Title: Analysis of Cropland Connectivity

Version: 1.0.1

Date: 2023-10-29

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 Xing et al (2021) <doi:10.1093/biosci/biaa067>. The primary function is sean() and is indicative of how sensitive the risk analysis is to parameters using kernel models.


This analysis produces 3 maps - mean, variance, and difference for the crop risk index. It applies distance functions and graph operations on a network to calculate 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.

sean() will set some global variables which can be accessed using $ prefix. These values are propagated to other functions for performing operations such as distance matrix calculation. parameters.yaml stores the parameters and values and can be accessed using get_parameters(). Refer it's usage.

The objective of this package is to support risk analysis using cropland connectivity on 10 parameters - host crops, density threshold, aggregation and distance method, resolution, geographic extent, link threshold, kernel models, network metrics and maps.

These parameters serves as an input and are used different phases of analysis workflow.

License: GPL-3

Encoding: UTF-8

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, stringr (>= 1.5.0), memoise (>= 2.0.1), graphics, rlang (>= 1.1.1), viridisLite (>= 0.4.2), beepr (>= 1.3), rnaturalearth (>= 0.3.3), tools

**Suggests** devtools, knitr, lintr (>= 3.0.2), mockthat (>= 0.2.8), pkgdown, rmarkdown, testthat (>= 3.1.7)

**URL** https://garrettlab.github.io/CroplandConnectivity/,
https://CRAN.R-project.org/package=geohabnet/,
https://github.com/GarrettLab/CroplandConnectivity/tree/main/geohabnet/,
https://www.garrettlab.com/

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

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Krishna Keshav [aut, cre],
Garrett Lab [cph] (https://garrettlab.com),
Karen Garrett [aut] (<https://orcid.org/0000-0002-6578-1616>),
Aaron Plex [aut] (<https://orcid.org/0000-0001-7317-3090>),
University of Florida [cph, fn] (https://www.ufl.edu)

**Maintainer** Krishna Keshav <kkeshav@ufl.edu>

**Repository** CRAN

**Date/Publication** 2023-10-31 11:30:03 UTC

**R topics documented:**

- ccrit_diff .............. 3
- ccrit_mean .............. 4
- ccrit_variance ........... 4
- connectivity ............ 5
- cropharvest_rast ........ 6
- crops_rast .............. 7
- dist_methods ............ 7
- geoscale_param ........... 8
- get_parameters .......... 8
- get_param_metrics ........ 9
- get_rasters ............. 9
- get_supported_sources ... 10
- global_scales ........... 11
- load_parameters .......... 11
- model_powerlaw ........... 12
- nn_sum .................. 13
- reset_params ............ 14
- reso .................. 15
- sa_onrasters ........... 15
**ccri_diff**

`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**

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

**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.
- `outdir`  
  Character. Output directory for saving raster in TIFF format. Default is `tempdir()`.

**Value**

Invisible NULL.
### ccri_mean

**Calculate mean of raster objects**

**Description**

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

**Usage**

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

**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`.
- `outdir`: Character. Output directory for saving raster in TIFF format. Default is `tempdir()`.

**Value**

Invisible NULL.

### ccri_variance

**Calculate variance of CCRI**

**Description**

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

**Usage**

```r
ccri_variance(
  indexes,
  rast,
  global,
  geoscale,
  res = reso(),
  outdir = tempdir()
)
```
connectivity

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.
outdir Character. Output directory for saving raster in TIFF format. Default is tempdir().

Value

Invisible NULL.

Calculate mean, variance and difference. The result is produced in form of maps plotted with predefined settings. 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,
  outdir = tempdir()
)

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.
outdir Character. Output directory for saving raster in TIFF format. Default is tempdir().
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 provided value of outdir or tempdir(). If interactive() is TRUE, then plots can be seen in active plot window. E.g. Rstudio

Value

Invisible NULL.

References


---

cropharvest_rast

Get raster object for crop

description

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

Usage

cropharvest_rast(crop_name, data_source)

Arguments

crop_name Name of the crop
data_source Data source for cropland information

Value

Raster.

Examples

cropharvest_rast("avocado", "monfreda")
**crops_rast**  
*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**

```r
crops_rast(crop_names)
```

**Arguments**

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

**Value**

SpatRaster. Raster object which is sum of all the individual crop raster

**Examples**

```r
crops_rast(list(monfreda = c("wheat", "barley"), mapspam = c("wheat", "potato")))
```

---

**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**

```r
dist_methods()
```

**Value**

vector

**Examples**

```r
dist_methods()
```
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

Vector. A set of geographical scales

get_parameters  Get Parameters

Description

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

Usage

get_parameters(out_path = tempdir(), iwindow = FALSE)

Arguments

out_path  character. The output path where the parameter file will be copied. Default is temporary directory tempdir()

iwindow  logical. If TRUE, prompts the user to select the output directory using a file chooser window. Default is FALSE

Details

Using configuration file is an alternative to sean()

Value

character. The path to the copied parameter file.

See Also

set_parameters()
get_param_metrics

Examples

get_parameters()
get_parameters(out = tempdir())

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. 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(). This is also a wrapper of crops_rast(). Function creates 2 raster object - one is a sum of all the crops specified under sources and other using the provided raster file. See tiff_torast()

Usage

get_rasters(hosts)

Arguments

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

Value

List of SpatRaster.

See Also

load_parameters(), get_parameters(), tiff_torast(), cropharvest_rast()

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, cropharvest_rast() will look for cropland data in this specific source.

Usage

get_supported_sources()

Value

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

Examples

# Get currently supported sources
get_supported_sources()
global_scales

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()

Details

Separate analysis on geographical scales of eastern and western hemisphere are combined to run global analysis.

Value

List. Named list with scales for eastern and western hemisphere

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 = .param_fp())

Arguments

filepath Path to the YAML file containing the parameters. By default, it takes the value of parameters.yaml in R user’s directory.

Value

object with parameters and values
Examples

# Load parameters from default file
load_parameters()

model_powerlaw

Description

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

Usage

```r
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.
**nn_sum**

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("betweenness"), weights = c(100))`. Default values are fetched from parameters.yaml and arguments uses the same structure.

**Value**

risk index

**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()`.
- `pagerank()`: 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)
pagerank(crop_dm, we)

Arguments

crop_dm Distance matrix. In the internal workflow, the distance matrix comes as 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()

Description

Resets the values in the parameters.yaml file to the default initial values.

Usage

reset_params()

Value

Logical. TRUE if function was successfully executed.

Examples

reset_params()
reso

Get resolution value

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

Usage
reso()

Value
Numeric. Resolution from parameters.yaml. Default is 24.

See Also
set_reso()

sa_onrasters
Run sensitivity 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,
  outdir = tempdir()
)
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.
- **outdir**: Character. Output directory for saving raster in TIFF format. Default is `tempdir()`.

Details

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

Value

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

References


See Also

Use `get_rasters()` to obtain raster object.

Examples

```r
rr <- get_rasters(list(monfreda = c("avocado"))),
res1 <- sa_onrasters(rr[[1]],
    global = FALSE,
    geoscale = c(-115, -75, 5, 32),
    c(0.0001, 0.00004),
    c(0.0001, 0.00005),
```
```r
res = 24)
res2 <- sa_onrasters(rr[[1]],
  global = TRUE,
  link_thresholds = c(0.000001),
  host_density_thresholds = c(0.00015),
  agg_methods = c("sum"),
  res = 24)
```

#### Describe

This function calculates sensitivity analysis on cropland harvested area fraction based on provided parameters. Some parameters are only accessible from parameters.yaml and uses value from here.

#### Usage

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

#### 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).
- **agg_methods**: Vector. Aggregation methods.
- **dist_method**: character. One of the values from dist_methods().
- **link_threshold**: numeric. A threshold value for link.
- **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.
- **outdir**: Character. Output directory for saving raster in TIFF format. Default is tempdir().
Details

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

Value

A list of calculated CCRI values using negative exponential

References


See Also

Uses `connectivity()`

Examples

```r
avocado <- cropharvest_rast("avocado", "monfreda")
risk_indexes <- sean(avocado)
```

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

Value

Logical. Sources in crop data is available.
sensitivity_analysis

See Also

get_supported_sources()

Examples

search_crop("coffee")
search_crop("wheat")
search_crop("jackfruit")

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 scales global_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

Value

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

References


set_global_scales

See Also

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

Examples

# Run analysis on specified parameters.yaml
sensitivity_analysis()
sensitivity_analysis(FALSE, FALSE)
sensitivity_analysis(TRUE, FALSE)

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.

Value

List. Named list with scales for eastern and western hemisphere

See Also

global_scales() terra::ext()

Examples

set_global_scales(list(east = c(-24, 180, -58, 60), west = c(-140, -34, -58, 60)))
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_params, iwindow = FALSE)
```

**Arguments**

- `new_params` : 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

**Examples**

```
param_fp <- get_parameters()
set_parameters(param_fp)
```

---

**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.

**Value**

Invisible TRUE
Examples

    set_reso(24)

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 \texttt{geodata::spamCrops()} for supported crops.

Value

  SpatRaster

References


See Also

  \texttt{geodata::spamCrops()} \texttt{search_crop()}

Examples

  sp_rast("rice")
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()

tiff_torast

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

tiff_torast(path_to_tif)

Arguments

path_to_tif  TIFF file. This is an encoding of map in raster format.

Value

SpatRaster.
Examples

# Generate raster for usage
fp <- paste(tempfile(), ".tif", sep = "")
ret <- utils::download.file(
  "https://geohabnet.s3.us-east-2.amazonaws.com/util-rasters/avocado_HarvestedAreaFraction.tif",
  destfile = fp, method = "auto", mode = "wb")
tiff_torast(fp)
Index

* metrics
  - `nn_sum`, 13
  - `supported_metrics`, 23

betweeness (`nn_sum`), 13

`ccri_diff`, 3
`ccri_mean`, 4
`ccri_variance`, 4

closeness (`nn_sum`), 13

connectivity, 5
connectivity(), 18, 20

cropharvest_rast, 6
cropharvest_rast(), 10
crops_rast, 7
crops_rast(), 9

degree (`nn_sum`), 13
dist_methods, 7
dist_methods(), 16, 17

ev (`nn_sum`), 13

geodata::monfredaCrops(), 7, 18
geodata::spamCrops(), 7, 18, 22
geoscale_param, 8
get_param_metrics, 9
get_parameters, 8
get_parameters(), 9, 10, 20, 21
get_rasters, 9
get_rasters(), 16
get_supported_sources, 10
global_scales, 11
global_scales(), 16, 18–20

igraph, 13
igraph::betweenness(), 13
igraph::closeness(), 13
igraph::degree(), 13
igraph::evcent(), 13

load_parameters, 11
load_parameters(), 9, 10

model_neg_exp (model_powerlaw), 12
model_neg_exp(), 12
model_powerlaw, 12
model_powerlaw(), 12

`nn_sum`, 13, 23
node_strength (`nn_sum`), 13

pagerank (`nn_sum`), 13

reset_params, 14
reso, 15
reso(), 16, 17

sa_onrasters, 15
sa_onrasters(), 20
sean, 17
sean(), 6–8, 12, 14, 16, 20
search_crop, 18
search_crop(), 22
sensitivity_analysis, 19
sensitivity_analysis(), 17, 15
set_global_scales, 20
set_global_scales(), 11
set_parameters, 21
set_parameters(), 8, 19, 20
set_reso, 21
set_reso(), 15
sp_rast, 22

supported_metrics, 14, 23
tempdir, 3–6, 8, 16, 17
terra::ext(), 20
terra::mean(), 4
terra::rast(), 23
tiff_torast, 23
tiff_torast(), 9, 10