Package 'nprotreg'

October 14, 2018

Title Nonparametric Rotations for Sphere-Sphere Regression

Version 1.0.0

Description Fits sphere-sphere regression models by estimating locally weighted rotations.
Simulation of sphere-sphere data according to non-rigid rotation models.
Provides methods for bias reduction applying iterative
procedures within a Newton-Raphson learning scheme.
Cross-validation is exploited to select smoothing parameters.
See Marco Di Marzio, Agnese Panzera & Charles C. Taylor (2018)
<doi:10.1080/01621459.2017.1421542>.

Depends R (>= 3.3.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Suggests testthat

Imports methods, expm, stats

NeedsCompilation no

Author Charles C. Taylor [aut],
Giovanni Lafratta [aut, cre],
Stefania Fensore [aut]

Maintainer Giovanni Lafratta <giovanni.lafratta@unich.it>

Repository CRAN

Date/Publication 2018-10-14 17:20:06 UTC

R topics documented:

---

convert_cartesian_to_spherical

Converts Cartesian to Spherical Coordinates.

Description

The Cartesian coordinates of points on a 3-dimensional sphere with unit radius and center at the origin are converted to the equivalent longitude and latitude coordinates, measured in radians.

Usage

```
convert_cartesian_to_spherical(cartesian_coords)
```

Arguments

cartesian_coords

A matrix whose rows contain the Cartesian coordinates of the specified points.

Value

A matrix of rows containing the longitude and latitude of specific points on a 3-dimensional sphere.

See Also

http://mathworld.wolfram.com/SphericalCoordinates.html.

Other Conversion functions: convert_spherical_to_cartesian

Examples

```
library(nprotreg)

# Define the Cartesian coordinates of the North and South Poles.

north_pole <- cbind(0, 0, 1)
south_pole <- cbind(0, 0, -1)
cartesian_coords <- rbind(north_pole, south_pole)

# Get the corresponding Spherical coordinates.

spherical_coords <- convert_cartesian_to_spherical(cartesian_coords)
```

convert_spherical_to_cartesian

Converts Spherical to Cartesian Coordinates.

Description

The longitude and latitude coordinates of points on a 3-dimensional sphere with unit radius and center at the origin are converted to the equivalent Cartesian coordinates.

Usage

```
convert_spherical_to_cartesian(spherical_coords)
```

Arguments

spherical_coords

A matrix of rows containing the longitude and latitude, measured in radians, of specific points on a 3-dimensional sphere.

Value

A matrix whose rows contain the Cartesian coordinates of the specified points.

See Also

http://mathworld.wolfram.com/SphericalCoordinates.html.

Other Conversion functions: convert_cartesian_to_spherical

Examples

```
library(nprotreg)

# Define the Spherical coordinates of the North and South Poles.

north_pole <- cbind(0, pi / 2)
south_pole <- cbind(0, - pi / 2)
spherical_coords <- rbind(north_pole, south_pole)

# Get the corresponding Cartesian coordinates.

cartesian_coords <- convert_spherical_to_cartesian(spherical_coords)
```

---

cross_validate_concentration

> *Cross-validates The Concentration Parameter In A 3D Spherical Regression.*

Description

Returns a cross-validated value for the concentration parameter in a 3D regression, relating specific explanatory points to response ones, given a weighting scheme for the observed data set. This function supports the method for sphere-sphere regression proposed by Di Marzio et al. (2018).

Usage

```
cross_validate_concentration(concentration_upper_bound = 10,
  explanatory_points, response_points,
  weights_generator = weight_explanatory_points,
  number_of_expansion_terms = 1, number_of_iterations = 1,
  allow_reflections = FALSE)
```

Arguments

concentration_upper_bound

A scalar numeric value representing the upper end-point of the interval to be searched for the required minimizer. Defaults to `10`.

explanatory_points

An m-by-3 matrix whose rows contain the Cartesian coordinates of the explanatory points used to calculate the regression estimators.

response_points

An m-by-*3* matrix whose rows contain the Cartesian coordinates of the response points corresponding to the explanatory points.

weights_generator

A function that, given a matrix of n evaluation points, returns an m-by-n matrix whose j-th column contains the weights assigned to the explanatory points while analyzing the j-th evaluation point. Defaults to `weight_explanatory_points`.

number_of_expansion_terms

The number of terms to be included in the expansion of the matrix exponential applied while approximating a local rotation matrix. Must be 1 or 2. Defaults to 1.

number_of_iterations

The number of rotation fitting steps to be executed. At each step, the points estimated during the previous step are exploited as the current explanatory points. Defaults to 1.

allow_reflections

A logical scalar value. If set to `TRUE` signals that reflections are allowed. Defaults to `FALSE`. It is ignored if `number_of_expansion_terms` is 2.

Details

Function `weights_generator` must be prototyped as having the following three arguments:

`evaluation_points` a matrix whose *n* rows are the Cartesian coordinates of given evaluation points.

`explanatory_points` a matrix whose *m* rows are the Cartesian coordinates of given explanatory points.

`concentration` A non negative scalar whose reciprocal value is proportional to the bandwidth applied while estimating a spherical regression model.

It is also expected that `weights_generator` will return a non NULL numerical *m*-by-*n* matrix whose *j*-th column contains the weights assigned to the explanatory points while analyzing the *j*-th evaluation point.

Value

A list having two components, `concentration`, a scalar, numeric value representing the cross-validated concentration for the specified 3D regression, and `objective`, the value of the cross-validating objective function at argument `concentration`.

References

Marco Di Marzio, Agnese Panzera & Charles C. Taylor (2018) Nonparametric rotations for sphere-sphere regression, Journal of the American Statistical Association, DOI: 10.1080/01621459.2017.1421542

See Also

Other Regression functions: `fit_regression`, `get_equally_spaced_points`, `get_skew_symmetric_matrix`, `simulate_regression`, `simulate_rigid_regression`, `weight_explanatory_points`

Examples

```
library(nprotreg)

# Define a matrix of explanatory points.

number_of_explanatory_points <- 50

explanatory_points <- get_equally_spaced_points(
  number_of_explanatory_points)

# Define a matrix of response points by simulation.

local_rotation_composer <- function(point) {
  independent_components <- (1 / 2) *
    c(exp(2.0 * point[3]), - exp(2.0 * point[2]), exp(2.0 * point[1]))
}

local_error_sampler <- function(point) {
  rnorm(3)
```

```
    }

    response_points <- simulate_regression(explanatory_points,
                                            local_rotation_composer,
                                            local_error_sampler)

    # Define an upper bound for concentration.

    concentration_upper_bound <- 1

    # Use default weights generator.

    weights_generator <- weight_explanatory_points

    # Cross-validate concentration parameter.

    cv_info <- cross_validate_concentration(
      concentration_upper_bound,
      explanatory_points,
      response_points,
      weights_generator,
      number_of_expansion_terms = 1,
      number_of_iterations = 2,
      allow_reflections = FALSE
    )

    # Get the cross-validated concentration value.

    cat("cross-validated concentration value: \n")
    print(cv_info$concentration)
```

---

fit_regression   *Fits a 3D Spherical Regression.*

Description

Returns 3D spherical points obtained by locally rotating the specified evaluation points, given an approximated model for local rotations and a weighting scheme for the observed data set. This function implements the method for sphere-sphere regression proposed by Di Marzio et al. (2018).

Usage

```
fit_regression(evaluation_points, explanatory_points, response_points,
  concentration, weights_generator = weight_explanatory_points,
  number_of_expansion_terms = 1, number_of_iterations = 1,
  allow_reflections = FALSE)
```

Arguments

evaluation_points
> An *n*-by-3 matrix whose rows contain the Cartesian coordinates of the points at which the regression will be estimated.

explanatory_points
> An *m*-by-3 matrix whose rows contain the Cartesian coordinates of the explanatory points used to calculate the regression estimators.

response_points
> An *m*-by-*3* matrix whose rows contain the Cartesian coordinates of the response points corresponding to the explanatory points.

concentration
> A non negative scalar whose reciprocal value is proportional to the bandwidth applied while estimating a spherical regression model.

weights_generator
> A function that, given a matrix of *n* evaluation points, returns an *m*-by-*n* matrix whose *j*-th column contains the weights assigned to the explanatory points while analyzing the *j*-th evaluation point. Defaults to [weight_explanatory_points](#).

number_of_expansion_terms
> The number of terms to be included in the expansion of the matrix exponential applied while approximating a local rotation matrix. Must be 1 or 2. Defaults to 1.

number_of_iterations
> The number of rotation fitting steps to be executed. At each step, the points estimated during the previous step are exploited as the current explanatory points. Defaults to 1.

allow_reflections
> A logical scalar value. If set to TRUE signals that reflections are allowed. Defaults to FALSE. It is ignored if number_of_expansion_terms is 2.

Details

Function weights_generator must be prototyped as having the following three arguments:

evaluation_points a matrix whose *n* rows are the Cartesian coordinates of given evaluation points.

explanatory_points a matrix whose *m* rows are the Cartesian coordinates of given explanatory points.

concentration A non negative scalar whose reciprocal value is proportional to the bandwidth applied while estimating a spherical regression model.

It is also expected that weights_generator will return a non NULL numerical *m*-by-*n* matrix whose *j*-th column contains the weights assigned to the explanatory points while analyzing the *j*-th evaluation point.

Value

A number_of_iterations-length vector of lists, with the s-th list having two components, fitted_response_points, an *n*-by-*3* matrix whose rows contain the Cartesian coordinates of the fitted points at iteration s, and explanatory_points, an *m*-by-*3* matrix whose rows contain the Cartesian coordinates of the points exploited as explanatory at iteration s.

References

Marco Di Marzio, Agnese Panzera & Charles C. Taylor (2018) Nonparametric rotations for sphere-sphere regression, Journal of the American Statistical Association, DOI: 10.1080/01621459.2017.1421542

See Also

Other Regression functions: cross_validate_concentration, get_equally_spaced_points, get_skew_symmetric_matrix, simulate_regression, simulate_rigid_regression, weight_explanatory_points

Examples

```
library(nprotreg)

# Create 100 equally spaced design points on the sphere.

number_of_explanatory_points <- 100

explanatory_points <- get_equally_spaced_points(
  number_of_explanatory_points
)

# Define the regression model, where the rotation for a given "point"
# is obtained from the exponential of a skew-symmetric matrix with the
# following components.

local_rotation_composer <- function(point) {
  independent_components <- (1 / 8) *
    c(exp(2.0 * point[3]), - exp(2.0 * point[2]), exp(2.0 * point[1]))
}

# Define an error term given by a small rotation, similarly defined
# from a skew-symmetric matrix with random entries.

local_error_sampler <- function(point) {
  rnorm(3, sd = .01)
}

# Generate the matrix of responses, using the regression model
# and the error model.

response_points <- simulate_regression(
  explanatory_points,
  local_rotation_composer,
  local_error_sampler
)

# Create some "test data" for which the response will be predicted.

evaluation_points <- rbind(
  cbind(.5, 0, .8660254),
  cbind(-.5, 0, .8660254),
  cbind(1, 0, 0),
```

```
  cbind(0, 1, 0),
  cbind(-1, 0, 0),
  cbind(0, -1, 0),
  cbind(.5, 0, -.8660254),
  cbind(-.5, 0, -.8660254)
)

# Define a weight function for nonparametric fit.

weights_generator <- weight_explanatory_points

# Set the concentration parameter.

concentration <- 5

# Or obtain this by cross-validation: see
# the `cross_validate_concentration` function.

# Fit regression.

fitted_model <- fit_regression(
  evaluation_points,
  explanatory_points,
  response_points,
  concentration,
  weights_generator,
  number_of_expansion_terms = 1,
  number_of_iterations = 2
)

# Extract the point corresponding to the
# second evaluation point fitted at
# the first iteration.

cat("Point fitted at iteration 1 corresponding to the second evaluation point: \n")
cat(fitted_model[[1]]$fitted_response_points[2, ])

## Not run:
# Create some plots to view the results.

# 3D plot.

library(rgl)

plot3d(
  explanatory_points,
  type = "n",
  xlab = "x",
  ylab = "y",
  zlab = "z",
  box = TRUE,
  axes = TRUE
)
```

```
spheres3d(0, 0, 0, radius = 1, lit = FALSE, color = "white")
spheres3d(0, 0, 0, radius = 1.01, lit = FALSE, color = "black", front = "culled")
text3d(c(0, 0, 1), text = "N", adj = 0)

ll <- 10
vv1 <- (ll - (0:(ll))) / ll
vv2 <- 1 - vv1
plot3d(explanatory_points, add = TRUE, col = 2)
for (i in 1:dim(explanatory_points)[1]) {
  m <- outer(vv1, explanatory_points[i,], "*") +
    outer(vv2, response_points[i,], "*")
  m <- m / sqrt(apply(m ^ 2, 1, sum))
  lines3d(m, col = 3)
}

plot3d(evaluation_points, add = TRUE, col = 4)

for (i in 1:dim(evaluation_points)[1]) {
  m <- outer(vv1, evaluation_points[i,], "*") +
    outer(vv2, fitted_model[[1]]$fitted_response_points[i,], "*")
  m <- m / sqrt(apply(m ^ 2, 1, sum))
  lines3d(m, col = 1)
}

# 2D plot.

explanatory_spherical_coords <- convert_cartesian_to_spherical(explanatory_points)
response_spherical_coords <- convert_cartesian_to_spherical(response_points)

plot(
  x = explanatory_spherical_coords[, 1],
  y = explanatory_spherical_coords[, 2],
  pch = 20,
  cex = .7,
  col = 2,
  xlab = "longitude",
  ylab = "latitude"
)

for (i in 1:dim(explanatory_spherical_coords)[1]) {
  column <- 1
  if ((explanatory_spherical_coords[i, 1] - response_spherical_coords[i, 1]) ^ 2 +
      (explanatory_spherical_coords[i, 2] - response_spherical_coords[i, 2]) ^ 2 > 4)
        column <- "grey"
  lines(
    c(explanatory_spherical_coords[i, 1], response_spherical_coords[i, 1]),
    c(explanatory_spherical_coords[i, 2], response_spherical_coords[i, 2]),
    col = column
  )
}

evaluation_spherical_coords <- convert_cartesian_to_spherical(
  evaluation_points
```

```
)

fitted_response_spherical_coords <- convert_cartesian_to_spherical(
  fitted_model[[1]]$fitted_response_points
)

points(
  x = evaluation_spherical_coords[, 1],
  y = evaluation_spherical_coords[, 2],
  pch = 20,
  cex = .7,
  col = 4
)

for (i in 1:dim(evaluation_spherical_coords)[1]) {
  column <- 3
  if ((evaluation_spherical_coords[i, 1] - fitted_response_spherical_coords[i, 1]) ^ 2 +
      (evaluation_spherical_coords[i, 2] - fitted_response_spherical_coords[i, 2]) ^ 2 > 4)
        column <- "grey"
  lines(
    c(evaluation_spherical_coords[i, 1], fitted_response_spherical_coords[i, 1]),
    c(evaluation_spherical_coords[i, 2], fitted_response_spherical_coords[i, 2]),
    col = column
  )
}


## End(Not run)
```

---

get_equally_spaced_points

Generates Equally Spaced Points On A 3D Sphere.

Description

Generates points approximately equally spaced on a 3D sphere.

Usage

```
get_equally_spaced_points(number_of_points)
```

Arguments

number_of_points

> A scalar, positive integer representing the number of points to get.

Value

A `number_of_points`-by-3 matrix whose rows contain the Cartesian coordinates of the equally spaced points.

See Also

Other Regression functions: cross_validate_concentration, fit_regression, get_skew_symmetric_matrix, simulate_regression, simulate_rigid_regression, weight_explanatory_points

Examples

```
library(nprotreg)

# Define the number of points to get.

number_of_points <- 5

# Get the Cartesian coordinates of the equally spaced points.

equally_spaced_points <- get_equally_spaced_points(number_of_points)
```

---

get_skew_symmetric_matrix

Gets a 3-by-3 Skew Symmetric Matrix.

Description

Returns the 3-by-3 skew symmetric matrix having the specified independent components.

Usage

```
get_skew_symmetric_matrix(independent_components)
```

Arguments

independent_components

A vector containing the independent components of the matrix to get.

Details

Given a vector of components, say $[x, y, z]$, this function will return matrix

$$
\begin{matrix}
0 & -z & y \\
z & 0 & -x \\
-y & x & 0
\end{matrix}
$$

Value

The 3-by-3 skew symmetric matrix corresponding to the specified independent components.

See Also

https://en.wikipedia.org/wiki/Skew-symmetric_matrix.

Other Regression functions: cross_validate_concentration, fit_regression, get_equally_spaced_points, simulate_regression, simulate_rigid_regression, weight_explanatory_points

Examples

```
library(nprotreg)

# Define a vector of independent components.

independent_components <- cbind(1, 2, 3)

# Get the corresponding 3-by-3 skew symmetric matrix.

m <- get_skew_symmetric_matrix(independent_components)
```

---

nprotreg	*nprotreg: Nonparametric Rotations for Sphere-Sphere Regression.*

Description

The nprotreg package provides several categories of functions.

Regression functions

Regression functions provide support for simulating and fitting 3-dimensional spherical regression models.

- cross_validate_concentration
- fit_regression
- get_equally_spaced_points
- get_skew_symmetric_matrix
- simulate_regression
- simulate_rigid_regression
- weight_explanatory_points

Conversion functions

Conversion functions transform coordinates of points on a 3-dimensional sphere with unit radius and center at the origin.

- convert_cartesian_to_spherical
- convert_spherical_to_cartesian

---

simulate_regression   *Simulates a 3D Spherical Regression.*

Description

Returns the response points corresponding to the specified explanatory points, given a model for local rotations and an error term sampler.

Usage

```
simulate_regression(explanatory_points, local_rotation_composer,
  local_error_sampler)
```

Arguments

explanatory_points

        An *m*-by-3 matrix whose rows contain the Cartesian coordinates of the points at which the regression will be simulated.

local_rotation_composer

        A function that returns a 3-length numeric vector representing the independent components of a skew symmetric matrix local to an explanatory point, given its Cartesian coordinates.

local_error_sampler

        A function that returns a 3-length numeric vector representing a sampled error term local to an explanatory point, given its Cartesian coordinates.

Details

Let E be the *m*-by-3 matrix of explanatory points. This function will return an *m*-by-*3* matrix whose i-th row is obtained by transposition of the following expression:

$$exp(\Phi(\epsilon(x)))exp(\Phi(s(x)))x$$

where x is the transpose of the i-th row of E. Terms $\epsilon(x)$ and $s(x)$ are obtained by evaluating at x functions `local_error_sampler` and `local_rotation_composer`, respectively, while matrix $\Phi(c)$, for a 3-length numeric vector c, is the skew symmetric matrix having its independent components represented by the entries of c (for a thorough discussion, see function `get_skew_symmetric_matrix`).

Functions `local_error_sampler` and `local_rotation_composer` must be prototyped as having one argument, `point`, representing the Cartesian coordinates of a point on a 3D sphere, and returning a non `NULL` numerical object having length equal to 3.

Value

An *m*-by-*3* matrix whose rows contain the Cartesian coordinates of the response points corresponding to the explanatory points.

See Also

Other Regression functions: cross_validate_concentration, fit_regression, get_equally_spaced_points, get_skew_symmetric_matrix, simulate_rigid_regression, weight_explanatory_points

Examples

```
library(nprotreg)

# Define a matrix of explanatory points.

explanatory_points <- rbind(
  cbind(.5, 0, .8660254),
  cbind(-.5, 0, .8660254),
  cbind(1, 0, 0),
  cbind(0, 1, 0),
  cbind(-1, 0, 0),
  cbind(0, -1, 0),
  cbind(.5, 0, -.8660254),
  cbind(-.5, 0, -.8660254)
)

# Define a local rotation composer.

local_rotation_composer <- function(point) {
  independent_components <- (1 / 2) *
    c(exp(2.0 * point[3]), - exp(2.0 * point[2]), exp(2.0 * point[1]))
}

# Define a local error sampler.

local_error_sampler <- function(point) {
  rnorm(3)
}

# Get the corresponding 8-by-3 matrix of response points.
# Rows corresponds to explanatory points,
# columns to Cartesian coordinates.

response_points <- simulate_regression(explanatory_points,
                                       local_rotation_composer,
                                       local_error_sampler)

# Get the response point corresponding to the second
# explanatory point.

cat("Response point corresponding to the second explanatory point: \n")
cat(response_points[2, ])
```

---

simulate_rigid_regression

Simulates a Rigid 3D Spherical Regression.

Description

Returns the response points corresponding to the specified explanatory points, given a rigid rotation model and an error term sampler.

Usage

```
simulate_rigid_regression(explanatory_points, rotation_matrix,
  local_error_sampler)
```

Arguments

explanatory_points

> An *m*-by-3 matrix whose rows contain the Cartesian coordinates of the points at which the regression will be simulated.

rotation_matrix

> A 3-by-3 rotation matrix.

local_error_sampler

> A function that returns a 3-length numeric vector representing a sampled error term local to an explanatory point, given its Cartesian coordinates.

Details

Let E be the *m*-by-3 matrix of explanatory points. This function will return an *m*-by-*3* matrix whose i-th row is obtained by transposition of the following expression:

$$exp(\Phi(\epsilon(x)))Rx$$

where x is the transpose of the i-th row of E and R is rotation_matrix. Term $\epsilon(x)$ is obtained by evaluating at x function local_error_sampler, while matrix $\Phi(c)$, for a 3-length numeric vector c, is the skew symmetric matrix having its independent components represented by the entries of c (for a thorough discussion, see function get_skew_symmetric_matrix).

Function local_error_sampler must be prototyped as having one argument, point, representing the Cartesian coordinates of a point on a 3D sphere, and returning a non NULL numerical object having length equal to 3.

Value

An *m*-by-*3* matrix whose rows contain the Cartesian coordinates of the response points corresponding to the explanatory points.

See Also

Other Regression functions: cross_validate_concentration, fit_regression, get_equally_spaced_points, get_skew_symmetric_matrix, simulate_regression, weight_explanatory_points

Examples

```
library(nprotreg)
library(expm)

# Define a matrix of explanatory points.

explanatory_points <- rbind(
  cbind(.5, 0, .8660254),
  cbind(-.5, 0, .8660254),
  cbind(1, 0, 0),
  cbind(0, 1, 0),
  cbind(-1, 0, 0),
  cbind(0, -1, 0),
  cbind(.5, 0, -.8660254),
  cbind(-.5, 0, -.8660254)
)

# Define a rotation matrix.

rotation_matrix <- expm(
  get_skew_symmetric_matrix(
    cbind(-0.36, 0.48, -0.8)
  )
)

# Define a local error sampler.

local_error_sampler <- function(point) {
  rnorm(3)
}

# Get the corresponding 8-by-3 matrix of response points.
# Rows corresponds to explanatory points,
# columns to Cartesian coordinates.

response_points <- simulate_rigid_regression(explanatory_points,
                                              rotation_matrix,
                                              local_error_sampler)

# Get the response point corresponding to the second
# explanatory point.

cat("Response point corresponding to the second explanatory point: \n")
cat(response_points[2, ])
```

weight_explanatory_points

Weights the Specified Explanatory Points in a 3D Spherical Regression.

Description

Returns the weights assigned to the specified explanatory points for each evaluation point under study, given a concentration parameter.

Usage

```
weight_explanatory_points(evaluation_points, explanatory_points, concentration)
```

Arguments

evaluation_points

An *n*-by-3 matrix whose rows contain the Cartesian coordinates of the points on which the regression will be estimated.

explanatory_points

An *m*-by-3 matrix whose rows contain the Cartesian coordinates of the explanatory points used to calculate the regression estimators.

concentration   A non negative scalar whose reciprocal value is proportional to the bandwidth applied while estimating a spherical regression model.

Details

Let X be the *m*-by-3 matrix of explanatory points, and E the *n*-by-3 matrix of evaluation points, and κ the concentration parameter. This function will return an *m*-by-*n* matrix whose (i, j) entry is defined as follows:

$$exp(\kappa(s(i, j) - 1))$$

where $s(i, j)$ is the scalar product of the i-th row of X and the j-th row of E.

Value

An *m*-by-*n* matrix whose *j*-th column contains the weights assigned to the explanatory points while analyzing the *j*-th evaluation point.

See Also

Other Regression functions: cross_validate_concentration, fit_regression, get_equally_spaced_points, get_skew_symmetric_matrix, simulate_regression, simulate_rigid_regression

Examples

```
library(nprotreg)

# Define a matrix of evaluation points.

north_pole <- cbind(0, 0, 1)
south_pole <- cbind(0, 0, -1)
evaluation_points <- rbind(north_pole, south_pole)

# Define a matrix of explanatory points

explanatory_points <- rbind(
  cbind(.5, 0, .8660254),
  cbind(-.5, 0, .8660254),
  cbind(1, 0, 0),
  cbind(0, 1, 0),
  cbind(-1, 0, 0),
  cbind(0, -1, 0),
  cbind(.5, 0, -.8660254),
  cbind(-.5, 0, -.8660254)
)

# Define a value for the concentration parameter.

concentration <- 1.0

# Get the corresponding 8-by-2 matrix of weights.
# Columns corresponds to evaluation points,
# rows to explanatory ones.

weights <- weight_explanatory_points(evaluation_points,
                                     explanatory_points,
                                     concentration)

# Get the weights assigned to the explanatory points
# while analyzing the second evaluation point.

cat("Weights assigned while analyzing the second evaluation point: \n")
cat(weights[, 2])
```

Index