Package 'ohigui'

November 23, 2013

Version 0.9.13

Date 2013-06-15

Title Ocean Health Index - Graphical User Interface

Author Ben Best <bbest@nceas.ucsb.edu>, with past contributions from Darren Hardy <drh@stanford.edu>

Maintainer Ben Best <bbest@nceas.ucsb.edu>

URL http://ohi-science.org

Depends R (>= 3.0.1)

Imports
ggplot2 (>= 0.9),knitr (>= 1.1),markdown (>= 0.5.4),pander (>= 0.3.1),plyr (>= 1.8),RColorBrewer (>= 1.0),RJSONIO

Description ohigui is an interface through the web browser for exploring and calculating the Ocean Health Index.

License MIT

LazyData true

Collate
'ohi.R''launchApp.R''load.config.R''config.summary.R''ohi.file.R''allregions.R''aster.R''ohi.model.goal.R''ohi.mod
package.R''assemble.layers_data.R''check.layers_navigation.R''ohi-vars.R''ohi.extras.R''AggregateLayers.R''scoring.R''ohi.nature2012.R'

R topics documented:

`ohigui-package` *Ocean Health Index - Graphical User Interface*

Description

The ohigui is an interface through the web browser for exploring and calculating the Ocean Health Index.

Author(s)

Ben Best <bbest@nceas.ucsb.edu>

`AggregateLayers` *Aggregate Layers*

Description

Aggregate by country, country and year, or some other weight.

Usage

```
aggregate_by_country(df, col.value = "value",
  col.country = "country_id",
  lyrs.dat.csv = layers_data.csv)
```

Arguments

`df`	Input data frame.
`col.value`	Column in data frame containing the value to be aggregated.
`col.country`	Column in data frame containing the country_id.
`col.weight`	Column in data frame containing the weight.

Value

These functions aggregate to region by either country, country and year, or just a weight. These are mostly only used for aggregating a goal's status or trend calculations to region.

Examples

```
## Not run:
    aggregate_by_country(df, col.value='value', col.country='country_id')
    aggregate_by_country_weighted(df, w, col.value='value', col.country='country_id', col.weight='weight')
    aggregate_by_country_year(df, col.value='value', col.country='country_id')

## End(Not run)
```

`aggregate_to_region` *Check and Assemble Layers*

Description

Check all the input layers as defined by layers_navigation.csv, found in dir.layers and assembled out to a consolidated layers_data.csv for easy data extraction.

Usage

```
aggregate_by_country(df, col.value='value', col.country='country_id')
aggregate_weighted(df, w, col.value='value', col.country='country_id', col.weight='weight')
aggregate_by_country_year(df, col.value='value', col.country='country_id')
```

Arguments

`df`	Input data frame.
`col.value`	Column in data frame containing the value to be aggregated.
`col.country`	Column in data frame containing the country_id.
`col.weight`	Column in data frame containing the weight.

Value

These functions aggregate to region by either country, country and year, or just a weight. These are mostly only used for aggregating a goal's status or trend calculations to region.

Examples

```
## Not run:
  aggregate_by_country(df, col.value='value', col.country='country_id')
  aggregate_weighted(df, w, col.value='value', col.country='country_id', col.weight='weight')
  aggregate_by_country_year(df, col.value='value', col.country='country_id')

## End(Not run)
```

allregions *Ocean Health Index: Global regions*

Description

Functions to join with global regions

Usage

```
allregions(d = NULL, scope = "all")
```

Arguments

d	The data frame to join with. If NULL, returns all regions.
scope	'all' for all global regions, or 'eez' for EEZ/country regions.

Value

Returns a data.frame containing a left join of regions.

See Also

merge

Examples

```
d <- data.frame(id=1:50, status=runif(50, 0, 1))
allregions(d) # returns status=NA for all but the first 50 regions
ohi.load('regions_details')
```

assemble.layers_data *Assemble Layers*

Description

Check all the input layers as defined by layers_navigation.csv, found in dir.layers and assembled out to a consolidated layers_data.csv for easy data extraction.

Usage

```
assemble.layers_data(layers_navigation.csv,
  layers_data.csv, layers_id_fields, verbose = T,
  msg.indent = "  ")
```

Arguments

`layers_navigation.csv`
: Full path to the layers_navigation.csv file.

`dir.layers`
: Full path to the directory containing the layers files.

`layers_data.csv`
: Combined data table output of all layers dir.layers based on descriptions in layers_navigation.

`layers_id_fields`
: Character vector of unique identifiers typically spatial (eg region_id, country_id, saup_id).

Value

All of these parameters should be defined in config.R.

The check.layers_navigation() function iterates through the layers_navigation.csv and checks for the existence of all the input files and looks for matching fields. Any unused fields should be dealt with before moving onto using assemble.layers_data.

The assemble.layers_data() function reads in all data layers and combines them into a single data table layers_data.csv.

Examples

```
## Not run:
  config.R = '~/ohi/scenarios/global_2012_nature/conf/config.R'
  source(config.R)
  check.layers_navigation(layers_navigation.csv, dir.layers, layers_id_fields)
  assemble.layers_data(layers_navigation.csv, dir.layers, layers_data.csv, layers_id_fields)

## End(Not run)
```

---

aster *Plot Aster*

Description

Plot flower plot. Created by Jim Regetz. Slight modifications by Darren and Ben.

Usage

```
aster(lengths, widths, labels, disk = 0.5, max.length,
  center = NULL, main = NULL, fill.col = NULL,
  plot.outline = TRUE, label.offset = 0.15,
  xlim = c(-1.2, 1.2), ylim = c(-1.2, 1.2), uin = NULL,
  tol = 0.04, cex = 1, bty = "n", lty = 1,
  label.col = "black", label.font = 3, label.cex = NULL,
  ...)
```

Arguments

lengths	length of petal outward to extent of circle
widths	width of petal
labels	petal label outside of circel
disk	relative radius of a central donut hole
max.length	...
center	center value
main	
fill.col	
plot.outline	
label.offset	
xlim	
ylim	
uin	
tol	
cex	
bty	
lty	
label.col	
label.font	
label.cex	

Value

Generate something akin to a rose plot in which the width and length of each petal are directly specified by the user. Or to put it differently, this is somewhat like a pie chart in which the radius of each wedge is allowed to vary (along with the angular width, as pie charts do). As an additional enhancement, one can specify a central disk of arbitrary radius (from 0 to 1, assuming that the plot itself is scaled to the unit circle), in which case the petal heights are always measured from the edge of the disk rather than the center of the circle; if desired, text can be added in the center.

Although this kind of plot may already be well known in some circles (no pun intended), I haven't seen it clearly defined or labeled anywhere, so I'm anointing it an 'aster' plot because its component parts are reminiscent of composite flower morphology.

The 'lengths' dictates how far out each petal extends, 'widths' dictates the (angular) width of each petal, and 'disk' gives the relative radius of a central donut hole. If no widths are provided, all petals will have equal widths. Additional function arguments can also control whether petals are labeled, whether the petal lengths are rescaled to the maximum score or to a user-input score, whether spokes delineating each petal are extended to an outer circle, and more. I also wrote a quick convenience wrapper for creating a legend plot.

Note that the function here is a repurposed and very heavily modified version of the windrose() function contained in the 'circular' package, although sufficiently rewritten so as not to depend on any functionality in that package.

Examples

```
## Not run:
# generate some fake data
set.seed(1)
scores <- sample(1:10)
weights <- sample(1:10)
labels <- paste(LETTERS[1:10], "X", sep="")

# do some plots
par(mfrow=c(2,2), xpd=NA)
aster(lengths=scores, widths=weights, disk=0, main="Example 1",
    plot.outline=FALSE)
aster(lengths=scores, widths=weights, labels=labels, main="Example 2",
    lty=2, fill.col="gray", plot.outline=FALSE)
aster.legend(labels=labels, widths=weights)
aster(lengths=scores, widths=weights, disk=0.5, main="Example 3",
    center="Hello world")

## End(Not run)
```

`check.layers_navigation` *Check Layers*

Description

Check all the input layers as defined by layers_navigation.csv, found in dir.layers and assembled out to a consolidated layers_data.csv for easy data extraction.

Usage

```
check.layers_navigation(layers_navigation.csv,
  layers_id_fields, verbose = T, msg.indent = "  ")
```

Arguments

`layers_navigation.csv`	Full path to the layers_navigation.csv file.
`dir.layers`	Full path to the directory containing the layers files.
`layers_data.csv`	Combined data table output of all layers dir.layers based on descriptions in layers_navigation.
`layers_id_fields`	Character vector of unique identifiers typically spatial (eg region_id, country_id, saup_id).

Value

All of these parameters should be defined in config.R.

The check.layers_navigation() function iterates through the layers_navigation.csv and checks for the existence of all the input files and looks for matching fields. Any unused fields should be dealt with before moving onto using assemble.layers_data.

The assemble.layers_data() function reads in all data layers and combines them into a single data table layers_data.csv.

Examples

```
## Not run:
  config.R = '~/ohi/scenarios/global_2012_nature/conf/config.R'
  source(config.R)
  check.layers_navigation(layers_navigation.csv, dir.layers, layers_id_fields)
  assemble.layers_data(layers_navigation.csv, dir.layers, layers_data.csv, layers_id_fields)

## End(Not run)
```

---

config.summary *Print summary of configuration*

Description

This function outputs the summary of the required files in the configuration file.

Usage

```
config.summary(config.R, indent = "  ")
```

Arguments

config.R	path to configuration file
indent	level for indenting output before required input file

---

launchApp *Launch the browser application*

Description

This function loads the specified configuration with load.config and launches the web browser using shiny::runApp. [TODO: specify needed config.R format in seperate help file.]

Usage

```
launchApp(config.R, ...)
```

Arguments

config.R	path to configuration file
...	arguments passed to shiny::runApp

Examples

```
## Not run:
launchApp('/usr/local/ohi/src/toolbox/scenarios/global_2012_nature/conf/config.R')

## End(Not run)
```

load.config *Load the configuration file*

Description

This function loads the specified configuration file into the global namespace. [TODO: specify needed config.R format in seperate help file.]

Usage

```
load.config(config.R)
```

Arguments

config.R	path to configuration file

Examples

```
## Not run:
load.config('/usr/local/ohi/src/toolbox/scenarios/global_2012_nature/conf/config.R')

## End(Not run)
```

ohi-vars *Ocean Health Index: Global variables*

Description

Various variable constants for goals, dimensions, labels, etc.

Usage

```
ohi.goal.all
```

Format

1. ohi.subgoal.all is the list of only the subgoals, or the goal if only 1 subgoal.
2. ohi.goal.subgoal.unique is the list of only the subgoals, or the goal if only 1 subgoal.
3. ohi.goal.subgoal.all the union of all goals and subgoals.
4. ohi.subgoal.parent is the list of subgoals and their parent goal.
5. labels is a list to map codes into text labels.

Examples

```
ohi.labels[['AO']]
ohi.labels[ohi.goal.all]
ohi.labels[ohi.goal.subgoal.all]
ohi.subgoal.parent[ohi.subgoal.all]
```

ohi.extras *Ocean Health Index: TBD*

Description

TBD

Source

tbd

References

tbd

Examples

```
Sys.info()
```

ohi.model.goal *Ocean Health Index: Goal Model*

Description

The goal model function.

Usage

```
ohi.model.goal(id, status, trend, resilience, pressure,
  DISCOUNT = 1, BETA = 0.67, default.trend = 0)
```

Arguments

id	Region identifiers.
status	Status scores.
trend	Trend values for 5 year outlook.
resilience	Resilience scores.
pressure	Pressures scores.
DISCOUNT	Discount multiplier.
BETA	Multiplier used in likely future status calculation.
default.trend	The default trend value (0) if region has NA.

Value

Returns a data.frame with the input data, and a likely future status and OHI score.

Examples

```
## Not run:
## run a model with 50 regions using random data,
## using 5 year 1-percent discount rate and beta=0.67
require(ohi)
d <- ohi.model.goal(id=1:50,
                    status=runif(50, 0, 1),
                    trend=runif(50, -1, 1),
                    resilience=runif(50, 0, 1),
                    pressure=runif(50, 0, 1),
                    DISCOUNT = (1 + 0.01)^-5,
                    BETA = 0.67,
                    default.trend = 0.0)
## view model output
names(d)
d[,c('id','score','xF')]

## End(Not run)
```

`ohi.model.pressures` *Ocean Health Index: Pressures Model*

Description

The pressures model function computes a pressures score for each region given a weighting matrix for a goal and the individual pressures values.

Usage

```
ohi.model.pressures(p, w, GAMMA = 0.5, browse = F)
```

Arguments

p	the pressures value matrix [region_id x pressure]. Each score must be a real number between 0 and 1 inclusive, or NA. The pressure names must be of the form *category_pressure* where *category* is one of the categories listed in `ohi.pressure.category`. Use `ss` to denote the social category.

```
pressure region_id cc_acid cc_sst cc_uv
 fp_art_hb 1 0.879 0.360 0.764 NA 2 0.579 0.396 0.531 NA 3
 0.926 0.235 0.769 NA 4 0.914 0.554 0.795 NA 5 0.860 0.609
 0.802 0.001 6 0.871 0.325 0.788 0.001 7 0.846 0.410 0.677
 0.000 8 0.806 0.671 0.752 NA 9 0.844 0.595 0.678 NA 10
 0.860 0.575 0.781 0.109
```

w	the weighting matrix of the form [region_id x pressure]. Each rank weight must be a real number between 0 and 3 inclusive, or NA.

```
pressure
 region_id cc_acid cc_sst cc_uv fp_art_hb 1 2 1 0.6 NA 2 2
 1 0.5 NA 3 2 1 2.1 NA 4 2 1 3.0 NA 5 2 1 2.8 1 6 2 1 2.2
 1 7 2 1 1.3 1 8 2 1 1.7 NA 9 2 1 3.0 NA 10 2 1 1.2 1
```

GAMMA	Multiplier used to combine environmental and social pressures.

Details

Each pressure layer $p(i, j)$ is either environmental or social, belongs to a pressures category $K \in \{cc, fp, hd, po, sp, ss\}$, and has a value $(0..1)$ for each region *i* and pressures layer *j*. Each goal has a weight matrix w that has a rank weight between 0 and 3 inclusive, or NoData, for each region *i* and each pressure layer *j* on a per goal *g* basis.

The pressures scores calculations go through 5 steps, using a complex weighting scheme that varies across goals, subgoals, pressures categories, and regions:

- *g* is the goal or subgoal (e.g., AO, CW, LIV, ECO, ...),
- *i* is the region (e.g., 1, 2, 3, ...),
- *j* is the pressures layer or stressor (e.g., `cc_acid`, `fp_art_lb`, etc.).

Calculations

1. Apply weights for each goal *g*, region *i*, and pressure layer *j*: Each weighted pressure $p_w(g, i, j)$ is the pressure layer value $p(i, j)$ per region *i* and pressure layer *j* multiplied by the rank weight $w(g, i, j)$ for that goal *g*, region *i*, and pressure layer *j*. If the $w(g, i, j)$ is NoData or 0, the weighted pressure $p_w(g, i, j)$ is NoData.

$$p_w(g, i, j) = w(g, i, j) * p(i, j)$$

2. Category-level aggregation: The pressures category score p_K is the sum of all p_w within each category, then rescaled to 0..1 using a linear scale range transformation (from 0..3 to 0..1). Any score p_K greater than 1 is capped to 1:

$$p_K(g, i) = \frac{\min(\sum_{j \in K} p_w(g, i, j), 3)}{3}$$

3. Environmental aggregation: The environmental pressures score $p_E(g, i)$ is the weighted sum of p_K(g,i), where each weight is the maximum weight in the pressure category K, and then divided by the sum of the maximum weights:

$$w_{K,max}(g, i) = max(\{\forall_j \in K | w(g, i, j)\})$$

$$p_E(g, i) = \frac{\sum_K w_{K,max}(g, i) p_K(g, i)}{\sum_K w_{K,max}(g, i)}$$

4. Social aggregation: The social pressures score $p_S(g, i)$ is the mean of the *unweighted* social pressure scores $p(i, j)$:

$$p_S(g, i) = \frac{\sum_{j \in S} p(i, j)}{N}$$

5. Gamma combination: The pressures score $p_X(g, i)$:

$$p_X(g, i) = \gamma p_E(g, i) + (1 - \gamma) p_S(g, i)$$

Value

Returns a named vector with the pressures score for each named region.

See Also

ohi.model.pressures.matrix

Examples

```
## Not run:
  > ohi.pressure.category
$environmental
[1] "po" "hd" "fp" "sp" "cc"

$social
[1] "ss"
> p
         pressure
region_id fp_art_hb fp_art_lb fp_com_hb fp_com_lb hd_intertidal
       1      0.122      0.25      0.35     0.395         0.954
       2      0.096      0.94      0.85     0.252         0.649
       3      0.858      0.46      0.84     0.097         0.425
       4      0.814      0.63      0.60     0.672         0.659
       5      0.247      0.51      0.58     0.941         0.046
       6      0.853      0.34      0.15     0.370         0.385
       7      0.601      0.31      0.39     0.873         0.064
       8      0.355      0.89      0.74     0.159         0.273
       9      0.289      0.94      0.52     0.743         0.094
       10     0.887      0.89      0.87     0.660         0.746
         pressure
region_id hd_subtidal_hb hd_subtidal_sb po_chemicals po_nutrients
       1           0.535          0.651        0.042        0.931
       2           0.454          0.069        0.234        0.025
       3           0.297          0.428        0.970        0.679
       4           0.953          0.485        0.063        0.565
       5           0.963          0.045        0.552        0.828
       6           0.598          0.213        0.907        0.220
       7           0.476          0.641        0.980        0.214
       8           0.285          0.858        0.447        0.793
       9           0.591          0.702        0.719        0.472
       10          0.072          0.431        0.685        0.102
         pressure
region_id sp_alien sp_genetic ss_wgi
       1     0.979      0.761  0.181
       2     0.345      0.091  0.631
       3     0.223      0.986  0.646
       4     0.035      0.078  0.559
       5     0.992      0.643  0.432
       6     0.963      0.416  0.221
       7     0.752      0.627  0.257
       8     0.100      0.245  0.333
       9     0.316      0.373  0.347
       10    0.283      0.224  0.031
> w
         pressure
region_id fp_art_hb fp_art_lb fp_com_hb fp_com_lb hd_intertidal
       1          2         1      0.92         1             1
       2          2         1      0.48         1             1
       3          2         1      2.81         1             1
       4          2         1      1.19         1             1
```

```
        5          2         1      2.82         1             1
        6          2         1      1.07         1             1
        7          2         1      1.48         1             1
        8          2         1      0.46         1             1
        9          2         1      0.56         1             1
        10         2         1      0.90         1             1
          pressure
region_id hd_subtidal_hb hd_subtidal_sb po_chemicals po_nutrients
        1               2              2         1.00            1
        2               2              2         0.79            1
        3               2              2         0.37            1
        4               2              2         0.91            1
        5               2              2         1.06            1
        6               2              2         0.72            1
        7               2              2         0.49            1
        8               2              2         1.18            1
        9               2              2         0.18            1
        10              2              2         0.28            1
          pressure
region_id sp_alien sp_genetic ss_wgi
        1         1          1      1
        2         1          1      1
        3         1          1      1
        4         1          1      1
        5         1          1      1
        6         1          1      1
        7         1          1      1
        8         1          1      1
        9         1          1      1
        10        1          1      1
> p_x <- ohi.model.pressures(p, w)
> p_x
   1    2    3    4    5    6    7    8    9   10
0.40 0.53 0.68 0.63 0.60 0.43 0.48 0.47 0.50 0.30
> data.frame(region_id=names(p_x), pressure=p_x)
   region_id pressure
1          1     0.40
2          2     0.53
3          3     0.68
4          4     0.63
5          5     0.60
6          6     0.43
7          7     0.48
8          8     0.47
9          9     0.50
10        10     0.30
>
>


## End(Not run)
```

---

ohi.model.pressures.matrix

Ocean Health Index: Pressures Matrix Model

Description

The pressures matrix model function computes a pressures weighting matrix based on regional attributes per category.

Usage

```
ohi.model.pressures.matrix(alpha, beta, calc = "avg")
```

Arguments

`alpha`	the weighting matrix of the form [category x pressure]. Each rank weight must be an integer between 0 and 3 inclusive, or NA.
`beta`	the aggregation matrix of the form [region_id x category] to collapse across each category.
`calc`	type of calculation, whether avg (default), mean (diff't from avg?) or presence (results in 1 or 0).

Details

Given:

- *g* is the goal or subgoal (e.g., AO, CW, LIV, ECO, ...),
- *i* is the region (e.g., 1, 2, 3, ...),
- *j* is the pressures layer or stressor (e.g., `cc_acid`, `fp_art_lb`, etc.).
- *k* is the category (e.g., habitat, sector, product, etc.)

There may be a component *k* for a given goal *g* such that $p_w(g, i, j, k)$ and $w(g, i, j, k)$.

$$p_w(g, i, j, k) = w(g, i, j, k) * p(i, j)$$

In these cases where there is a component *k* for goal *g*, there's an additional aggregation or formula to calculate $w(g, i, j)$ based on the core rank weight $\alpha(g, j, k)$ from the original pressures matrix (as written in Halpern et al. (2012)) and some region-specific data for each category k $\beta(i, k)$.

This function `ohi.model.pressures.matrix` will aggregate a category-specific weighting matrix $\alpha(g, j, k)$ [category x pressure] using region-specific data $\beta(g, i, k)$ into a [region_id x pressure] matrix $w(g, i, j)$ used in `ohi.model.pressures`, such that:

$$w(g, i, j) = \frac{\sum_k \alpha(g, j, k) * \beta(g, i, k)}{\sum_k \beta(g, i, k)}$$

1. For the CP, CS goals, the weight depends on the extent *A* of habitat *k* in region *i*:

$$\beta(i, k) = A(i, k)$$

2. For the HAB goal, the weight depends on the presence of habitat *k* (i.e., if $A(i, k) > 0$) in region *i*:

$$\beta(i, k) = hasHabitat(i, k)$$

3. For the LIV and ECO goals, the weight depends on the presence of sector *k* if data available for region *i* and sector *k*:

$$\beta(i,k) = hasSector(i,k)$$

4. For the NP goal, the weight depends on the peak dollar value of each product *k* across all years (see w_p from SI Equation S27) if data available for region *i* and product *k*:

$$\beta(i,k) = w_p(i,k)$$

Value

Returns a weight matrix *w* [region_id x pressure] suitable for `ohi.model.pressures`.

See Also

`ohi.model.pressures`

`ohi.model.resilience` *Ocean Health Index: Resilience Model*

Description

The resilience model function computes a resilience score for each region given a weighting matrix for a goal and the individual resilience values.

Usage

```
ohi.model.resilience(r, t, w = NA, gamma = 0.5)
```

Arguments

`r`	the resilience value matrix `[region_id x layer]`. Each score must be a real number between 0 and 1 inclusive, or NA.
`t`	the typing vector t[layer] where values are from `ohi.resilience.category`.
`w`	the weighting matrix of the form `[region_id x layer]`. Each rank weight must be a real number >= 0, or NA for even weighting.
`w.layers`	the weighting vector of the form `[layer]`. Each rank weight must be a real number >= 0, or NA for even weighting.
`gamma`	the gamma constant for $r_{i,x}$ calculation.
`b`	a boolean value matrix `[region_id x layer]` which is `TRUE` if the given region_id should include layer, and `FALSE` otherwise.

Details

To calculate Resilience for each goal g and region i ($r(g,i)$) we assess three types of resilience measures *j*: ecological integrity ($Y_E(g,i)$), goal-specific regulations aimed at addressing ecological pressures ($G(g,i)$), and social integrity ($Y_S(g,i)$). The first two measures address ecological resilience while the third addresses social resilience. When all three aspects are relevant to a goal, Resilience is calculated for each goal g and each region *i*:

$$r(g,i) = \gamma * \left(\frac{Y_E(g,i) + G(g,i)}{2} \right) + (1 - \gamma) * Y_S(g,i)$$

where each goal *g* is comprised of several resilience layers *j* where w_j is a configuration-time weight to aggregate across resilience categories:

$$G(g,i) = \frac{\sum_{j \in g} w_j G(i,j)}{\sum_{j \in g} w_j}$$

$$Y_E(g,i) = \frac{\sum_{j \in g} Y_E(i,j)}{N}$$

$$Y_S(g,i) = \frac{\sum_{j \in g} Y_S(i,j)}{N}$$

Value

`ohi.model.resilience` returns resilience score for each region. `ohi.model.resilience.matrix` returns a weighting matrix suitable for `ohi.model.resilience`.

Examples

```
## Not run:
> ohi.resilience.category
[1] "environmental" "regulatory"    "social"
> b
         layer
region_id fishing-v1 habitat-combo species-diversity-3nm wgi-all
      104       TRUE          TRUE                  TRUE    TRUE
      105       TRUE          TRUE                  TRUE    TRUE
      106       TRUE          TRUE                  TRUE    TRUE
      107       TRUE          TRUE                  TRUE    TRUE
      108       TRUE          TRUE                  TRUE    TRUE
      109       TRUE          TRUE                  TRUE    TRUE
      110       TRUE          TRUE                  TRUE    TRUE
      111       TRUE          TRUE                  TRUE    TRUE
      112       TRUE          TRUE                  TRUE    TRUE
      113       TRUE          TRUE                  TRUE    TRUE
      114       TRUE          TRUE                  TRUE    TRUE
> w
           fishing-v1         habitat-combo species-diversity-3nm
                    2                     2                     1
              wgi-all
                    1
> w < -ohi.model.resilience.matrix(b, w)
> w
         layer
region_id fishing-v1 habitat-combo species-diversity-3nm wgi-all
```

```
      104          2             2                     1       1
      105          2             2                     1       1
      106          2             2                     1       1
      107          2             2                     1       1
      108          2             2                     1       1
      109          2             2                     1       1
      110          2             2                     1       1
      111          2             2                     1       1
      112          2             2                     1       1
      113          2             2                     1       1
      114          2             2                     1       1

> r
         layer
region_id fishing-v1 habitat-combo species-diversity-3nm wgi-all
      104     0.4870        0.4495                0.8679  0.4385
      105     0.5162        0.5905                0.8748  0.2460
      106     0.4811        0.4051                0.8852  0.6465
      107     0.3618        0.2583                0.8260  0.8007
      108     0.5322        0.4703                0.9318  0.5579
      109     0.5053        0.4703                0.9313  0.5579
      110     0.6491        0.5690                0.9239  0.5703
      111     0.3629        0.1562                0.9230  0.6375
      112     0.5670        0.5000                0.9273  0.5718
      113     0.3807        0.2530                0.9339  0.4484
      114     0.6508        0.5690                0.9275  0.5703
> t
           fishing-v1         habitat-combo species-diversity-3nm
         "regulatory"          "regulatory"       "environmental"
              wgi-all
            "social"

> ohi.model.resilience(r, t, w)
   104    105    106    107    108    109    110    111    112    113
0.5533 0.4800 0.6553 0.6844 0.6372 0.6337 0.6684 0.6144 0.6511 0.5369
   114
0.6695

## End(Not run)
```

`ohi.nature2012` *Ocean Health Index: Global Results*

Description

This data set contains the scores and model outputs from the Ocean Health Index global study (Halpern et al. 2012).

Format

A list containing:

1. `regions` is a data frame with the id and label for each reporting region (country EEZs).
2. `global` is a data frame with area-weighted and unweighted global OHI scores, including per-goal scores.

3. `valueset` is a data frame with the 5 value set (weighting combinations) scores per region.
4. `goal` is an array of scores as *goal ~ dimension ~ region*.
5. `labels` is a list to map codes into text labels.
6. `layers` is a list of layers data from the file archive.
7. `countries` is a list of country-level data to aggregate into regions.

References

Halpern, BS, C Longo, D Hardy, KL McLeod, JF Samhouri, SK Katona, K Kleisner, SE Lester, J O'Leary, M Ranelletti, AA Rosenberg, C Scarborough, LR Selig, BD Best, DR Brumbaugh, FS Chapin III, LB Crowder, KL Daly, SC Doney, C Elfes, MJ Fogarty, SD Gaines, K Jacobsen, LB Karrer, HM Leslie, E Neeley, D Pauly, S Polasky, B Ris, K St. Martin, GS Stone, UR Sumaila, and D Zeller. 2012. *An index to assess the health and benefits of the global ocean*. **Nature** 488, 615–620 (30 August 2012). http://dx.doi.org/10.1038/nature11397doi:10.1038/nature11397.

See Also

File archive

Examples

```
options(max.print=10)
options(digits=3)
data(ohi.nature2012, package='ohigui')

## Find the Region ID number for the United States
region_id <- with(ohi.nature2012$regions, id[label == "United States"])
region_id

## Show the Artisanal Fishing Opportunities goal score
with(ohi.nature2012, goal['AO', 'score', region_id])

## Show the status and trend for the Food Provision: Fisheries (FIS) subgoal
with(ohi.nature2012, goal['FIS', c('status','trend'), region_id])

## Show all goals and dimensions
with(ohi.nature2012, goal[ , , region_id])

## Show the unweighted OHI score for Region 116
subset(ohi.nature2012[['valueset']], id == region_id)[['unweighted']]

## Show the attributes for Region 116
subset(ohi.nature2012[['regions']], id == region_id)

## Show all countries part of Region 116
subset(ohi.nature2012[['countries']][['regions']], region_id == region_id)

## Show data for CP Extent layer for Region 116
subset(ohi.nature2012[['layers']][['data']],
       layer_id == 'i_cp_extent' & region_id == region_id)
```

`ohi.read.csv` *Ocean Health Index: Data file format*

Description

Simple read/write utility functions for the CSV and RData data file format.

Usage

```
ohi.read.csv(file, na.strings = "", row.names = NULL,
  ...)
```

Arguments

`file`	Full path to input/output file.
`x`	A data frame with data to write.
`digits`	Use to restrict ASCII representation of doubles.
`row.names`	Do not use them by default
`na.strings`	Use blanks for NA.
`na`	Use blanks for NA.
`dir`	The directory to use.
`method`	The data file format method.
`envir`	Environment in which to assign name
`name`	The variable to which data will be assigned, and used for the filename – e.g., name = 'regions' will look for 'regions.csv', etc.
`xdim`	The name of a dimension, and expects get(dimension) to return a valid data frame with 2 columns: region_code, and dimension – e.g., ('region_code', 'status').
`...`	Arguments passed onto read.csv, write.csv, load, save.

Value

Returns a data.frame with the input data.

See Also

`read.csv`, `write.csv`

Examples

```
## Not run:
  d <- ohi.read.csv('data/regions.csv')
  names(d)
  head(d)

  d$label <- toupper(d$label)
  ohi.write.csv(d, 'data/regions.veryloud.csv')

  status <- data.frame(region_code=d$region_code, status=rnorm(nrow(d)))
```

```
  ohi.save.status()

## End(Not run)
```

`scoring` *Ocean Health Index: Scoring*

Description

Scoring functions

Usage

```
score.rescale(x, xlim = NULL, method = "linear", ...)
```

Arguments

`x`	A numeric vector of data.
`xlim`	The scoring range. If null, derives range from data.
`p`	A percentage buffer to add to the maximum value.
`method`	Only 'linear' is supported.
`...`	Arguments for min, max, pmin, pmax.

Value

Returns scores.

See Also

min, max, pmin, pmax

Examples

```
score.max(c(0.5, 1, 2))
score.max(c(0.5, 1, 2), p=0.25)
score.rescale(c(0.5, 1, 2))
score.clamp(c(-0.5, 1, 2))
score.clamp(c(-0.5, 1, 2), xlim=c(-1, 1))
```

Index