Package ‘ohicore’

December 8, 2013

Version 0.1

Date 2013-09-25

Title Ocean Health Index calculation package

Author Ben Best, Steve Hastings, Darren Hardy

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

Depends R (>= 2.14.0),plyr,reshape2,RJSONIO

Description A collection of functions for generically calculating the Ocean Health Index scores as well as individual goals and sub-goals.

License MIT

Collate
'CalculatePressuresComponent.R''CalculateResilienceComponent.R''CalculateStatusComponent.R''CalculateSubgo

LazyData TRUE

R topics documented:

`CalculatePressures` *Calculate the pressures component of each (sub)goal.*

Description

Calculate the pressures component of each (sub)goal.

Usage

```
CalculatePressures(layers, conf, gamma, debug = F)
```

Arguments

`layers`	object `Layers`
`conf`	object `Conf`
`gamma`	(optional) if not specified defaults to 0.5

Value

data.frame containing columns 'region_id' and per subgoal pressures score

---

`CalculatePressuresComponent`
Calculate the pressures component of each (sub)goal.

Description

Calculate the pressures component of each (sub)goal.

Usage

```
CalculatePressuresComponent(eco.pressures,
  social.pressures, c.name = "category",
  s.name = "region", gamma = 0.5)
```

Arguments

`eco.pressures`	data.frame containing columns 'region', 'category', 'weight', and 'value'
`social.pressures`	data.frame containing columns 'region', and 'value'
`gamma`	(optional) if not specified defaults to 0.5

Value

data.frame containing columns 'region', 'p_E', 'p_S', and 'p_x'

`CalculatePressuresMatrix`
Calculate Pressures Matrix

Description

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

Usage

```
CalculatePressuresMatrix(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 `CalculatePressuresMatrix` 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 `CalculatePressuresScore`, 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 `CalculatePressuresScore`.

See Also

`CalculatePressuresScore`

CalculatePressuresScore

Calculate Pressures Score

Description

The pressures score is calculated for each region given a weighting matrix for a goal and the individual pressures values.

Usage

```
CalculatePressuresScore(p, w, GAMMA = 0.5, browse = F,
  pressures_categories = list(environmental = c("po", "hd", "fp", "sp", "cc"), social = "ss"))
```

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

CalculatePressuresMatrix

Examples

```
## Not run:
  > conf$config$pressures_categories
$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 <- CalculatePressuresScore(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)
```

CalculateResilienceComponent
Calculate the Resilience component of each (sub)goal.

Description

Calculate the Resilience component of each (sub)goal.

Usage

```
CalculateResilienceComponent(goal.specific.regulations,
  ecological.integrity, social.integrity,
  c.name = "category", s.name = "region", gamma = 0.5)
```

Arguments

`goal.specific.regulations`
: (data.frame) contains columns 'region', 'weight', and 'value'

`gamma`
: (numeric) represents the weighting between ecological and social aspects of resilience, defaults to 0.5 (equal weights)

Value

(data.frame)

`CalculateStatusComponent` *Compute a single subgoal.*

Description

Compute a single subgoal.

Usage

```
CalculateStatusComponent(DATA, fun, trend.Years = 5,
  c.name = "year", s.name = "region")
```

Arguments

`DATA`	data.frame containing columns 'region', 'value', and (optionally) 'w'
`fun`	(optional) function for calculating the subgoal value, if not specified it will default to a weighted average
`w`	(optional) numeric vector describing the

Value

stuff

`CalculateSubgoal` *Compute a single subgoal.*

Description

Compute a single subgoal.

Usage

```
CalculateSubgoal(current.data, eco.pressures,
  social.pressures, gs.regulations, social.integrity,
  eco.integrity, fun = stats::weighted.mean,
  trend.Years = 5)
```

Arguments

`DATA`	data.frame containing columns 'region', 'value', and (optionally) 'w'
`fun`	(optional) function for calculating the subgoal value, if not specified it will default to a weighted average
`w`	(optional) numeric vector describing the

Value

stuff

`CheckLayers` *Check Layers*

Description

Check all the input layers as defined by layers.csv and update required fields

Usage

```
CheckLayers(layers.csv, layers.dir, flds_id, verbose = T,
  msg.indent = "  ")
```

Arguments

`layers.csv`	full path to the layers.csv file.
`layers.dir`	full path to the directory containing the layers files.
`flds_id`	character vector of unique identifiers, typically spatial, eg c('region_id', 'country_id', 'saup_id'), described in your `Conf$layers_id_fields`.
`if`	True (default), extra diagnostics are output

Details

The CheckLayers() function iterates through all the layers in layers.csv and updates the following field names:

- *fld_id_num* - numeric unique identifier
- *fld_id_chr* - character unique identifier
- *fld_category* - category
- *fld_year* - year
- *fld_val_num* - numeric value
- *fld_val_chr* - character value

Additional diagnostic fields are updated:

- *file_exists* - input filename exists
- *val_min* - minimum value, if numeric
- *val_max* - maximum value, if numeric
- *val_0to1* - TRUE if value ranges between 0 and 1
- *flds_unused* - unused fields from input file when guessing prescribed field names (aboves)
- *flds_missing* - fields expected, as given by Layers units, and not found
- *rows_duplicated* - given the combination of all row-identifying fields (and excluding value fields), the number of rows which are duplicates
- *num_ids_unique* - number of unique ids, as provided by just the unique instances of the fld_id

Value

warning messages

Examples

```
## Not run:
  CheckLayers(layers.csv, layers.dir, c('rgn_id','cntry_key','saup_id'))

## End(Not run)
```

`Conf` *Conf reference class.*

Description

Conf reference class.

Usage

```
Conf(...)
```

Arguments

`dir`	path to directory containing necessary files

Details

To create this object, `Conf(dir)`. The `dir` is expected to have the following files:

- *config.R*
- *functions.R*
- *goals.csv*
- *pressures_matrix.csv*
- *resilience_matrix.csv*
- *resilienceweights.csv*

See also `Conf_write() to write the configuration back to disk.`

Value

object reference class of Config containing:

- *config*
- *functions*
- *goals*
- *pressures_matrix*
- *resilience_matrix*
- *resilienceweights*

`Conf_write` *Write the Conf to disk*

Description

Write the Conf to disk

Arguments

`dir` path to directory where the Conf files should be output

Details

Use this function to write the configuration to disk, like so `conf$write(dir)`. This is useful for modifying and then reloading with `Conf(dir)`.

`Halpern2012.` *Calculate Biodiversity.*

Description

Calculate Biodiversity.

Usage

```
Halpern2012.(A, G, w, Cc, Cr, ...)
```

Arguments

`placeholder` placeholder

Value

1

`Halpern2012.AO` *Calculate Artisanal Fishing Opportunities.*

Description

Calculate Artisanal Fishing Opportunities.

Usage

```
Halpern2012.AO(Sao, Oao, PPPpcGDP, ...)
```

Arguments

`placeholder`	placeholder Sao
`placeholder`	placeholder Oao
`placeholder`	placeholder PPPpcGDP

Value

1

`Halpern2012.BD.HAB` *Calculate Habitats subgoal of Biodiversity.*

Description

Calculate Habitats subgoal of Biodiversity.

Usage

```
Halpern2012.BD.HAB(Cc, Cr, ...)
```

Arguments

`placeholder`	placeholder

Value

1

`Halpern2012.BD.SPP` *Calculate Species subgoal of Biodiversity.*

Description

Calculate Species subgoal of Biodiversity.

Usage

```
Halpern2012.BD.SPP(A, G, w, ...)
```

Arguments

`placeholder`	placeholder

Value

1

`Halpern2012.CP` *Calculate Coastal Protection*

Description

Calculate Coastal Protection

Usage

```
Halpern2012.CP(Cc, Cr, w, A, ...)
```

Arguments

`placeholder`	placeholder Cc current 'condition' of habitat k
`placeholder`	placeholder Cr reference 'condition' of habitat k
`placeholder`	placeholder A amount of area covered by habitat k
`placeholder`	placeholder w rank weight of habitat protective ability

Value

1

`Halpern2012.CS` *Calculate Carbon Storage*

Description

Calculate Carbon Storage

Usage

```
Halpern2012.CS(Cc, Cr, A, ...)
```

Arguments

`placeholder`	placeholder Cc current 'condition' of habitat k
`placeholder`	placeholder Cr reference 'condition' of habitat k
`placeholder`	placeholder A amount of area covered by habitat k

Value

1

`Halpern2012.CW` *Calculate Clean Waters.*

Description

Calculate Clean Waters.

Usage

```
Halpern2012.CW(a, u, l, d, ...)
```

Arguments

`placeholder`	placeholder a number of coastal people without access to sanitation rescaled to global maximum
`placeholder`	placeholder u 1 - (nutrient input)
`placeholder`	placeholder l 1 - (chemical input)
`placeholder`	placeholder d 1 - (marine debris input)

Value

1

`Halpern2012.FP` *Calculate Food Provision.*

Description

Calculate Food Provision.

Usage

```
Halpern2012.FP(w, dBt, mMSY, Bt, Tc, k, Smk, Ac, Yk, ...)
```

Arguments

`placeholder`	placeholder k each mariculture species
`placeholder`	placeholder Smk sustainability score for each species k
`placeholder`	placeholder Ac area of coastal waters (3nm strip)
`placeholder`	placeholder Yl yield of each species k

Value

1

`Halpern2012.FP.FIS` *Calculate Fisheries subgoal of Food Provision.*

Description

Calculate Fisheries subgoal of Food Provision.

Usage

```
Halpern2012.FP.FIS(mMSY, Bt, Tc, ...)
```

Arguments

`placeholder`	placeholder dBt absolute difference between landed biomass and mMSY
`placeholder`	placeholder mMSY multi-species maximum sustainable yield
`placeholder`	placeholder Tc taxonomic report quiality correction factor
`placeholder`	placeholder Bt wild-caught fishing yield

Value

1

`Halpern2012.FP.MAR` *Calculate Mariculture subgoal of Food Provision.*

Description

Calculate Mariculture subgoal of Food Provision.

Usage

```
Halpern2012.FP.MAR(k, Smk, Ac, Yk, ...)
```

Arguments

`placeholder`	placeholder k each mariculture species
`placeholder`	placeholder Smk sustainability score for each species k
`placeholder`	placeholder Ac area of coastal waters (3nm strip)
`placeholder`	placeholder Yl yield of each species k

Value

1

`Halpern2012.ICO` *Calculate Iconic Species subgoal of Sense of Place.*

Description

Calculate Iconic Species subgoal of Sense of Place.

Usage

```
Halpern2012.ICO(S, w, ...)
```

Arguments

`placeholder`	placeholder S number of assessed species in each category
`placeholder`	placeholder w status weight assigned per threat category

Value

1

`Halpern2012.LE` *Calculate Coastal Livelihoods and Economies.*

Description

Calculate Coastal Livelihoods and Economies.

Usage

```
Halpern2012.LE(jc, jr, gc, gr, ec, er, ...)
```

Arguments

`placeholder`	placeholder jc total adjusted jobs per sector at current time
`placeholder`	placeholder jr total adjusted jobs per sector at reference time
`placeholder`	placeholder gc average PPP-adjusted per-capita annual wages per sector in current region
`placeholder`	placeholder gr average PPP-adjusted per-capita annual wages per sector in reference region
`placeholder`	placeholder ec total adjusted revenue generated per sector at current time
`placeholder`	placeholder er total adjusted revenue generated per sector at reference time

Value

1

`Halpern2012.LE.ECO` *Calculate Economies subgoal of Coastal Livelihoods and Economies.*

Description

Calculate Economies subgoal of Coastal Livelihoods and Economies.

Usage

```
Halpern2012.LE.ECO(ec, er, ...)
```

Arguments

`placeholder`	placeholder ec total adjusted revenue generated per sector at current time
`placeholder`	placeholder er total adjusted revenue generated per sector at reference time

Value

1

`Halpern2012.LE.LIV` *Calculate Livelihoods subgoal of Coastal Livelihoods and Economies.*

Description

Calculate Livelihoods subgoal of Coastal Livelihoods and Economies.

Usage

```
Halpern2012.LE.LIV(jc, jr, gc, gr, ...)
```

Arguments

`placeholder`	placeholder jc total adjusted jobs per sector at current time
`placeholder`	placeholder jr total adjusted jobs per sector at reference time
`placeholder`	placeholder gc average PPP-adjusted per-capita annual wages per sector in current region
`placeholder`	placeholder gr average PPP-adjusted per-capita annual wages per sector in reference region

Value

1

`Halpern2012.LSP` *Calculate Lasting Special Places subgoal of Sense of Place.*

Description

Calculate Lasting Special Places subgoal of Sense of Place.

Usage

```
Halpern2012.LSP(CMPA, tCMPA, CP, tCP, ...)
```

Arguments

`placeholder`	placeholder CMPA coastal marine protected area
`placeholder`	placeholder tCMPA total coastal marine area
`placeholder`	placeholder CP coastline protected
`placeholder`	placeholder tCP total coastline

Value

1

`Halpern2012.NP` *Calculate Natural Products. (Needs work)*

Description

Calculate Natural Products. (Needs work)

Usage

```
Halpern2012.NP(N, wp, Hp, E, R, Nv, Nk, w, ...)
```

Arguments

`placeholder`	placeholder N number of products that have ever been harvested
`placeholder`	placeholder wp proportional peak dollar value of each product relative to the total peak dollar value of all products
`placeholder`	placeholder Hp harvest of a product relative to its buffered peak reference point
`placeholder`	placeholder E exposure term
`placeholder`	placeholder R risk term
`placeholder`	placeholder Nv 1 or 2, depending on whether or not a viability term is used
`placeholder`	placeholder Nk number of species in each k category of exploitation
`placeholder`	placeholder w weight assigned to each k category of exploitation status

Value

1

Halpern2012.SP *Calculate Sense of Place.*

Description

Calculate Sense of Place.

Usage

```
Halpern2012.SP(S, w, CMPA, tCMPA, CP, tCP, ...)
```

Arguments

`placeholder`	placeholder S number of assessed species in each category
`placeholder`	placeholder w status weight assigned per threat category
`placeholder`	placeholder CMPA coastal marine protected area
`placeholder`	placeholder tCMPA total coastal marine area
`placeholder`	placeholder CP coastline protected
`placeholder`	placeholder tCP total coastline

Value

1

Halpern2012.TR *Calculate Tourism and Recreation.*

Description

Calculate Tourism and Recreation.

Usage

```
Halpern2012.TR(D, t, V, S, ...)
```

Arguments

`placeholder`	placeholder D number of tourist-days
`placeholder`	placeholder t most recent year
`placeholder`	placeholder V total region population size
`placeholder`	placeholder S sustainability factor

Value

1

`Layers` *Layers reference class.*

Description

Layers reference class.

Usage

```
Layers(...)
```

Arguments

`layers.csv`	path to comma-seperated value file with row of metadata per layer
`layers.dir`	path of directory containing individual layer files

Details

To instantiate this object, `Layers(layers.csv,  layers.dir)` is used. The `layers.csv` is expected to have the following columns:

- *layer* - unique identifier (no spaces or special characters)
- *targets* - the pipe and space (' | ') delimited list of targets (goal name, 'Pressures' or 'Resilience') to feed this data layer
- *title* - full title of the variable
- *description* detailed description
- *citation* - reference for documentation
- *units* - indicating units and required column name in the layer csv file
- *filename* - the csv data file for the layer

The layers.dir directory should contain all the csv filenames listed in the layers.csv file.

Value

object (non-instantiated) reference class of Layers containing

- *meta* - metadata data frame of original layers.csv
- *data* - named list of data frames, one per layer
- *targets* - named list of character vector indicating a layer's targets, goal (status, trend) or dimension (pressures, resilience)

`layers.Global2012.Nature2012ftp` *Layers accompanying Nature 2012 publication on the FTP site for Global 2012 analysis.*

Description

These layers get used to calculate the Ocean Health Index.

Format

a `Layers` object

References

http://ohi-science.org

`layers.Global2012.www2013` *Layers used for the 2013 web launch applied to Global 2012 analysis.*

Description

These layers get used to calculate the Ocean Health Index.

Format

a `Layers` object

References

http://ohi-science.org

`layers.Global2013.www2013` *Layers used for the 2013 web launch applied to Global 2013 analysis.*

Description

These layers get used to calculate the Ocean Health Index.

Format

a `Layers` object

References

http://ohi-science.org

Scores *Scores reference class.*

Description

Scores reference class.

Usage

```
Scores(...)
```

Arguments

`scores.csv`	path to comma-seperated results file, long style

Details

To instantiate this object, `Scores(results.csv)` is used. The `results.csv` is expected to have the following columns:

- *region_id* - unique numeric region identifier, reserving 0 as the region_id for the area-weighted average of the entire study area
- *goal* - the goal code or Index
- *dimension* - the dimension code, one of: status, trend, pressures, resilience, future, score
- *score* - the numeric score: 0-100 for all dimensions, except trend (-1 to 1)

To get the wide view (many columns, with one row per region and columns having combination of goal and dimension), use something like: `reshape2::dcast(.self$long, region_id ~ goal + dimension, value.var='score')`.

Value

object reference class of Layers containing

- *data* - long view (many rows) of score results with columns: region_id, goal, dimension, score

---

scores.Global2012.www2013 *Scores resulting from the 2013 web launch applied to Global 2012 analysis.*

Description

These scores are the results of the Ocean Health Index.

Format

a `Scores` object

References

http://ohi-science.org

`scores.Global2013.www2013` *Scores resulting from the 2013 web launch applied to Global 2013 analysis.*

Description

These scores are the results of the Ocean Health Index.

Format

a `Scores` object

References

http://ohi-science.org

`ScoreScaling` *Score Scaling Functions*

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))
```

`SelectLayersData` *Select Layers to Data*

Description

Select Layers to Data

Usage

```
SelectLayersData(object, targets = NULL, layers = NULL,
  cast = TRUE, narrow = FALSE,
  expand.time.invariant = FALSE)
```

Arguments

`object`	instance of Layers class
`targets`	specifies the targets of layers to be selected, defaulting to `c('regions')`
`layers`	specifies the layers to be selected. If given as a named character vector, then layers get renamed with new names as values, and old names as names per `plyr::rename`
`narrow`	narrow the resulting data frame to just the fields containing data (as described by *flds* in the default wide result) #
`expand.time.invariant`	for layers without a year column, populate the same value throughout all years where available in other layer(s) #
`cast`	whether to cast the resulting dataset, or leave it melted, defaults to TRUE

Details

If neither targets or layers are specified then all layers are returned. If targets and layers are specified, then the union of the two sets of layers are returned, with any renamed layers renamed.

Value

data.frame with the merged data of selected layers having the following fields:

- *layer* - layer name, possibly renamed
- *layer0* - original layer name, if fed a named character vector to layers
- *id_num* - numeric id
- *id_chr* - character id
- *id_name* - fieldname of id
- *category* - category
- *category_name* - fieldname of character
- *year* - year
- *val_num* - numeric value
- *val_chr* - character value
- *val_name* - fieldname of value, usually in units as specified in Layers
- *flds* - data fields used for the layer

SpatialSchemes *SpatialSchemes reference class.*

Description

SpatialSchemes reference class.

Usage

```
SpatialSchemes(...)
```

Value

object (non-instantiated) reference class of SpatialSchemes

TransformSpatialScheme *Transform data*

Description

Transform data

Usage

```
TransformSpatialScheme(object, data, target, origin,
  categories)
```

Arguments

`object`	instance of SpatialSchemes class
`data`	data.frame such as returned from 'SelectLayersData' function
`target`	single spatial scheme to which data should be transformed
`origin`	spatial schemes from which to transform, can be vector
`categories`	layers for which transformation should be done (to be safe, for now this should be all the layers in param data)

Value

data.frame transformed data

Index