

# Package ‘geohabnet’

October 31, 2023

**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](https://doi.org/10.1093/biosci/biaa067)>.

The primary function is sean() and is indicative of how sensitive the risk analysis is to parameters using kernel models.

The Package currently supports crops sourced from Monfreda, C., N. Ramankutty, and J. A. Foley (2008) <[doi:10.1029/2007gb002947](https://doi.org/10.1029/2007gb002947)> ``Farming the planet: 2. Geographic distribution of crop areas, yields, physiological types, and net primary production in the year 2000, Global Biogeochem. Cycles, 22, GB1022" and International Food Policy Research Institute (2019) <[doi:10.7910/DVN/PRFF8V](https://doi.org/10.7910/DVN/PRFF8V)> ``Global Spatially-Disaggregated Crop Production Statistics Data for 2010 Version 2.0, Harvard Dataverse, V4".

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

**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 ( $\geq 1.7.29$ ),  
 easycsv ( $\geq 1.0.8$ ),  
 yaml ( $\geq 2.3.7$ ),  
 stats,  
 stringr ( $\geq 1.5.0$ ),  
 memoise ( $\geq 2.0.1$ ),  
 graphics,  
 rlang ( $\geq 1.1.1$ ),  
 viridisLite ( $\geq 0.4.2$ ),  
 beepr ( $\geq 1.3$ ),  
 rnaturalearth ( $\geq 0.3.3$ ),  
 tools

**Suggests** devtools,  
 knitr,  
 lintr ( $\geq 3.0.2$ ),  
 mockthat ( $\geq 0.2.8$ ),  
 pkgdown,  
 rmarkdown,  
 testthat ( $\geq 3.1.7$ )

**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

## R topics documented:

ccri_diff . . . . .	3
ccri_mean . . . . .	3
ccri_variance . . . . .	4
connectivity . . . . .	5
cropharvest_rast . . . . .	6
crops_rast . . . . .	6
dist_methods . . . . .	7
geoscale_param . . . . .	7
get_parameters . . . . .	8
get_param_metrics . . . . .	8
get_rasters . . . . .	9
get_supported_sources . . . . .	10
global_scales . . . . .	10
load_parameters . . . . .	11
model_powerlaw . . . . .	11
nn_sum . . . . .	12
reset_params . . . . .	13
reso . . . . .	14
sa_onrasters . . . . .	14
sean . . . . .	16
search_crop . . . . .	17

sensitivity_analysis . . . . .	18
set_global_scales . . . . .	19
set_parameters . . . . .	19
set_reso . . . . .	20
sp_rast . . . . .	20
supported_metrics . . . . .	21
tiff_torast . . . . .	22

ccri_diff	<i>Calculate difference map</i>
-----------	---------------------------------

## 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(), 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 <a href="#">tempdir()</a> .

## Value

Invisible NULL.

ccri_mean	<i>Calculate mean of raster objects</i>
-----------	---

## Description

Wrapper for [terra::mean\(\)](#). Calculates mean of list of rasters.

## Usage

```
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 <code>tempdir()</code> .

**Value**

Invisible NULL.

---

ccri_variance	<i>Calculate variance of CCRI</i>
---------------	-----------------------------------

---

**Description**

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

**Usage**

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

**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 <code>tempdir()</code> .

**Value**

Invisible NULL.

---

`connectivity`*Calculate and plot maps*

---

### Description

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

<code>indexes</code>	list of rasters. See details.
<code>global</code>	logical. TRUE if global analysis is required, FALSE otherwise. When TRUE, <code>geoscale</code> is ignored. Default is TRUE.
<code>geoscale</code>	vector. geographical scale
<code>res</code>	numeric. map resolution.
<code>pmean</code>	TRUE if map of mean should be plotted, FALSE otherwise.
<code>pvar</code>	TRUE if variance map should be plotted, FALSE otherwise.
<code>pdiff</code>	TRUE if difference map should be plotted, FALSE otherwise.
<code>outdir</code>	Character. Output directory for saving raster in TIFF format. Default is <code>tempdir()</code> .

### 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

Yanru Xing, John F Hernandez Nopsa, Kelsey F Andersen, Jorge L Andrade-Piedra, Fenton D Beed, Guy Blomme, Mónica Carvajal-Yepes, Danny L Coyne, Wilmer J Cuellar, Gregory A Forbes, Jan F Kreuze, Jürgen Kroschel, P Lava Kumar, James P Legg, Monica Parker, Elmar Schulte-Geldermann, Kalpana Sharma, Karen A Garrett, *Global Cropland Connectivity: A Risk Factor for Invasion and Saturation by Emerging Pathogens and Pests*, BioScience, Volume 70, Issue 9, September 2020, Pages 744–758, [doi:10.1093/biosci/biaa067](https://doi.org/10.1093/biosci/biaa067)

Hijmans R (2023). *terra: Spatial Data Analysis*. R package version 1.7-46, <https://CRAN.R-project.org/package=terra>

---

cropharvest_rast	<i>Get raster object for crop</i>
------------------	-----------------------------------

---

## 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	<i>Get sum of rasters for individual crops</i>
------------	--

---

## 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

```
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**

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

---

dist_methods	<i>Distance methods supported</i>
--------------	-----------------------------------

---

**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

**Examples**

```
dist_methods()
```

---

geoscale_param	<i>Get geographical scales from the parameters</i>
----------------	--

---

**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	<i>Get Parameters</i>
----------------	-----------------------

---

### 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 <a href="#">tempdir()</a>
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\(\)](#)

### Examples

```
get_parameters()
get_parameters(out = tempdir())
```

---

get_param_metrics	<i>Get metrics from parameters</i>
-------------------	------------------------------------

---

### Description

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

### Usage

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

### Arguments

params	R object of <a href="#">load_parameters()</a> . Default is <code>load_parameters()</code> .
--------	---



**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	<i>Get rasters object from parameters</i>
-------------	---

---

**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

**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	<i>Get supported sources of crops</i>
-----------------------	---------------------------------------

---

### 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	<i>Global geographical extent</i>
---------------	-----------------------------------

---

### 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

Seperate 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

*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 <a href="#">sean()</a> .
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("betweenness"), weights = c(100))`. Default values are fetched from parameters.yaml and arguments uses the same structure.

## Value

risk index

---

nn_sum	<i>Calculation on network matrix.</i>
--------	---------------------------------------

---

## 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\(\)](#).
- `betweenness()`: 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)

betweenness(crop_dm, we)

ev(crop_dm, we)

degree(crop_dm, we)

closeness(crop_dm, we)

pagerank(crop_dm, we)
```

### Arguments

<code>crop_dm</code>	Distance matrix. In the internal workflow, the distance matrix comes is a result of operations within <a href="#">sean()</a> and risk functions.
<code>we</code>	Weight in percentage.

### Value

Matrix with the mean value based on the assigned weight.

### See Also

Other metrics: [supported\\_metrics\(\)](#)

---

reset_params	<i>Reset parameters.yaml</i>
--------------	------------------------------

---

### 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	<i>Get resolution value</i>
------	-----------------------------

---

**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	<i>Run sensitivity analysis</i>
--------------	---------------------------------

---

**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 <a href="#">dist_methods()</a>
res	numeric. resolution at which operations will run. Default is <a href="#">reso()</a>
maps	logical. TRUE if maps are to be plotted, FALSE otherwise
outdir	Character. Output directory for saving raster in TIFF format. Default is <a href="#">tempdir()</a> .

## 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.

## References

Yanru Xing, John F Hernandez Nopsa, Kelsey F Andersen, Jorge L Andrade-Piedra, Fenton D Beed, Guy Blomme, Mónica Carvajal-Yepes, Danny L Coyne, Wilmer J Cuellar, Gregory A Forbes, Jan F Kreuze, Jürgen Kroschel, P Lava Kumar, James P Legg, Monica Parker, Elmar Schulte-Geldermann, Kalpana Sharma, Karen A Garrett, *Global Cropland Connectivity: A Risk Factor for Invasion and Saturation by Emerging Pathogens and Pests*, BioScience, Volume 70, Issue 9, September 2020, Pages 744–758, [doi:10.1093/biosci/biaa067](https://doi.org/10.1093/biosci/biaa067)

Hijmans R (2023). *terra: Spatial Data Analysis*. R package version 1.7-46, <https://CRAN.R-project.org/package=terra>

## See Also

Use [get\\_rasters\(\)](#) to obtain raster object.

## Examples

```
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),
  c("sum", "mean"),
  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)

```

sean

*Calculate sensitivity analysis on cropland harvested area fraction*

## Description

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

```

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

<code>rast</code>	Raster object which will be used in analysis.
<code>global</code>	Logical. TRUE if global analysis, FALSE otherwise. Default is TRUE
<code>geoscale</code>	Vector. Geographical coordinates in the form of <code>c(Xmin, Xmax, Ymin, Ymax)</code>
<code>agg_methods</code>	vector. Aggregation methods
<code>dist_method</code>	character. One of the values from <code>dist_methods()</code>
<code>link_threshold</code>	numeric. A threshold value for link
<code>host_density_threshold</code>	A host density threshold value
<code>res</code>	numeric. resolution at which operations will run. Default is <code>reso()</code>
<code>maps</code>	logical. TRUE if maps are to be plotted, FALSE otherwise
<code>outdir</code>	Character. Output directory for saving raster in TIFF format. Default is <code>tempdir()</code> .

## Details

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

## Value

A list of calculated CCRI values using negative exponential



## References

Yanru Xing, John F Hernandez Nopsa, Kelsey F Andersen, Jorge L Andrade-Piedra, Fenton D Beed, Guy Blomme, Mónica Carvajal-Yepes, Danny L Coyne, Wilmer J Cuellar, Gregory A Forbes, Jan F Kreuze, Jürgen Kroschel, P Lava Kumar, James P Legg, Monica Parker, Elmar Schulte-Geldermann, Kalpana Sharma, Karen A Garrett, *Global Cropland Connectivity: A Risk Factor for Invasion and Saturation by Emerging Pathogens and Pests*, BioScience, Volume 70, Issue 9, September 2020, Pages 744–758, doi:10.1093/biosci/biaa067

Hijmans R (2023). *terra: Spatial Data Analysis*. R package version 1.7-46, <https://CRAN.R-project.org/package=terra>

## See Also

Uses [connectivity\(\)](#)

## Examples

```
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 which crop data is available.

## 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

<code>maps</code>	logical. TRUE if maps are to be plotted, FALSE otherwise
<code>alert</code>	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

Yanru Xing, John F Hernandez Nopsa, Kelsey F Andersen, Jorge L Andrade-Piedra, Fenton D Beed, Guy Blomme, Mónica Carvajal-Yepes, Danny L Coyne, Wilmer J Cuellar, Gregory A Forbes, Jan F Kreuze, Jürgen Kroschel, P Lava Kumar, James P Legg, Monica Parker, Elmar Schulte-Geldermann, Kalpana Sharma, Karen A Garrett, *Global Cropland Connectivity: A Risk Factor for Invasion and Saturation by Emerging Pathogens and Pests*, BioScience, Volume 70, Issue 9, September 2020, Pages 744–758, [doi:10.1093/biosci/biaa067](https://doi.org/10.1093/biosci/biaa067)

Hijmans R (2023). *terra: Spatial Data Analysis*. R package version 1.7-46, <https://CRAN.R-project.org/package=terra>

## 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	<i>Set global geographical extent</i>
-------------------	---------------------------------------

---

**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	<i>Set Parameters</i>
----------------	-----------------------

---

**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	<i>Set resolution value</i>
----------	-----------------------------

---

**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	<i>raster for mapspam crop.</i>
---------	---------------------------------

---

**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**

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

**See Also**

[geodata::spamCrops\(\)](#) [search\\_crop\(\)](#)

**Examples**

```
sp_rast("rice")
```

---

supported_metrics	<i>Returns metrics currently supported in the analysis.</i>
-------------------	---

---

**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	<i>Get raster object from tif file</i>
-------------	--

---

**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)
```