Package ‘sdcSpatial’

October 14, 2022

Title  Statistical Disclosure Control for Spatial Data

Version  0.5.2


License  GPL-2

Encoding  UTF-8

LazyData  true

URL  https://github.com/edwindj/sdcSpatial

BugReports  https://github.com/edwindj/sdcSpatial/issues

RoxygenNote  7.1.2

Suggests  testthat, knitr, rmarkdown, sp, sf, FNN

Imports  raster, methods

Depends  R (>= 3.5.0)

VignetteBuilder  knitr

NeedsCompilation no

Author  Edwin de Jonge [aut, cre] (<https://orcid.org/0000-0002-6580-4718>), Peter-Paul de Wolf [aut], Douwe Hut [ctb], Sapphire Han [ctb]

Maintainer  Edwin de Jonge <edwindjonge@gmail.com>

Repository  CRAN

Date/Publication  2022-03-24 13:30:02 UTC
sdcSpatial-package

R topics documented:

- sdcSpatial-package .................................................. 2
- disclosure_risk ...................................................... 3
- dwellings ............................................................. 4
- enterprises ........................................................... 5
- is_sensitive ............................................................ 6
- is_sensitive_at ........................................................ 8
- mask_grid ............................................................ 9
- mask_random ......................................................... 10
- mask_voronoi ....................................................... 11
- mask_weighted_random ............................................. 12
- plot.sdc_raster ....................................................... 13
- plot_sensitive ....................................................... 14
- protect_neighborhood .............................................. 15
- protect_quadtree ................................................... 16
- protect_smooth ..................................................... 18
- remove_sensitive ................................................... 20
- sdc_raster .......................................................... 21
- sensitivity_score ................................................... 23
- smooth_raster ....................................................... 23

Index 25

sdcSpatial-package  Privacy Protected maps

Description
sdcSpatial contains functions to create spatial distribution maps, assess the risk of disclosure on a location and to suppress or adjust revealing values at certain locations.

Details
sdcSpatial working horse is the sdc_raster() object upon which the following methods can be applied:

Sensitivity assessment
- plot.sdc_raster(), plot_sensitive()
- print
- is_sensitive()

Protection methods
- remove_sensitive()
- protect_smooth()
- protect_quadtree()
The disclosure risk function is used by `is_sensitive()` to determine the risk of a raster cell. It returns a score between 0 and 1 for cells that have a finite value (otherwise NA).

### Usage

```r
 disclosure_risk(x, risk_type = x$risk_type)
```
Arguments

x  
    sdc_raster object.

risk_type  
    character: "external", "internal", "discrete".

Details

Different risk functions include:

- external (numeric variable), calculates how much the largest value comprises the total sum within a cell
- internal (numeric variable), calculates how much the largest value comprises the sum without the second largest value
- discrete (logical variable), calculates the fraction of TRUE vs FALSE

Value

raster::raster object with the disclosure risk.

See Also

Other sensitive: is_sensitive_at(), is_sensitive(), plot_sensitive(), remove_sensitive(), sdc_raster(), sensitivity_score()
**enterprises**  

**Source**  

Basisregistratie Adressen en Gebouwen https://www.kadaster.nl/zakelijk/registraties/basisregistraties/bag/bag-producten

**Examples**

```r
# dwellings is a data.frame, the best way is to first turn it  
# into a sf or sp object.

# create an sf object from our data
if (requireNamespace("sf")){
  dwellings_sf <- sf::st_as_sf(dwellings, coords=c("x", "y"), crs=28992)

  unemployed <- sdc_raster( dwellings_sf
    , "unemployed"
    , r=200
    , max_risk = 0.9
  )

  plot(unemployed)
  sensitivity_score(unemployed)

  unemployed_smoothed <- protect_smooth(unemployed, bw = 0.4e3)
  plot(unemployed_smoothed, main="Employment rate")
  plot(unemployed_smoothed, "sum", main = "Employment")
} else {
  message("Package 'sf' was not installed."
}

dwellings_sp <- dwellings  
# or change a data.frame into a sp object
sp::coordinates(dwellings_sp) <- ~ x + y
tryCatch(
  # not working on some OS versions.
  sp::proj4string(dwellings_sp) <- +init=epsg:28992"
)
consumption <- sdc_raster(dwellings_sp, dwellings_sp$consumption, r = 500)
consumption

plot(consumption)

# but we can also create a raster directly from a data.frame
unemployed <- sdc_raster( dwellings[c("x","y")], dwellings$unemployed)
```

**enterprises**  

*Simulated data set with enterprise locations.*
Description

Enterprises is generated from the Dutch open data BAG register. The locations are realistic, but the associated data is simulated.

Usage

enterprises

Format

An object of class SpatialPointsDataFrame with 8348 rows and 2 columns.

production numeric simulated production (lognormal).

fined logical simulated variable if an enterprise is fined or not.

Source

Basisregistratie Adressen en Gebouwen: https://www.kadaster.nl/zakelijk/registraties/basisregistraties/bag/bag-producten

Examples

library(sdcSpatial)
library(raster)
data("enterprises")

production <- sdc_raster(enterprises, "production", min_count = 10)
print(production)

# show the average production per cell
plot(production, "mean")
production$min_count <- 2 # adjust norm for sdc
plot(production)

production_safe <- remove_sensitive(production)
plot(production_safe)

is_sensitive Return raster with sensitive locations.

Description

Create a binary raster with sensitive locations.
### Usage

```r
is_sensitive(
  x,
  max_risk = x$max_risk,
  min_count = x$min_count,
  risk_type = x$risk_type
)
```

### Arguments

- **x**  
  *sdc_raster* object.

- **max_risk**  
  A risk value higher than `max_risk` will be sensitive.

- **min_count**  
  A count lower than `min_count` will be sensitive.

- **risk_type**  
  What kind of measure should be used (see details).

### Details

By default the risk settings are taken from `x`, but they can be overridden.

Different risk functions can be used:

- **external** (numeric variable), calculates how much the largest value comprises the total sum
- **internal** (numeric variable), calculates how much the largest value comprises the sum without the second largest value
- **discrete** (logical variable), calculates the fraction of sensitive values.

### See Also

Other sensitive: `disclosure_risk()`, `is_sensitive_at()`, `plot_sensitive()`, `remove_sensitive()`, `sdc_raster()`, `sensitivity_score()`

### Examples

```r
dwellings_sp <- dwellings
sp::coordinates(dwellings_sp) <- ~ x + y
tryCatch(
  # does not work on some OS versions
  sp::proj4string(dwellings_sp) <- "+init=epsg:28992"
)
# create a 1km grid
unemployed <- sdc_raster(dwellings_sp, dwellings_sp$unemployed, r = 1e3)
print(unemployed)

# retrieve the sensitive cells
is_sensitive(unemployed)
```
is_sensitive_at

Calculate sensitivity from a sdc_raster at x,y locations.

Description

Calculate sensitivity from a sdc_raster at x,y locations. A typical use is to calculate the sensitivity for each of the locations x was created with (see example).

Usage

is_sensitive_at(x, xy, ...)

Arguments

x sdc_raster()
xy matrix of x and y coordinates, or a SpatialPoints or SpatialPointsDataFrame object
... Arguments passed on to is_sensitive

max_risk a risk value higher than max_risk will be sensitive.
min_count a count lower than min_count will be sensitive.
risk_type what kind of measure should be used (see details).

Value

logical vector with

See Also

Other sensitive: disclosure_risk(), is_sensitive(), plot_sensitive(), remove_sensitive(), sdc_raster(), sensitivity_score()

Examples

production <- sdc_raster(enterprises, "production")

# add the sensitive variable to original data set.
enterprises$sensitive <- is_sensitive_at(production, enterprises)
mask_grid

Mask coordinates using a grid

Description

Perturbates coordinates by rounding coordinates to grid coordinates

Usage

mask_grid(x, r, plot = FALSE)

Arguments

x coordinates
r grid resolution
plot if TRUE the points (black) and the pertubation (red) will be plotted

See Also

Other point perturbation: mask_random(), mask_voronoi(), mask_weighted_random()

Examples

x <- cbind(
  x = c(2.5, 3.5, 7.2, 1.5),
  y = c(6.2, 3.8, 4.4, 2.1)
)

# plotting is only useful from small datasets!

# grid masking
x_g <- mask_grid(x, r=1, plot=TRUE)

# random perturbation
set.seed(3)
x_r <- mask_random(x, r=1, plot=TRUE)

if (requireNamespace("FNN", quietly = TRUE)){
  # weighted random perturbation
  x_wr <- mask_weighted_random(x, k = 2, r = 4, plot=TRUE)
  }

if ( requireNamespace("FNN", quietly = TRUE)
  && requireNamespace("sf", quietly = TRUE) ){
  # voronoi masking, plotting needs package `sf`
  x_vor <- mask_voronoi(x, r = 1, plot=TRUE)
}
mask_random  

Mask coordinates using random pertubation

Description

Pertubates points with a uniform pertubation in a circle. Note that \( r \) can either be one distance, of a distance per data point.

Usage

\[
\text{mask_random}(x, r, \text{plot} = \text{FALSE})
\]

Arguments

- \( x \) : coordinates, matrix or data.frame (first two columns)
- \( r \) : numeric maximum pertubation distance (vectorized)
- \( \text{plot} \) : if TRUE points will be plotted.

Value

adapted \( x \) with perturbed coordinates

See Also

Other point pertubation: \texttt{mask_grid()}, \texttt{mask_voronoi()}, \texttt{mask_weighted_random()}

Examples

\[
x <- \text{cbind}(
  x = c(2.5, 3.5, 7.2, 1.5),
  y = c(6.2, 3.8, 4.4, 2.1)
)
\]

# plotting is only useful from small datasets!

# grid masking
\[
x_g <- \text{mask_grid}(x, r=1, \text{plot}=\text{TRUE})
\]

# random pertubation
\[
\text{set.seed}(3)\hspace{1cm} x_r <- \text{mask_random}(x, r=1, \text{plot}=\text{TRUE})
\]

if (requireNamespace("FNN", quietly = \text{TRUE})){
  # weighted random pertubation
  \[
x_{wr} <- \text{mask_weighted_random}(x, k = 2, r = 4, \text{plot}=\text{TRUE})
  \]
}

if (requireNamespace("FNN", quietly = \text{TRUE})
  \&\& requireNamespace("sf", quietly = \text{TRUE})
mask_voronoi

mask_voronoi

Description

Perturbs points by using voronoi masking. Each point is moved at its nearest voronoi boundary.

Usage

mask_voronoi(x, r = 0, k = 10, plot = FALSE)

Arguments

x coordinates
r tolerance, nearest voronoi should be at least r away.
k number of neighbors to consider when determining nearest neighbors
plot if TRUE plots the voronoi tessellation, points (black), and perturbated points (red), needs package sf.

Value

adapted x with perturbed coordinates

See Also

Other point perturbation: mask_grid(), mask_random(), mask_weighted_random()

Examples

x <- cbind(
  x = c(2.5, 3.5, 7.2, 1.5),
  y = c(6.2, 3.8, 4.4, 2.1)
)

# plotting is only useful from small datasets!

# grid masking
x_g <- mask_grid(x, r=1, plot=TRUE)

# random perturbation
set.seed(3)
x_r <- mask_random(x, r=1, plot=TRUE)
mask_weighted_random

Mask coordinates using weighted random perturbation

Description

This method uses per point the distance to the kth neighbor as the maximum pertubation distance. Parameter r can be used to restrict the maximum distance of the kth neighbor.

Usage

mask_weighted_random(x, k = 5, r = NULL, plot = FALSE)

Arguments

x coordinates, matrix or data.frame (first two columns)
k integer number of neighbors to be used as the maximum distance
r numeric maximum pertubation distance (vectorized)
plot if TRUE points will be plotted.

Value

adapted x with perturbed coordinates

References

Spatial obfuscation methods for privacy protection of household-level data

See Also

Other point pertubation: mask_grid(), mask_random(), mask_voronoi()
Examples

```r
x <- cbind(
  x = c(2.5, 3.5, 7.2, 1.5),
  y = c(6.2, 3.8, 4.4, 2.1)
)

# plotting is only useful from small datasets!

# grid masking
x_g <- mask_grid(x, r=1, plot=TRUE)

# random pertubation
set.seed(3)
x_r <- mask_random(x, r=1, plot=TRUE)

if (requireNamespace("FNN", quietly = TRUE)){
  # weighted random pertubation
  x_wr <- mask_weighted_random(x, k = 2, r = 4, plot=TRUE)
}

if (requireNamespace("FNN", quietly = TRUE) && requireNamespace("sf", quietly = TRUE)){
  # voronoi masking, plotting needs package `sf`
  x_vor <- mask_voronoi(x, r = 1, plot=TRUE)
}
```

---

**plot.sdc_raster**  
*Plot a sdc_raster object*

**Description**

Plot a sdc_raster object together with its sensitivity.

**Usage**

```r
## S3 method for class 'sdc_raster'
plot(
  x,
  value = "mean",
  sensitive = TRUE,
  ...
  main = paste(substitute(x)),
  col
)
```
plot_sensitive

Arguments

- **x**: `sdc_raster` object to be plotted
- **value**: character which value layer to be used for plotting, e.g. "sum", "count", "mean" (default).
- **sensitive**: logical show the sensitivity in the plot?
- **main**: title of plot
- **col**: color palette to be used, passed on to `raster::plot()`.

Details

When sensitive is set to TRUE, a side-by-side plot will be made of the value and its sensitivity.

See Also

Other plotting: `plot_sensitive()`

---

Plot the sensitive cells of the sdc_raster.

Description

Plots the sensitive cells of the sdc_raster. The sensitive cells are plotted in red. The sensitive cells are determined using `is_sensitive`.

Usage

`plot_sensitive(x, value = "mean", main = "sensitive", col, ...)`

Arguments

- **x**: `sdc_raster` object
- **value**: character which value layer to be used for values, e.g. "sum", "count", "mean" (default).
- **main**: character title of map.
- **col**: color palette to be used, passed on to `raster::plot()`.
- **...**: passed on to `plot.sdc_raster`.

See Also

Other plotting: `plot.sdc_raster()`
Other sensitive: `disclosure_risk()`, `is_sensitive_at()`, `is_sensitive()`, `remove_sensitive()`, `sdc_raster()`, `sensitivity_score()`
**Description**

protects raster by summing over the neighborhood

**Usage**

```
protect_neighborhood(x, radius = 10 * raster::res(x$value)[1], ...)
```

**Arguments**

- `x` : sdc_raster() object to be protected
- `radius` : of the neighborhood to take
- `...` : not used at the moment

**Value**

sdc_raster object

**Examples**

```r
data(enterprises)

# create a sdc_raster from point data with raster with
# a resolution of 200m
production <- sdc_raster(enterprises, variable = "production",
                     , r = 200, min_count = 3)

print(production)

# plot the raster
zlim <- c(0, 3e4)
# show which raster cells are sensitive
plot(production, zlim=zlim)

# let's smooth to reduce the sensitivity
smoothed <- protect_smooth(production, bw = 400)
plot(smoothed)

neighborhood <- protect_neighborhood(production, radius=1000)
plot(neighborhood)

# what is the sensitivity fraction?
sensitivity_score(neighborhood)
```
Protect a raster with a quadtree method.

**Description**

`protect_quadtree` reduces sensitivity by aggregating sensitive cells with its three neighbors, and does this recursively until no sensitive cells are left or when the maximum zoom levels has been reached.

**Usage**

```r
protect_quadtree(x, max_zoom = Inf, ...)
```

**Arguments**

- `x` 
  - `sdc_raster` object to be protected.
- `max_zoom` 
  - numeric, restricts the number of zoom steps and thereby the max resolution for the blocks. Each step will zoom with a factor of 2 in x and y so the max resolution = resolution * 2^max_zoom.
- `...` 
  - Arguments passed on to `is_sensitive`
  - `max_risk` a risk value higher than `max_risk` will be sensitive.
  - `min_count` a count lower than `min_count` will be sensitive.
  - `risk_type` what kind of measure should be used (see details).

**Details**

This implementation generalizes the method as described by Suñé et al., in which there is no risk function, and only a `min_count` to determine sensitivity. Furthermore the method the article only handles count data (`x$value$count`), not mean or summed values. Currently the translation feature of the article is not (yet) implemented, for the original method does not take the `disclosure_risk` into account.

**Value**

- a `sdc_raster` object, in which sensitive cells have been recursively aggregated until not sensitive or when `max_zoom` has been reached.

**References**


**See Also**

Other protection methods: `protect_smooth()`, `remove_sensitive()`
Examples

```r
# library(raster)
#
# fined <- sdc_raster(enterprises, enterprises$fined)
# plot(fined)
# fined_qt <- protect_quadtree(fined)
# plot(fined_qt)
#
# fined <- sdc_raster(enterprises, enterprises$fined, r=50)
# plot(fined)
# fined_qt <- protect_quadtree(fined)
# plot(fined_qt)
#
# library(sf)
# st_crs(gemeente_2019) <- 28992
# nbl <- stTouches(gemeente_2019)
#
# coords <- st_coordinates(st_centroid(gemeente_2019))
# l <- lapply(seq_along(nbl), function(i){
# nb <- nbl[[i]]
# st_sfc(lapply(nb, function(j){
# st_linestring(coords[c(i,j),])
# }))
# })
# l2 <- do.call(c, l)
#
# edge_list <- as.data.frame(nbl)
# library(data.table)
# el <- as.data.table(edge_list)
# names(el) <- c("from", "to")
#
# edge_list$from <- gemeente_2019$id[edge_list$row.id]
# edge_list$to <- gemeente_2019$id[edge_list$col.id]
# edge_list <- subset(edge_list, row.id < col.id)
# edge_list <- edge_list[,c("from", "to")]
#
# g <- igraph::graph_from_data_frame(edge_list, directed = FALSE)
# plot(g)
# library(igraph)
# i <- match(names(V(g)), gemeente_2019$id)
#
# c2 <- igraph::layout_with_fr(g, coords[i,])
# plot(g, layout = c2)
#
# st_crs(buurt_2019) <- 28992
# system.time({
# nbl <- stTouches(buurt_2019)
# })
```

```r
# coords <- st_coordinates(st_centroid(buurt_2019))
# l <- lapply(seq_along(nbl), function(i){
# nb <- nbl[[i]]
# st_sfc(lapply(nb, function(j){
# st_linestring(coords[c(i,j),]))
# })
# })
# l2 <- do.call(c, l)
# plot(l2)
```

---

**protect_smooth**  
*Protect a sdc_raster by smoothing*

**Description**

protect_smooth reduces the sensitivity by applying a Gaussian smoother, making the values less localized.

**Usage**

```r
protect_smooth(x, bw = raster::res(x$value), ...)
```

**Arguments**

- **x**: raster object
- **bw**: bandwidth
- **...**: passed through to `focal`

**Details**

The sensitivity of a raster can be decreased by applying a kernel density smoother as argued by de Jonge et al. (2016) and de Wolf et al. (2018). Smoothing spatially spreads localized values, reducing the risk for location disclosure. Note that smoothing often visually enhances detection of spatial patterns. The kernel applied is a Gaussian kernel with a bandwidth `bw` supplied by the user. The smoother acts upon the `x$value$mean` from which a new `x$value$mean` is derived.

**References**


See Also

Other protection methods: `protect_quadtree()`, `remove_sensitive()`

Examples

```r
library(sdcSpatial)
library(raster)

data(enterprises)

# create a sdc_raster from point data with raster with
# a resolution of 200m
production <- sdc_raster(enterprises, variable = "production",
                         r = 200, min_count = 3)

print(production)

# plot the raster
zlim <- c(0, 3e4)
# show which raster cells are sensitive
plot(production, zlim=zlim)

# but we can also retrieve directly the raster
sensitive <- is_sensitive(production, min_count = 3)
plot(sensitive, col = c("white", "red"))

# what is the sensitivity fraction?
sensitivity_score(production)
# or equally
cellStats(sensitive, mean)

# let's smooth to reduce the sensitivity
smoothed <- protect_smooth(production, bw = 400)
plot(smoothed)

# let's smooth to reduce the sensitivity, with higher resolution
smoothed <- protect_smooth(production, bw = 400, smooth_fact=4, keep_resolution=FALSE)
plot(smoothed)

# what is the sensitivity fraction?
sensitivity_score(smoothed)

# let's remove the sensitive data.
smoothed_safe <- remove_sensitive(smoothed, min_count = 3)
plot(smoothed_safe)

# let's communicate!
production_mean <- mean(smoothed_safe)
production_total <- sum(smoothed_safe)

# and create a contour plot
```
remove_sensitive

remove_sensitive removes sensitive cells from a sdc_raster. The sensitive cells, as found by is_sensitive() are set to NA.

Usage

remove_sensitive(x, max_risk = x$max_risk, min_count = x$min_count, ...)

mask_sensitive(x, max_risk = x$max_risk, min_count = x$min_count, ...)

Arguments

x sdc_raster object.
max_risk a risk value higher than max_risk will be sensitive.
min_count a count lower than min_count will be sensitive.
... passed on to is_sensitive.

Details

Removing sensitive cells is a protection method, which often is useful to finalize map protection after other protection methods have been applied. mask_sensitive and remove_sensitive are synonyms, to accommodate both experienced raster users as well as sdc users.

Value

sdc_raster object with sensitive cells set to NA.

See Also

Other sensitive: disclosure_risk(), is_sensitive_at(), is_sensitive(), plot_sensitive(), sdc_raster(), sensitivity_score()
Other protection methods: protect_quadtree(), protect_smooth()
Examples

```r
library(raster)

unemployed <- sdc_raster(dwellings[1:2], dwellings$unemployed, r=200)

# plot the normally rastered data
plot(unemployed, zlim=c(0,1))
plot_sensitive(unemployed)

unemployed_safe <- remove_sensitive(unemployed, risk_type="discrete")
plot_sensitive(unemployed_safe, zlim=c(0,1))
print(unemployed)
unemployed$value
```

### sdc_raster

Create a raster map with privacy awareness

#### Description

`sdc_raster` creates multiple `raster::raster` objects ("count", "mean", "sum") from supplied point data `x` and calculates the sensitivity to privacy disclosure for each raster location.

#### Usage

```r
sdc_raster(
  x, 
  variable, 
  r = 200, 
  max_risk = 0.95, 
  min_count = 10, 
  risk_type = c("external", "internal", "discrete"), 
  ...,
  field = variable
)
```

#### Arguments

- **x** `sp::SpatialPointsDataFrame, sf::sf` or a two column matrix or `data.frame` that is used to create a raster map.
- **variable** name of data column or numeric with same length as `x` to be used for the data in the raster map.
- **r** either a desired resolution or a pre-existing `raster` object. In the first case, the `crs` of `x` (if present) will be used, in the latter the properties of the `r` will be kept.
- **max_risk** numeric, the maximum risk score (`disclosure_risk`) before a cell in the map is considered sensitive.
min_count numeric, a raster cell with less than \texttt{min\_count} observations is considered sensitive.

\texttt{risk\_type} passed on to \texttt{disclosure\_risk()}. 

... passed through to \texttt{raster::rasterize()}

\texttt{field} synonym for variable. If both supplied, \texttt{field} has precedence.

**Details**

A \texttt{sdc\_raster} object is the vehicle that does the book keeping for calculating sensitivity. Protection methods work upon a \texttt{sdc\_raster} and return a new \texttt{sdc\_raster} in which the sensitivity is reduced. The sensitivity of the map can be assessed with \texttt{sensitivity\_score}, \texttt{plot.sdc\_raster()}, \texttt{plot\_sensitive()} or \texttt{print}. Reducing the sensitivity can be done with \texttt{protect\_smooth()}, \texttt{protect\_quadtree()} and \texttt{remove\_sensitive()}. Raster maps for mean, sum and count data can be extracted from the \texttt{$value} (\texttt{brick()}).

**Value**

object of class "\texttt{sdc\_raster}":

- \texttt{$value}: \texttt{raster::brick()} object with different layers e.g. count, sum, mean, scale.
- \texttt{$max\_risk}: see above.
- \texttt{$min\_count}: see above. of protection operation \texttt{protect\_smooth()} or \texttt{protect\_quadtree()}
- \texttt{$type}: data type of variable, either numeric or logical
- \texttt{$risk\_type}, "external", "internal" or "discrete" (see \texttt{disclosure\_risk()})

**See Also**

Other sensitive: \texttt{disclosure\_risk()}, \texttt{is\_sensitive\_at()}, \texttt{is\_sensitive()}, \texttt{plot\_sensitive()}, \texttt{remove\_sensitive()}, \texttt{sensitivity\_score()}

**Examples**

```r
library(raster)
prod <- sdc_raster(enterprises, field = "production", r = 500)
print(prod)

prod <- sdc_raster(enterprises, field = "production", r = 1e3)
print(prod)

# get raster with the average production per cell averaged over the enterprises
prod_mean <- mean(prod)
summary(prod_mean)

# get raster with the total production per cell
prod_total <- sum(prod)
summary(prod_total)
```
sensitivity_score

| sensitivity_score | Mean sensitivity for raster |

**Description**

`sensitivity_score` calculates the fraction of cells (with a value) that are considered sensitive according to the used `disclosure_risk`.

**Usage**

`sensitivity_score(x, max_risk = x$max_risk, min_count = x$min_count, ...)

**Arguments**

- **x**: `sdc_raster` object.
- **max_risk**: a risk value higher than `max_risk` will be sensitive.
- **min_count**: a count lower than `min_count` will be sensitive.
- **...**: passed on to `is_sensitive`

**See Also**

Other sensitive: `disclosure_risk()`, `is_sensitive_at()`, `is_sensitive()`, `plot_sensitive()`, `remove_sensitive()`, `sdc_raster()`

**Examples**

```r
consumption <- sdc_raster(dwellings[1:2], variable = dwellings$consumption, r = 500)
sensitivity_score(consumption)
# same as
print(consumption)

# change the rules! A higher norm generates more sensitive cells
sensitivity_score(consumption, min_count = 20)
```

smooth_raster

| smooth_raster | Create kde density version of a raster |

**Description**

Create kde density version of a raster
Usage

smooth_raster(
  x,
  bw = raster::res(x),
  smooth_fact = 5,
  keep_resolution = TRUE,
  na.rm = TRUE,
  pad = TRUE,
  padValue = NA,
  threshold = NULL,
  type = c("Gauss", "circle", "rectangle"),
  ...)

Arguments

x raster object
bw bandwidth
smooth_fact integer, disaggregate factor to have a better smoothing
keep_resolution integer, should the returned map have same resolution as x or keep the disaggregated raster resulting from smooth_fact?
na.rm should the NA value be removed from the raster?
pad should the data be padded?
padValue what should the padding value be?
threshold cells with a lower (weighted) value of this threshold will be removed.
type what is the type of smoothing (see raster::focal())
... passed through to focal.
Index

* datasets
  dwellings, 4
  enterprises, 5

* plotting
  plot.sdc_raster, 13
  plot_sensitive, 14

* point pertubation
  mask_grid, 9
  mask_random, 10
  mask_voronoi, 11
  mask_weighted_random, 12

* protection methods
  protect_quadtree, 16
  protect_smooth, 18
  remove_sensitive, 20

* sensitive
  disclosure_risk, 3
  is_sensitive, 6
  is_sensitive_at, 8
  plot_sensitive, 14
  remove_sensitive, 20
  sdc_raster, 21
  sensitivity_score, 23

brick(), 22
character, 14
data.frame, 21
disclosure_risk, 3, 7, 8, 14, 20–23
disclosure_risk(), 22
dwellings, 4

enterprises, 5

focal, 18, 24

is_sensitive, 4, 6, 8, 14, 16, 20, 22, 23
is_sensitive(), 2, 3, 20
is_sensitive_at, 4, 7, 8, 14, 20, 22, 23

mask_grid, 9, 10–12
mask_random, 9, 10, 11, 12
mask_sensitive(remove_sensitive), 20
mask_voronoi, 9, 10, 11, 12
mask_weighted_random, 9–11, 12
plot.sdc_raster, 13, 14
plot.sdc_raster(), 2, 22
plot_sensitive, 4, 7, 8, 14, 14, 20, 22, 23
plot_sensitive(), 2, 22
protect_neighborhood, 15
protect_quadtree, 16, 19, 20
protect_quadtree(), 2, 22
protect_smooth, 16, 18, 20
protect_smooth(), 2, 22
raster, 21
raster::brick(), 22
raster::plot(), 14
raster::raster, 4, 21
raster::rasterize(), 22
remove_sensitive, 4, 7, 8, 14, 16, 19, 20, 22, 23
remove_sensitive(), 2, 22
sdc_raster, 4, 7, 8, 14, 16, 20, 21, 23
sdc_raster(), 2, 8
sdcSpatial (sdcspatial-package), 2
sdcSpatial-package, 2
sensitivity_score, 4, 7, 8, 14, 20, 22, 23
sf::sf, 21
smooth_raster, 23
sp::SpatialPointsDataFrame, 21