Package 'geohabnet'

August 7, 2023

Title Analysis of Cropland Connectivity

Version 0.2.0

Description Geographical spatial analysis of cropland connectivity.,
Allows users to visualize risk index plots for a given set of crops.
The functions are developed as an extension to analysis from 10.1093/biosci/biaa067.
Package currently supports crops sourced from monfreda and mapspam.
This analysis produces 3 maps - mean, variance, and difference for the crop risk index.
There are multiple ways in which functions can be used -
generate final outcome and then the intermediate outcomes for more sophisticated use cases.
Refer to vignettes.
\code{\link{sean}()} will set some global variables which can be accessed us-
ing \code{the$} prefix. These values are propagated to other functions for performing opera-
tions such as distance matrix calculation.
\code{parameters.yaml} stores the parameters and values and can be accessed us-
ing \code{\link{get_parameters}()}. Refer it's usage.

License file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports config (>= 0.3.1),
geodata (>= 0.5.8),
geosphere (>= 1.5.18),
igraph (>= 1.4.2),
terra (>= 1.7.29),
easycsv (>= 1.0.8),
yaml (>= 2.3.7),
stats (>= 4.2.3),
stringr (>= 1.5.0),
memoise (>= 2.0.1),
rnaturalearthdata,
graphics (>= 4.2.3),
stringi,
rlang (>= 1.1.1),
viridisLite (>= 0.4.2),
beepr (>= 1.3)

Suggests knitr,
lintr (>= 3.0.2),
mockthat (>= 0.2.8),

rmarkdown,
testthat (>= 3.1.7)

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

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

VignetteBuilder knitr

R topics documented:

`ccri_diff` *Calculate difference map*

Description

This function produces a map of difference b/w mean and sum indexes in rank of cropland harvested area fraction.

Usage

```
ccri_diff(rast, x, y, global, geoscale, res = reso())
```

Arguments

`rast`	A raster object for mean index raster difference
`x`	A raster object for cropland harvest
`y`	A raster object for cropland harvest
`global`	logical. `TRUE` if global analysis is required, `FALSE` otherwise. When `TRUE`, `geoscale` is ignored. Default is `TRUE`.
`geoscale`	vector. geographical scale
`res`	numeric. map resolution

`ccri_mean` *Calculate mean of raster objects*

Description

Wrapper for `terra::mean()`. Calculates mean of list of rasters.

Usage

```
ccri_mean(indexes, global = TRUE, geoscale = NULL, plt = TRUE)
```

Arguments

`indexes`	list of rasters. See details.
`global`	logical. `TRUE` if global analysis is required, `FALSE` otherwise. When `TRUE`, `geoscale` is ignored. Default is `TRUE`.
`geoscale`	vector. geographical scale
`plt`	`TRUE` if need to plot mean map, `FALSE` otherwise and `geoscale`.

`ccri_variance` *Calculate variance of CCRI*

Description

This function produces a map of variance of CCRI based on input parameters

Usage

```
ccri_variance(indexes, rast, global, geoscale, res = reso())
```

Arguments

`indexes`	list of rasters. See details.
`rast`	A raster object. It will be used in calculating variance.
`global`	logical. `TRUE` if global analysis is required, `FALSE` otherwise. When `TRUE`, `geoscale` is ignored. Default is `TRUE`.
`geoscale`	vector. geographical scale
`res`	numeric. map resolution

connectivity *Calculate and plot maps*

Description

```
Calculate mean, variance and difference. The result is produced in form of maps plotted with predefin
Currently, the settings for plot cannot be customized.
Default value is `TRUE` for all logical arguments
```

Usage

```
connectivity(
  indexes,
  global = TRUE,
  geoscale,
  res = reso(),
  pmean = TRUE,
  pvar = TRUE,
  pdiff = TRUE
)
```

Arguments

`indexes`	list of rasters. See details.
`global`	logical. `TRUE` if global analysis is required, `FALSE` otherwise. When `TRUE`, `geoscale` is ignored. Default is `TRUE`.
`geoscale`	vector. geographical scale
`res`	numeric. map resolution
`pmean`	`TRUE` if map of mean should be plotted, `FALSE` otherwise.
`pvar`	`TRUE` if variance map should be plotted, `FALSE` otherwise
`pdiff`	`TRUE` if difference map should be plotted, `FALSE` otherwise

Details

`indexes` are actually risk resulting from operations on crop's raster and parameters provided in either `parameters.yaml` or `sean()`.

It will save all the opted plots using - `pmean`, `pvar` and `pdiff`. File will be saved in `getwd()`.If `interactive()` is `TRUE`, then plots can be seen in active plot window. E.g. Rstudio

`dist_methods` *Distance methods supported*

Description

Contains supported strategies to calculate distance between two points. Use of one the methods in `sean()` or `sensitivity_analysis()`.

Usage

```
dist_methods()
```

Value

vector

`geoscale_param` *Get geographical scales from the parameters*

Description

This function returns a list of geographical scales set in global and custom extent in `parameters.yaml`. If `global` is `TRUE`, the `CustomExt` is ignored.

Usage

```
geoscale_param()
```

Value

A list of geographical scales

`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"), mapspam = c("wheat", "potato")))

## End(Not run)
```

`get_crop_raster_fromtif` *Get raster object from tif file*

Description

This is a wrapper of `terra::rast()` 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
fp <- .get_helper_filepath("avocado")
get_crop_raster_fromtif(fp)
get_crop_raster_fromtif("avocado_HarvestedAreaFraction.tif")


## End(Not run)
```

`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_param_metrics` *Get metrics from parameters*

Description

Get metrics and parameters stored in `parameters.yaml`.

Usage

```
get_param_metrics(params = load_parameters())
```

Arguments

`params`	R object of `load_parameters()`. Default is `load_parameters()`.

Value

List of metrics - parameters and values. See usage.

Examples

```
# Get metrics from parameters
get_param_metrics()
get_param_metrics(load_parameters())
```

`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(mapspam = c("wheat"), monfreda = c("avocado"), file = "some_raster.tif"))

## End(Not run)
```

`get_supported_sources` *Get supported sources of crops*

Description

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_scales` *Global geographical extent*

Description

See geographical extents used in global analysis. Returns eastern and western hemisphere extents. Each extent is in the form of c(Xmin, Xmax, Ymin, Ymax).

Usage

```
global_scales()
```

See Also

`set_global_scales()`

`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()
```

`model_powerlaw` *Calculate risk index using inbuilt models.*

Description

- `model_powerlaw()` calculates risk index using power law.
- `model_neg_exp()` calculates risk index using negative exponential.

Usage

```
model_powerlaw(
  beta,
  link_threshold,
  distance_matrix = the$distance_matrix,
  thresholded_crop_values,
  adj_mat = NULL,
  crop_raster,
  crop_cells_above_threshold,
 metrics = the$parameters_config$`CCRI parameters`$NetworkMetrics$InversePowerLaw
)

model_neg_exp(
  gamma_val,
  link_threshold,
  distance_matrix = the$distance_matrix,
  thresholded_crop_values,
  adj_mat = NULL,
  crop_raster,
  crop_cells_above_threshold,
 metrics = the$parameters_config$`CCRI parameters`$NetworkMetrics$InversePowerLaw
)
```

Arguments

`beta`	A list of beta values. `DispersalParameterBeta` in `parameters.yaml`.
`link_threshold`	A threshold value for link.
`distance_matrix`	distance matrix, generated during `sean()`.
`thresholded_crop_values`	crop values above threshold.
`adj_mat`	Adjacency matrix(optional) representing un-directed graph network. If this is provided, then gamma_val, distance_matrix, link_threshold and thresholded_crop_values are ignored. These ignored parameters are used to generate adjacency matrix internally. This is the only way to use custom adjacency matrix.
`crop_raster`	A raster object for cropland harvest.
`crop_cells_above_threshold`	crop cells above threshold. Only contains cells and not the the values.
`metrics`	A list 2 vectors - metrics and weights.
`gamma_val`	A list of beta values. `DispersalParameterGamma` in `parameters.yaml`.

Details

Network metrics should be passed as a list of vectors e.g. `list(metrics = c("betweeness"), weights = c(100))`. Default values are fetched from `parameters.yaml` and arguments uses the same structure.

Value

risk index

`nn_sum` *Calculation on network matrix.*

Description

These are basically an abstraction of functions under the igraph package. The functions included in this abstraction are:

- `nn_sum()`: Calculates the sum of nearest neighbors `igraph::graph.knn()`.
- `node_strength()`: Calculates the sum of edge weights of adjacent nodes `igraph::graph.strength()`.
- `betweeness()`: Calculates the vertex and edge betweenness based on the number of geodesics `igraph::betweenness()`.
- `ev()`: Calculates the eigenvector centrality of positions within the network `igraph::evcent()`.

- `closeness()`: measures how many steps is required to access every other vertex from a given vertex `igraph::closeness()`.
- `degree()`: number of adjacent edges `igraph::degree()`.
- `page_rank()`: page rank score for vertices `igraph::page_rank()`.

Usage

```
nn_sum(crop_dm, we)

node_strength(crop_dm, we)

betweeness(crop_dm, we)

ev(crop_dm, we)

degree(crop_dm, we)

closeness(crop_dm, we)

page_rank(crop_dm, we)
```

Arguments

`crop_dm`	Distance matrix. In the internal workflow, the distance matrix comes is a result of operations within `sean()` and risk functions.
`we`	Weight in percentage.

Value

Matrix with the mean value based on the assigned weight.

See Also

Other metrics: `supported_metrics()`

`reso` *Get resolution value*

Description

Resolution stored in `parameter.yaml`. If not present it will result default value.

Usage

```
reso()
```

See Also

`set_reso()`

sa_onrasters *Run senstivity analysis*

Description

Same as sensitivity_analysis() but it takes raster object and other parameters as an input.

Usage

```
sa_onrasters(
  rast,
  global = TRUE,
  geoscale,
  link_thresholds,
  host_density_thresholds,
  agg_methods = c("sum", "mean"),
  dist_method = "geodesic",
  res = reso(),
  maps = TRUE
)
```

Arguments

rast	Raster object which will be used in analysis.
global	Logical. TRUE if global analysis, FALSE otherwise. Default is TRUE
geoscale	Vector. Geographical coordinates in the form of c(Xmin, Xmax, Ymin, Ymax)
link_thresholds	vector. link threshold values
host_density_thresholds	vector. host density threshold values
agg_methods	vector. Aggregation methods
dist_method	character. One of the values from dist_methods()
res	numeric. resolution at which operations will run. Default is reso()
maps	logical. TRUE if maps are to be plotted, FALSE otherwise

Details

When global = TRUE, geo_scale is ignored. Instead uses scales from global_scales().

Value

A list of calculated CCRI indices after operations. An index is generated for each combination of paramters. One combination is equivalent to sean() function.

See Also

Use get_rasters() to obtain raster object.

Examples

```
## Not run:
rr <- get_rasters(list(monfreda = c("avocado")))
sa_onrasters(rr[[1]],
            global = FALSE,
            geoscale = c(-115, -75, 5, 32),
            c(0.0001, 0.00004),
            c(0.0001, 0.00005),
            c("sum", "mean"),
            res = 24)
sa_onrasters(rr[[1]],
            global = TRUE,
            link_thresholds = c(0.000001),
            host_density_thresholds = c(0.00015),
            agg_methods = c("sum"),
            res = 24)

## End(Not run)
```

---

sean — *Calculate sensitivity analysis on cropland harvested area fraction*

Description

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

Usage

```
sean(
  link_threshold = 0,
  global = TRUE,
  geoscale,
  agg_methods = c("sum", "mean"),
  dist_method = "geodesic",
  rast,
  host_density_threshold = 0,
  res = reso(),
  maps = TRUE
)
```

Arguments

`link_threshold`	numeric. A threshold value for link
`global`	Logical. `TRUE` if global analysis, `FALSE` otherwise. Default is `TRUE`
`geoscale`	Vector. Geographical coordinates in the form of c(Xmin, Xmax, Ymin, Ymax)
`agg_methods`	vector. Aggregation methods
`dist_method`	character. One of the values from `dist_methods()`
`rast`	Raster object which will be used in analysis.
`host_density_threshold`	A host density threshold value

`res`	numeric. resolution at which operations will run. Default is `reso()`
`maps`	logical. `TRUE` if maps are to be plotted, `FALSE` otherwise

Details

When `global` = `TRUE`, `geoscale` is ignored and `global_scales()` is used

Value

A list of calculated CCRI values using negative exponential

See Also

Uses `connectivity()`

`search_crop` *Search for crop*

Description

It returns the dataset sources in which crop data is available. It's a wrapper around `geodata::spamCrops()` and `geodata::monfredaCrops()`

Usage

```
search_crop(name)
```

Arguments

`name`	name of crop

See Also

`get_supported_sources()`

Examples

```
search_crop("coffee")
search_crop("wheat")
## Not run:
search_crop("jackfruit")

## End(Not run)
```

sensitivity_analysis *Calculate sensitivity analysis on parameters*

Description

This function runs sensitivity analysis on parameters based on parameters provided through set_parameters(). It can be used as an entry point for CCRI. By default, it runs analysis on global sclaesglobal_scales(). After analysis is complete, it will suppress maps for outcomes if maps = FALSE or interactive() is FALSE.

Usage

```
sensitivity_analysis(maps = TRUE, alert = TRUE)
```

Arguments

maps	logical. TRUE if maps are to be plotted, FALSE otherwise
alert	logical. TRUE if beep sound is to be played, FALSE otherwise

Details

vignette("global_analysis", package = "geohabnet")

Value

logical. TRUE if analysis is completed, FALSE otherwise. Errors are not handled.

See Also

sa_onrasters() sean() global_scales() get_parameters() set_parameters() connectivity()

Examples

```
## Not run:
# Run analysis on specified parameters.yaml
sensitivity_analysis()
sensitivity_analysis(FALSE, FALSE)
sensitivity_analysis(TRUE, FALSE)

## End(Not run)
```

`set_global_scales` *Set global geographical extent*

Description

Set the geographical extents used in global analysis. Each extent should be in the form of c(Xmin, Xmax, Ymin, Ymax)

Usage

```
set_global_scales(value)
```

Arguments

`value`	list. Named list of eastern and western hemisphere extents. See usage.

See Also

`global_scales()` `terra::ext()`

Examples

```
## Not run:
set_global_scales(list(east = c(-24, 180, -58, 60), west = c(-140, -34, -58, 60)))

## 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_reso *Set resolution value*

Description

Set resolution to be used in analysis. It doesn't modify the parameters.yaml but instead a currently loaded instance of it. Must be greater than 0 and less than or equal to 48.

Usage

```
set_reso(value)
```

Arguments

value	numeric. Resolution value.

Examples

```
## Not run:
set_reso(24)

## End(Not run)
```

sp_rast *raster for mapspam crop.*

Description

get raster for crop in mapspam dataset

Usage

```
sp_rast(crp)
```

Arguments

crp	character. name of a crop. Case-insensitive.

Details

See geodata::spamCrops() for supported crops.

Value

spatRaster

References

www.mapspam.com/data International Food Policy Research Institute, 2020. Spatially-Disaggregated Crop Production Statistics Data in Africa South of the Sahara for 2017. https://doi.org/10.7910/DVN/FSSKBW, Harvard Dataverse, V2

See Also

`geodata::spamCrops()` `search_crop()`

Examples

```
## Not run:
sp_rast("rice")

## End(Not run)
```

`supported_metrics` *Returns metrics currently supported in the analysis.*

Description

Returns metrics currently supported in the analysis.

Usage

```
supported_metrics()
```

Value

vector of supported metrics.

See Also

Other metrics: `nn_sum()`

Examples

```
supported_metrics()
```