Package 'ClimaRep'

July 21, 2025

Title Estimating Climate Representativeness

Version 0.6

Description Offers tools to estimate the climate representativeness of defined areas and quantifies and analyzes its transformation under future climate change scenarios. Approaches described in Mingarro and Lobo (2018) <doi:10.32800/abc.2018.41.0333> and Mingarro and Lobo (2022) <doi:10.1017/S037689292100014X>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Imports ggplot2, terra, utils, stats, sf, tidyterra

Config/testthat/edition 3

NeedsCompilation no

Author Mario Mingarro Lopez [aut, cre] (ORCID: <https://orcid.org/0000-0003-3977-7944>)

Maintainer Mario Mingarro Lopez <mario_mingarro@mncn.csic.es>

Repository CRAN

Date/Publication 2025-06-27 13:30:13 UTC

Contents

mh_overlay *Overlay Mahalanobis-based Climate Representativeness Classifications*

Description

Combines multiple single-layer rasters (`tif`), outputs from `mh_rep` or `mh_rep_ch` for different input polygons, into a multi-layered `SpatRaster`.

This function handles inputs from both `mh_rep` (which primarily contains **Represented** areas) and `mh_rep_ch` (which includes **Retained**, **Lost**, and **Novel** areas). The output layers consistently represent counts of each input.

Usage

```
mh_overlay(folder_path)
```

Arguments

`folder_path`	`character`. The path to the directory containing the classification rasters (`.tif`) generated by `mh_rep` or `mh_rep_ch`. These rasters should primarily contain the categories: `1` (Retained/Represented), `2` (Lost), and `3` (Novel). Category `0` (Non-represented) will be ignored for the RGB output.

Details

This function streamlines the aggregation of Climate Representativeness classifications. It is designed to work with outputs from both `mh_rep` and `mh_rep_ch`.

For each of the three key categories (Lost, Retained/Represented, Novel), the function:

1. Identifies and reads all `.tif` files within the `folder_path`.
2. For each input raster, it creates a binary layer: `1` if the cell's value matches the target category (e.g., `2` for 'Lost'), and `0` otherwise.
3. Sums these binary layers to generate a cumulative count for that specific category at each grid cell.

The three resulting count layers (Lost, Retained, Novel) are then consistently stacked in the following order:

- First layer (Red): Cumulative count of **Lost**.
- Second layer (Green): Cumulative count of **Retained**.
- Third layer (Blue): Cumulative count of **Novel**.

This fixed order ensures that the output `SpatRaster` is immediately ready for direct RGB visualization using `terra::plotRGB()`, where the color mixtures will intuitively reflect the spatial agreement of these change types.

The output SpatRaster contains raw counts. While terra::plotRGB() often handles stretching for visualization, users might normalize these counts manually (e.g., to 0-255) for finer control over visual contrast.

A new subfolder named overlay/ will be created within the folder_path. The resulting three-layered RGB will be saved as ClimaRep_overlay.tif inside this new overlay/ subfolder.

Value

Writes the multi-layered (ClimaRep_overlay.tif) outputs to disk in a new overlay subfolder within the folder_path. When mh_rep_ch results are used, the output layers consistently represent counts for **Lost** (Red), **Retained** (Green), and **Novel** (Blue) categories across all input rasters. Designed for direct RGB plotting. When mh_rep results are used, the output layers consistently represent counts for **Represented** categories across all input rasters.

Examples

```
ClimaRep_overlay <- ClimaRep::mh_overlay(folder_path = system.file("extdata", package = "ClimaRep"))
terra::plotRGB(ClimaRep_overlay)
terra::plot(ClimaRep_overlay)
```

---

mh_rep *Multivariate Climate Representativeness Analysis*

Description

This function calculates Mahalanobis-based Climate Representativeness for input polygon within a defined area.

Representativeness is assessed by comparing the multivariate climate conditions of each cell, of the reference climate space (climate_variables), with the climate conditions within each specific input polygon.

Usage

```
mh_rep(
  polygon,
  col_name,
  climate_variables,
  th = 0.95,
  dir_output = file.path(tempdir(), "ClimaRep"),
  save_raw = FALSE
)
```

Arguments

polygon	An sf object containing the defined areas. **Must have the same CRS as** climate_variables.
col_name	character. Name of the column in the polygon object that contains unique identifiers for each polygon.

climate_variables
: A SpatRaster stack of climate variables representing the conditions. Its CRS will be used as the reference system.

th
: numeric (0-1). Percentile threshold used to define representativeness. Cells with a Mahalanobis distance below or equal to the th are classified as representative (default: 0.95).

dir_output
: character. Path to the directory where output files will be saved. The function will create subdirectories within this path.

save_raw
: logical. If TRUE, saves the intermediate continuous Mahalanobis distance rasters calculated for each polygon before binary classification. The final binary classification rasters are always saved (default: FALSE).

Details

This function performs a multivariate analysis using Mahalanobis distance to assess the Climate Representativeness of input polygons for a single time period.

Crucially, this function assumes that all spatial inputs (polygon, climate_variables) are already correctly aligned and share the same Coordinate Reference System (CRS). If inputs do not meet these criteria, the function will stop with an informative error.

Here are the key steps:

1. Checking of spatial inputs: Ensures that polygon and climate_variables have matching CRSs.
2. Calculate the multivariate covariance matrix using climate data from all cells.
3. For each polygon in the polygon object:
    - Crop and mask the climate variables raster (climate_variables) to the boundary of the current polygon.
    - Calculate the multivariate mean using the climate data from the previous step. This defines the climate centroid for the current polygon.
    - Calculate the Mahalanobis distance for each cell relative to the centroid and covariance matrix.
    - Apply the specified threshold (th) to Mahalanobis distances to determine which cells are considered representative. This threshold is a percentile of the Mahalanobis distances within the current polygon.
    - Classify each cell as Representative = 1 (Mahalanobis distance $\leq$ th) or Non-Representative = 0 (Mahalanobis distance $>$ th).
4. Saves the binary classification raster (.tif) and generates a corresponding visualization map (.jpeg) for each polygon. These are saved within the specified output directory (dir_output).

It is important to note that Mahalanobis distance is sensitive to collinearity among variables. While the covariance matrix accounts for correlations, it is strongly recommended that the climate_variables are not strongly correlated. Consider performing a collinearity analysis beforehand, perhaps using the vif_filter function from this package.

Value

Writes the following outputs to disk within subdirectories of `dir_output`:

- Classification (`.tif`) rasters: Binary rasters (`0` for **Non-representative** and`1` for **Representative**) for each input polygon are saved in the `Representativeness/` subdirectory.
- Visualization (`.jpeg`) maps: Image files visualizing the classification results for each `polygon` are saved in the `Charts/` subdirectory.
- Raw Mahalanobis distance rasters: Optionally saved as `.tif` files in the `Mh_Raw/` subdirectory if `save_raw = TRUE`.

Examples

```
library(terra)
library(sf)
set.seed(2458)
n_cells <- 100 * 100
r_clim_present <- terra::rast(ncols = 100, nrows = 100, nlyrs = 7)
values(r_clim_present) <- c(
   (terra::rowFromCell(r_clim_present, 1:n_cells) * 0.2 + rnorm(n_cells, 0, 3)),
   (terra::rowFromCell(r_clim_present, 1:n_cells) * 0.9 + rnorm(n_cells, 0, 0.2)),
   (terra::colFromCell(r_clim_present, 1:n_cells) * 0.15 + rnorm(n_cells, 0, 2.5)),
   (terra::colFromCell(r_clim_present, 1:n_cells) +
     (terra::rowFromCell(r_clim_present, 1:n_cells)) * 0.1 + rnorm(n_cells, 0, 4)),
   (terra::colFromCell(r_clim_present, 1:n_cells) /
     (terra::rowFromCell(r_clim_present, 1:n_cells)) * 0.1 + rnorm(n_cells, 0, 4)),
   (terra::colFromCell(r_clim_present, 1:n_cells) *
     (terra::rowFromCell(r_clim_present, 1:n_cells) + 0.1 + rnorm(n_cells, 0, 4))),
   (terra::colFromCell(r_clim_present, 1:n_cells) *
     (terra::colFromCell(r_clim_present, 1:n_cells) + 0.1 + rnorm(n_cells, 0, 4)))
)
names(r_clim_present) <- c("varA", "varB", "varC", "varD", "varE", "varF", "varG")
terra::crs(r_clim_present) <- "EPSG:4326"

vif_result <- ClimaRep::vif_filter(r_clim_present, th = 5)
print(vif_result$summary)
r_clim_present_filtered <- vif_result$filtered_raster
hex_grid <- sf::st_sf(
   sf::st_make_grid(
     sf::st_as_sf(
       terra::as.polygons(
         terra::ext(r_clim_present_filtered))),
     square = FALSE)
     )
sf::st_crs(hex_grid) <- "EPSG:4326"
polygons <- hex_grid[sample(nrow(hex_grid), 2), ]
polygons$name <- c("Pol_A", "Pol_B")
terra::plot(r_clim_present_filtered[[1]])
terra::plot(polygons, add = TRUE, color = "transparent", lwd = 3)

ClimaRep::mh_rep(
   polygon = polygons,
```

```
  col_name = "name",
  climate_variables = r_clim_present_filtered,
  th = 0.95,
  dir_output = file.path(tempdir(), "ClimaRep"),
  save_raw = TRUE
  )
```

`mh_rep_ch` *Multivariate Temporal Climate Representativeness Change Analysis*

Description

This function calculates Mahalanobis-based Climate Representativeness (or forward climate analogs) for input polygon across two time periods (present and future) within a defined area.

The function identifies areas of climate representativeness **Retained**, **Lost**, or **Novel**.

Representativeness is assessed by comparing the multivariate climate conditions of each cell, of the reference climate space (`present_climate_variables` and `future_climate_variables`), with the climate conditions within each specific input `polygon`.

Usage

```
mh_rep_ch(
  polygon,
  col_name,
  present_climate_variables,
  future_climate_variables,
  study_area,
  th = 0.95,
  model,
  year,
  dir_output = file.path(tempdir(), "ClimaRep"),
  save_raw = FALSE
)
```

Arguments

`polygon`	An `sf` object containing the defined areas. **Must have the same CRS as** `present_climate_variables`.
`col_name`	`character`. Name of the column in the `polygon` object that contains unique identifiers for each polygon.
`present_climate_variables`	A `SpatRaster` stack of climate variables representing present conditions. Its CRS will be used as the reference system.
`future_climate_variables`	A `SpatRaster` stack containing the same climate variables as `present_climate_variables` but representing future projected conditions. **Must have the same CRS, extent, and resolution as** `present_climate_variables`.

study_area	A single sf polygon. **Must have the same CRS as** present_climate_variables.
th	numeric (0-1). Percentile threshold used to define representativeness. Cells with a Mahalanobis distance below or equal to the th are classified as representative (default: 0.95).
model	character. Name or identifier of the climate model used (e.g., "MIROC6"). This parameter is used in output filenames and subdirectory names, allowing for better file management.
year	character. Year or period of future climate data (e.g., "2070"). This parameter is used in output filenames and subdirectory names, allowing for better file management.
dir_output	character. Path to the directory where output files will be saved. The function will create subdirectories within this path.
save_raw	logical. If TRUE, saves the intermediate continuous Mahalanobis distance rasters calculated for each polygon before binary classification. The final binary classification rasters are always saved (default: FALSE).

Details

This function extends the approach used in mh_rep to assess Changes in Climate Representativeness (or forward climate analogs) over time. While mh_rep() calculates representativeness in a single scenario, mh_rep_ch() adapts this by using the mean from the present polygon but a covariance matrix derived from the overall climate space across both present and future periods combined.

Crucially, this function assumes that all spatial inputs (polygon, present_climate_variables, future_climate_variables, study_area) are already correctly aligned and share the same Coordinate Reference System (CRS) and consistent spatial properties (extent, resolution). If inputs do not meet these criteria, the function will stop with an informative error.

Here are the key steps:

1. Checking of spatial inputs: Ensures that present_climate_variables, future_climate_variables, polygon, and study_area all have matching CRSs, and that present_climate_variables and future_climate_variables share identical extents and resolutions.
2. Calculate the multivariate covariance matrix using climate data from all cells for both present and future time periods combined.
3. For each polygon in the polygon object:
   - Crop and mask the current climate variables raster (present_climate_variables) to the boundary of the current polygon.
   - Calculate the multivariate mean using the climate data from the previous step. This defines the climate centroid for the current polygon. Calculate the Mahalanobis distance for each cell relative to the centroid and the overall present and future covariance matrix. This results in a Mahalanobis distance raster for the present period and another for the future period.
   - Apply the specified threshold (th) to Mahalanobis distances to determine which cells are considered representative. This threshold is a percentile of the Mahalanobis distances within the current polygon.
   - Classify each cells, for both present and future periods, as Representative = 1 (Mahalanobis distance $\leq$ th) or Non-Representative = 0 (Mahalanobis distance $>$ th).

4. Compares the binary representativeness of each cell between the present and future periods and determines cells where conditions are:
   - `0`: **Non-represented**: Cells that are outside the defined Mahalanobis threshold in both present and future periods.
   - `1`: **Retained**: Cells that are within the defined Mahalanobis threshold in both present and future periods.
   - `2`: **Lost**: Cells that are within the defined Mahalanobis threshold in the present period but outside it in the future period.
   - `3`: **Novel**: Cells that are outside the defined Mahalanobis threshold in the present period but within it in the future period.
5. Saves the classification raster (`.tif`) and generates a corresponding visualization map (`.jpeg`) for each polygon. These are saved within the specified output directory (`dir_output`). All files are saved using the `model` and `year` parameters for better file management.

It is important to note that Mahalanobis distance assumes is sensitive to collinearity among variables. While the covariance matrix accounts for correlations, it is strongly recommended that the climate variables (`present_climate_variables`) are not strongly correlated. Consider performing a collinearity analysis beforehand, perhaps using the `vif_filter` function from this package.

Value

Writes the following outputs to disk within subdirectories of `dir_output`:

- Classification (`.tif`) change rasters: Change category rasters (`0` for **Non-representative**, `1` for **Retained**, `2` for **Lost** and `3` for **Novel**) for each input polygon are saved in the `Change/` subdirectory.
- Visualization (`.jpeg`) maps: Image files visualizing the change classification results for each `polygon` are saved in the `Charts/` subdirectory.
- Raw Mahalanobis distance rasters: Optionally, they are saved as `.tif` files in the `Mh_Raw_Pre/` and `Mh_Raw_Fut/` subdirectories if `save_raw = TRUE`.

Examples

```
library(terra)
library(sf)
set.seed(2458)
n_cells <- 100 * 100
r_clim_present <- terra::rast(ncols = 100, nrows = 100, nlyrs = 7)
values(r_clim_present) <- c(
   (terra::rowFromCell(r_clim_present, 1:n_cells) * 0.2 + rnorm(n_cells, 0, 3)),
   (terra::rowFromCell(r_clim_present, 1:n_cells) * 0.9 + rnorm(n_cells, 0, 0.2)),
   (terra::colFromCell(r_clim_present, 1:n_cells) * 0.15 + rnorm(n_cells, 0, 2.5)),
   (terra::colFromCell(r_clim_present, 1:n_cells) +
     (terra::rowFromCell(r_clim_present, 1:n_cells)) * 0.1 + rnorm(n_cells, 0, 4)),
   (terra::colFromCell(r_clim_present, 1:n_cells) /
     (terra::rowFromCell(r_clim_present, 1:n_cells)) * 0.1 + rnorm(n_cells, 0, 4)),
   (terra::colFromCell(r_clim_present, 1:n_cells) *
     (terra::rowFromCell(r_clim_present, 1:n_cells) + 0.1 + rnorm(n_cells, 0, 4))),
   (terra::colFromCell(r_clim_present, 1:n_cells) *
```

```
     (terra::colFromCell(r_clim_present, 1:n_cells) + 0.1 + rnorm(n_cells, 0, 4)))
)
names(r_clim_present) <- c("varA", "varB", "varC", "varD", "varE", "varF", "varG")
terra::crs(r_clim_present) <- "EPSG:4326"

vif_result <- ClimaRep::vif_filter(r_clim_present, th = 5)
print(vif_result$summary)
r_clim_present_filtered <- vif_result$filtered_raster
r_clim_future <- r_clim_present_filtered + 2
names(r_clim_future) <- names(r_clim_present_filtered)
hex_grid <- sf::st_sf(
   sf::st_make_grid(
     sf::st_as_sf(
       terra::as.polygons(
         terra::ext(r_clim_present_filtered))),
     square = FALSE))
sf::st_crs(hex_grid) <- "EPSG:4326"
polygons <- hex_grid[sample(nrow(hex_grid), 2), ]
polygons$name <- c("Pol_1", "Pol_2")
study_area_polygon <- sf::st_as_sf(terra::as.polygons(terra::ext(r_clim_present_filtered)))
sf::st_crs(study_area_polygon) <- "EPSG:4326"
terra::plot(r_clim_present_filtered[[1]])
terra::plot(polygons, add = TRUE, color = "transparent", lwd = 3)
terra::plot(study_area_polygon, add = TRUE, col = "transparent", lwd = 3, border = "red")

ClimaRep::mh_rep_ch(
   polygon = polygons,
   col_name = "name",
   present_climate_variables = r_clim_present_filtered,
   future_climate_variables = r_clim_future,
   study_area = study_area_polygon,
   th = 0.95,
   model = "ExampleModel",
   year = "2070",
   dir_output = file.path(tempdir(), "ClimaRepChange"),
   save_raw = TRUE)
```

`vif_filter` *Filter SpatRaster Layers based on Variance Inflation Factor (VIF)*

Description

This function iteratively filters layers from a `SpatRaster` object by removing the one with the highest Variance Inflation Factor (VIF) that exceeds a specified threshold (`th`).

Usage

```
vif_filter(x, th = 5)
```

Arguments

x	A SpatRaster object containing the layers (variables) to filter. Must contain two or more layers.
th	A numeric value specifying the Variance Inflation Factor (VIF) threshold. Layers whose VIF exceeds this threshold are candidates for removal in each iteration (default: 5).

Details

This function implements a common iterative procedure to reduce multicollinearity among raster layers by removing variables with high Variance Inflation Factor (VIF). The VIF for a specific predictor indicates how much the variance of its estimated coefficient is inflated due to its linear relationships with all other predictors in the model. Conceptually, it is based on the proportion of variance that predictor shares with the other independent variables. A high VIF value suggests a high degree of collinearity with other predictors (values exceeding 5 or 10 are often considered problematic; see O'Brien, 2007). In this context, the function also provides the Pearson correlation matrix between all initial variables.

Key steps:

1. Validate inputs: Ensures x is a SpatRaster with at least two layers and th is a valid numeric value.
2. Convert the input SpatRaster (x) to a data.frame, retaining only unique rows if x has many cells and few unique climate values.
3. Remove rows containing any NA values across all variables from the data.frame.
4. In each iteration, calculate the VIF for all variables currently remaining in the dataset.
5. Identify the variable with the highest VIF among the remaining variables.
6. If this highest VIF value is greater than the threshold (th), remove the variable with the highest VIF from the dataset, and the loop continues with the remaining variables.
7. This iterative process repeats until the highest VIF among the remaining variables is less than or equal to $\leq$ th, or until only one variable remains in the dataset.

The output of vif_filter returns a list object with a filtered SpatRaster object and a statistics summary.

The SpatRaster object containing only the variables that were kept and also provides a comprehensive summary printed to the console. The summary list including:

- The original Pearson's correlation matrix between all initial variables.
- The variables names that were kept and those that were excluded.
- The final VIF values for the variables retained after the process.

The internal VIF calculation includes checks to handle potential numerical instability, such as columns with zero or near-zero variance and cases of perfect collinearity among variables, which could otherwise lead to errors (e.g., infinite VIFs or issues with matrix inversion). Variables identified as having infinite VIF due to perfect collinearity are prioritized for removal.

References: O'brien (2007) A Caution Regarding Rules of Thumb for Variance Inflation Factors. Quality & Quantity, 41: 673–690. doi:10.1007/s11135-006-9018-6

Value

A SpatRaster object containing only the layers retained by the VIF filtering process.

Examples

```
library(terra)
library(sf)

set.seed(2458)
n_cells <- 100 * 100
r_clim <- terra::rast(ncols = 100, nrows = 100, nlyrs = 7)
values(r_clim) <- c(
   (rowFromCell(r_clim, 1:n_cells) * 0.2 + rnorm(n_cells, 0, 3)),
   (rowFromCell(r_clim, 1:n_cells) * 0.9 + rnorm(n_cells, 0, 0.2)),
   (colFromCell(r_clim, 1:n_cells) * 0.15 + rnorm(n_cells, 0, 2.5)),
   (colFromCell(r_clim, 1:n_cells) +
     (rowFromCell(r_clim, 1:n_cells)) * 0.1 + rnorm(n_cells, 0, 4)),
   (colFromCell(r_clim, 1:n_cells) /
     (rowFromCell(r_clim, 1:n_cells)) * 0.1 + rnorm(n_cells, 0, 4)),
   (colFromCell(r_clim, 1:n_cells) *
     (rowFromCell(r_clim, 1:n_cells) + 0.1 + rnorm(n_cells, 0, 4))),
   (colFromCell(r_clim, 1:n_cells) *
     (colFromCell(r_clim, 1:n_cells) + 0.1 + rnorm(n_cells, 0, 4))))
names(r_clim) <- c("varA", "varB", "varC", "varD", "varE", "varF", "varG")
terra::crs(r_clim) <- "EPSG:4326"
terra::plot(r_clim)

vif_result <- ClimaRep::vif_filter(r_clim, th = 5)
print(vif_result$summary)
r_clim_filtered <- vif_result$filtered_raster
terra::plot(r_clim_filtered)
```

Index