Package 'MRG'

July 11, 2025

Type Package

Title Create Non-Confidential Multi-Resolution Grids

Version 0.3.10

Maintainer Jon Olav Skoien <jon.skoien@gmail.com>

Imports magrittr, parallel, terra, sf, stars, plyr, dplyr, rlang, sjmisc, vardpoor, purrr, tidyr, tidyselect, methods, ggplot2, viridis, grDevices, utils

Suggests patchwork, knitr, rmarkdown, giscoR, bookdown, units, ggforce, kableExtra

Description The need for anonymization of individual survey responses often leads to many suppressed grid cells in a regular grid. Here we provide functionality for creating multi-resolution gridded data, respecting the confidentiality rules, such as a minimum number of units and dominance by one or more units for each grid cell. The functions also include the possibility for contextual suppression of data. For more details see Skoien et al. (2025) <doi:10.48550/arXiv.2410.17601>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Depends R (>= 3.5.0)

LazyData true

Author Jon Olav Skoien [aut, cre], Nicolas Lampach [aut]

Repository CRAN

Date/Publication 2025-07-11 13:00:09 UTC

Contents

`createMRGobject` *Create a single object containing all necessary objects for multiResGrid functions*

Description

Create a single object containing all necessary objects for multiResGrid functions

Prints MRG-objects

Usage

```
createMRGobject(
  ifg,
  ress = c(1, 5, 10, 20, 40) * 1000,
  geovar = c("GEO_LCT", "geometry"),
  lnames = NULL,
  vars = NULL,
  weights = NULL,
  mincount = 10,
  countFeatureOrTotal = "feature",
  nlarge = 2,
  plim = 0.85,
  verbose = FALSE,
  nclus = 1,
  clusType = NULL,
  domEstat = TRUE,
  consistencyCheck = FALSE,
  outfile = NULL,
  splitlim = 5e+07,
  checkDominance = TRUE,
  checkReliability = FALSE,
  userfun = NULL,
  strat = NULL,
```

```
  confrules = "individual",
  suppresslim = 0,
  sumsmall = FALSE,
  suppresslimSum = 0,
  reliabilitySplit = TRUE,
  pseudoreg = NULL,
  plotIntermediate = FALSE,
  addIntermediate = FALSE,
  locAdj = "LL",
  postProcess = TRUE,
  rounding = -1,
  remCols = TRUE,
  ...
)

## S3 method for class 'MRG'
print(x, ...)
```

Arguments

ifg	Either a data.frame or tibble or sf-object with the locations and the data of the survey or census data, or a list of such objects.
ress	A vector with the different resolutions
geovar	Name of geodata variable in the objects. Must me the same for all of the surveys/censuses, if the data sets are not submitted as sf-objects
lnames	Names for the different surveys or censuses if ifg is a list. Typically it could be survey years
vars	Variable(s) of interest that should be aggregated (necessary when ifg is used for individual farm specific anonymization rules)
weights	Extrapolation factor(s) (weights) wi of unit i in the sample of units nc falling into a specific cell c. Weights are used for disclosure control measures. A weight of 1 will be used if missing. If only one weight is given, it will be used for all variables. If the length is more than one, the length has to be equal to the number of variables. If the same weight is used for several variables, it must be repeated in the weights-vector
mincount	The minimum number of farms for a grid cell (threshold rule)
countFeatureOrTotal	Should the frequency limit be applied on records with a positive value for a certain feature, or on all records, independent of value of feature
nlarge	Parameter to be used if the nlarge(st) farms should count for maximum plim percent of the total value for the variable in the grid cell (see details of gridData)
plim	See nlarge
verbose	Indicates if some extra output should be printed. Usually TRUE/FALSE, but can also have a value of 2 for multiResGrid for even more output.
nclus	Number of clusters to use for parallel processing. No parallelization is used for nclus = 1.

clusType — The type of cluster; see makeCluster for more details. The default of makeCluster is used if type is missing or NA

domEstat — Should the dominance rule be applied as in the IFS handbook (TRUE), where the weights are rounded before finding the first nlarge contributors, or should it be the first nlarge contributors*weight, where also fractions are considered (FALSE)?

consistencyCheck — logical; whether consistency between the gridded values and the similar values from ifg should be checked. The gridded value is derived from rasterize and the second one from st_join. The two methods can in some cases treat border cases between grid cells differently.

outfile — File to direct the output in case of parallel processing, see makeCluster for more details.

splitlim — For large dataset - split the data set in batches of more or less splitlim size

checkDominance — Logical - should the dominance rule be applied?

checkReliability — Logical - should the prediction variance be checked, and used for the aggregation? This considerably increases computation time

userfun — This gives the possibility to add a user defined function with additional confidentiality rules which the grid cell has to pass

strat — Column name defining the strata for stratified sampling, used if checkReliability is TRUE

confrules — Should the frequency rule (number of holdings) refer to the number of holdings with a value of the individual vars above zero ("individual") or the total number of holdings in the data set ("total")?

suppresslim — Parameter that can be used to avoid that almost empty grid cells are merged with cells with considerably higher number of observations. The value is a minimum share of the total potential new cell for a grid cell to be aggregated. See below for more details.

sumsmall — Logical; should the suppresslimSum value be applied on the sum of small grid cells within the lower resolution grid cell? Note that different combinations of suppreslim and suppreslimSum values might not give completely intuitive results.For instance, if both are equal, then a higher value can lead to more grid cells being left unaggregated for smaller grid sizes, leading to aggregation for a large grid cell

suppresslimSum — Parameter similar to suppreslim, but affecting the total of grid cells to be suppressed

reliabilitySplit — Logical or number - parameter to be used in calculation of the reliability (if checkReliability = TRUE). It can either give the number of groups, or if TRUE, it will create groups of approdcimately 50,000 records per group. If FALSE, the data set will not be split, independent on the size.

pseudoreg — A column with regions to be used to define pseudostrata if checkReliability is TRUE. This is used for the cases when one or more strata only has a single record (and the weight is different from one). This makes variance calculation

impossible, so such strata are merged into a pseudostrata. If pseudoreg is given (for example a column with the country name, or NUTS2 region), the pseudostrata will be created separately for each pseudoreg region.

`plotIntermediate`
: Logical or number - make a simple plot showing which grid cells have already passed the frequency rule. plotintermediate = TRUE, the function will wait 5 seconds after plotting before continuing, otherwise it will wait plotintermediate seconds.

`addIntermediate`
: Logical; will add a list of all intermediate himgs and lohs (overlay of himg and the lower resolution grid) as an attribute to the object to be returned

`locAdj`
: parameter to adjust the coordinates if they are exactly on the borders between grid cells. The values can either be FALSE, or "jitter" (adding a small random value to the coordinates, essentially spreading them randomly around the real location), "UR", "UL", "LR" or "LL", to describe which corner of the grid cell the location belong (upper right, upper left, lower right or lower left).

`postProcess`
: Logical; should the postprocessing be done as part of creation of the multiresolution grid (TRUE), or be done in a separate step afterwards (FALSE). The second option is useful when wanting to check the confidential grid cells of the final map

`rounding`
: either logical (FALSE) or an integer indicating the number of decimal places to be used. Negative values are allowed (such as the default value rounding to the closest 10). See also the details for `digits` in `round`.

`remCols`
: Logical; Should intermediate columns be removed? Can be set to FALSE for further analyses. Temporary columns will not be removed if their names partly match the variable names of `vars`

`...`
: Other parameters to underlying print functions

`x`
: MRG-object, created by call to `createMRGobject`

Details

The function creates a single object, containing both the mapped data and the parameters for for further processing. This assures that all processing is done with the same variables.

Value

A list containing the necessary elements for further processing with the `MRG`-package, referred to as being of class `MRG`.

Examples

```
library(sf)

# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data

# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")
```

```
ress = 1000*2^(1:7)
MRGobject = createMRGobject(ifg = ifg, ress = ress, var = "UAA")
# Run the adaptive grid function only with farm number as con, then plot results
himg1 = multiResGrid(MRGobject)

himg1 = multiResGrid(MRGobject)
# Parameters can be updated in the object or in the call to multiResGrid
MRGobject$suppresslim = 0.02
himg2 = multiResGrid(MRGobject)
himg3 = multiResGrid(MRGobject, suppresslim = 0.05)
```

`fssgeo` *Function that creates an sf-object from IFS data*

Description

Function that creates an sf-object from IFS data

Usage

```
fssgeo(ifs, crsOut = 3035, locAdj = FALSE)
```

Arguments

`ifs`	A data.frame or tibble with the locations and the data of the survey or census data
`crsOut`	The coordinate reference system (crs) to be used
`locAdj`	parameter to adjust the coordinates if they are exactly on the borders between grid cells. The values can either be FALSE, or "jitter" (adding a small random value to the coordinates, essentially spreading them randomly around the real location), "UR", "UL", "LR" or "LL", to describe which corner of the grid cell the location belong (upper right, upper left, lower right or lower left).

Details

The geo-location in the FSS file has a particular format. For 2020, it includes country, coordinate reference system (CRS), resolution (precision of coordinates) and coordinates in one attribute ("GEO_LCT"). For past years, the FSS data structure differs and it includes three separate columns, like latitudes, longitudes and coordinate reference system. This function splits the attribute in its individual parts, and creates an sf-object with the correct coordinates and CRS.

Value

An `sf`-object with the locations of the survey or census data

Examples

```
data(ifs_dk)
ifg = fssgeo(ifs_dk)
```

`gridData` *Function that converts point data to gridded data (polygon values) or a list of gridded data*

Description

Function that converts point data to gridded data (polygon values) or a list of gridded data

Usage

```
gridData(
  ifg,
  res = 1000,
  vars = NULL,
  weights = NULL,
  nclus = 1,
  confrules = "individual",
  crsOut = 3035,
  verbose = FALSE,
  locAdj = FALSE
)
```

Arguments

`ifg`	Either a data.frame or tibble or sf-object with the locations and the data of the survey or census data, or a list of such objects.
`res`	A resolution or a vector with the different resolutions
`vars`	Variable(s) of interest that should be aggregated (necessary when ifg is used for individual farm specific anonymization rules)
`weights`	Extrapolation factor(s) (weights) wi of unit i in the sample of units nc falling into a specific cell c. Weights are used for disclosure control measures. A weight of 1 will be used if missing. If only one weight is given, it will be used for all variables. If the length is more than one, the length has to be equal to the number of variables. If the same weight is used for several variables, it must be repeated in the weights-vector
`nclus`	Number of clusters to use for parallel processing. No parallelization is used for `nclus = 1`.
`confrules`	Should the frequency rule (number of holdings) refer to the number of holdings with a value of the individual vars above zero ("individual") or the total number of holdings in the data set ("total")?

crsOut	The coordinate reference system (crs) to be used
verbose	Indicates if some extra output should be printed. Usually TRUE/FALSE, but can also have a value of 2 for multiResGrid for even more output.
locAdj	parameter to adjust the coordinates if they are exactly on the borders between grid cells. The values can either be FALSE, or \"jitter\" (adding a small random value to the coordinates, essentially spreading them randomly around the real location), \"UR\", \"UL\", \"LR\" or \"LL\", to describe which corner of the grid cell the location belong (upper right, upper left, lower right or lower left). Please use with care in this function. It will make it possible to produce the grid,but notice that the coordinates of ifg will be left untouched, which can cause problems if this is used in other functions.

Details

This will create hierarchical grids of the selected variable(s), at the requested resolution(s), and using the requested function. In reality, the function will usually be sum, mean or max3, where the last one gives the average of the three highest numbers in the grid cell.

Additionally, the function will always return the extrapolated number of farms per grid unit. The result will either be a set of sf-polygons (default) or a stars object.

Value

A hierarchical list of gridded data, in the different resolutions requested. Each grid also includes the count of records used for the gridding, and the sum of the weights.

Examples

```
library(sf)
if (!require(ggplot2)) print("Plotting of results will not work without installation of ggplot2")
if (!require(viridis)) print("Some of the plots will not work without installation of ggplot2")
if (require(giscoR)) {
  useBorder = TRUE
} else {
  useBorder = FALSE
  print("You need to install giscoR for plotting borders and clipping the gridded maps")
}

# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data
# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")

if (useBorder) {
# Read country borders, only used for plotting
  borders = gisco_get_nuts(nuts_level = 0)
  dkb = borders[borders$CNTR_CODE == "DK",] %>% st_transform(crs = 3035)
}

ress = c(1,5,10,20,40,80)*1000
ifl = gridData(ifg, vars = c("UAA", "UAAXK0000_ORG"), weights = "EXT_CORE",
```

```
                res = ress)
ifl2 = gridData(ifg, vars = c("UAA", "UAAXK0000_ORG"), weights = "EXT_CORE",
                res = ress, nclus = 2)
all.equal(ifl, ifl2)
if (require(ggplot2)) {
ifall = do.call("rbind", ifl)
g1 = ggplot() + geom_sf(data = ifall, aes(fill = count, color = count)) +
 scale_fill_viridis( name = "number of \n holdings", trans = "log10") +
 scale_color_viridis( name = "number of \n holdings", trans = "log10") +
 coord_sf(crs = 3035) +
 theme_bw() +
 ggtitle("Number of holdings for different resolutions") +
 facet_wrap(vars(res))
 if (useBorder) g1 = g1 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)
 g1
}
#'

MRGcluster(action = "stop")
```

`ifs_dk` *Test data sets for the MRG package*

Description

Synthetic data set of Danish farming data, similar to the structure of the real Farm Structure Survey (FSS) data. It contains more than 37,000 synthetic records - generated in a way that should replicate the structure and the distribution of real data, but where the individual data are different from the real data.

Usage

```
data(ifs_dk)
```

Format

A data frame with 37,088 rows and 14 variables

- COUNTRY The name of the country
- YEAR The year of the survey data
- ID_SYNTH Unique ID of the record
- FARMTYPE Farm typology. Farms are classified into different types according to their dominant activity and standard output value (proxy for farm income). For further information see https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Farm_typology

- HLD_FEF Not used. Farm is included in frame extension (HLD_FEF=1) or main frame (HLD_FEF=0)
- REGIONS NUTS2 region
- GEO_LCT The geolocation in typical FSS-format, including both country, CRS and xy coordinates
- EXT_CORE The extrapolation weights for core data (1 in this data set)
- STRA_ID_CORE Which stratum the record belongs to - only used for the reliability checking
- UAA The utilized agricultural area of the farm
- UAAXK0000_ORG The organic utilized agricultural area, excluding kitchen gardens of the farm. UAAXK0000_ORG includes fully certified area and area under conversion
- Sample Whether the record should be included as a weighted subsample
- EXT_MODULE The extrapolation weights for the sample data

A data frame with 37088 rows and 14 variables

Details

The variables are as follows:

For practical purposes, we have derived a synthetic data set from the original 2020 agricultural census micro data. Although synthetic data sets are a feasible way to provide public access to the data by mitigating any confidentiality concerns, there have only been a few attempts made to create synthetic public files of micro data collected by official statistical institutes.

The attached data set has been produced by application of a hot-deck procedure - originally developed to impute missing information - to substitute a data entry from the original data (i.e., the recipient) by using a value from a similar record (i.e., the donor) within the same classification group (Andridge and Little, 2010; Ford, 1983; Joenssen and Bankhofer, 2012).

A single hot deck imputed data set is computed for each country individually. First, records are partitioned into homogeneous groups so that the donors follow the same distribution as the recipients. Data points from the recipients are substituted sequentially based on a value from a varying pool of donors. Furthermore, the nearest neighbour matching technique using distance metrics is applied to select the most appropriate donor from the pool of donors. For a few of the discrete variables, such as $FARMTYPE$, SO_EUR, HLD_FEF and $NUTS2$, a donor was chosen randomly by preserving the original empirical distribution or they were simply randomly decoded (i.e., renamed). The variable containing information about the geographical location (GEO_LCT) of the agricultural holding was imputed by restricting the donor to the same country. To assess the quality rating system (i.e., the reliability), we created an artificial sample ($SAMPLE$) with the respective extrapolation factors (EXT_MODULE) based on stratification. The sample size consists of approximately one third of the synthetic 2020 census for Denmark.

The empirical distribution of the two main variables of interest of the synthetic data, UAA and $UAAXK0000_ORG$ are widely preserved within the different economic size classes.

References

Andridge RR, Little RJ (2010). A review of hot deck imputation for survey non-response. International statistical review, 78(1), 40–64.

Ford BL (1983). An overview of hot-deck procedures." Incomplete data in sample surveys, 2(Part IV), 185–207.

Joenssen DW, Bankhofer U (2012). Hot deck methods for imputing missing data. In P Perner (ed.), Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Computer Science, pp. 63–75. Springer, Berlin, Heidelberg. ISBN 978-3-642-31537-4. doi:10.1007/978-3-642-31537-4_6

`locAdjFun` *Function that modifies the observation locations, to reduce the risk that they are on grid cell boundaries*

Description

Function that modifies the observation locations, to reduce the risk that they are on grid cell boundaries

Usage

```
locAdjFun(ifg, locAdj, ress)
```

Arguments

`ifg`	Either a data.frame or tibble or sf-object with the locations and the data of the survey or census data, or a list of such objects.
`locAdj`	parameter to adjust the coordinates if they are exactly on the borders between grid cells. The values can either be FALSE, or "jitter" (adding a small random value to the coordinates, essentially spreading them randomly around the real location), "UR", "UL", "LR" or "LL", to describe which corner of the grid cell the location belong (upper right, upper left, lower right or lower left).
`ress`	A vector with the different resolutions

Details

This can be used as a pre-processing step before creating a multi-resolution grid. The gridding procedure will have problems if the points are located exactly on grid cell boundaries. The locations should therefore be slightly modified, to better control to which grid cells they are associated. This can either be a systematic modification, or a random modification.

In the case of FSS data, the coordinates have been reported as the lower left corner of a 1 km grid.

Value

An `sf`-object with slightly modified locations for the survey or census data, according to the `locAdj`-parameter

Examples

```
data(ifs_dk)

ifg = fssgeo(ifs_dk, locAdj = FALSE)
ifg = locAdjFun(ifg, "LL")
```

`MRGcluster` *Function that allows to apply parallel processing*

Description

Function that allows to apply parallel processing

Usage

```
MRGcluster(nclus, ..., action = "start", clusType, outfile = NULL)
```

Arguments

`nclus`	Number of clusters to use for parallel processing. No parallelization is used for `nclus = 1`.
`...`	arguments that should be evaluated in the cluster (can also be called later)
`action`	Defines the action of the function. There are three options: **"start"** Starts a new cluster if necessary, reuses an existing if it has already been started **"restart"** Stops the cluster and starts it again. To be used in case there are difficulties with the cluster, or if the user wants to change the type of the cluster
`clusType`	The type of cluster; see `makeCluster` for more details. The default of makeCluster is used if type is missing or NA.
`outfile`	File to direct the output, `makeCluster` for more details.

Value

The function will either return a cluster for parallel computation, or stop a cluster (returning NULL)

MRGfromDF *Function to create a gridded (usually multi-resolution grid) from a data.frame or csv file with information about the corners and resolution, as typically can be downloaded from Eurostat. The function can also save the grid as a geo-object.*

Description

Function to create a gridded (usually multi-resolution grid) from a data.frame or csv file with information about the corners and resolution, as typically can be downloaded from Eurostat. The function can also save the grid as a geo-object.

Usage

```
MRGfromDF(
  df,
  coords = c("x", "y"),
  coordscale,
  crs = NA,
  res = "res",
  Estat = TRUE,
  cignore = FALSE,
  dsn,
  layer,
  ...
)
```

Arguments

df	A data.frame or name of a csv file with multi-resolution data, only specifying the lower left corner of the grid cells
coords	Names of the numeric columns holding coordinates
coordscale	Multiplication scaling factor for coordinates
crs	The coordinate reference system (CRS) into which all data should be projected before plotting. If not specified, will use the CRS defined in the first sf layer of the plot.
res	A resolution or a vector with the different resolutions
Estat	Indicate if Eurostat is the source of the data set. This is currently the default, but this might be changed in the future if other providers will follow the same conventions
cignore	Logical; Should the function ignore if parameters appear to be neither lat-lon or projected
dsn	Source name to be used by st_write. Interpretation varies by driver: can be a filename, a folder, a database name, or a Database Connection.

layer	Layer name to be used by st_write. Varies by driver, may be a file name without extension; for database connection, it is the name of the table. If layer is missing, the basename of dsn is taken.
...	Additional parameters to read.csv if csv is a file name

Details

This function is mainly for handling csv files downloaded from Eurostat, but can also be used for data from other sources, which adapt the same csv-convention as Eurostat.

The Eurostat-files have x- and y-coordinates that have been projected in the epsg:3035 projection. However, the coordinates show kilometers, not meters, so they have to be multiplied with 1000. Similar data sets might also be offered by other providers. The multiplication can be done with coordscale, or with `Estat = TRUE` (which also sets `crs = 3035`)

The function will also check the coordinates, if the range of both x- and y-coordinates are between 360 and 20000, it would often indicate that the coordinates should be multiplied. The function will suggest to correct this. If the coordinates are actually correct, the check can be overrun with `cignore = TRUE`

If writing to file, it is necessary to add the dsn and potentially layer to the input.

Value

The function produces a multiresolution grid, which is a sf-object with polygons.

Examples

```
library(MRG)
library(dplyr)
library(tidyr)
# C19.csv is an example file from Eurostat, including Utilized Agricultural Area
# (UAAXK0000) and organic UAA (UAAXK0000_ORG)
csvFile = system.file("ex/C19.csv", package="MRG")
C19 = MRGfromDF(csvFile, crs = 3035)  %>% mutate(orgShare = UAAXK0000_ORG/UAAXK0000)
MRGplot(C19, var = orgShare, xlim = c(2600000, 5800000), ylim = c(1500000, 5200000))
```

MRGmerge *Merge two or more multi-resolution grids to a common resolution*

Description

Merge two or more multi-resolution grids to a common resolution

Usage

```
MRGmerge(himg1, himg2, vars1, vars2, na.rm = TRUE, postProcess = FALSE, ...)
```

Arguments

himg1	Either a grid resulting from a call to multiResGrid, or a list of such grids
himg2	A grid resulting from a call to multiResGrid
vars1	Variable(s) of interest that should be merged from the first grid, or a list of variables, one for each grid in the list himg1
vars2	Variable(s) of interest that should be merged from the second grid
na.rm	Should NA values be removed when summing values (essentially treating them equal to zero)
postProcess	Logical; should the postprocessing be done as part of creation of the multiresolution grid (TRUE), or be done in a separate step afterwards (FALSE). The second option is useful when wanting to check the confidential grid cells of the final map
...	Additional grids (himg3, himg4, ...) and variables (vars3, vars4, ...) to be merged. Additional grids and variables must be named.

Details

This function can merge different multi-resolution grids to a common resolution, i.e., it will select the grid cells with the lowest resolution, as these are the ones defining the restrictions.

The function will merge the variable names in `vars1, vars2, ...` if they exist. If they are missing, the function will look for variable names in the attributes of the grids (`attr(himg, "vars")`). These are added by `multiResGrid`, but will often disappear if the grid has been manipulated, or has been exported to another format for transmission.

If the variables are not given as `vars` or attributes, the function will try to guess them from the column names. Typical column names used by MRG (mostly temporary variables such as `small`, `confidential` etc) will be ignored. If variable names partly coincide with any of these names, or with `count`, `res`, `geometry`, it is necessary to specify vars.

The multi-resolution grids must be passed as named parameters if more than two are given.

Common variable names in different grids should be avoided.

The default of the function is to treat NA-values as zeroes when merging (through `na.rm` in sums). It will therefore not be possible to separate restricted grid cells from grid cells with zero observations after merging, except for the ones that have been left as they were. The alternative would be a much higher number of NA-values in the merged grids.

The resulting grid will most likely not have exactly the same values as a multi-resolution grid produced directly from the microdata. If the input-grids have been post-processed (the normal situation when not having access to the microdata), the grid cell values have usually been rounded, and some might have been suppressed. As these rounded and potentially suppressed values are summed, their values are likely to deviate from those that are computed directly from the microdata through a joint gridding process.

Value

The function produces a new multiresolution grid, which is a `sf`-object with polygons.

Examples

```
library(sf)
library(dplyr)

# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data
ifs_weight = ifs_dk %>% dplyr::filter(Sample == 1) # Extract weighted subsample

# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")
fsg = fssgeo(ifs_weight, locAdj = "LL")

# We use the numeric part of the farmtype to create a third variable. This
# is done for the an example, the value does not have any meaning when treated
# like this
ifg$ft = as.numeric(substr(ifg$FARMTYPE, 3, 4))^2

ress = c(1,5,10,20,40, 80, 160)*1000
# Create regular grid of the variables
ifl = gridData(ifg, vars = c("UAA", "UAAXK0000_ORG", "ft"), res = ress)

# Create the different multi-resolution grids
himg1 = multiResGrid(ifl, vars = "UAA", ifg = ifg, postProcess = FALSE)
himg2 = multiResGrid(ifl, vars = "UAAXK0000_ORG", ifg = ifg, postProcess = FALSE)
himg3 = multiResGrid(ifl, vars = "ft", ifg = ifg, postProcess = FALSE)

# The grids have different number of polygons
dim(himg1)
dim(himg2)
dim(himg3)

hh1 = MRGmerge(himg1, himg2, himg3 = himg3)
dim(hh1)
# Postprocessing can also be done on the merged object
hh11 = MRGmerge(himg1, himg2, himg3 = himg3, postProcess = TRUE, rounding = -1)
dim(hh11)
summary(hh1$UAA-hh11$UAA)

# If two data sets share the same variable, one of them has to be renamed.
# (A comparison of the two can act as a indication of possible errors
# introduced through the post-processing)

himg21 = multiResGrid(ifl, vars = c("UAA", "UAAXK0000_ORG"), ifg = ifg, postProcess = FALSE)
hh3 = try(MRGmerge(himg1, himg21, himg3 = himg3))
himg21 = himg21 %>% rename(UAA2 = UAA, weight_UAA2 = weight_UAA)
hh3 = MRGmerge(himg1, himg21, himg3 = himg3)


summary(hh3[, c("UAA", "UAA2")])

himg4 = multiResGrid(ifl, vars = c("UAA", "ft", "UAAXK0000_ORG"), ifg = ifg, postProcess = FALSE)
summary(hh1[, c("UAA", "UAAXK0000_ORG", "ft")])
```

```
summary(himg4[, c("UAA", "UAAXK0000_ORG", "ft")])
```

`MRGoverlap` *Function that finds and merges overlapping grid cells in a multi-resolution grid The need for this function comes from an error in the gridding process, and it can be seen as symptom solving rather than solving the issue. This function will either just show the problematic grid cells or remove the overlaps.*

Description

Function that finds and merges overlapping grid cells in a multi-resolution grid The need for this function comes from an error in the gridding process, and it can be seen as symptom solving rather than solving the issue. This function will either just show the problematic grid cells or remove the overlaps.

Usage

```
MRGoverlap(himg, vars, himg2, action = "sum")
```

Arguments

`himg`	The grid resulting from a call to multiResGrid
`vars`	Variable(s) of interest that should be aggregated (necessary when ifg is used for individual farm specific anonymization rules)
`himg2`	A grid resulting from a call to multiResGrid
`action`	How to treat the values of overlapping grid cells. Possible values are:

none return an `sf` data.frame just with the overlapping grid cells

sum sum the values of the overlapping grid cells - NAs are ignored unless both cells are NA or one is NA and one is 0

sumna sum the values of the overlapping grid cells - sum is NA if any of them is NA

avg avg of the grid cells - NAs are ignored unless both cells are NA or one is NA and one is 0

avgna avg of the grid cells - avg is NA if any of them is NA

replace replace the problematic grid cells with grid cells from `himg2`

Details

A multi-resoluion grids should not have overlapping grid cells, by definition. However, this could happen through stitching different grids together. Although this should rather have been taken care of during the gridding process, this is not always possible to redo for an end-user.

This function can first of all be used to identify and show the overlapping grid cells. It can also be used to create a valid multi-resolution grid for these cases, as long as the grd cells all have the same base grid (i.e. no overlapping grid cells are partly overlapping, whether the grid cells have the same size or not). However, there will be some additional errors introduced in this process.

Except for `action = "none"` and `action = "replace"`, the function will try to create a valid grid based on the values in the grid. This means that it has to find a sensible value for the merged grid cells. Frequently one of the grid cells will have an NA value, meaning that it has been suppressed. This means that there are observations in the grid cell, but relatively few. If the overlapping grid cell has a value, this is most likely larger, it is non-confidential. The default action is therefore to sum values, but ignore NA-values unless both are NA or one is NA and the other is 0. The last case means that one of the grid cells have a confidential number of records, whereas the other one has zero records. The total is then a confidential number of records.

if `action = "restart"`, there must be a second grid which includes the updated values of the overlapping grid cells. This could typically happen if the data set is too large to be processed as a single batch. There could then be overlapping grid cells on the border between different batches. Instead of reprocessing the entire grid, it is possible to reprocess the border regions (in one more more batches, as long as they are not overlapping). The grid cells from the border regions have to be passed as `himg2`.

The function uses `st_join` to check for overlaps. If the grid has a very high number of grid cells (a few tens of thousands), this process can be rather slow. In that case, it might be better to check parts of the grid separately.

Value

The function will return different objects, depending on "action". The returned object will for different values of "action" be:

none An `sf` data.frame with the overlapping grid cells

sum A multi-resolution grid containing the sum the values of the overlapping grid cells. NAs are ignored unless both cells are NA or one is NA and one is 0

sumna A multi-resolution grid containing the sum the values of the overlapping grid cells. The sum will be NA if any of them is NA

avg A multi-resolution grid containing the average of the grid cells. NAs are ignored unless both cells are NA or one is NA and one is 0

avgna A multi-resolution grid containing the average of the grid cells. The average will be NA if any of them is NA

replace A multi-resolution grid containing, where the problematic grid cells area replaced with grid cells from `himg2`

Examples

```
library(sf)
```

```
library(giscoR)
library(dplyr)

# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data

# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")

nuts2 = gisco_get_nuts(nuts_level = 2)
nuts2 = nuts2 %>% filter(CNTR_CODE == "DK") %>% st_transform(crs = st_crs(ifg))

ifg$NUTS2 = st_join(ifg, nuts2, join = st_nearest_feature)$NUTS_ID
ress = c(1,5,10,20,40, 80, 160)*1000
# Create regular grid of the variables, for three regions
ifl = gridData(ifg[ifg$NUTS2 %in% c("DK03", "DK04", "DK05"),], vars = c("UAA"), res = ress)
ifl3 = gridData(ifg[ifg$NUTS2 == "DK03",], vars = c("UAA"), res = ress)
ifl4 = gridData(ifg[ifg$NUTS2 == "DK04",], vars = c("UAA"), res = ress)
ifl5 = gridData(ifg[ifg$NUTS2 == "DK05",], vars = c("UAA"), res = ress)

# Create the different multi-resolution grids for different nuts regions
himg3 = multiResGrid(ifl3, vars = "UAA", ifg = ifg[ifg$NUTS2 == "DK03",], suppresslim = 0.02)
himg4 = multiResGrid(ifl4, vars = "UAA", ifg = ifg[ifg$NUTS2 == "DK04",], suppresslim = 0.02)
himg5 = multiResGrid(ifl5, vars = "UAA", ifg = ifg[ifg$NUTS2 == "DK05",], suppresslim = 0.02)

# Bind them together and create new consecutive IDs for the grid cells
himg = rbind(himg3, himg4, himg5)
himg$ID = 1:dim(himg)[1]

# Find the overlapping grid cells, and show some examples.
himgd = MRGoverlap(himg, action = "none")
dim(himgd)
himgd[himgd$ID.y %in% 932:940,]

# Remove overlapping grid cells
himgnew = MRGoverlap(himg, action = "sum")

# Check that there are no more overlaping grid cells
himgd2 = MRGoverlap(himgnew, action = "none")
himgd2


# Create a new multi-resolution grid which has the correct grid cells
# at the border. In this example, the region of interest is so small that
# it is difficult to reprocess just the border grid cells, so
# we make a new complete grid

himg1 =  multiResGrid(ifl, vars = "UAA", ifg = ifg[ifg$NUTS2 %in% c("DK03", "DK04", "DK05"),],
                      suppresslim = 0.02)
himgnew2 = MRGoverlap(himg, himg2 = himg1, action = "replace")
himgd12 = MRGoverlap(himgnew2, action = "none")
himgd12
```

MRGplot *Convenience function based on ggplot2 to plot multi-resolution grids with some default suggestions For full flexibility it is better to use ggplot2 directly.The function can also be used for ordinary grids*

Description

Convenience function based on ggplot2 to plot multi-resolution grids with some default suggestions For full flexibility it is better to use ggplot2 directly.The function can also be used for ordinary grids

Usage

```
MRGplot(
  himg,
  var,
  linecolor,
  option = "D",
  lwd = 0,
  lwdb = 1,
  borders,
  name = waiver(),
  title = NULL,
  xlim,
  ylim,
  crs,
  clip = TRUE,
  transform = "identity",
  show.legend = TRUE
)
```

Arguments

himg	The grid resulting from a call to multiResGrid
var	Which variable to plot
linecolor	Which column or color to use for lines between grid cells. The default is not to plot lines
option	The color map option to use, see scale_color_viridis for more details
lwd	Line width for the grid cells. Default is zero, to remove or minimize (for pdf) the line width
lwdb	The line width for the border

borders	A polygon object with borders than can be drawn on top of the multi-resolution grid. The object will also be used to clip the grid if clip = TRUE.
name	Name to be used for color scale. The default is to use the name of the fill/color column. name = NULL will give no name.
title	The title of the plot
xlim	The limits for the x-axis. The default is to use the bounding box of the grid.
ylim	The limits for the y-axis. The default is to use the bounding box of the grid.
crs	The coordinate reference system (CRS) into which all data should be projected before plotting. If not specified, will use the CRS defined in the first sf layer of the plot.
clip	Logical; should the grid be clipped to the borders object (if exsisting)?
transform	Possible transformation of the color scale, typical values can be "log", "log10" or "sqrt", based on available transformations in the scales package. See for example transform_log and other transformations for more details.
show.legend	Logical; should the legend be shown or not.

Details

The function is a wrapper around ggplot, possibly calling geom_sf twice, for the grid itself and for the borders. The function uses the scale_color_viridis color scale.

Value

The function will plot the object, and also return a valid ggplot-object that can be further customized.

Examples

```
library(sf)
library(ggplot2)

if (require(giscoR)) {
  useBorder = TRUE
} else {
  useBorder = FALSE
  print("You need to install giscoR for plotting borders and clipping the gridded maps")
}
# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data

# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")

if (useBorder) {
# Read country borders, only used for plotting
  borders = gisco_get_nuts(nuts_level = 0)
}
```

```
ress = c(1,5,10,20,40, 80, 160)*1000
# Gridding Utilized agricultural area (UAA)
ifl = gridData(ifg, "UAA",res = ress)

# Create a multi-resolution grid of UAA
himg1 = multiResGrid(ifl, vars = "UAA", ifg = ifg)

if (useBorder) {
  p1 = MRGplot(himg1, UAA, transform = "log10", borders = borders)
} else {
  p1 = MRGplot(himg1, UAA, transform = "log10")
}
p1

# Plot can be customized further (reverting to ggplot default color scale in this case)
p1 + scale_color_continuous() + scale_fill_continuous()
```

`MRGpostProcess` *Make some final adjustments to the multiresolution grids*

Description

Make some final adjustments to the multiresolution grids

Usage

```
MRGpostProcess(himg, vars, remCols = TRUE, rounding = "varying")
```

Arguments

`himg`	The grid resulting from a call to multiResGrid
`vars`	Variable(s) of interest that should be aggregated (necessary when ifg is used for individual farm specific anonymization rules)
`remCols`	Logical; Should intermediate columns be removed? Can be set to FALSE for further analyses. Temporary columns will not be removed if their names partly match the variable names of `vars`
`rounding`	either logical (FALSE) or an integer indicating the number of decimal places to be used. Negative values are allowed (such as the default value rounding to the closest 10). See also the details for `digits` in `round`.

Details

The postprocessing function is normally called directly from `multiResGrid`. However, it might be useful to check the values of the grid cells that will be suppressed, and the values before rounding. In that case `multiResGrid` can be called with the argument `postProcess = FALSE`, and the post processing be done separately.

Value

The function will return a post-processed multi-resolution grid with non-confidential gridded data. See `multiResGrid` for more information.

Examples

```
library(sf)

# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data
# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")

# Set the base resolutions, and create a hierarchical list with gridded data
ress = 1000*2^(1:7)
ifl = gridData(ifg, "UAA", res = ress)
himg = multiResGrid(ifl, ifg = ifg, var = "UAA", weight = "EXT_CORE", postProcess = FALSE)
himgp = MRGpostProcess(himg, var = "UAA")

# Confidential grid cells, being suppressed in postProcessing
himg[himg$confidential,]
```

`multiResGrid` *Create multi-resolution grids based on confidentiality or reliability restrictions*

Description

Function that creates a multi-resolution grid with larger grid cells in regions with lower resolution of data, or where data needs to be anonymized for disclosure control reasons. The function can also be used to create a grid of new variables, using an existing multi-resolution grid as template. The possible restrictions that will lead to aggregation of a grid cell are:

1. Frequency rule (Aggregate to reach a minimum number of counts)
2. Dominance rule (Aggregate because of dominance by one or more units)
3. p-percent rule (Aggregate because the second largest producer could identify the production of the largest producer with less than p percent uncertainty.)
4. Reliability rule (Aggregate because the uncertainty is too high)
5. User defined rule (Aggregate because a grid cell does not respect a user defined criteria)

Usage

```
multiResGrid(MRGinp, ...)

## S3 method for class 'MRG'
multiResGrid(MRGinp, ...)

## S3 method for class 'sf'
multiResGrid(MRGinp, ..., ifg, vars)

## S3 method for class 'list'
multiResGrid(
  MRGinp,
  ifg,
  vars,
  weights,
  countFeatureOrTotal = "feature",
  mincount = 10,
  nlarge = 2,
  plim = 0.85,
  verbose = FALSE,
  domEstat = TRUE,
  outfile = NULL,
  checkDominance = TRUE,
  checkPpercent = FALSE,
  pPercent = 20,
  checkReliability = FALSE,
  userfun,
  strat = NULL,
  confrules = "individual",
  suppresslim = 0,
  sumsmall = FALSE,
  suppresslimSum = NULL,
  reliabilitySplit = TRUE,
  pseudoreg = NULL,
  plotIntermediate = FALSE,
  addIntermediate = FALSE,
  postProcess = TRUE,
  rounding = "varying",
  remCols = TRUE,
  ...
)
```

Arguments

MRGinp	Either an MRGobject (from a call to createMRGobject) or a list of gridded data with different resolutions (from a call to gridData or a gridded sf-object (typically from an earlier call to multiResGrid)
...	Possible arguments to underlying functions

ifg	Either a data.frame or tibble or sf-object with the locations and the data of the survey or census data, or a list of such objects.
vars	Variable(s) of interest that should be aggregated (necessary when ifg is used for individual farm specific anonymization rules)
weights	Extrapolation factor(s) (weights) wi of unit i in the sample of units nc falling into a specific cell c. Weights are used for disclosure control measures. A weight of 1 will be used if missing. If only one weight is given, it will be used for all variables. If the length is more than one, the length has to be equal to the number of variables. If the same weight is used for several variables, it must be repeated in the weights-vector
countFeatureOrTotal	Should the frequency limit be applied on records with a positive value for a certain feature, or on all records, independent of value of feature
mincount	The minimum number of farms for a grid cell (threshold rule)
nlarge	Parameter to be used if the nlarge(st) farms should count for maximum plim percent of the total value for the variable in the grid cell (see details of gridData)
plim	See nlarge
verbose	Indicates if some extra output should be printed. Usually TRUE/FALSE, but can also have a value of 2 for multiResGrid for even more output.
domEstat	Should the dominance rule be applied as in the IFS handbook (TRUE), where the weights are rounded before finding the first nlarge contributors, or should it be the first nlarge contributors*weight, where also fractions are considered (FALSE)?
outfile	File to direct the output in case of parallel processing, see makeCluster for more details.
checkDominance	Logical - should the dominance rule be applied?
checkPpercent	Logical - should the p-percent rule be applied?
pPercent	Which limit to use for the p-Percent rule?
checkReliability	Logical - should the prediction variance be checked, and used for the aggregation? This considerably increases computation time
userfun	This gives the possibility to add a user defined function with additional confidentiality rules which the grid cell has to pass
strat	Column name defining the strata for stratified sampling, used if checkReliability is TRUE
confrules	Should the frequency rule (number of holdings) refer to the number of holdings with a value of the individual vars above zero ("individual") or the total number of holdings in the data set ("total")?
suppresslim	Parameter that can be used to avoid that almost empty grid cells are merged with cells with considerably higher number of observations. The value is a minimum share of the total potential new cell for a grid cell to be aggregated. See below for more details.

`sumsmall` Logical; should the suppresslimSum value be applied on the sum of small grid cells within the lower resolution grid cell? Note that different combinations of suppreslim and suppreslimSum values might not give completely intuitive results.For instance, if both are equal, then a higher value can lead to more grid cells being left unaggregated for smaller grid sizes, leading to aggregation for a large grid cell

`suppresslimSum` Parameter similar to suppreslim, but affecting the total of grid cells to be suppressed

`reliabilitySplit`
Logical or number - parameter to be used in calculation of the reliability (if checkReliability = TRUE). It can either give the number of groups, or if TRUE, it will create groups of approdcimately 50,000 records per group. If FALSE, the data set will not be split, independent on the size.

`pseudoreg` A column with regions to be used to define pseudostrata if checkReliability is TRUE. This is used for the cases when one or more strata only has a single record (and the weight is different from one). This makes variance calculation impossible, so such strata are merged into a pseudostrata. If pseudoreg is given (for example a column with the country name, or NUTS2 region), the pseudostrata will be created separately for each pseudoreg region.

`plotIntermediate`
Logical or number - make a simple plot showing which grid cells have already passed the frequency rule. plotintermediate = TRUE, the function will wait 5 seconds after plotting before continuing, otherwise it will wait plotintermediate seconds.

`addIntermediate`
Logical; will add a list of all intermediate himgs and lohs (overlay of himg and the lower resolution grid) as an attribute to the object to be returned

`postProcess` Logical; should the postprocessing be done as part of creation of the multiresolution grid (TRUE), or be done in a separate step afterwards (FALSE). The second option is useful when wanting to check the confidential grid cells of the final map

`rounding` either logical (FALSE) or an integer indicating the number of decimal places to be used. Negative values are allowed (such as the default value rounding to the closest 10). See also the details for `digits` in `round`.

`remCols` Logical; Should intermediate columns be removed? Can be set to FALSE for further analyses. Temporary columns will not be removed if their names partly match the variable names of `vars`

Details

This function will find the highest resolution data set that fulfills the confidentiality rules and potential reliability rules for variable(s) of interest. Starting with the second highest resolution (5 km in the default settings), the function will check if any of the 1 km sub pixels will have values not fulfilling any of the confidentiality rules (number of farms, values of the 2 largest compared to values of the entire grid cell). If all values are above the confidentiality limits, the grid cells will be kept at a 1 km resolution, otherwise only the 5 km grid cell will be kept. This will again be tested against

the confidentiality rules in the next iteration, when grid cells will possibly be merged to 10 km grid cells.

The function can also be called if it is necessary to create a grid of a new variable for the same grid as an already existing variable. The confidentiality rules will then be applied to the new variables for the existing grid cells, and mask the ones that do not respect the rules. The function will not do any further merging of grid cells, for this it is necessary to grid the variables together. This feature is useful when the new data set has a similar resolution as the original data set. It will give a high number of missing values if the resolution of the new data is more sparse than the original. In the examples below, this means that it is possible to copy the grid of organic organic agricultural area to a grid of all agricultural area, whereas the opposite will not work well.

The standard threshold rule for spatial data is at least 10 units (mincount).

The parameters nlarge and plim are used for determining the dominance treatment for the variable of interest, with default values of `nlarge = 2` and `plim = 0.85`. If more than plim of the values of the grid cell (e.g. UAA, arable land, number of livestock) is explained by 1-nlarge weighted holdings, the grid cell will not pass the confidentiality rule.

It is also possible to apply the p-percent rule. This rule defines a minimum percentage for how close the second largest produces could be of estimating the production of the largest producer by subtracting its own production from the total value of the cell.

$$Y_{cell} - Y_2 - Y_1)/Y_1 < pPercent$$

where Y_{cell}, Y_2, Y_1 represent the total production value of the cell, the value of the second largest production, and the value of the largest production, respectively.

The concept of reliability is explained in details in section 4.6 in the integrated farm survey handbook for 2023: https://wikis.ec.europa.eu/display/IFS/Integrated+Farm+Statistics+Manual+ In short, it is an estimate of the coefficient of variation for an estimate (a grid cell in this case), based on the number in the sample relative to the number in the population, and taking into account possible stratified sampling approaches. The number is zero if all holdings in the population in a grid cell has been sampled, and the default requirement is that the CV is less than 35

The computation can be time and memory intensive, particularly for the first iteration. The method involves creation (and inversion) of a matrix of size `nr*ng`, where `nr` is the number of records and `ng` is the number of grid cells. it is therefore sometimes necessary to split the data set into smaller parts, to reduce the computational challenges. The parameter `reliabilitySplit` is used for this. It will split the area of interest into several subsets. This will have some impact on the reliability calculations. The `reliabilitySplit` value might be set temporarily higher for the first iterations, as it will also depend on the number of grid cells.

Reliability cannot be calculated for records belonging to strata with only one record. The function will therefore attempt to merge these into pseudostrata, if there is more than one of these strata. The `pseudoreg`-parameter can be used to define the regions within which the pseudostrata are created (for example NUTS2-region). If there are still strata with only one record, these will cause a printed warning.

There are some cases where aggregation might not be desired. In the situation where a relatively large single grid cell does not respect the confidentiality rules, it is fine to aggregate it if the neighbouring grid cells are also relatively large. However, it can be seen as unfortunate if the single cell was aggregated with many smaller grid cells that could otherwise be disseminated at a high resolution. The added value of being able to present a value for a region with very few farms is perhaps lower than what is lost by having to aggregate to a lower resolution. The parameter `suppresslim`

indicates the minimum value in a grid cell relative to the possible lower resolution grid cell before it is necessary to aggregate. If the limit is 0.05, a grid cell would only cause an aggregation to lower resolution if the value in the grid cell is more than 5% of the value in the lower resolution grid cell. Instead, it would be left as it is, and will be suppressed in the post-processing step.

There are cases when the built-in confidentiality checks are not what the user needs. That is why it is possible to submit a user defined function. This function needs to follow certain rules.

1. The first argument must be a data.frame with name `df`. This is a data.frame with the individual records for a particular grid cell. It has three columns:
   (a) himgid - the ID of the current grid cell. This is the grouping variable and is constant for the data.frame
   (b) gridvar - a new common name for the current variable to be gridded
   (c) weight - the weight of the variable to be gridded
2. The function can include additional parameters for calculation of confidentiality (or reliability, or suitability, if the meaning of the function refers to something else). This can be new parameters to this particular function (through the ellipsis argument (...) of `multiResGrid`), existing parameters to `multiResGrid`, or potentially internal variables of `multiResGrid.`)
3. The result of the function must be a logical, either the rule was passed for the records of this grid cell, or not (TRUE/FALSE)
4. The function can potentially use internal variables of `multiResGrid`, however, the meaning of these will have to be understood from the code

A simple example of a `userfun` is given in the example section below (the one producing `himg6`)

Value

The function will return a multi-resolution grid with observations gridded to different grid cell sizes according to the confidentiality rules to be applied. It can also include some additional columns that indicates which of the different confidentiality rules that have been applied.

Note that the function might (if `postProcess = FALSE`) return values also for the confidential grid-cells. This is for the case where the owner of the data wants to examine data that will be suppressed during post-processing.

Examples

```
library(sf)
if (!require(ggplot2)) print("Plotting of results will not work
                     without installation of ggplot2")
if (!require(viridis)) print("Some of the plots will not work
                     without installation of viridis package")
if (!require(patchwork)) print("Some of the plots will not work
                     without installation of patchwork")

if (require(giscoR)) {
  useBorder = TRUE
} else {
  useBorder = FALSE
  print("You need to install giscoR for plotting borders and clipping the gridded maps")
}
```

```
# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data
ifs_weight = ifs_dk %>% dplyr::filter(Sample == 1) # Extract weighted subsample

# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")
fsg = fssgeo(ifs_weight, locAdj = "LL")

if (useBorder) {
# Read country borders, only used for plotting
  borders = gisco_get_nuts(nuts_level = 0)
  dkb = borders[borders$CNTR_CODE == "DK",] %>% st_transform(crs = 3035)
}

ress = c(1,5,10,20,40, 80, 160)*1000
# Gridding Utilized agricultural area (UAA)
ifl = gridData(ifg, "UAA",res = ress)
# Gridding organic utilized agricultural area
ifl2 = gridData(ifg, vars = "UAAXK0000_ORG", res = ress)

# Gridding UAA and organic UAA together
ifl3 = gridData(ifg, vars = c("UAA", "UAAXK0000_ORG"), res = ress)

# Gridding the UAA from the survey - the survey weights are in the column EXT_MODULE
fsl = gridData(fsg,  vars = c("UAA"), weights = "EXT_MODULE",  res = ress)

# Create a multi-resolution grid only with farm number as confidentiality rule, then plot results
himg0 = multiResGrid(ifl, checkReliability = FALSE, suppresslim = 0)
ggplot(himg0) + geom_sf(aes(fill = count))

# Create a multi-resolution grid of UAA, also based on the dominance rule (default)
himg1 = multiResGrid(ifl, vars = "UAA", ifg = ifg)
  p1 = ggplot(himg1) + geom_sf(aes(fill = UAA))
  p1

# Create a multi-resolution grid of UAA, also based on the p-percent rule
himg101 = multiResGrid(ifl, vars = "UAA", ifg = ifg, checkPpercent = TRUE)
  p11 = ggplot(himg101) + geom_sf(aes(fill = UAA))
  p11

# Create multi-resolution grid of organic UAA
himg2 = multiResGrid(ifl2, vars = "UAAXK0000_ORG", ifg = ifg)
himg21 = multiResGrid(ifl2, vars = "UAAXK0000_ORG", ifg = ifg, postProcess = FALSE)

ggplot(himg2) + geom_sf(aes(fill = UAAXK0000_ORG))

# Create joint multi-resolution grid of organic UAA and total UAA
himg3 = multiResGrid(ifl3, vars = c("UAA", "UAAXK0000_ORG"), ifg = ifg,
                  checkReliability = FALSE, suppresslim = 0)
# Create multi-resolution grid of organic UAA, based on the UAA grid
# The large number of missing values indicates that this feature should
# mainly be used for data that have similar or higher resolution as the
# original data set.
```

```
himg33 = multiResGrid(himg1, vars = c("UAAXK0000_ORG"), ifg = ifg,
                  checkReliability = FALSE, suppresslim = 0)
p31 = ggplot(himg3) + geom_sf(aes(fill = UAA))
p32 = ggplot(himg3) + geom_sf(aes(fill = UAAXK0000_ORG))
p33 = ggplot(himg33) + geom_sf(aes(fill = UAAXK0000_ORG))
p31 + p32 + p33

# Create multi-resolution grid of UAA, based on survey data,
# with and without applying reliability check
# This is a relatively slow functionality
# rounding is set to FALSE, to be better able to visualize the few records
# (Not recommended for data to be published)
himg4 = multiResGrid(fsl,  vars = c("UAA"), weights = "EXT_MODULE", ifg = fsg,
                     strat = "STRA_ID_CORE", checkReliability = FALSE, rounding = FALSE)
# The parameter reliabilitySplit = 15 will divide the data set in 15 groups for the
# reliabilityCheck.
# A lower value would be recommended, but a high value speeds up the computation for this example
himg5 = multiResGrid(fsl,  vars = c("UAA"), weights = "EXT_MODULE", ifg = fsg,
                      strat = "STRA_ID_CORE", checkReliability = TRUE,
                      reliabilitySplit = TRUE, rounding = FALSE, pseudoreg = "REGIONS")

# Apply suppreslim to suppress insignificant grid cells
# Show intermediate maps of confidential cells (wait 5 seconds)
pint = ifelse(interactive(), 5, FALSE)
#himg11 = multiResGrid(ifl, vars = "UAA", ifg = ifg,
#                   suppresslim = 0, plotIntermediate = pint)
himg11 = himg1
himg12 = multiResGrid(ifl, vars = "UAA", ifg = ifg,
                  suppresslim = 0.02, plotIntermediate = pint)
himg13 = multiResGrid(ifl, vars = "UAA", ifg = ifg,
                  suppresslim = 0.05, plotIntermediate = pint)
himg14 = multiResGrid(ifl, vars = "UAA", ifg = ifg,
                  suppresslim = 0.1, plotIntermediate = pint)


 # This is an example of a userfun that can be used for alternative restrictions
 # for a grid cell. This particular toy example assures that there are at least
 # \code{nabove} records with a value (UAA in this case) above a certain "limit".
 ufun = function(df, nabove, limit) {
   sum(df$gridvar > limit) < nabove
 }

himg6 = multiResGrid(ifl, vars = "UAA", ifg = ifg,
            suppresslim = 0.2, plotIntermediate = pint, userfun = ufun, nabove = 5, limit = 10)



if (useBorder) himg00 = st_intersection(dkb, himg0) else himg00 = himg0
p00 = ggplot() + geom_sf(data = himg00, aes(fill = count, color = count)) +
  scale_fill_viridis( name = "number of farms", trans = "log10") +
  scale_color_viridis( name = "number of farms", trans = "log10") +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("Number of farms for variable grid cell size, only frequency confidentiality") +
```

```
  theme_bw()
if (useBorder) p00 = p00 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)
p00

if (useBorder) himg01 = st_intersection(dkb, himg1) else himg01 = himg1
p01 = ggplot() + geom_sf(data = himg01, aes(fill = count, color = count)) +
  scale_fill_viridis( name = "number of farms", trans = "log10") +
  scale_color_viridis( name = "number of farms", trans = "log10") +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
 ggtitle("Number of farms for variable grid cell size, frequency and dominance confidentiality") +
  theme_bw()
if (useBorder) p01 = p01 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)
p01

# Plot the density of organic agriculture, as hectares per square km
if (useBorder)himg02 = st_intersection(dkb, himg2) else himg02 = himg2
himg02$orgarea = himg02$UAAXK0000_ORG/units::set_units(st_area(himg02), "km^2")
units(himg02$orgarea) = NULL
p02 = ggplot() + geom_sf(data = himg02, aes(fill = orgarea), lwd = 0) +
  scale_fill_viridis( name = "ha / km2") +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("Organic UAA density")  +
  theme_bw()
if (useBorder) p02 = p02 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)
p02

# Plot the relative abundance of organic UAA relative to total UAA
if (useBorder) himg03 = st_intersection(dkb, himg3) else himg03 = himg3
himg03$ouaashare = himg03$UAAXK0000_ORG/himg03$UAA*100
p03 = ggplot() + geom_sf(data = himg03, aes(fill = ouaashare), lwd = 0) +
  scale_fill_viridis( name = "% Organic") +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("Organic share")  +
  theme_bw()
if (useBorder) p03 = p03 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)
p03


# Plot maps from survey data before and after adding the reliability constraint
# The percentage of UAA can be above 100% due to farm area being registered at the location
# of the administration building, but the map without reliability check has too high values
# for too many cells

if (useBorder) himg04 = st_intersection(dkb, himg4) else himg04 = himg4
himg04$area = st_area(himg04)/1e6
units(himg04$area) = NULL
himg04$uaashare = himg04$UAA/himg04$area
himg04$uaashare[himg04$uaashare > 1000] = 1000
p04 = ggplot() + geom_sf(data = himg04, aes(fill = uaashare), lwd = 0) +
  scale_fill_viridis( name = "% UAA",  trans = "log10", limits = c(1,1000)) +
  geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("UAA share (sample without reliability check)")  +
```

```
  theme_bw()
if (useBorder) p04 = p04 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)
p04

if (useBorder) himg05 = st_intersection(dkb, himg5) else himg05 = himg5
himg05$area = st_area(himg05)/1e6
units(himg05$area) = NULL
himg05$uaashare = himg05$UAA/himg05$area
himg05$uaashare[himg05$uaashare > 1000] = 1000
p05 = ggplot() + geom_sf(data = himg05, aes(fill = uaashare), lwd = 0) +
  scale_fill_viridis( name = "% UAA",  trans = "log10", limits = c(1,1000)) +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("UAA share (sample with reliability check)")  +
  theme_bw()
if (useBorder) p05 = p05 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)

if (require(patchwork)) p04 + p05 + plot_layout(guides = "collect")

if (useBorder) himg06 = st_intersection(dkb, himg6) else himg06 = himg6
p06 = ggplot() + geom_sf(data = himg06, aes(fill = UAA), lwd = 0) +
  scale_fill_viridis( name = "ha") +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("UAA, with additional user defined function")  +
  theme_bw()
if (useBorder) p06 = p06 + geom_sf(data = dkb, fill = NA, colour='black', lwd = 1)
p06


# Plot the different maps from using different suppreslim values
himgs = list(himg11, himg12, himg13, himg14)
slims = c(0, 0.02, 0.05, 0.1, 0.2)
plots = list()
uaas = c(himg11$UAA, himg12$UAA, himg13$UAA, himg14$UAA)
lims = range(uaas[uaas > 0], na.rm = TRUE)
for (ii in 1:4) {
  if (useBorder) himg = st_intersection(dkb, himgs[[ii]]) else himg = himgs[[ii]]
  plots[[ii]] =
   ggplot() + geom_sf(data = himg, aes(fill = UAA), lwd = 0) +
   scale_fill_viridis( name = "UAA (ha)", trans = "log10", limits = lims, na.value="red") +
    ggtitle(paste("Suppresslim = ", slims[[ii]])) +
    xlab("") + ylab("") +
    theme_bw()
   if (useBorder) plots[[ii]] = plots[[ii]] +
                     geom_sf(data = dkb, fill = NA, colour='black', lwd = 0.5)
}

if (require(patchwork)) plots[[1]]  + plots[[2]] + plots[[3]]  + plots[[4]] +
                               plot_layout(guides = "collect")



#' @rdname multiResGrid
```

`remSmall` *Function that will move values from grid cells with small values to the ones with larger values for disclosure control reasons*

Description

Two main confidentiality rules are considered: - Threshold rule (suppression due to a minimum number of counts) - Dominance rule (suppression due to dominance by one or more units)

Usage

```
remSmall(
  gdl,
  ress,
  ires0,
  mincount = 10,
  ifg,
  var,
  weight,
  nlarge = 2,
  plim = 0.85,
  sampleRandom = TRUE,
  domEstat = TRUE,
  verbose = FALSE,
  nclus = 1,
  clusType,
  outfile = NULL,
  checkDominance = TRUE,
  checkReliability = TRUE
)
```

Arguments

`gdl`	A list of gridded data with different resolutions (from a call to `gridData`
`ress`	A vector with the different resolutions
`ires0`	Which resolution level to use as base for the downscaling
`mincount`	The minimum number of farms for a grid cell (threshold rule)
`ifg`	Either a data.frame or tibble or sf-object with the locations and the data of the survey or census data, or a list of such objects.
`var`	Variable of interest that should be aggregated (necessary when ifg is used for individual farm specific confidence rules)
`weight`	Extrapolation factor (weight) wi of unit i in the sample of units nc falling into a specific cell c. Weights are used for disclosure control measures.
`nlarge`	Parameter to be used if the nlarge(st) farms should count for maximum plim percent of the total value for the variable in the grid cell (see details of `gridData`)

plim	See nlarge
sampleRandom	Logical; if the value is TRUE, values from grid cells with values under the limit will be moved to a random neighbour if there are more neighbours above the limit. False will always pick the largest (and the first one in the list if they are equal)
domEstat	Should the dominance rule be applied as in the IFS handbook (TRUE), where the weights are rounded before finding the first nlarge contributors, or should it be the first nlarge contributors*weight, where also fractions are considered (FALSE)?
verbose	Indicates if some extra output should be printed. Usually TRUE/FALSE, but can also have a value of 2 for multiResGrid for even more output.
nclus	Number of clusters to use for parallel processing. No parallelization is used for nclus = 1.
clusType	The type of cluster; see makeCluster for more details. The default of makeCluster is used if type is missing or NA
outfile	File to direct the output in case of parallel processing, see makeCluster for more details.
checkDominance	Logical - should the dominance rule be applied?
checkReliability	Logical - should the prediction variance be checked, and used for the aggregation? This considerably increases computation time

Details

This function uses the hierarchy of gridded data to associate values from grid cells that need to be anonymized to the grid cell with the highest values, within increasingly larger sub-grids.

The parameters nlarge and plim are used for setting value dependent confidentiality rules. If the rule is that the largest two holdings in a grid cell should not count for more than 85 of the total value (UAA, number of livestock, ...), then nlarge = 2 and plim = 0.85

The function will create set the value to NA for the grid cells where the content has been moved to a neighbouring grid cells.

Value

A gridded data set, where each grid cell respects the confidentiality rules.

Examples

```
library(sf)
library(sf)
if (!require(ggplot2)) print("Plotting of results will not work without installation of ggplot2")
if (!require(viridis)) print("Some of the plots will not work without installation of ggplot2")
if (!require(patchwork)) print("Some of the plots will not work without installation of patchwork")

if (require(giscoR)) {
  useBorder = TRUE
} else {
```

```
  useBorder = FALSE
  print("You need to install giscoR for plotting borders and clipping the gridded maps")
}
# These are SYNTHETIC agricultural FSS data
data(ifs_dk) # Census data
ifs_weight = ifs_dk %>% dplyr::filter(Sample == 1) # Extract weighted subsample

# Create spatial data
ifg = fssgeo(ifs_dk, locAdj = "LL")
fsg = fssgeo(ifs_weight, locAdj = "LL")

if (useBorder) {
# Read country borders, only used for plotting
  borders = gisco_get_nuts(nuts_level = 0)
  dkb = borders[borders$CNTR_CODE == "DK",] %>% st_transform(crs = 3035)
}

# Set the base resolutions, and create a hierarchical list with gridded data
ress = c(1,5,10,20,40,80, 160, 320, 640, 1280, 2560)*1000
# Create the grid with UAA as variable and EXT_CORE as weight
# These can be dropped if only the number of farms are of interest in the analyses
ifl = gridData(ifg, "UAA", weight = "EXT_CORE", res = ress)

# Run the procedure for the third resolution level (10 km), only using number of holdings
# as confidentiality rule
# himg1 and himg2 should give the same result, but only when sampleRandom = FALSE
himg1 <- remSmall(ifl, ress, 3, sampleRandom = FALSE)
plot(himg1[, "count"])
himg12 <- remSmall(ifl, ress, 3, sampleRandom = FALSE, nclus = 2)
# Run the procedure for UAA, using the defaults for variable
# confidentiality rule (nlarge = 2 and plim = 0.85)

himg2 <- remSmall(ifl, ress, weight = "EXT_CORE", ires0 = 3, var = "UAA", ifg = ifg)
plot(himg2[, "count"])
plot(himg2[, "UAA"])

# Run the procedure for organic UAA, but still requiring 10 holdings of any kind per grid cell
# Using resolution level 5 (40 km)
 iflOuaaAll = gridData(ifg, "UAAXK0000_ORG", res = ress)
himg3 = remSmall(iflOuaaAll, ress, 5, ifg = ifg, var = "UAAXK0000_ORG")
plot(himg3[, "count"])
plot(himg3[, "UAAXK0000_ORG"])

# Run the procedure for organic UAA, but require at least 10 organic holdings per grid cell
# Using resolution level 5 (40 km)
ifgOuaa = ifg[ifg$UAAXK0000_ORG > 0, ]
iflOuaa = list()
iflOuaa = gridData(ifgOuaa, "UAAXK0000_ORG",  res = ress)
himg4 = remSmall(iflOuaa, ress, 5, ifg = ifg, var = "UAAXK0000_ORG")
plot(himg4[, "count"])
plot(himg4[, "UAAXK0000_ORG"])

himg4l = list()
```

```
# Run the proceduure for organic UAA for different resolution levels
for (ipl in 1:6) himg4l[[ipl]] = remSmall(iflOuaa, ress, ipl, ifg = ifg, var = "UAAXK0000_ORG")


 # Create proper plots
  breaks = c(1,3,10,30,100)
  labels = breaks
  p1 = ggplot() + geom_sf(data = himg1, aes(fill = count, color = count)) +
  scale_fill_viridis( name = "number of \nholdings", trans = "log10",
                  breaks = breaks, labels = labels, limits = c(1,100)) +
  scale_color_viridis( name = "number of \nholdings", trans = "log10",
                 breaks = breaks, labels = labels, limits = c(1,100)) +
  geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("Number of holdings after swapping") +
  theme_bw()

# For comparison the number of organic farms and organic UAA, without taking any
# confidentiality into account
gcompOfarms = ggplot() + geom_sf(data = ifl[[3]], aes(fill = count, color = count)) +
 scale_fill_viridis( name = "number of \nholdings", trans = "log10",
                         breaks = breaks, labels = labels, limits = c(1,100)) +
 scale_color_viridis( name = "number of \nholdings", trans = "log10",
                         breaks = breaks, labels = labels, limits = c(1,100)) +
 geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
 coord_sf(crs = 3035) +
 ggtitle("Number of holdings - ordinary gridded data") +
 theme_bw()

  gcompOfarms + p1 + plot_layout(guides = "collect")

  p2 = ggplot() + geom_sf(data = himg2, aes(fill = count, color = count)) +
  scale_fill_viridis( name = "number of \nholdings", trans = "log10") +
  scale_color_viridis( name = "number of \nholdings", trans = "log10") +
  geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("Number of farms - corrected for farm size") +
  theme_bw()

  p3 = ggplot() + geom_sf(data = himg2, aes(fill = UAA, color = UAA)) +
  scale_fill_viridis( name = "UAA", trans = "log10") +
  scale_color_viridis( name = "UAA", trans = "log10") +
  geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("UAA - corrected for farm size") +
  theme_bw()

  p4 = ggplot() + geom_sf(data = himg3, aes(fill = count, color = count)) +
  scale_fill_viridis( name = "number of \nholdings", trans = "log10") +
  scale_color_viridis( name = "number of \nholdings", trans = "log10") +
  geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
  coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
  ggtitle("Number of farms - based on number of organic farms and organic farm size") +
```

```
 theme_bw()

p5 = ggplot() + geom_sf(data = himg3, aes(fill = UAAXK0000_ORG, color = UAAXK0000_ORG)) +
 scale_fill_viridis( name = "UAA organic", trans = "log10") +
 scale_color_viridis( name = "UAA organic", trans = "log10") +
 geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
 coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
 ggtitle("UAA organic - based on organic farm numbers and size") +
 theme_bw()

 p6 = ggplot() + geom_sf(data = himg4, aes(fill = count, color = count)) +
 scale_fill_viridis( name = "number of \nholdings", trans = "log10") +
 scale_color_viridis( name = "number of \nholdings", trans = "log10") +
 geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
 coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
 ggtitle("Number of organic farms - based on organic farm numbers and size") +
 theme_bw()

 uaalims = c(min(c(himg4$UAAXK0000_ORG, iflOuaa[[5]]$UAAXK0000_ORG), na.rm = TRUE),
             max(c(himg4$UAAXK0000_ORG, iflOuaa[[5]]$UAAXK0000_ORG), na.rm = TRUE))
 p7 = ggplot() + geom_sf(data = himg4, aes(fill = UAAXK0000_ORG, color = UAAXK0000_ORG)) +
 scale_fill_viridis( name = "UAA organic", trans = "log10", limits = uaalims) +
 scale_color_viridis( name = "UAA organic", trans = "log10", limits = uaalims) +
 geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
 coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
 ggtitle("UAA organic after swapping  ") +
 theme_bw()

# For comparison the number of organic farms and organic UAA, without taking any
# confidentiality into account

gcompOUAA = ggplot() + geom_sf(data = iflOuaa[[5]],
                  aes(fill = UAAXK0000_ORG, color = UAAXK0000_ORG)) +
 scale_fill_viridis( name = "UAA organic", trans = "log10", limits = uaalims) +
 scale_color_viridis( name = "UAA organic", trans = "log10", limits = uaalims) +
 geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
 coord_sf(crs = 3035) +
 ggtitle("Organic UAA - ordinary gridded data") +
 theme_bw()

 print(gcompOUAA) + p7 +  plot_layout(guides = "collect")

 ppl = list()
 counts = do.call("rbind", himg4l[1:5])$count
 clim = c(min(counts, na.rm = TRUE), max(counts, na.rm = TRUE))
 for (ipl in 1:length(himg4l)) {
  ppl[[ipl]] = ggplot() + geom_sf(data = himg4l[[ipl]], aes(fill = count, color = count)) +
 scale_fill_viridis( name = "number of \nholdings", trans = "log10", limits = clim) +
 scale_color_viridis( name = "number of \nholdings", trans = "log10", limits = clim) +
 geom_sf(data = dkb, fill = NA, colour='black', lwd = 1) +
 coord_sf(crs = 3035) +#, xlim = c(2377294, 6400000), ylim = c(1313597, 5628510)) +
 ggtitle(paste("Base resolution", ress[ipl]/1000, "km")) +
 theme_bw()
```

```
  }
  ppl[[1]] + ppl[[2]] + ppl[[3]] + ppl[[4]] + plot_layout(guides = "collect")

MRGcluster(action = "stop")
```

Index