"` indicating if a fully connected graph (`"full"`) or a random effect (random intercepts; `"individual"`) penalty is created.
`node_labels`	character; a vector of alternative labels for the levels of the factor.
`buffer`	numeric; buffer distance for all or for individual elements of the geometry. See argument `dist` in sf::st_buffer for details.
`delta`	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
`...`	arguments passed to other methods.

mrf_penalty.SpatialPolygons

MRF penalty from a SpatialPolygons

Description

MRF penalty from a SpatialPolygons

Usage

```
## S3 method for class 'SpatialPolygons'
mrf_penalty(
  object,
  model = "icar",
  node_labels = NULL,
  buffer = NULL,
  delta = FALSE,
  ...
)
```

Arguments

object	an R object to create the MRF penalty from.
model	character; one of "full" or "individual" indicating if a fully connected graph ("full") or a random effect (random intercepts; "individual") penalty is created.
node_labels	character; a vector of alternative labels for the levels of the factor.
buffer	numeric; buffer distance for all or for individual elements of the geometry. See argument dist in sf::st_buffer for details.
delta	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
...	arguments passed to other methods.

mrf_penalty.SpatialPolygonsDataFrame

MRF penalty from a SpatialPoylgonsDataFrame

Description

MRF penalty from a SpatialPoylgonsDataFrame

Usage

```
## S3 method for class 'SpatialPolygonsDataFrame'
mrf_penalty(
  object,
  model = "icar",
  node_labels = NULL,
  buffer = NULL,
  delta = FALSE,
  ...
)
```

Arguments

object	an R object to create the MRF penalty from.
model	character; one of "full" or "individual" indicating if a fully connected graph ("full") or a random effect (random intercepts; "individual") penalty is created.
node_labels	character; a vector of alternative labels for the levels of the factor.
buffer	numeric; buffer distance for all or for individual elements of the geometry. See argument dist in sf::st_buffer for details.
delta	numeric or logical; either the numeric value to add to the diagonal of the MRF penalty matrix, or a logical value indicating if such an adjustment should be made. The default is to not alter the diagonal of the penalty matrix.
...	arguments passed to other methods.

---

vis *Visualize a data set or object*

Description

Synonyms of the generics::visualize() method. Alternatives are vis() and visualise().

Usage

```
vis(x, ...)

visualise(x, ...)
```

Arguments

x	A data frame or other object.
...	Other arguments passed to methods

visualize.cyclic_mrf_penalty

Visualizing penalty matrix or graph object for a cyclic 1D MRF

Description

Visualizing penalty matrix or graph object for a cyclic 1D MRF

Usage

```
## S3 method for class 'cyclic_mrf_penalty'
visualize(x, graph = TRUE, layout = "linear", circular = TRUE, ...)
```

Arguments

x	an object of class "cyclic_mrf_penalty"
graph	logical;
layout	character;
circular	logical;
...	arguments passed to other methods and ultimately on to ggraph::create_layout() if graph = TRUE.

Examples

```
# example code
mrf_penalty(1:10, type = "linear") |>
  visualize()
```

visualize.fully_connected_graph_mrf_penalty

Plot a thing

Description

Plot a thing

Usage

```
## S3 method for class 'fully_connected_graph_mrf_penalty'
visualize(
  x,
  graph = TRUE,
  layout = "stress",
  circular = FALSE,
```

```
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  fill_scale = NULL,
  ...
)
```

Arguments

x	an object of class "fully_connected_graph_mrf_penalty"
graph	logical;
layout	character;
circular	logical;
xlab, ylab, title, subtitle, caption	character; labels for plots. If xlab or ylab are not supplied, a suitable default is used.
fill_scale	a suitable fill scale to use if plotting the penalty matrix
...	arguments passed to other methods and ultimately on to ggraph::create_layout() if graph = TRUE.

Examples

```
# example code
mrf_penalty(1:10, type = "linear") |>
  visualize()
```

---

visualize.sequential_mrf_penalty
Plot a thing

Description

Plot a thing

Usage

```
## S3 method for class 'sequential_mrf_penalty'
visualize(
  x,
  graph = TRUE,
  layout = "linear",
  circular = FALSE,
  xlab = NULL,
  ylab = NULL,
```

```
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  fill_scale = NULL,
  ...
)
```

Arguments

x	an object of class "sequential_mrf_penalty"
graph	logical;
layout	character;
circular	logical;
xlab, ylab, title, subtitle, caption	character; labels for plots. If xlab or ylab are not supplied, a suitable default is used.
fill_scale	a suitable fill scale to use if plotting the penalty matrix
...	arguments passed to other methods and ultimately on to ggraph::create_layout() if graph = TRUE.

Examples

```
# example code
mrf_penalty(1:10, type = "linear") |>
  visualize()
```

---

visualize.tree_mrf_penalty

Plot a thing

Description

Plot a thing

Usage

```
## S3 method for class 'tree_mrf_penalty'
visualize(
  x,
  graph = TRUE,
  layout = "stress",
  circular = FALSE,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
```

```
  caption = NULL,
  fill_scale = NULL,
  ...
)
```

Arguments

x	an object of class "dendrogram_mrf_penalty"
graph	logical;
layout	character;
circular	logical;
xlab, ylab, title, subtitle, caption	character; labels for plots. If xlab or ylab are not supplied, a suitable default is used.
fill_scale	a suitable fill scale to use if plotting the penalty matrix
...	arguments passed to other methods and ultimately on to ggraph::create_layout() if graph = TRUE.

Examples

```
# example code
hc <- hclust(dist(USArrests), "complete")
mrf_penalty(hc, internal_nodes = FALSE) |>
  visualize()
```

Index