Package ‘geohabnet’

June 23, 2023

Title Analysis of cropland connectivity

Version 0.0.0.9000

Description c("Geographical spatial analysis of cropland connectivity. Allows users to visualize risk index plots for a given set of crops. Xing et al. (2021) <https://doi.org/10.1093/biosci/biaa067>. Package currently support crops sourced from monfreda and spam. The analysis produces 3 maps, mean, variance and difference for the risk index.

License file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports colorspace (>= 2.1.0),
config (>= 0.3.1),
geodata (>= 0.5.8),
geosphere (>= 1.5.18),
igraph (>= 1.4.2),
raster (>= 3.6.20),
rworldmap (>= 1.3.6),
terra (>= 1.7.29),
easycsv (>= 1.0.8),
yaml (>= 2.3.7),
stats (>= 4.2.3),
stringr (>= 1.5.0)

Suggests lintr (>= 3.0.2),
mockthat (>= 0.2.8),
rmarkdown,
testthat (>= 3.1.7)

URL https://github.com/GarrettLab/CroplandConnectivity

BugReports https://github.com/GarrettLab/CroplandConnectivity/issues

R topics documented:

`calculate_ccri` *Calculate Cropland Connectivity Risk Index (CCRI) This function calculates CCRI for given parameters using power law and negative exponential. It's required to call `initialize_cropland_data()` before calling this function. It returns a list of CCRI values.*

Description

Calculate Cropland Connectivity Risk Index (CCRI) This function calculates CCRI for given parameters using power law and negative exponential. It's required to call `initialize_cropland_data()` before calling this function. It returns a list of CCRI values.

Usage

```
calculate_ccri(
  link_threshold = 0,
  power_law_metrics =
    the$parameters_config$`CCRI parameters`$NetworkMetrics$InversePowerLaw,
  negative_exponential_metrics =
    the$parameters_config$`CCRI parameters`$NetworkMetrics$NegativeExponential,
  crop_cells_above_threshold,
  thresholded_crop_values
)
```

Arguments

`link_threshold`	A threshold value for link
`power_law_metrics`	A list of power law metrics
`negative_exponential_metrics`	A list of negative exponential metrics
`crop_cells_above_threshold`	A list of crop cells above threshold
`thresholded_crop_values`	A list of crop values above threshold

Value

A list of calculated CCRI values

`calculate_difference_map`
Calculate difference map This function produces a map of difference in rank of cropland harvested area fraction

---

Description

Calculate difference map This function produces a map of difference in rank of cropland harvested area fraction

Usage

```
calculate_difference_map(
  mean_index_raster_diff,
  cropharvest_aggtm_crop,
  cropharvest_agglm_crop,
  zero_extent_raster,
  map_grey_background_ext,
  resolution = the$parameters_config$`CCRI parameters`$Resolution
)
```

Arguments

`mean_index_raster_diff`	A raster object for mean index raster difference
`cropharvest_aggtm_crop`	A raster object for cropland harvest
`cropharvest_agglm_crop`	A raster object for cropland harvest
`zero_extent_raster`	A raster object for zero extent raster
`map_grey_background_ext`	A raster object for map grey background extent
`resolution`	resolution to plot raster and map

`calculate_metrics_weight` *calculate weights for each metric*

Description

calculate weights for each metric

Usage

```
calculate_metrics_weight(
  betweenness_metric = FALSE,
  node_strength = FALSE,
  sum_of_nearest_neighbors = FALSE,
  eigenvector_centrality = FALSE
)
```

Arguments

`betweenness_metric`	A logical value indicating if the betweenness metric should be used
`node_strength`	A logical value indicating if the node strength metric should be used
`sum_of_nearest_neighbors`	A logical value indicating if the sum of nearest neighbors metric should be used
`eigenvector_centrality`	A logical value indicating if the eigenvector centrality metric should be used

Value

A named vector of weights for each metric

Examples

```
# return weights for each metric
calculate_metrics_weight(betweenness_metric = TRUE, node_strength = TRUE,
                         sum_of_nearest_neighbors = TRUE, eigenvector_centrality = TRUE)
```

`calculate_zero_raster` *Calculate raster objects for given extent and resolution This function returns a list of zero raster and map grey background extent*

Description

Calculate raster objects for given extent and resolution This function returns a list of zero raster and map grey background extent

Usage

```
calculate_zero_raster(
  geoscale,
  mean_index_raster,
  resolution = the$parameters_config$`CCRI parameters`$Resolution
)
```

Arguments

`geoscale`	A list of longitude and latitude values for cropland analysis
`mean_index_raster`	A raster object for mean index raster
`resolution`	resolution to plot raster and map

Value

A list of zero raster and map grey background extent

`ccri_negative_exponential` *Calculate negative exponential*

Description

Calculate negative exponential

Usage

```
ccri_negative_exponential(
  dispersal_parameter_gamma_vals,
  link_threshold = 0,
  betweenness_metric = FALSE,
  node_strength = FALSE,
  sum_of_nearest_neighbors = FALSE,
  eigenvector_centrality = FALSE,
  crop_cells_above_threshold = NULL,
  thresholded_crop_values = NULL
)
```

Arguments

`dispersal_parameter_gamma_vals`	A list of gamma values
`link_threshold`	A threshold value for link
`betweenness_metric`	A boolean value to calculate betweenness metric
`node_strength`	A boolean value to calculate node strength
`sum_of_nearest_neighbors`	A boolean value to calculate sum of nearest neighbors

`eigenvector_centrality`	A boolean value to calculate eigenvector centrality
`crop_cells_above_threshold`	A list of crop cells above threshold
`thresholded_crop_values`	A list of crop values above threshold

Value

A list of calculated negative exponential

ccri_neg_exponential_function

Calculate CCRI using negative exponential for given parameters This function calculates CCRI using negative exponential for given parameters based on provided metrics and parameters.

Description

Calculate CCRI using negative exponential for given parameters This function calculates CCRI using negative exponential for given parameters based on provided metrics and parameters.

Usage

```
ccri_neg_exponential_function(
  dispersal_parameter_gamma_val,
  link_threshold,
  distance_matrix = the$distance_matrix,
  thresholded_crop_values,
  crop_raster,
  crop_cells_above_threshold,
  betweenness_metric = FALSE,
  node_strength = FALSE,
  sum_of_nearest_neighbors = FALSE,
  eigenvector_centrality = FALSE
)
```

Arguments

`dispersal_parameter_gamma_val`	A list of gamma values
`link_threshold`	A threshold value for link
`distance_matrix`	distance matrix calculated during initialize_crop_data().
`thresholded_crop_values`	crop values above threshold
`crop_raster`	A raster object for crop raster
`crop_cells_above_threshold`	A list of crop cells above threshold

`betweenness_metric`	A boolean value to calculate betweenness metric
`node_strength`	A boolean value to calculate node strength
`sum_of_nearest_neighbors`	A boolean value to calculate sum of nearest neighbors
`eigenvector_centrality`	A boolean value to calculate eigenvector centrality

Value

A list of calculated CCRI values using negative exponential

`ccri_powerlaw` *Calculate inverse power law*

Description

Calculate inverse power law

Usage

```
ccri_powerlaw(
  dispersal_parameter_beta_vals,
  link_threshold = 0,
  betweenness_metric = FALSE,
  node_strength = FALSE,
  sum_of_nearest_neighbors = FALSE,
  eigenvector_centrality = FALSE,
  crop_cells_above_threshold = NULL,
  thresholded_crop_values = NULL
)
```

Arguments

`dispersal_parameter_beta_vals`	A list of beta values
`link_threshold`	A threshold value for link
`betweenness_metric`	A boolean value to calculate betweenness metric
`node_strength`	A boolean value to calculate node strength
`sum_of_nearest_neighbors`	A boolean value to calculate sum of nearest neighbors
`eigenvector_centrality`	A boolean value to calculate eigenvector centrality
`crop_cells_above_threshold`	A list of crop cells above threshold
`thresholded_crop_values`	A list of crop values above threshold

Value

A list of calculated inverse power law

ccri_powerlaw_function

Calculate CCRI using powerlaw for given parameters This function calculates CCRI using powerlaw for given parameters based on provided metrics and parameters.

Description

Calculate CCRI using powerlaw for given parameters This function calculates CCRI using powerlaw for given parameters based on provided metrics and parameters.

Usage

```
ccri_powerlaw_function(
  dispersal_parameter_beta,
  link_threshold,
  distance_matrix = the$distance_matrix,
  thresholded_crop_values,
  crop_raster,
  crop_cells_above_threshold,
  betweenness_metric = FALSE,
  node_strength = FALSE,
  sum_of_nearest_neighbors = FALSE,
  eigenvector_centrality = FALSE
)
```

Arguments

`dispersal_parameter_beta`	A list of beta values
`link_threshold`	A threshold value for link
`distance_matrix`	distance matrix, generated during initialize_crop_data()
`thresholded_crop_values`	crop values above threshold
`crop_raster`	A raster object for cropland harvest
`crop_cells_above_threshold`	crop cells above threshold. Only contains cells and not the the values.
`betweenness_metric`	A boolean value to calculate betweenness metric
`node_strength`	A boolean value to calculate node strength
`sum_of_nearest_neighbors`	A boolean value to calculate sum of nearest neighbors
`eigenvector_centrality`	A boolean value to calculate eigenvector centrality

Value

A list of calculated CCRI values using powerlaw

`ccri_variance` *Calculate variance of CCRI This function produces a map of variance of CCRI based on inpt parameters*

Description

Calculate variance of CCRI This function produces a map of variance of CCRI based on inpt parameters

Usage

```
ccri_variance(
  indexes,
  variance_mean_index_raster,
  zero_extent_raster,
  map_grey_background_ext,
  resolution = the$parameters_config$`CCRI parameters`$Resolution
)
```

Arguments

`indexes`	A list of index values
`variance_mean_index_raster`	A raster object for variance mean index raster
`zero_extent_raster`	A raster object for zero extent raster
`map_grey_background_ext`	A raster object for map grey background extent
`resolution`	resolution to plot raster and map

`check_metrics` *Check if metrics in the list are valid*

Description

Check if metrics in the list are valid

Usage

```
check_metrics(metrics_list)
```

Arguments

`metrics_list`	A character vector of metrics to check

Value

A named logical vector indicating if each metric is valid or not

Examples

```
# return list of valid metrics
check_metrics(list("betweeness", "invalid_metric"))
```

---

get_cropharvest_raster

Get raster object for crop

Description

Get cropland information in a form of raster object from data source for crop

Usage

```
get_cropharvest_raster(crop_name, data_source)
```

Arguments

`crop_name`	Name of the crop
`data_source`	Data source for cropland information

Value

Raster object

Examples

```
get_cropharvest_raster("avocado", "monfreda")
```

---

get_cropharvest_raster_sum

Get sum of rasters for individual crops

Description

Takes crop names and returns raster object which is sum of raster of individual crops. Currently, only supports crops listed in `geodata::monfredaCrops()`, `geodata::spamCrops()` If crop is present in multiple sources, then their mean is calculated.

Usage

```
get_cropharvest_raster_sum(crop_names)
```

Arguments

`crop_names`	A named list of source along with crop names

Value

Raster object which is sum of all the individual crop rasters

Examples

```
## Not run:
get_cropharvest_raster_sum(list(monfreda = c("wheat", "barley"), spam = c("wheat", "potato")))

## End(Not run)
```

---

get_crop_raster_fromtif *Get raster object from tif file*

Description

This is a wrapper of `raster::raster()` and generates a raster object if provided with a TIF file.

Usage

```
get_crop_raster_fromtif(path_to_tif)
```

Arguments

`path_to_tif`	TIF file

Value

Raster object

Examples

```
## Not run:
# Generate raster for usage
get_crop_raster_fromtif(system.file("avocado_HarvestedAreaFraction.tif", "tifs",
  package = "geohabnet", mustWork = TRUE
))

## End(Not run)
```

---

get_geographic_scales *Get geographical scales from the paramters This function returns a list of geographical scales set global and custom extent in parameters.yaml*

Description

Get geographical scales from the paramters This function returns a list of geographical scales set global and custom extent in parameters.yaml

Usage

```
get_geographic_scales()
```

Value

A list of geographical scales

`get_parameters` *Get Parameters*

Description

Retrieves the parameters and copies the parameter file to the specified output path.

Usage

```
get_parameters(iwindow = FALSE, out_path = getwd())
```

Arguments

`iwindow`	Logical. If TRUE, prompts the user to select the output directory using a file chooser window.
`out_path`	Character. The output path where the parameter file will be copied.

Value

Character. The path to the copied parameter file.

`get_rasters` *Get rasters object from parameters*

Description

Takes named list of hosts as an input. See host object in `get_parameters()` or `load_parameters()`. Function creates 2 raster object - one is a sum of all the crops specified under sources and other using the provided raster file. See `get_crop_raster_fromtif()`

Usage

```
get_rasters(hosts)
```

Arguments

`hosts`	List of hosts and values. It is synonym to Hosts object in parameters

Value

List of rasters

See Also

`load_parameters()`, `get_parameters()`, `get_crop_raster_fromtif()`, `get_cropharvest_raster()`

Examples

```
# Get default rasters
## Not run:
get_rasters(list(spam = c("wheat"), monfreda = c("avocado"), file = "some_raster.tif"))

## End(Not run)
```

`get_supported_sources` *Get supported sources of crops When provided, `get_cropharvest_raster()` will look for cropland data in this specific source.*

Description

Get supported sources of crops When provided, `get_cropharvest_raster()` will look for cropland data in this specific source.

Usage

```
get_supported_sources()
```

Value

return vector of supported sources. Also used as a lookup to find get raster object.

Examples

```
# Get currently supported sources
get_supported_sources()
```

`global_analysis` *Global cropland density map Only when user has enabled global analysis*

Description

Global cropland density map Only when user has enabled global analysis

Usage

```
global_analysis(
  map_grey_background_extent,
  resolution = the$parameters_config$`CCRI parameters`$Resolution
)
```

Arguments

`map_grey_background_extent`	A raster object for map's grey background
`resolution`	resolution to plot raster and map

`initialize_cropland_data` *intialize cropland data with geiven paramters, it will be later used to calculate CCRI and other functions*

Description

intialize cropland data with geiven paramters, it will be later used to calculate CCRI and other functions

Usage

```
initialize_cropland_data(
  cropharvest_raster,
  resolution = 12,
  geo_scale,
  host_density_threshold = 0,
  agg_method = "sum"
)
```

Arguments

`cropharvest_raster`	A raster object for cropland harvest
`resolution`	resolution to plot raster and map (default: 12)
`geo_scale`	A list of longitude and latitude values for cropland analysis
`host_density_threshold`	A threshold value for cropland density (default: 0)
`agg_method`	A method to aggregate cropland raster (default: "sum")

`load_parameters` *Load Parameters from YAML File*

Description

This function loads parameters from a YAML file and stores them in an object.

Usage

```
load_parameters(filepath = .get_helper_filepath(.kparameters_file_type))
```

Arguments

`filepath`	Path to the YAML file containing the parameters. By default, it takes the value of ".kparameters_file_type" which is set to "parameters.yaml".

Value

object with parameters and values

Examples

```
# Load parameters from default file
load_parameters()
```

`sa_onrasters` *Run analysis*

Description

Run analysis

Usage

```
sa_onrasters(
  cropharvest_raster,
  geo_scales,
  link_thresholds,
  host_density_thresholds,
  aggregate_methods = c("sum", "mean"),
  resolution
)
```

Arguments

`cropharvest_raster`	Raster object which will be used in analysis.
`geo_scales`	List of geographical scales to be used in analysis. The rasters will be cropped to provided geographical scale. Independent analysis is run on each sale.
`link_thresholds`	A list of threshold values for link
`host_density_thresholds`	A list of host density threshold values
`aggregate_methods`	A list of aggregation methods
`resolution`	resolution to plot raster and map

Value

A list of calculated CCRI values using negative exponential

See Also

Use `get_rasters()` to obtain raster object.

sensitivity_analysis_on_cropland_threshold

Calculate sensitivity analysis on cropland harvested area fraction This function calculates sensitivity analysis on cropland harvested area fraction based on provided parameters. It can be used as entry point for sensitivity analysis.

Description

Calculate sensitivity analysis on cropland harvested area fraction This function calculates sensitivity analysis on cropland harvested area fraction based on provided parameters. It can be used as entry point for sensitivity analysis.

Usage

```
sensitivity_analysis_on_cropland_threshold(
  link_thresholds,
  host_density_thresholds,
  geo_scale,
  aggregate_methods = c("sum", "mean"),
  cropharvest_raster,
  resolution
)
```

Arguments

`link_thresholds`	A list of threshold values for link
`host_density_thresholds`	A list of host density threshold values
`geo_scale`	longitude and latitude values for cropland analysis
`aggregate_methods`	A list of aggregation methods
`cropharvest_raster`	A raster object for cropland harvest
`resolution`	resolution to plot raster and map

Value

A list of calculated CCRI values using negative exponential

`sensitivity_analysis_on_geoextent_scale`
Calculate sensitivity analysis on cropland harvested area fraction This function calculates sensitivity analysis on cropland harvested area fraction based on provided parameters. It can be used as entry point for sensitivity analysis.

Description

Calculate sensitivity analysis on cropland harvested area fraction This function calculates sensitivity analysis on cropland harvested area fraction based on provided parameters. It can be used as entry point for sensitivity analysis.

Usage

```
sensitivity_analysis_on_geoextent_scale(
  link_threshold = 0,
  geo_scale,
  aggregate_methods = c("sum", "mean"),
  cropharvest_raster,
  host_density_threshold = 0,
  resolution = 24
)
```

Arguments

`link_threshold`	A threshold value for link
`geo_scale`	A list of longitude and latitude values for cropland analysis
`aggregate_methods`	A list of aggregation methods. It can be sum or mean.
`cropharvest_raster`	A raster object for cropland harvest
`host_density_threshold`	A host density threshold value
`resolution`	resolution to plot raster and map

Value

A list of calculated CCRI values using negative exponential

`sensitivity_analysis_on_link_weight`
Calculate sensitivity analysis on cropland harvested area fraction This function calculates sensitivity analysis on cropland harvested area fraction based on provided parameters. It can be used as entry point for sensitivity analysis.

Description

Calculate sensitivity analysis on cropland harvested area fraction This function calculates sensitivity analysis on cropland harvested area fraction based on provided parameters. It can be used as entry point for sensitivity analysis.

Usage

```
sensitivity_analysis_on_link_weight(
  link_threshold = 0,
  host_density_thresholds,
  geo_scale,
  aggregate_methods,
  cropharvest_raster,
  resolution
)
```

Arguments

`link_threshold`	A threshold value for link
`host_density_thresholds`	A list of host density threshold values
`geo_scale`	A list of longitude and latitude values for cropland analysis
`aggregate_methods`	A list of aggregation methods
`cropharvest_raster`	A raster object for cropland harvest
`resolution`	resolution to plot raster and map

Value

A list of calculated CCRI values using negative exponential

`senstivity_analysis` *Calculate sensitivity analysis on parameters*

Description

This function runs sensitivity analysis on parameters based on provided parameters through `set_parameters()`. It can be used as entry point for sensitivity analysis. Plots results of sensitivity analysis.

Usage

```
senstivity_analysis()
```

Examples

```
## Not run:
# Run analysis on specified parameters.yaml
senstivity_analysis()

## End(Not run)
```

`set_parameters` *Set Parameters*

Description

This function allows you to set the parameters by replacing the existing parameters file with a new one. Use `get_parameters()` to modify the parameter values.

Usage

```
set_parameters(new_parameters_file, iwindow = FALSE)
```

Arguments

`new_parameters_file`	The path to the new parameters file.
`iwindow`	Logical indicating whether to prompt the user to select the new parameters file using a file selection window. Defaults to FALSE.

Value

None

`set_parameters_object` *Set Parameters function*

Description

This function allows you to override existing parameters with new values.

Usage

```
set_parameters_object(
  dispersal_parameter_beta = c(0.5, 1, 1.5),
  dispersal_parameter_gamma = c(0.05, 1, 0.2, 0.3),
  aggregation_strategy = c("sum", "mean"),
  hosts = c("avocado"),
  host_density_threshold = c(0.0015, 0.002, 0.0025),
  link_threshold = c(0, 1e-06, 6e-04),
  resolution = 24,
  global_analysis = FALSE,
  west_extent = c(-24, -180, -58, 60),
  east_extent = c(-140, -34, --58, 60),
  custom_extent = list(c(-115, -75, -5, 32)),
 metrics_inv_powerlaw = c("betweeness", "node_strength", "sum_of_nearest_neighbors",
    "eigenvector_centrality"),
 metrics_neg_exponential = c("betweeness", "node_strength", "sum_of_nearest_neighbors",
    "eigenvector_centrality")
)
```

Arguments

`dispersal_parameter_beta`	Numeric vector of dispersal parameter beta values
`dispersal_parameter_gamma`	Numeric vector of dispersal parameter gamma values
`aggregation_strategy`	Character vector of aggregation strategies
`hosts`	Character vector of hosts
`host_density_threshold`	Numeric vector of host density threshold values
`link_threshold`	Numeric vector of link threshold values
`resolution`	Numeric vector of resolution values
`global_analysis`	Logical vector of global analysis values
`west_extent`	Numeric vector of west extent values
`east_extent`	Numeric vector of east extent values
`custom_extent`	List of custom extent values
`metrics_inv_powerlaw`	Character vector of inv_powerlaw metrics
`metrics_neg_exponential`	Character vector of neg_exponential metrics

Value

TRUE if the parameters were set successfully, FALSE otherwise

See Also

`load_parameters()` `set_parameters()`

Examples

```
## Not run:
# Set parameters
set_parameters_object()
# Set parameters with custom beta values
set_parameters_object(dispersal_parameter_beta = c(0.5, 1, 1.5))

## End(Not run)
```