Package 'prevR'

May 15, 2023

Type Package

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

Version 5.0.0

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

Acknowledgments 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

Language en-US

LazyData yes

Depends R (>= 3.5.0)

Imports directlabels, ggplot2, gstat, KernSmooth, fields, foreign, methods, sf, stars

Suggests grDevices, knitr, rmarkdown, spelling, terra

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2023-05-15 18:50:03 UTC

R topics documented:

`prevR-package` *Estimating regional trends of a prevalence from a DHS.*

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
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 adaptative bandwidths of equal number of persons surveyed (a variant of the nearest neighbor 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 visualization
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 visualizing 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.

Acknowledgment

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).

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>
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://theses.hal.science/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)
plot(
  prev.N300["k.wprev.N300.RInf"],
  pal = prevR.colors.red,
  lty = 0,
  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)
plot(
  radius.N300,
  pal = prevR.colors.blue,
  lty = 0,
  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 prevR.
...	not used, for compatibility with the generic method base::as.data.frame().
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: • "id" (optional) cluster's identifier. • "x" cluster's longitude. • "y" cluster's latitude. • "n" number of valid observations in the cluster. • "pos" number of positive cases in the cluster. • "wn" (optional) sum of observations weight. • "wpos" (optional) sum of positive cases weight. • "c.type" (optional) type of cluster (used only by `plot()`). See examples.
`boundary`	object of class sf::sf defining the studied area.
`proj`	projection of clusters coordinates used in `data` (longitude and latitude in decimal degrees by default). One of (i) character: a string accepted by GDAL, (ii) integer, a valid EPSG value (numeric), or (iii) an object of class `crs`, see `sf::st_crs()`.

Details

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

If `boundary` is not defined (`NULL`), a rectangle 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)
```

---

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. One of (i) character: a string accepted by GDAL, (ii) integer, a valid EPSG value (numeric), or (iii) an object of class crs, see sf::st_crs().

Details

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

sf::st_transform(), 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 = FALSE,
  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	projection of clusters coordinates used in data (longitude and latitude in decimal degrees by default). One of (i) character: a string accepted by GDAL, (ii) integer, a valid EPSG value (numeric), or (iii) an object of class crs, see sf::st_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 labeling (see directlabels::direct.label()).
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 prevR.
`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 sf::st_write, `foreign::write.dbf()` or `utils::write.table()`.

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
"dbf"	DBASE format
"txt"	tabulated text
"csv"	'comma separated values'
"csv2"	CSV variant using a semicolon as field separator

`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

`sf::st_write()`, `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\ surveyed, distributed in 401 clusters. 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 `https://www.dhsprogram.com/`. Could also be directly a data.frame.
`file.dbf`	GPS position of clusters in DATABASE format (.dbf), downloaded from `https://www.dhsprogram.com/`. 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` *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`.

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 sf::sf object 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 prevR.
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 cases on density surface of observed cases.

This method is a variant of the nearest neighbor technique. Surfaces are estimated using gaussian kernel estimators with adaptative bandwidths, bandwidth size being determined by a minimum number of observations in the neighborhood (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 references 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) available in object. 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 sf::sf. Surfaces are named according to the name of the corresponding 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).

Note

Results could be plotted with `sf::plot()` or with **ggplot2** using `ggplot2::geom_sf()`. See examples.

prevR provides several continuous color palettes (see prevR.colors).

Results could be turned into a **stars** raster using `stars::st_rasterize()`.

To export to ASCII grid, rasterize the results with `stars::st_rasterize()`, convert to `SpatRast` with `terra::rast()`, extract the desired layer with `[[]]` and then use `terra::writeRaster()`. See examples.

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

`KernSmooth::bkde2D()`, `rings()`, `Noptim()`.

Examples

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

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

plot(prev.N300, lty = 0)

library(ggplot2)
ggplot(prev.N300) +
  aes(fill = k.wprev.N300.RInf) +
  geom_sf(colour = "transparent") +
  scale_fill_gradientn(colors = prevR.colors.red()) +
  theme_prevR_light()
```

```
# Export k.wprev.N300.RInf surface in ASCII Grid
r <- terra::rast(stars::st_rasterize(prev.N300))
# writeRaster(r[[2]], "kprev.N300.asc")

## 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 prevR.
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 gstat::vgm().
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, only if model is not defined (NULL).
keep.variance	return variance of estimates?
show.variogram	plot the variogram?
...	additional arguments transmitted to gstat::krige() or gstat::idw().
idp	inverse distance weighting power (see gstat::idw()).

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" or "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 complex 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 a exponential variogram to the sample variogram (fit = "auto"). You can also specify directly the variogram to use with the parameter model.

Interpolations are calculated on a spatial grid obtained with make.grid.prevR().

Value

Object of class sf::sf. 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*.

Note

Results could be plotted with sf::plot() or with **ggplot2** using ggplot2::geom_sf(). See examples.

prevR provides several continuous color palettes (see prevR.colors).

Results could be turned into a **stars** raster using `stars::st_rasterize()`.

To export to ASCII grid, rasterize the results with `stars::st_rasterize()`, convert to `SpatRast` with `terra::rast()`, extract the desired layer with `[[]]` and then use `terra::writeRaster()`. See examples.

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 = 50)
  prev.krige <- krige(r.wprev ~ 1, dhs, N = c(100, 300, 500))

  plot(prev.krige, lty = 0)

  library(ggplot2)
  ggplot(prev.krige) +
    aes(fill = r.wprev.N300.RInf) +
    geom_sf(colour = "transparent") +
    scale_fill_gradientn(colors = prevR.colors.red()) +
    theme_prevR_light()

  # Export r.wprev.N300.RInf surface in ASCII Grid
  r <- terra::rast(stars::st_rasterize(prev.krige))
  # writeRaster(r[[2]], "wprev.N300.asc")

## End(Not run)
```

---

make.grid.prevR *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

```
make.grid.prevR(object, nb.cells = 100, cell.size = NULL)
```

Arguments

`object`	object of class prevR.
`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 corresponding to the center of 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 sf::sfc (simple feature geometry list column).

See Also

`sf::st_make_grid()`

Examples

```
make.grid.prevR(fdhs)
make.grid.prevR(fdhs, nb.cells = 200)
```

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 = FALSE,
  ...
)
```

Arguments

`x`	object of class prevR.
`type`	graph to plot: • "position" clusters position. • "c.type" clusters per `c.type`. • "count" number of observations per cluster. • "flower" number of positive cases per cluster.
`add.legend`	add a legend?
`legend.location`	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 graphics::title().

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")
```

---

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 %) in the cluster (pos/n).
- "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 %) in the cluster (wpos/wn).
- "c.type" (optional) cluster type.

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

`proj` object of class sf::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 %) inside the ring (r.wpos/r.wn).

Note: the list `rings` is named, the name of each element is N*N_value*.R*R_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 usable with R graphical functions. These palettes are continuous, contrast being accentuated by darkening and lightening extreme 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 = 10)

prevR.colors.blue.inverse(n = 10)

prevR.colors.gray(n = 10)

prevR.colors.gray.inverse(n = 10)
```

```
prevR.colors.green(n = 10)

prevR.colors.green.inverse(n = 10)

prevR.colors.red(n = 10)

prevR.colors.red.inverse(n = 10)

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 prevR.
`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 `cell.size` 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 *Calculation of rings of equal number of observation and/or equal radius.*

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 prevR.
`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 row of the data frame `clusters` of `object`, `rings()` determines a ring, centered on the cluster. It could be:

- rings of equal 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 both 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 first 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 simultaneously 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 %) inside the ring (r.pos/r.n).
- "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 %) inside the ring (r.wpos/r.wn).

Note: the list `rings` is named, the name of each element is N*N_value*.R*R_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://theses.hal.science/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)
```

st_filter_prevR *Spatial filter*

Description

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

Usage

```
st_filter_prevR(x, y)
```

Arguments

x	object of class sf::sf
y	object of class sf::sf

Details

The function try to apply `sf::st_filter()`. In case it fails, it will try to rebuild y according to spherical geometry (see `sf::st_as_s2()`) before filtering. If it still fail, it will return x unfiltered.

Value

Return x filtered by y

See Also

`sf::st_filter()`.

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 prevR.
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 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["NAME"])
```

---

update_prevR *Update a prevR object*

Description

Update an object of class prevR created with a previous version of the package to the last version. In particular, it will convert any boundary slot defined with the sp package to sf class.

Usage

```
update_prevR(object)
```

Arguments

object	a prevR object

Value

a prevR object

xyz2dataframe *Convert a surface in xyz to a data frame.*

Description

Several functions (for example `KernSmooth::bkde2D()`) 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 <- matrix(c(2, 4, 6, 8, 10, 2, 4, 6, 8, 10), ncol = 2)
op <- KernSmooth::bkde2D(x, bandwidth = 1)
str(op)

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

Index