Package 'ClimaRep'

May 7, 2026

Title Estimating Climate Representativeness

Version 1.0

Description Offers tools to estimate the climate representativeness of reference polygons and quantifies 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 [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3977-7944>),
Gabriel del Barrio [ctb] (ORCID:
<https://orcid.org/0000-0001-6146-9433>),
Jorge M. Lobo [ctb] (ORCID: <https://orcid.org/0000-0002-3152-4769>)

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

Repository CRAN

Date/Publication 2025-10-22 19:30:07 UTC

Contents

mh_rep 2
mh_rep_ch 4
rep_overlay 8
vif_filter 10

Index **12**

mh_rep *Multivariate climate representativeness analysis*

Description

This function calculates Mahalanobis-based climate representativeness for input polygon within a defined area.

The function categorizes cells based on how their climate representativeness, labeling them as Non-representative and Representative.

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,
  study_area,
  th = 0.95,
  dir_output = file.path(tempdir(), "ClimaRep"),
  save_raw = FALSE
)
```

Arguments

`polygon`	An `sf` object containing the defined areas. **Its CRS will be used as the reference system.**
`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.
`study_area`	An `sf` object defining the study area. The analysis will be limited to this area.
`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.

Key steps:

1. Ensures that `climate_variables` and `study_area` have matching CRSs with `polygon` by automatically transforming them if needed. The CRS of `polygon` will be used as the reference system.
2. Crops and masks the `climate_variables` raster to the `study_area` to limit all subsequent calculations to the area of interest.
3. Calculate the multivariate covariance matrix using climate data from all cells.
4. 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`).
5. 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")
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(
   polygon = polygons,
   col_name = "name",
   climate_variables = r_clim_present_filtered,
   study_area = study_area_polygon,
   th = 0.95,
   dir_output = file.path(tempdir(), "ClimaRep"),
   save_raw = TRUE
   )
```

---

mh_rep_ch *Multivariate temporal climate tepresentativeness 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 categorizes cells based on how their climate representativeness changes, labeling them as Stable, 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. **Its CRS will be used as the reference system.**
`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 current conditions.
`future_climate_variables`	A `SpatRaster` stack containing the same climate variables as `present_climate_variables` but representing future projected conditions.
`study_area`	A single `sf` polygon.
`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.

Here are the key steps:

1. Checking CRS: Ensures that `future_climate_variables`, `climate_variables`, and `study_area` have matching CRSs with `polygon` by automatically transforming them if needed. The CRS of `polygon` will be used as the reference system.
2. Crops and masks the `climate_variables` and `future_climate_variables` raster to the `study_area` to limit all subsequent calculations to the area of interest.
3. Calculate the multivariate covariance matrix using climate data from all cells for both present and present-future time periods combined.
4. For each polygon in the `polygon` object:
   - Crop and mask the present 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 present-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 Unsuitable = `0` (Mahalanobis distance $>$ `th`).
5. Compares the binary representativeness of each cell between the present and future periods and determines cells where conditions are:
   - `0`: **Unsuitable**: Cells that are outside the defined Mahalanobis threshold in both present and future periods.
   - `1`: **Stable**: Cells that are within the defined Mahalanobis threshold in both present and future periods. **Representative** if `Climarep::mh_rep()` is used
   - `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.

6. 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 **Unsuitable**, `1` for **Stable**, `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)
```

`rep_overlay` *Overlay ClimaRep classifications*

Description

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

This function handles inputs from both `ClimaRep::mh_rep()` (which primarily contains **Representative** cells) and `ClimaRep::mh_rep_ch()` (which includes **Stable**, **Lost**, and **Novel** cells). The output layers consistently represent counts of each input.

Usage

```
rep_overlay(folder_path, output_dir = file.path(tempdir(), "ClimaRep_overlay"))
```

Arguments

`folder_path`	`character`. The path to the directory containing the classification rasters (`.tif`) generated by `ClimaRep::mh_rep()` or `ClimaRep::mh_rep_ch()`. These rasters should primarily contain the categories: `1` (Stable/Representative), `2` (Lost), and `3` (Novel). Category `0` (Unsuitable) will be ignored for the RGB output.
`output_dir`	`character`. Path to the directory where the output file will be saved.

Details

This function streamlines the aggregation of ClimaRep classifications. It is designed to work with outputs from both `ClimaRep::mh_rep()` and `ClimaRep::mh_rep_ch`.

For each of the three key categories (Lost, Stable/Representative, 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, Stable, Novel) are then consistently stacked in the following order:

- First layer (Red): Cumulative count of Lost.
- Second layer (Green): Cumulative count of Stable.
- 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-number of `polygons`) for finer 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 following outputs within the directory specified by `output_dir`: When `ClimaRep::mh_rep()` results are used, the output layers consistently represent counts for **Representative** categories across all input rasters. When `ClimaRep::mh_rep_ch()` results are used, the output layers consistently represent counts for **Lost** (Red), **Stable** (Green), and **Novel** (Blue) categories across all input rasters. Designed for direct RGB plotting.

- A multi-layered `SpatRaster` (`ClimaRep_overlay.tif`) for RGB visualization.
- Individual `.tif` files for each band (Lost, Stable, Novel) in `Individual_Bands/` subdirectory.
- `Additional note`: The `ClimaRep::mh_rep()` function analyzes a single period. When its output is used, representative cells are all categorized as Stable (value 1), while other categories (Lost and Novel) have a value of zero.

Examples

```
ClimaRep_overlay <- ClimaRep::rep_overlay(folder_path = system.file("extdata",
                                                                    package = "ClimaRep"),
                                  output_dir = file.path(tempdir(), "rep_overlay_output"))
terra::plotRGB(ClimaRep_overlay)
terra::plot(ClimaRep_overlay)
```

`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 a 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. 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; Legendre & Legendre, 2012).

The filtering process is fully automated and robust:

1. Validates the input and converts the `SpatRaster` to a `data.frame` for calculations.
2. In each step, the function attempts to calculate VIF efficiently using matrix inversion. If perfect collinearity is detected (resulting in a singular matrix that cannot be inverted), the function automatically switches to a more robust method based on linear regressions to handle the situation without an error.
3. The function identifies the variable with the highest VIF among the remaining variables. If its VIF is greater than the threshold (`th`), that variable is removed. The process repeats until all remaining variables are below the threshold or until only one variable remains.

The output is a `list` containing two main components:

- `SpatRaster` object with the variables that were retained after the filtering process.
- A list with a detailed summary of the process, including the names of the kept and excluded variables, the original Pearson's correlation matrix, and the final VIF values for the retained variables.

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). 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(5), 673–690. https://doi.org/10.1007/s11135-006-9018-6 Legendre and Legendre (2012) Numerical Ecology (3a ed.). Elsevier, Netherlands.

Value

A `list` object containing the filtered `SpatRaster` and a summary of the filtering process.

Examples

```
library(terra)

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

mh_rep, 2
mh_rep_ch, 4

rep_overlay, 8

vif_filter, 10