

# Package ‘prevR’

August 28, 2020

**Type** Package

**Title** Estimating Regional Trends of a Prevalence from a DHS and Similar Surveys

**Version** 3.4.1

**Maintainer** Joseph Larmarange <joseph.larmarange@ird.fr>

**Acknowledgements** funding from ANRS and IRD, and technical support from LYSIS (info@lysis-consultants.fr)

**Description** Spatial estimation of a prevalence surface or a relative risks surface, using data from a Demographic and Health Survey (DHS) or an analog survey, see Larmarange et al. (2011) <doi:10.4000/cybergeo.24606>.

**License** CeCILL

**URL** <https://github.com/larmarange/prevR/>

**BugReports** <https://github.com/larmarange/prevR/issues>

**LazyLoad** yes

**LazyData** yes

**Depends** R (>= 3.5.0)

**Imports** sp, rgdal (>= 0.7), ggplot2, directlabels, GenKern, fields, gstat, foreign, mapproj, methods

**Suggests** tcltk, geoR, knitr, rmarkdown

**Encoding** UTF-8

**Collate** 'prevR-package.r' 'setClass.prevR.r' 'is.prevR.r' 'as.prevR.r' 'onAttach.r' 'isInputOk.prevR.r' 'changeproj.prevR.r' 'colors.prevR.r' 'as.data.frame.prevR.r' 'plot.prevR.r' 'show.prevR.r' 'print.prevR.r' 'summary.prevR.r' 'rings.prevR.r' 'as.SpatialGrid.prevR.r' 'create.boundary.prevR.r' 'export.prevR.r' 'import.dhs.prevR.r' 'point.in.SpatialPolygons.prevR.r' 'NA.outside.SpatialPolygons.prevR.r' 'xyz2dataframe.prevR.r' 'init.exp.model.variogram.r' 'eyefit.prevR.r' 'krige.prevR.r' 'kde.prevR.r' 'Noptim.r' 'theme\_prevR.r' 'direct.label\_prevR.r' 'quick.prevR.r'

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Joseph Larmarange [aut, cre] (<<https://orcid.org/0000-0001-7097-700X>>)

**Repository** CRAN

**Date/Publication** 2020-08-28 12:20:03 UTC

## R topics documented:

prevR-package . . . . .	3
as.data.frame.prevR . . . . .	6
as.prevR . . . . .	7
as.SpatialGrid,prevR-method . . . . .	8
changeproj,prevR-method . . . . .	9
create.boundary . . . . .	10
direct.label_prevR . . . . .	11
export,prevR-method . . . . .	12
fdhs . . . . .	14
import.dhs . . . . .	14
is.prevR . . . . .	15
kde,prevR-method . . . . .	16
krige,ANY,prevR-method . . . . .	19
NA.outside.SpatialPolygons . . . . .	22
Noptim . . . . .	22
plot,prevR,missing-method . . . . .	23
point.in.SpatialPolygons . . . . .	24
prevR-class . . . . .	25
prevR.colors . . . . .	27
print,prevR-method . . . . .	28
quick.prevR . . . . .	29
rings,prevR-method . . . . .	31
show,prevR-method . . . . .	33
summary,prevR-method . . . . .	33
theme_prevR . . . . .	34
TMWorldBorders . . . . .	35
xyz2dataframe . . . . .	36
<b>Index</b>	<b>37</b>

## Description

**prevR** allows spatial estimation of a prevalence surface or a relative risks surface, using data from a Demographic and Health Survey (DHS) or an analog survey.

## Details

Package: prevR  
Type: Package  
Licence: CeCILL-C - [https://cecill.info/licences/Licence\\_CeCILL-C\\_V1-en.html](https://cecill.info/licences/Licence_CeCILL-C_V1-en.html)  
Website: <https://larmarange.github.io/prevR/>

This package performs a methodological approach for spatial estimation of regional trends of a prevalence using data from surveys using a stratified two-stage sample design (as Demographic and Health Surveys). In these kind of surveys, positive and control cases are spatially positioned at the centre of their corresponding surveyed cluster.

This package provides functions to estimate a prevalence surface using a kernel estimator with adaptive bandwidths of equal number of persons surveyed (a variant of the nearest neighbour technique) or with fixed bandwidths. The prevalence surface could also be calculated using a spatial interpolation (kriging or inverse distance weighting) after a moving average smoothing based on circles of equal number of observed persons or circles of equal radius.

With the kernel estimator approach, it's also possible to estimate a surface of relative risks.

For a quick demo, enter `quick.prevR(fdhs)`.

For a full demo, enter `demo(prevR)`.

The content of **prevR** can be broken up as follows:

### *Datasets*

`fdhs` is a fictive dataset used for testing the package.

`TMWorldBorders` provides national borders of every countries in the World and could be used to define the limits of the studied area.

### *Creating objects*

**prevR** functions takes as input objects of class `prevR`.

`import.dhs()` allows to import easily, through a step by step procedure, data from a DHS (Demographic and Health Surveys) downloaded from <http://www.measuredhs.com>.

`as.prevR()` is a generic function to create an object of class `prevR`.

`create.boundary()` could be used to select borders of a country and transfer them to `as.prevR()` in order to define the studied area.

### *Data visualisation*

Methods `show()`, `print()` and `summary()` display a summary of a object of class `prevR`.

The method `plot()` could be used on a object of class `prevR` for visualising the studied area, spatial position of clusters, number of observations or number of positive cases by cluster.

#### *Data manipulation*

The method `changeproj()` changes the projection of the spatial coordinates.

The method `as.data.frame()` converts an object of class `prevR` into a data frame.

The method `export()` export data and/or the studied area in a text file, a dbf file or a shapefile.

#### *Data analysis*

`rings()` calculates rings of equal number of observations and/or equal radius.

`kde()` calculates a prevalence surface or a relative risks surface using gaussian kernel density estimators (kde) with adaptative bandwidths.

`krige()` executes a spatial interpolation using an ordinary kriging.

`idw()` executes a spatial interpolation using an inverse distance weighting (idw) technique.

#### *Results visualisation and export*

Outputs of `kde()`, `krige()` and `idw()` are objects of class `sp::SpatialPixelsDataFrame`.

Results could be plotted using the function `sp::spplot()`.

`prevR` provides several continuous color palettes (see `prevR.colors`) compatible with `sp::spplot()`.

Calculated surfaces could be export using the function `mapproj::writeAsciiGrid()`.

## Acknowledgement

`prevR` has been developed with funding from the French National Agency for Research on AIDS and Viral Hepatitis (ANRS - <http://www.anrs.fr>) and the French Research Institute for Sustainable Development (IRD - <https://www.ird.fr>), and technical support from LYSIS ([info@lysis-consultants.fr](mailto:info@lysis-consultants.fr)).

## Citation

To cite `prevR`:

Larmarange Joseph, Vallo Roselyne, Yaro Seydou, Msellati Philippe and Meda Nicolas (2011) "Methods for mapping regional trends of HIV prevalence from Demographic and Health Surveys (DHS)", *Cybergeo: European Journal of Geography*, no 558, <https://journals.openedition.org/cybergeo/24606>, DOI: 10.4000/cybergeo.24606.

## Author(s)

Joseph Larmarange <[joseph.larmarange@ird.fr](mailto:joseph.larmarange@ird.fr)>  
IRD - CEPED (UMR 196 Université Paris Descartes Ined IRD)

## References

Larmarange Joseph and Bendaud Victoria (2014) "HIV estimates at second subnational level from national population-based survey", *AIDS*, n° 28, p. S469-S476, DOI: 10.1097/QAD.0000000000000480

Larmarange Joseph, Vallo Roselyne, Yaro Seydou, Msellati Philippe and Meda Nicolas (2011) "Methods for mapping regional trends of HIV prevalence from Demographic and Health Surveys (DHS)", *Cybergeo: European Journal of Geography*, n° 558, <https://journals.openedition.org/cybergeo/24606>, DOI: 10.4000/cybergeo.24606

Larmarange Joseph (2007) *Prévalences du VIH en Afrique : validité d'une mesure*, PhD thesis in demography, directed by Benoît Ferry, université Paris Descartes, <https://tel.archives-ouvertes.fr/tel-00320283>.

Larmarange Joseph, Vallo Roselyne, Yaro Seydou, Msellati Philippe Meda Nicolas and Ferry Benoît (2006), "Cartographier les données des enquêtes démographiques et de santé à partir des coordonnées des zones d'enquête", *Chaire Quételet, 29 novembre au 1er décembre 2006*, Université Catholique de Louvain, Louvain-la-Neuve, Belgique.

## Examples

```
## Not run:
par(ask = TRUE)
# Creating an object of class prevR
col <- c(id = "cluster",
        x = "x",
        y = "y",
        n = "n",
        pos = "pos",
        c.type = "residence",
        wn = "weighted.n",
        wpos = "weighted.pos"
)
dhs <- as.prevR(fdhs.clusters,col, fdhs.boundary)

str(dhs)
print(dhs)

plot(dhs, main="Clusters position")
plot(dhs, type="c.type", main="Clusters by residence")
plot(dhs, type="count", main="Observations by cluster")
plot(dhs, type="flower", main="Positive cases by cluster")

# Changing coordinates projection
plot(dhs,axes=TRUE)
dhs <- changeproj(dhs,
                  "+proj=utm +zone=30 +ellps=WGS84 +datum=WGS84 +units=m +no_defs")
print(dhs)
plot(dhs, axes=TRUE)

# Calculating rings of equal number of observations for different values of N
dhs <- rings(dhs,N=c(100,200,300,400,500))
print(dhs)
summary(dhs)

# Prevalence surface for N=300
prev.N300 <- kde(dhs, N=300, nb.cells=200)
splot(prev.N300, 'k.wprev.N300.RInf',
      cuts=100, col.regions=prevR.colors.red(101),
      main="Regional trends of prevalence (N=300)"
)

# Smoothing ring radii surface (spatial interpolation by kriging)
```

```

radius.N300 <- krige('r.radius', dhs, N=300, nb.cells=200)
splot(radius.N300,
      cuts=100, col.regions=prevR.colors.blue(101),
      main="Radius of circle (N=300)"
)
par(ask = FALSE)

## End(Not run)

```

---

as.data.frame.prevR    *Convert an object of class prevR into a data.frame.*

---

### Description

This function merges the slots clusters et rings of a object of class [prevR](#).

### Usage

```

## S3 method for class 'prevR'
as.data.frame(x, ..., N = NULL, R = NULL, clusters.only = FALSE)

```

### Arguments

x	object of class <a href="#">prevR</a> .
...	not used, for compatibility with the generic method <code>base::as.data.frame()</code> .
N	integer or list of integers setting elements of rings to extract.
R	integer or list of integers setting elements of rings to extract.
clusters.only	return only the slot clusters of x?

### Value

If `clusters.only = TRUE`, the function will return only the slot clusters of x.

Otherwise, slots clusters and rings of x will be merged in a unique data frame. The columns of rings will be renamed adding a suffix like *.N300.RInf*.

N and R define the elements of rings to extract. If not specified (NULL), all the elements of rings will be included.

### See Also

[base::as.data.frame\(\)](#), [prevR](#).

**Examples**

```

str(fdhs)
str(as.data.frame(fdhs))
## Not run:
r.fdhs <- rings(fdhs, N=c(100,200,300))
str(r.fdhs)
str(as.data.frame(r.fdhs, clusters.only=TRUE))
str(as.data.frame(r.fdhs))
str(as.data.frame(r.fdhs, N=300))

## End(Not run)

```

---

as.prevR

*Create an object of class prevR.*


---

**Description**

This function creates an object of class `prevR` from a data frame.

**Usage**

```
as.prevR(data, col, boundary = NULL, proj = "+proj=longlat +datum=WGS84")
```

**Arguments**

data	data frame, each line corresponding to an observed cluster.
col	vector identifying the columns of data to use. clusters columns names are fixed: <ul style="list-style-type: none"> <li>• "id" (optional) cluster's identifier.</li> <li>• "x" cluster's longitude.</li> <li>• "y" cluster's latitude.</li> <li>• "n" number of valid observations in the cluster.</li> <li>• "pos" number of positive cases in the cluster.</li> <li>• "wn" (optional) sum of observations weight.</li> <li>• "wpos" (optional) sum of positive cases weight.</li> <li>• "c.type" (optional) type of cluster (used only by <code>plot()</code>).</li> </ul> See examples.
boundary	object of class <code>sp::SpatialPolygons</code> defining the studied area.
proj	projection of clusters coordinates used in data (longitude and latitude in decimal degrees by default).

**Details**

Only "x", "y" "n" and "pos" are required in col. If "id" is not specified, a numerical identifier will be automatically created.

proj defines projection used by data. It could be a character string corresponding to a *PROJ.4* projection (see <https://proj.org/> for more details) or an object of class `sp::CRS`.

If the projection of boundary is defined in a slot called `proj4string`, boundary will be projected according to `proj`. If the slot `proj4string` is missing, boundary will be considered to be already in the same projection as `proj`.

If `boundary` is not defined (NULL), a considered corresponding to minimal and maximal coordinates of data will be used.

`boundary` could be the result of the function `create.boundary()`.

It's not possible to change projection of data with `as.prevR()`. Use `changeproj()` instead.

**Value**

Object of class `prevR`

**See Also**

`prevR` class, `create.boundary()`, `changeproj()`, `import.dhs()`.

**Examples**

```
col <- c(id = "cluster",
        x = "x",
        y = "y",
        n = "n",
        pos = "pos",
        c.type = "residence",
        wn = "weighted.n",
        wpos = "weighted.pos"
)
dhs <- as.prevR(fdhs.clusters,col, fdhs.boundary)

str(dhs)
print(dhs)
```

---

as.SpatialGrid,prevR-method

*Create a spatial grid from an object of class prevR.*

---

**Description**

This function generates a spatial rectangular grid from the slot `boundary` of an object of class `prevR`; function used in particular by the methods `kde()`, `krige()` and `idw()`.



**Usage**

```
## S4 method for signature 'prevR'
as.SpatialGrid(object, nb.cells = 100, cell.size = NULL)
```

**Arguments**

object	object of class <a href="#">prevR</a> .
nb.cells	number of cells on the longest side of the studied area (unused if cell.size is defined).
cell.size	size of each cell (in the unit of the projection).

**Details**

This function generates a spatial rectangular grid, each cell being a square of side cell.size. If cell.size is not defined, side of cells will be calculated as the longest side of the slot boundary of object divided by nb.cells.

**Value**

Object of class [sp::SpatialGrid](#).

**See Also**

[sp::GridTopology\(\)](#), [sp::SpatialGrid](#).

**Examples**

```
str(as.SpatialGrid(fdhs))
str(as.SpatialGrid(fdhs, nb.cells=200))
```

---

changeproj,prevR-method

*Convert map projection of a object of class prevR.*

---

**Description**

This function converts map projection (and/or datum) used by an object of class [prevR](#) into another one.

**Usage**

```
## S4 method for signature 'prevR'
changeproj(object, proj)
```

**Arguments**

object            object of class [prevR](#).  
 proj             new map projection.

**Details**

proj could be a character string corresponding to a *PROJ.4* projection (see <https://proj.org/> for more details) or an object of class [sp::CRS](#).

changeproj transform the columns "x" and "y" of the slot clusters of object and convert boundary using the new map projection defined by proj.  
 If applicable, the slot rings will be recalculated.

**Value**

Return object expressed in the projection proj.

**See Also**

[rgdal::spTransform\(\)](#), [prevR](#).

**Examples**

```
print(fdhs)
plot(fdhs, axes=TRUE, main="Projection: longitude/latitude")

fdhs2 <- changeproj(fdhs,
                    "+proj=utm +zone=30 +ellps=WGS84 +datum=WGS84 +units=m +no_defs")
print(fdhs2)
plot(fdhs2, axes=TRUE, main="Projection: UTM Zone 30")
```

---

create.boundary            *Provide national boundaries of a country.*

---

**Description**

This function uses the data set [TMWorldBorders](#). One or several countries can be selected and will be returned as an object of class [sp::SpatialPolygons](#).

**Usage**

```
create.boundary(
  countries = NULL,
  multiple = F,
  proj = "+proj=longlat +datum=WGS84"
)
```

**Arguments**

countries	a vector of character string corresponding to the name of the countries you want to extract from the dataset. If NULL, a dialogue box will be appear in order to select the desired country.
multiple	should the dialog box allow multiple selection (unused if countries is specified)?
proj	map projection to use for the result (longitude and latitude in decimal degrees by default).

**Details**

proj could be a character string corresponding to a *PROJ.4* projection (see <https://proj.org/> for more details) or an object of class `sp::CRS`.

**Value**

Object of class `sp::SpatialPolygons`.

**Note**

The result will be automatically plotted.

**See Also**

[TMWorldBorders](#).

**Examples**

```
## Not run:
boundary <- create.boundary()

## End(Not run)

boundary <- create.boundary("Burkina Faso")
boundary <- create.boundary("Burkina Faso",
  proj="+proj=utm +zone=30 +ellps=WGS84 +datum=WGS84 +units=m +no_defs")
boundary <- create.boundary(countries = c("Burkina Faso", "Ghana", "Benin"))
```

---

direct.label\_prevR      *Direct label on a ggplot object*

---

**Description**

Direct label a ggplot2 grouped plot

**Usage**

```
direct.label_prevR(p, method = NULL, debug = FALSE)
```

**Arguments**

p	The ggplot object.
method	Method for direct labelling (see <a href="#">directlabels::direct.label()</a> ).
debug	Show debug output?

**Value**

The ggplot object with direct labels added.

**Note**

This function is based on and similar to [directlabels::direct.label\(\)](#) except that legend is not hidden.

**See Also**

[directlabels::direct.label\(\)](#)

---

export,prevR-method    *Export an object of class prevR.*

---

**Description**

This method could be used to export an object of class [prevR](#) in different formats (text, shapefile, dbase...)

**Usage**

```
## S4 method for signature 'prevR'  
export(  
  object,  
  element,  
  format,  
  file,  
  N = NULL,  
  R = NULL,  
  clusters.only = FALSE,  
  ext = NULL,  
  sep = NULL,  
  dec = NULL,  
  ...  
)
```

**Arguments**

object	object of class <a href="#">prevR</a> .
element	element to export: "clusters" or "boundary".
format	format: "dbf", "txt", "csv", "csv2" or "shp" (unused if element="boundary").
file	file name without extension.
N	integer or list of integers setting elements of rings to export (unused if element="boundary").
R	integer or list of integers setting elements of rings to export (unused if element="boundary").
clusters.only	export only the slot clusters of object (unused if element="boundary")?
ext	coerce the extension of the export file (unused if element="boundary" or if format="shp").
sep	coerce the field separator string (unused if element="boundary" or if format="shp" or if format="dbf").
dec	coerce the string to use for decimal point (unused if element="boundary" or if format="shp" or if format="dbf").
...	additional arguments transmitted to <a href="#">rgdal::writeOGR()</a> , <a href="#">foreign::write.dbf()</a> or <a href="#">utils::write.table()</a> .

**Details**

If element="boundary", the slot boundary of object will be exported as a *shapefile*.

Otherwise, the slot clusters, merged with the slot rings, will be exported.

See [as.data.frame\(\)](#) for details on the use of the parameters of N, R et clusters.only.

format specifies the export format of the data frame returned by [as.data.frame\(\)](#):

"shp"	Shape File (require the package <b>rgdal</b> )
"dbf"	DBASE format (extension: .dbf, require the package <b>foreign</b> )
"txt"	tabulated text (extension: .txt)
"csv"	'comma separated values' (extension: .csv)
"csv2"	CSV variant using a semicolon as field separator (extension: .csv)

ext could be used to coerce the extension of the output file, except for *shapefile* export, which will write four different files (.shp, .shx, .dbf and .prj).

The "txt" format uses by default a tabulation as field separator and a point "." for decimal point. The "csv" format uses a comma "," as field separator and a point "." as decimal point.

The "csv2" format is a variant using a semicolon ";" as field separator and a colon ":" for decimal point, the Excel convention for CSV files in some Western European locales.

sep and dec could be used to coerce the field separator and the decimal point (together with the "txt" format).

**See Also**

[mapproj::writePolyShape](#), [mapproj::writePointsShape\(\)](#), [foreign::write.dbf\(\)](#), [utils::write.table\(\)](#).

**Examples**

```
## Not run:
export(fdhs, element="boundary", file="area")
export(fdhs, element="clusters", format="shp", file="points")

dhs <- rings(fdhs,N=c(100,300,500))
export(dhs, element="clusters", format="csv", N=300, file="points")

## End(Not run)
```

---

 fdhs

*Fictitious data generated by a DHS simulation.*


---

**Description**

Data set generated by a Demographic and Health Survey (DHS) simulation on a fictitious country with a national prevalence of 10\ This dataset is composed of 3 objects:

- `fdhs.clusters`: data frame (one line per cluster).
- `fdhs.boundary`: object of class `sp::SpatialPolygons` corresponding to the borders of the fictitious country.
- `fdhs`: object of class `prevR` returned by `as.prevR()` using the two previous objects.

**Examples**

```
## Not run:
str(fdhs)
str(fdhs.clusters)
str(fdhs.boundary)
demo(prevR)

## End(Not run)
```

---

 import.dhs

*Import DHS data.*


---

**Description**

This step by step function guides users to import data from a Demographic and Health Survey (DHS) and create an object of class `prevR`.

**Usage**

```
import.dhs(file.sav, file.dbf)
```

**Arguments**

file.sav	DHS data (one individual per line) in SPSS format (.sav), downloaded from <a href="https://www.dhsprogram.com/">https://www.dhsprogram.com/</a> . Could also be directly a data.frame.
file.dbf	GPS position of clusters in DATABASE format (.dbf), downloaded from <a href="https://www.dhsprogram.com/">https://www.dhsprogram.com/</a> . Could also be directly a data.frame.

**Note**

If you don't provide the precise path of files, R will check the working directory (see `base::setwd()`). To specify the file path, see `base::file.path()`.

This function was developed specifically for importing DHS. For a generic function for creating an object of class `prevR`, see `as.prevR()`.

**See Also**

`as.prevR()`, `prevR` class.

**Examples**

```
## Not run:
imported_data <- import.dhs("data.sav", "gps.dbf")

## End(Not run)
```

---

is.prevR	<i>Test if an object is of class prevR. This function test if the class of an object is <code>prevR</code>. It could be used to test the slot rings or the slot boundary.</i>
----------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

**Description**

Test if an object is of class `prevR`. This function test if the class of an object is `prevR`. It could be used to test the slot rings or the slot boundary.

**Usage**

```
is.prevR(object, slot = NULL)
```

**Arguments**

object	object to test.
slot	"clusters", "rings", "boundary" or "proj".

**Details**

Slots rings and boundary are always present in an object of class `prevR`, but rings could be NULL and boundary a `sp::SpatialPolygons` with an attribute named `valid` with the value FALSE (when boundaries of the studied area have not been specified explicitly).

- If rings is NULL, `is.prevR(object, "rings")` will return FALSE.
- If boundary has an attribute `valid` equal to FALSE, `is.prevR(object, "boundary")` will return FALSE.

**Value**

TRUE or FALSE.

**See Also**

`prevR`.

**Examples**

```
col <- c(id = "cluster",
        x = "x",
        y = "y",
        n = "n",
        pos = "pos",
        c.type = "residence",
        wn = "weighted.n",
        wpos = "weighted.pos"
)
dhs <- as.prevR(fdhs.clusters,col, fdhs.boundary)

is.prevR(dhs)
is.prevR(dhs,"rings")
is.prevR(dhs,"boundary")

dhs <- rings(dhs,N=300)
is.prevR(dhs,"rings")
```

---

kde,prevR-method

*Kernel density estimation for prevR object.*

---

**Description**

This function allows to calculate a prevalence surface (ratio of two intensity surfaces) and/or a relative risks surface (ratio of two density surfaces) using gaussian kernel estimators with adaptative bandwidths of equal number of observations or equal radius.



**Usage**

```
## S4 method for signature 'prevR'
kde(
  object,
  N = NULL,
  R = NULL,
  weighted = TRUE,
  risk.ratio = FALSE,
  keep.details = FALSE,
  nb.cells = 100,
  cell.size = NULL,
  progression = TRUE,
  short.names = FALSE
)
```

**Arguments**

object	object of class <a href="#">prevR</a> .
N	integer or list of integers corresponding to the rings to use.
R	integer or list of integers corresponding to the rings to use.
weighted	use weighted data (TRUE, FALSE or "2")?
risk.ratio	calculate a relative risks surface instead of a prevalence surface (TRUE, FALSE or "2")?
keep.details	return surface of positive cases and surface of observed cases?
nb.cells	number of cells on the longest side of the studied area (unused if cell.size is defined).
cell.size	size of each cell (in the unit of the projection).
progression	show a progress bar?
short.names	should names of the output be short?

**Details**

This function calculates a prevalence surface as the ratio of the intensity surface (expressed in cases per surface unit) of positive cases on the intensity surface of observed cases and could also calculate a relative risks surface corresponding to the ratio of the density surface (whose integral has been normalized to one) of positive cas on density surface of observed cases.

This method is a variant of the nearest neighbour technique. Surfaces are estimated using gaussian kernel estimators with adaptative bandwidths, bandwidth size being determined by a minimum number of observations in the neighbourhood (see [rings\(\)](#) for more details). Fixed bandwidths could also be used. More precisely, the bandwidth used is half the radius of rings of equal number of observations or equal radius (parameters N and R) calculated by the function [rings\(\)](#). See referenes for a detailed explanation of the implemented methodology.

N and R determine the rings to use for the estimation. If they are not defined, surfaces will be estimated for each available couples (N,R). Several estimations could be simultaneously calculated if several values of N and R are defined.

A suggested value of N could be computed with [Noptim\(\)](#).

**Value**

Object of class `sp::SpatialPixelsDataFrame`. Surfaces are named according to the name of the corresponding variable, N and R (for example: `k.prev.N300.RInf`). If `short.names` is TRUE and if there is only one combination of couples (N, R), variable names will not be suffixed by the value of N and R.

Estimated variables are (depending on the function parameters) :

- "k.pos" unweighted intensity surface of positive cases.
- "k.obs" unweighted intensity surface of observed cases.
- "k.prev" unweighted surface of prevalence (k.pos/k.obs).
- "k.case" unweighted density surface of positive cases.
- "k.control" unweighted density surface of observed cases.
- "k.rr" unweighted surface of relative risks (k.case/k.control).
- "k.wpos" weighted intensity surface of positive cases.
- "k.wobs" weighted intensity surface of observed cases.
- "k.wprev" weighted surface of prevalence (k.wpos/k.wobs).
- "k.wcase" weighted density surface of positive cases.
- "k.wcontrol" weighted density surface of observed cases.
- "k.wrr" weighted surface of relative risks (k.wcase/k.wcontrol).

NA value is applied to cells of the grid located outside of the studied area (see `NA.outside.SpatialPolygons()`).

**Note**

Results could be plotted with `sp::spplot()`.

**prevR** provides several continuous color palettes (see `prevR.colors`) compatible with `sp::spplot()`. Calculated surfaces could be export using the function `mapproj::writeAsciiGrid()`.

See the package **sparr** for another methodology to estimate relative risks surfaces, adapted for other kind of data than Demographic and Health Surveys (DHS).

**References**

Larmarange Joseph, Vallo Roselyne, Yaro Seydou, Msellati Philippe and Meda Nicolas (2011) "Methods for mapping regional trends of HIV prevalence from Demographic and Health Surveys (DHS)", *Cybergeo: European Journal of Geography*, no 558, <https://journals.openedition.org/cybergeo/24606>, DOI: 10.4000/cybergeo.24606.

**See Also**

`GenKern::KernSur()`, `rings()`, `Noptim()`.

**Examples**

```
## Not run:
dhs <- rings(fdhs, N=c(100,200,300,400,500))

prev.N300 <- kde(dhs, N=300, nb.cells=200)

library(sp)
spplot(prev.N300, 'k.wprev.N300.RInf',
        cuts=100, col.regions=prevR.colors.red(101),
        main="Regional trends of prevalence (N=300)"
)

prev.krige <- kde(dhs, N=c(100,300,500), R=Inf,
                 nb.cells=200, risk.ratio=2, keep.details=FALSE
)
str(prev.krige)
spplot(prev.krige,
        c('k.wprev.N100.RInf', 'k.wprev.N300.RInf', 'k.wprev.N500.RInf'),
        cuts=100, col.regions=prevR.colors.red(101)
)

## End(Not run)
```

---

```
krige,ANY,prevR-method
```

*Spatial interpolation (kriging and inverse distance weighting) for objects of class prevR.*

---

**Description**

These functions execute a spatial interpolation of a variable of the slot rings of an object of class [prevR](#). The method `krige` implements the ordinary kriging technique. The method `idw` executes an inverse distance weighting interpolation.

**Usage**

```
## S4 method for signature 'ANY,prevR'
krige(
  formula,
  locations,
  N = NULL,
  R = Inf,
  model = NULL,
  nb.cells = 100,
  cell.size = NULL,
  fit = "auto",
  keep.variance = FALSE,
```

```

    show.variogram = FALSE,
    ...
)

## S4 method for signature 'ANY,prevR'
idw(
  formula,
  locations,
  N = NULL,
  R = Inf,
  nb.cells = 100,
  cell.size = NULL,
  idp = 2,
  ...
)

```

### Arguments

formula	variable(s) to interpolate (see details).
locations	object of class <a href="#">prevR</a> .
N	integer or list of integers corresponding to the rings to use.
R	integer or list of integers corresponding to the rings to use.
model	a variogram model returned by the function <a href="#">gstat::vgm()</a> .
nb.cells	number of cells on the longest side of the studied area (unused if cell.size is defined).
cell.size	size of each cell (in the unit of the projection).
fit	"auto" for using a variogram automatically fitted from the data, "manual" for using a variogram fitted through a graphic interface (unused if model is defined).
keep.variance	return variance of estimates?
show.variogram	plot the variogram?
...	additional arguments transmitted to <a href="#">gstat::krige()</a> or <a href="#">gstat::idw()</a> .
idp	inverse distance weighting power (see <a href="#">gstat::idw()</a> ).

### Details

formula specifies the variable(s) to interpolate. Only variables available in the slot rings of locations could be used. Possible values are "r.pos", "r.n", "r.prev", "r.radius", "r.clusters", "r.wpos", "r.wn" ou "r.wprev". Variables could be specified with a character string or a formula (example: list(r.pos~1,r.prev~1). Only formula like variable.name~1 are accepted. For more complexe interpolations, use directly functions [gstat::krige\(\)](#) and [gstat::idw\(\)](#) from **gstat**.

N and R determine the rings to use for the interpolation. If they are not defined, surfaces will be estimated for each available couples (N,R). Several interpolations could be simultaneously calculated if several variables and/or several values of N and R are defined.

A suggested value of N could be computed with [Noptim\(\)](#).

In the case of an ordinary kriging, the method `krige()` from **prevR** will try to fit automatically an exponential variogram to the sample variogram (`fit="auto"`). If you choose `fit="manual"`, the sample variogram will be plotted and a graphical dialog box (adapted from `geoR::eyefit()`) will appear for a manual and visual fitting. You can also specify directly the variogram to use with the parameter `model`. Packages **geoR** and **tcltk** are required for manual fit.

Interpolations are calculated on a spatial grid obtained with `as.SpatialGrid()`.

## Value

Object of class `sp::SpatialPixelsDataFrame`. The name of estimated surfaces depends on the name of the interpolated variable, N and R (for example: `r.radius.N300.RInf`). If you ask the function to return variance (`keep.variance=TRUE`), corresponding surfaces names will have the suffix `.var`.

NA value is applied to points located outside of the studied area (voir `NA.outside.SpatialPolygons()`).

## Note

Results could be plotted with `sp::spplot()`.

**prevR** provides several continuous color palettes (see `prevR.colors`) compatible with `sp::spplot()`. Calculated surfaces could be export using the function `maptools::writeAsciiGrid()`.

## References

Larmarange Joseph, Vallo Roselyne, Yaro Seydou, Msellati Philippe and Meda Nicolas (2011) "Methods for mapping regional trends of HIV prevalence from Demographic and Health Surveys (DHS)", *Cybergeo: European Journal of Geography*, no 558, <https://journals.openedition.org/cybergeo/24606>, DOI: 10.4000/cybergeo.24606.

## See Also

`gstat::krige()`, `gstat::idw()`, `rings()`, `Noptim()`.

## Examples

```
## Not run:
dhs <- rings(fdhs, N = c(100,200,300,400,500))
radius.N300 <- krige('r.radius', dhs, N = 300, nb.cells = 200)
prev.krige <- krige(r.wprev ~ 1, dhs, N = c(100, 300, 500))
library(sp)
spplot(prev.krige, c('r.wprev.N100.RInf', 'r.wprev.N300.RInf', 'r.wprev.N500.RInf'))

## End(Not run)
```

NA.outside.SpatialPolygons

*Affect NA value to points located outside a polygon.*

---

### Description

This function forces points of an object of class [sp::SpatialPixelsDataFrame](#) located outside the limits defined by an object of class [sp::SpatialPolygons](#) to NA.

### Usage

```
NA.outside.SpatialPolygons(sp.data, sp.poly)
```

### Arguments

sp.data            object of class [sp::SpatialPixelsDataFrame](#).  
sp.poly            object of class [sp::SpatialPolygons](#).

### Value

Return sp.data modified.

### See Also

[point.in.SpatialPolygons\(\)](#).

---

Noptim

*Suggested optimal value for N*

---

### Description

Based on previous simulation work, the function suggests an optimal value for the N parameter based on national prevalence, the total number of observations and the number of clusters. See Larmarange et al. 2011 for more details.

### Usage

```
Noptim(object)
```

### Arguments

object            object of class [prevR](#).

### Value

an integer.

## References

Larmarange Joseph, Vallo Roselyne, Yaro Seydou, Msellati Philippe and Meda Nicolas (2011) "Methods for mapping regional trends of HIV prevalence from Demographic and Health Surveys (DHS)", *Cybergeo: European Journal of Geography*, no 558, <https://journals.openedition.org/cybergeo/24606>, DOI: 10.4000/cybergeo.24606.

## Examples

```
Noptim(fdhs)
```

---

```
plot,prevR,missing-method
```

*Plot object of class prevR.*

---

## Description

Method plot for object of class `prevR`. Plot clusters, number of observations per cluster or number of positive cases per cluster.

## Usage

```
## S4 method for signature 'prevR,missing'
plot(
  x,
  type = "position",
  add.legend = TRUE,
  legend.location = "bottomright",
  factor.size = 0.2,
  new.window = FALSE,
  axes = F,
  ...
)
```

## Arguments

<code>x</code>	object of class <code>prevR</code> .
<code>type</code>	graph to plot: <ul style="list-style-type: none"> <li>• "position" clusters position.</li> <li>• "c.type" clusters per c. type.</li> <li>• "count" number of observations per cluster.</li> <li>• "flower" number of positive cases per cluster.</li> </ul>
<code>add.legend</code>	add a legend?
<code>legend.location</code>	legend location.

factor.size	scale factor of rings (for type="count").
new.window	plot in a new window?
axes	show axes?
...	additional arguments transmitted to <code>graphics::title()</code> .

**Details**

Available values for legend.location are: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" use "center".

Use main to define a title and sub for a subtitle (see `graphics::title()`).

**See Also**

`graphics::title()`, `graphics::legend()`.

**Examples**

```
plot(fdhs, type = "position", main="position", axes=TRUE)
plot(fdhs, type = "c.type", main="c.type")
plot(fdhs, type = "count", main="count", factor.size = 0.1)
plot(fdhs, type = "flower", main="flower")
```

---

point.in.SpatialPolygons

*Test if a point falls within a given polygon.*

---

**Description**

This function tests if one or more points are located inside an object of class `sp::SpatialPolygons`.

**Usage**

```
point.in.SpatialPolygons(point.x, point.y, SpP)
```

**Arguments**

point.x	numerical array of x-coordinates of points.
point.y	numerical array of y-coordinates of points.
SpP	object of class <code>sp::SpatialPolygons</code>

**Value**

Boolean array.



**See Also**

`sp::point.in.polygon()`, `[NA.outside.SpatialPolygons()]`.  
`[NA.outside.SpatialPolygons()]`: `R:NA.outside.SpatialPolygons()`

---

```
prevR-class
```

```
Objects of class prevR.
```

---

**Description**

Class used by the package **prevR**

**Slots**

`clusters` data.frame with observed data (one line per cluster). Columns names are:

- "id" cluster ID.
- "x" longitude.
- "y" latitude.
- "n" number of valid observations per cluster.
- "pos" number of positive cases per cluster.
- "prev" observed prevalence (in \
- "wn" (optional) sum of weights of observations per cluster.
- "wpos" (optional) sum of weights of positive cases per cluster.
- "wprev" (optional) weighted observed prevalence (in \
- "c.type" (optional) cluster type.

`boundary` object of class `sp::SpatialPolygons`, borders of the studied area.

`proj` object of class `sp::CRS`, map projection used.

`rings` list of results returned by `rings()`. Each entry is composed of 3 elements: N, minimum number of observations per ring; R, maximum radius of rings and estimates, a data frame with the following variables:

- "id" cluster ID.
- "r.pos" number of positive cases inside the ring.
- "r.n" number of valid observations inside the ring.
- "r.prev" observed prevalence (in \
- "r.radius" ring radius (in kilometers if coordinates in decimal degrees, in the unit of the projection otherwise).
- "r.clusters" number of clusters located inside the ring.
- "r.wpos" (optional) sum of weights of positive cases inside the ring.
- "r.wn" (optional) sum of weights of valid observations inside the ring.
- "r.wprev" (optional) weighted observed prevalence (in \

Note: the list `rings` is named, the name of each element is `NN_value.RR_value`, for example `N300.RInf`.

## Objects from the Class

Objects of this class could be created by the function `as.prevR()`.

## Methods

**as.data.frame** signature(`x = "prevR"`) converts an object of class `prevR` into a data frame.

**as.SpatialGrid** signature(`object = "prevR"`) generates a spatial grid.

**export** signature(`object = "prevR"`) exports a `prevR` object as a shapefile, a dbase file or a text file.

**idw** signature(`formula = "ANY"`, `locations = "prevR"`) calculates a spatial interpolation using an inverse distance weighting.

**kde** signature(`object = "prevR"`) estimates a prevalence surface using kernel density estimators.

**krige** signature(`formula = "ANY"`, `locations = "prevR"`) calculates a spatial interpolation by kriging.

**plot** signature(`x = "prevR"`, `y = "ANY"`) plots data of a `prevR` object.

**print** signature(`x = "prevR"`) shows a summary of a `prevR` object.

**rings** signature(`object = "prevR"`) calculates rings of equal number of observations and/or equal radius.

**show** signature(`object = "prevR"`) shows a summary of a `prevR` object.

**summary** signature(`object = "prevR"`) shows a summary of the variables of a `prevR` object.

**changeproj** signature(`object = "prevR"`) changes the map projection used.

## See Also

`as.prevR()`, `is.prevR()`, `changeproj()`, `rings()`, `print()`, `plot()`, `summary()`, `kde()`, `krige()`, `idw()`, `export()`.

## Examples

```
showClass("prevR")

col <- c(id = "cluster",
        x = "x",
        y="y",
        n="n",
        pos = "pos",
        c.type = "residence",
        wn="weighted.n",
        wpos="weighted.pos"
)
dhs <- as.prevR(fdhs.clusters,col, fdhs.boundary)
str(dhs)
print(dhs)

## Not run:
dhs <- rings(fdhs,N=c(100,300,500))
```

```
str(dhs)
print(dhs)

## End(Not run)
```

---

prevR.colors

*Continuous color palettes.*

---

## Description

Functions generating color palettes useable with R graphical functions, in particular with `sp::splot()`. These palettes are continuous, contrast being accentuated by darkening and lightening extrem values. `prevR.demo.pal` plot the available palettes. `prevR.colors.qgis.pal` export a palette in a text file readable by Quantum GIS, an open-source mapping software.

## Usage

```
prevR.colors.blue(n)

prevR.colors.blue.inverse(n)

prevR.colors.gray(n)

prevR.colors.gray.inverse(n)

prevR.colors.green(n)

prevR.colors.green.inverse(n)

prevR.colors.red(n)

prevR.colors.red.inverse(n)

prevR.demo.pal(n, border = if (n < 32) "light gray" else NA, main = NULL)

prevR.colors.qgis.pal(file, at, pal = "red", inverse = FALSE)
```

## Arguments

n	number of different colors in the palette.
border	border color.
main	title.
file	file name with extension.
at	list of values of the palette.
pal	color palette to use ("red", "green", "blue" or "gray").
inverse	use the inverse palette?

**Details**

[prevR.colors.red\(\)](#) produces a color gradation from white/yellow to red/dark red.  
[prevR.colors.blue\(\)](#) produces a color gradation from light blue to dark blue.  
[prevR.colors.green\(\)](#) produces a color gradation from light green to dark green.  
[prevR.colors.gray\(\)](#) produces a color gradation from white/light gray to dark gray/black.

Functions with a suffix *.inverse* produce the same color gradation, but from dark colors to light ones.

**Value**

[prevR.demo.pal\(\)](#) plot the color palettes.  
[prevR.colors.qgis.pal\(\)](#) export a color palette in a text file readable by Quantum GIS.  
 The other functions return a list of colors coded in hexadecimal.

**Note**

To obtain the list of colors in RGB (Red/Green/Blue), use the function [grDevices::col2rgb\(\)](#). The code of [prevR.demo.pal\(\)](#) was adapted from the function `demo.pal` presented in the examples of [grDevices::rainbow\(\)](#).

**See Also**

Other color palettes are available in R. See for example [grDevices::rainbow\(\)](#) or the package **RColorBrewer**.

**Examples**

```

prevR.demo.pal(25)
prevR.colors.red(5)
col2rgb(prevR.colors.red(5))

## Not run:
prevR.colors.qgis.pal('palette.txt', seq(0,25,length.out=100), 'red')

## End(Not run)
  
```

---

print,prevR-method      *Summary of a prevR object.*

---

**Description**

Method print for objects of class [prevR](#): shows a summary of the object's characteristics.

**Usage**

```

## S4 method for signature 'prevR'
print(x)
  
```

**Arguments**

x                      object of class `prevR`.

**Note**

Exactly the same as `show()`.

**See Also**

`summary()`.

**Examples**

```
print(fdhs)
## Not run:
dhs <- rings(fdhs, N=c(100, 300, 500))
print(dhs)

## End(Not run)
```

---

quick.prevR

*Quick prevR analysis and plot*

---

**Description**

This function performs several analysis in one go: (i) apply `rings()`; (ii) compute prevalence surface with `kde()`; (iii) compute the surface of rings radii with `krige()`; (iv) plot prevalence surface using `prevR.colors.red()` and add rings radii as a contour plot.

**Usage**

```
quick.prevR(
  object,
  N = Noptim(object),
  nb.cells = 100,
  cell.size = NULL,
  weighted = NULL,
  plot.results = TRUE,
  return.results = FALSE,
  return.plot = FALSE,
  legend.title = "%",
  cex = 0.7,
  progression = TRUE
)
```

### Arguments

object	object of class <a href="#">prevR</a> .
N	integer or list of integers corresponding to the rings to use.
nb.cells	number of cells on the longest side of the studied area (unused if <code>cell.size</code> is defined).
cell.size	size of each cell (in the unit of the projection).
weighted	use weighted data (TRUE, FALSE or "2")?
plot.results	plot the results?
return.results	return the results?
return.plot	return the plot within the results?
legend.title	title of the legend
cex	to control the text size on the graph
progression	show a progress bar?

### Details

N determine the rings to use for the estimation. By default, a suggested value of N will be computed with [Noptim\(\)](#).

### Value

A list of one or several elements, depending on the arguments: (i) `prev` is a `SpatialPixelsDataFrame` containing the prevalence surface; (ii) `radius` a `SpatialPixelsDataFrame` containing the kriged surface of the rings radii; (iii) `plot` a `ggplot` graph.

### See Also

[Noptim\(\)](#), [rings\(\)](#), [kde\(\)](#) and [krige\(\)](#).

### Examples

```
## Not run:  
quick.prevR(fdhs)  
  
## End(Not run)
```

---

rings,prevR-method	<i>Calculation of rings of equal number of observation and/or equal radius.</i>
--------------------	---------------------------------------------------------------------------------

---

## Description

For each cluster, this function determines a ring of equal number of observations and/or equal radius and calculates several indicators from observations located inside that ring.

## Usage

```
## S4 method for signature 'prevR'
rings(object, N = seq(100, 500, 50), R = Inf, progression = TRUE)
```

## Arguments

object	object of class <a href="#">prevR</a> .
N	minimum number of observations.
R	maximum rings radius (in kilometers if coordinates in decimal degrees, in the unit of the projection otherwise).
progression	show a progress bar?

## Details

For each ligne of the data frame `clusters` of `object`, `rings` determines a ring, centred on the cluster. It could be:

- rings of eaqul number of observations if N is finite and R=Inf;
- rings of equal radius if N=Inf and R is finite;
- a combination of both (see below) if N and R are finite.

For *rings of equal number of observations*, `rings` selects the smallest ring containing at least N valid observations.

For *rings of equal radius*, `rings` selects all clusters located at a lower distance than R from the central cluster.

For *combination of both*, `rings` calculates firts the ring with the minimum number of observations and test if its radius is lower than R or not. If so, the ring is kept, otherwise the ring of maximum radius is calculated.

Different series of rings could be simultaneoulsy calculated by providing different values for N and R. `rings` will calculate rings corresponding to each couple (N,R).

**Value**

Return object with the slot `rings` completed for each couple (N,R).

Each entry is composed of 3 elements: N, minimum number of observations per ring; R, maximum radius of rings and estimates, a data frame with the following variables:

- "id" cluster ID.
- "r.pos" number of positive cases inside the ring.
- "r.n" number of valid observations inside the ring.
- "r.prev" observed prevalence (in \
- "r.radius" ring radius (in kilometers if coordinates in decimal degrees, in the unit of the projection otherwise).
- "r.clusters" number of clusters located inside the ring.
- "r.wpos" (optional) sum of weights of positive cases inside the ring.
- "r.wn" (optional) sum of weights of valid observations inside the ring.
- "r.wprev" (optional) weighted observed prevalence (in \

Note: the list `rings` is named, the name of each element is `NN_value.RR_value`, for example `N300.RInf`.

Note 2: `r.wpos`, `r.wn` and `r.wprev` are calculated only if the slot `clusters` of object contains weighted data.

**References**

Larmarange Joseph, Vallo Roselyne, Yaro Seydou, Msellati Philippe and Meda Nicolas (2011) "Methods for mapping regional trends of HIV prevalence from Demographic and Health Surveys (DHS)", *Cybergeo : European Journal of Geography*, no 558, <https://journals.openedition.org/cybergeo/24606>, DOI: 10.4000/cybergeo.24606.

Larmarange Joseph (2007) *Prévalences du VIH en Afrique : validité d'une mesure*, PhD thesis in demography, directed by Benoît Ferry, université Paris Descartes, <https://tel.archives-ouvertes.fr/tel-00320283>.

**See Also**

[prevR](#).

**Examples**

```
## Not run:
print(fdhs)
dhs <- rings(fdhs,N=c(100,200,300,400,500))
print(dhs)

## End(Not run)
```



---

show,prevR-method      *Summary of a prevR object.*

---

### Description

Method show for objects of class [prevR](#): shows a summary of the object's characteristics.

### Usage

```
## S4 method for signature 'prevR'  
show(object)
```

### Arguments

object                  object of class [prevR](#).

### Note

Exactly the same as [print\(\)](#).

### See Also

[summary\(\)](#).

### Examples

```
fdhs  
## Not run:  
dhs <- rings(fdhs,N=c(100,300,500))  
dhs  
  
## End(Not run)
```

---

summary,prevR-method      *Detailed summary of the variables of a prevR object*

---

### Description

Method summary for objects of class [prevR](#): shows a summary of the variables of the object.

### Usage

```
## S4 method for signature 'prevR'  
summary(object, probs = c(0, 0.1, 0.25, 0.5, 0.75, 0.8, 0.9, 0.95, 0.99, 1))
```

**Arguments**

object	object of class <code>prevR</code> .
probs	vector of probabilities with values in [0,1] for computing quantiles of the rings radii (see examples).

**See Also**

`print()`.

**Examples**

```
summary(fdhs)
## Not run:
dhs <- rings(fdhs, N=c(100,300,500))
summary(dhs)
summary(dhs, c(0, 0.25, 0.5, 0.75, 1))

## End(Not run)
```

---

theme\_prevR

*prevR themes for ggplot2*

---

**Description**

Two custom themes for ggplot2 graphs, hiding axis.

**Usage**

```
theme_prevR(base_size = 12)
```

```
theme_prevR_light(base_size = 12)
```

**Arguments**

base_size	base font size
-----------	----------------

---

`TMWorldBorders`*Dataset "TM World Borders Dataset 0.3".*

---

### Description

This dataset provides boundaries of all countries in the world, in decimal degrees. Available variables are:

- "FIPS" FIPS 10-4 Country Code.
- "ISO2" ISO 3166-1 Alpha-2 Country Code.
- "ISO3" ISO 3166-1 Alpha-3 Country Code.
- "UN" ISO 3166-1 Numeric-3 Country Code.
- "NAME" Name of country/area.
- "AREA" Land area, FAO Statistics (2002).
- "POP2005" Population, World Population Prospects (2005).
- "REGION" Macro geographical (continental region), UN Statistics.
- "SUBREGION" Geographical sub-region, UN Statistics.
- "LON" Longitude.
- "LAT" Latitude.

### Format

Object of class `sp::SpatialPolygonsDataFrame`.

### Note

The boundaries, names designations used do not imply official endorsement or acceptance by the authors. Use this dataset with care, as several of the borders are disputed

### Source

Provided by Bjorn Sandvik on [http://thematicmapping.org/downloads/world\\_borders.php](http://thematicmapping.org/downloads/world_borders.php). The dataset was derived by Schuyler Erle from public domain sources. Sean Gilles did some clean up and made some enhancements. The dataset is available under a *Creative Commons Attribution-Share Alike License* (<https://creativecommons.org/licenses/by-sa/3.0/>).

### Examples

```
plot(TMWorldBorders)
```

---

xyz2dataframe	<i>Convert a surface in xyz to a data frame.</i>
---------------	--------------------------------------------------

---

### Description

Several functions (for example `GenKern::KernSur()`) return a surface as a list "xyz" composed of three elements: vector of ordinates in the x dimension, vector of ordinates in the y dimension and a matrix with the values of the surface in x and y. This function transforms a list "xyz" into a data frame.

### Usage

```
xyz2dataframe(xyz, xcol = 1, ycol = 2, zcol = 3)
```

### Arguments

xyz	a list with 3 elements: a vector with x-coordinates, a vector with y-coordinates and and matrix with value for each point of coordinates x[i], y[j].
xcol	x index.
ycol	y index.
zcol	z index.

### Value

A data.frame.

### Note

xyz could be a list like x,y,z1,z2,z3. If so, zcol should be equal to `c("z1", "z2", "z3")` or `c(3,4,5)`.

### Examples

```
x <- c(2,4,6,8,10)
y <- x
op <- GenKern::KernSur(x,y, xgridsize=50, ygridsize=50,
                      correlation=0,
                      xbandwidth=1, ybandwidth=1,
                      range.x=c(0,13), range.y=c(0,13)
)
str(op)

op.df <- xyz2dataframe(op)
str(op.df)
```

# Index

- \* **classes**
  - prevR-class, 25
- \* **class**
  - is.prevR, 15
- \* **color**
  - prevR.colors, 27
- \* **datasets**
  - fdhs, 14
  - TMWorldBorders, 35
- \* **hplot**
  - plot,prevR,missing-method, 23
- \* **manip**
  - as.data.frame.prevR, 6
  - as.prevR, 7
  - as.SpatialGrid,prevR-method, 8
  - changeproj,prevR-method, 9
  - create.boundary, 10
  - export,prevR-method, 12
  - import.dhs, 14
  - NA.outside.SpatialPolygons, 22
  - point.in.SpatialPolygons, 24
  - xyz2dataframe, 36
- \* **math**
  - rings,prevR-method, 31
- \* **package**
  - prevR-package, 3
- \* **plot**
  - quick.prevR, 29
- \* **smooth**
  - kde,prevR-method, 16
  - krige,ANY,prevR-method, 19
  - quick.prevR, 29
- \* **spatial**
  - as.SpatialGrid,prevR-method, 8
  - changeproj,prevR-method, 9
  - create.boundary, 10
  - export,prevR-method, 12
  - kde,prevR-method, 16
  - krige,ANY,prevR-method, 19
  - NA.outside.SpatialPolygons, 22
  - point.in.SpatialPolygons, 24
  - quick.prevR, 29
  - rings,prevR-method, 31
  - TMWorldBorders, 35
  - xyz2dataframe, 36
- \* **stat**
  - Noptim, 22
- as.data.frame(as.data.frame.prevR), 6
- as.data.frame(), 4, 13
- as.data.frame.prevR, 6
- as.prevR, 7
- as.prevR(), 3, 14, 15, 26
- as.SpatialGrid
  - (as.SpatialGrid,prevR-method), 8
- as.SpatialGrid(), 21
- as.SpatialGrid,prevR-method, 8
- as.SpatialGrid-methods
  - (as.SpatialGrid,prevR-method), 8
- base::as.data.frame(), 6
- base::file.path(), 15
- base::setwd(), 15
- changeproj(changeproj,prevR-method), 9
- changeproj(), 4, 8, 26
- changeproj,prevR-method, 9
- changeproj-methods
  - (changeproj,prevR-method), 9
- create.boundary, 10
- create.boundary(), 3, 8
- direct.label\_prevR, 11
- directlabels::direct.label(), 12
- export(export,prevR-method), 12
- export(), 4, 26
- export,prevR-method, 12

- export-methods (export,prevR-method), 12
- fdhs, 3, 14
- foreign::write.dbf(), 13
- GenKern::KernSur(), 18, 36
- geoR::eyefit(), 21
- graphics::legend(), 24
- graphics::title(), 24
- grDevices::col2rgb(), 28
- grDevices::rainbow(), 28
- gstat::idw(), 20, 21
- gstat::krige(), 20, 21
- gstat::vgm(), 20
- idw (krige,ANY,prevR-method), 19
- idw(), 4, 8, 26
- idw,ANY,prevR-method
  - (krige,ANY,prevR-method), 19
- idw,prevR-method
  - (krige,ANY,prevR-method), 19
- idw-methods (krige,ANY,prevR-method), 19
- import.dhs, 14
- import.dhs(), 3, 8
- is.prevR, 15
- is.prevR(), 26
- kde (kde,prevR-method), 16
- kde(), 4, 8, 26, 29, 30
- kde,prevR-method, 16
- kde-methods (kde,prevR-method), 16
- krige (krige,ANY,prevR-method), 19
- krige(), 4, 8, 21, 26, 29, 30
- krige,ANY,prevR-method, 19
- krige,prevR-method
  - (krige,ANY,prevR-method), 19
- krige-methods (krige,ANY,prevR-method), 19
- maptools::writeAsciiGrid(), 4, 18, 21
- maptools::writePointsShape(), 13
- maptools::writePolyShape, 13
- NA.outside.SpatialPolygons, 22
- NA.outside.SpatialPolygons(), 18, 21
- Noptim, 22
- Noptim(), 17, 18, 20, 21, 30
- plot (plot,prevR,missing-method), 23
- plot(), 4, 7, 26
- plot,prevR,missing-method, 23
- plot,prevR-method
  - (plot,prevR,missing-method), 23
- plot-methods
  - (plot,prevR,missing-method), 23
- point.in.SpatialPolygons, 24
- point.in.SpatialPolygons(), 22
- prevR, 3, 4, 6–10, 12–17, 19, 20, 22, 23, 28–34
- prevR-class, 25
- prevR-package, 3
- prevR.colors, 4, 18, 21, 27
- prevR.colors.blue(), 28
- prevR.colors.gray(), 28
- prevR.colors.green(), 28
- prevR.colors.qgis.pal(), 28
- prevR.colors.red(), 28, 29
- prevR.demo.pal (prevR.colors), 27
- prevR.demo.pal(), 28
- prevRsummary (summary,prevR-method), 33
- print (print,prevR-method), 28
- print(), 3, 26, 33, 34
- print,prevR-method, 28
- print-methods (print,prevR-method), 28
- quick.prevR, 29
- rgdal::spTransform(), 10
- rgdal::writeOGR(), 13
- rings (rings,prevR-method), 31
- rings(), 4, 17, 18, 21, 25, 26, 29, 30
- rings,prevR-method, 31
- rings-methods (rings,prevR-method), 31
- show (show,prevR-method), 33
- show(), 3, 29
- show,prevR-method, 33
- show-methods (show,prevR-method), 33
- sp::CRS, 8, 10, 11, 25
- sp::GridTopology(), 9
- sp::point.in.polygon(), 25
- sp::SpatialGrid, 9
- sp::SpatialPixelsDataFrame, 4, 18, 21, 22
- sp::SpatialPolygons, 7, 10, 11, 14, 16, 22, 24, 25
- sp::SpatialPolygonsDataFrame, 35
- sp::spplot(), 4, 18, 21, 27
- summary (summary,prevR-method), 33
- summary(), 3, 26, 29, 33
- summary,prevR-method, 33

summary-methods (summary, prevR-method),  
33

theme\_prevR, 34

theme\_prevR\_light (theme\_prevR), 34

TMWorldBorders, 3, 10, 11, 35

utils::write.table(), 13

xyz2dataframe, 36