Package 'lczexplore'

November 2, 2026

Title Imports, Visualizes, Compares and Explores Maps of Local Climate Zones

Version 0.1.2

Description Easily import and visualize local climate zone maps with standard encoding.
Compare two or more maps, and group some LCZ types according to user defined grouping vectors.
Explore their differences through circular visualization and indicators. For more Details see Gousseff et al (2023) ``lczexplore: an R package to explore Local Climate Zone classifications" <doi:10.21105/joss.05445>.

License LGPL (>= 3)

Encoding UTF-8

Imports circlize, collapse, data.table, dplyr, forcats, ggnewscale, ggplot2, grDevices, ggspatial, ggsankeyfier, jsonlite, magrittr, methods, patchwork, randomcoloR, RColorBrewer, sf, shades, terra, tidyr, units

Suggests knitr, markdown, rmarkdown, tinytest, png

VignetteBuilder knitr

LazyData true

Depends R (>= 4.0.0)

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Matthieu Gousseff [aut, cre, cph] (affiliation: Centre National de la Recherche Scientifique, Lab-STICC, ORCID: <https://orcid.org/0000-0002-7106-2677>)

Maintainer Matthieu Gousseff <matthieu.gousseff@cnrs.fr>

Repository CRAN

Date/Publication 2026-11-02 16:40:02 UTC

Contents

addMissingRSUs 3

aggregateRSUsByLCZ 4
areColors 5
barplotLCZaLocation 6
barplotLCZfromIntersect 7
checkColnameCase 8
checkDirSlash 9
checkWorkflowName 9
CohenKappa 10
compareLCZ 11
compareMultipleLCZ 14
completeDT 15
computeAgreements 16
computeConsensus 16
concatAllLocsWorkflows 17
concatAlocationWorkflows 18
concatIntersectedLocations 19
confidSensib 20
createIntersect 21
createWeightedFlux 23
drawChordDiagram 24
drawChordDiagramWithGrouping 25
drawSectors 27
fetchLCZ 27
geoClimateCall 28
geoClimateConfigFile 29
groupLCZ 31
groupLCZsuffix 32
importConcatMultipleLocsLCZvect 33
importLCZgc 34
importLCZraster 35
importLCZvect 36
importLCZvectFromFile 38
importLCZvectFromSf 39
importMultipleLCZvect 40
importMultipleLCZvectFromSession 41
importMultipleLocsLCZvect 42
importQualVar 43
intersectAlocation 44
LCZareas 45
LCZlevelToOrderedString 46
levCol 46
makeSectorsAndGroups 47
matConfLCZ 48
matConfLCZGlob 49
matConfPlot 51
plotSankeyfiedLCZ 52
plotSummarisedRSUs 53
prepareSankeyLCZ 54

redonBDT 55
redonOSM 55
showAlterLCZ 56
showLCZ 57
showStandardLCZ 59
standLevCol 60
summariseRSUs 61
workflowAgreeAreas 62

Index **63**

`addMissingRSUs` *adds missing Reference Spatial Units (geometries) from the difference with a reference zone file, and possibly add a value to the lcz column according to a reference workflow reference spatial units nor represented by their own geometries adding a column for location and workflow names*

Description

adds missing Reference Spatial Units (geometries) from the difference with a reference zone file, and possibly add a value to the lcz column according to a reference workflow reference spatial units nor represented by their own geometries adding a column for location and workflow names

Usage

```
addMissingRSUs(
  sfList,
  missingGeomsWf = "osm",
  zoneSf,
  refWf = "bdt",
  refLCZ = "107",
  residualLCZvalue = "105",
  column = "lcz_primary"
)
```

Arguments

`sfList`	the list of LCZ sf objects
`missingGeomsWf`	is the name of the column where some geometries are missing
`zoneSf`	is an sf file containing one geometry defining the zone where some geometries must be added
`refWf`	in case one wants to give the added geometries the value of a workflow of reference,
`refLCZ`	the LCZ type to report from the reference workflow to the missing geometries

residualLCZvalue
: the LCZ type to add when the reference workflow has a different value than refLCZ for the added geometries

column
: is the name of the LCZ column (to be explained)

Value

returns graphics of comparison and an object called matConfOut which contains : matConfLong, a confusion matrix in a longer form, matConfPlot is a ggplot2 object showing the confusion matrix. percAgg is the general agreement between the two sets of LCZ, expressed as a percentage of the total area of the study zone pseudoK is a heuristic estimate of a Cohen's kappa coefficient of agreement between classifications If saveG is not an empty string, graphics are saved under "saveG.png"

Examples

```
sfList<-importMultipleLCZvect(dirPath = paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
 workflowNames = c("osm","bdt","wudapt"), location = "Arville"  )
zoneSf <- sf::read_sf(
  paste0(
   system.file("extdata", package = "lczexplore"),
   "/multipleWfs/Arville/zone.fgb")
)
sfList<-addMissingRSUs(
sfList, zoneSf = zoneSf, refWf = NULL,
 refLCZ = "Unclassified", residualLCZvalue = "Unclassified")
```

---

aggregateRSUsByLCZ *For an sf object containing LCZ types, aggregates touching geometries with same value of LCZ*

Description

For an sf object containing LCZ types, aggregates touching geometries with same value of LCZ

Usage

```
aggregateRSUsByLCZ(
  sf,
  aggregateBufferSize = 0,
  column,
  wfColumn,
  locationColumn = "location"
)
```

Arguments

sf	contains the geometry and LCZ levels
aggregateBufferSize	is the size of a buffer you can add to the geometries before agregation. Useful when you suspect numeric precision to create false disjunction of geometries.
column	the name of the column containing LCZ types
wfColumn	is the column where the workflow used to produce LCZ classif is precised.
locationColumn	is the name of the column where the location is stored

Value

an sf object containing the agregated geometries, their LCZ types and the wf and location columns if present in the initial sf

Examples

```
dirPath<-paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs")
allLocAllWfs<-importConcatMultipleLocsLCZvect(
 dirPath = dirPath, locations = c("Redon", "Arville"),
workflowNames = c("osm","bdt","wudapt"),
 missingGeomsWf= "osm",
 refWf = NULL,
 refLCZ = "Unclassified",
 residualLCZvalue = "Unclassified",
 column = "lcz_primary"
)
ASUallLocAllWfs <- aggregateRSUsByLCZ(
allLocAllWfs,
column = "lcz_primary", wfColumn = "wf", locationColumn = "location", aggregateBufferSize = 0.5)
```

---

areColors *Checks if a string or a vector of strings define colors in R*

Description

Checks if a string or a vector of strings define colors in R

Usage

```
areColors(x)
```

Arguments

x	is the input string

Value

a vector of booleans indicting if the elements of x define a color in R (TRUE) or don't (FALSE)

Examples

```
areColors(c(NA, "black", "blackk", "1", "#00", "#000000"))
```

`barplotLCZaLocation` *in a given directory, if several LCZ files are present, plots the repartition of LCZ regarding of their source (workflow) NOTE: to represent the map of LCZ for a given file, use* `showLCZ` *function instead*

Description

in a given directory, if several LCZ files are present, plots the repartition of LCZ regarding of their source (workflow) NOTE: to represent the map of LCZ for a given file, use `showLCZ` function instead

Usage

```
barplotLCZaLocation(
  dirPath,
  location,
  workflowNames = c("osm", "bdt", "wudapt"),
  refWf = NULL,
  refLCZ = NA,
  residualLCZvalue = NA,
  missingGeom = "osm",
  plotNow = FALSE,
  plotSave = "/tmp"
)
```

Arguments

`dirPath`	is the path where the datasets are stored
`location`	is the name of the locations for the plot is produced
`workflowNames`	is a vector of prefixes. The LCZ files must be named workflow_rsu.fgb where workflow is on of the values in workflowNames vector
`refWf`	is a reference workflow name, passed to the function addMissingRSUs when needed
`refLCZ`	is a reference LCZtype, passed to the function addMissingRSUs when needed
`residualLCZvalue`	a LCZ default type, passed to the function addMissingRSUs when needed
`missingGeom`	allows to indicate which workflow is supposed to have missing geometries (not used most of the time)
`plotNow`	If TRUE, the boxplot of the repartition will be printed
`plotSave`	If TRUE, the plot will be saved in the directory pointed by dirPath

Value

A barplot of LCZ area percentage by LCZ type and workflow

Examples

```
barplotLCZaLocation(
dirPath = paste0(system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
refWf = NULL, refLCZ = NA, residualLCZvalue = "Unclassified",
location = "Arville", plotSave = FALSE, plotNow = TRUE)
```

---

barplotLCZfromIntersect

plots the repartition of LCZ types regarding of their source (map/workflow) from an sf object containing all spatial units and LCZ value, typically the output of createIntersect

---

Description

plots the repartition of LCZ types regarding of their source (map/workflow) from an sf object containing all spatial units and LCZ value, typically the output of createIntersect

Usage

```
barplotLCZfromIntersect(
  sfInt,
  workflowNames = NULL,
  columns = NULL,
  stat = "perc",
  plotNow = TRUE
)
```

Arguments

sfInt	is the file containing the spatial units and LCZ type per workflow
workflowNames	is a vector of prefixes. The LCZ files must be named workflow_rsu.fgb where workflow is on of the values in workflowNames vector
columns	is the vector of names of the columns containing the LCZ types. If not specified, the function will try to use the workflowNames instead
stat	can take the value "perc" to plot percentage of area for each LCZ type or "sum" for total area for each LCZ type
plotNow	If TRUE, the boxplot of the repartition will be printed

Value

A barplot of LCZ area percentage by LCZ type and workflow

Examples

```
# Two Locations
sfList2<-importMultipleLocsLCZvect(
 dirPath = paste0(
     system.file("extdata", package = "lczexplore"),
     "/multipleWfs"),
 workflowNames = c("osm","bdt","wudapt"))

twoLocsIntersect <- createIntersect(
 sfList = sfList2, columns = rep("lcz_primary", 3),
 workflowNames = c("osm","bdt","wudapt"))
example<-barplotLCZfromIntersect(sfInt = twoLocsIntersect,
                                 columns = c("osm", "bdt", "wudapt"),
                               workflowNames = c("osm", "bdt", "wudapt"))
```

`checkColnameCase` *Checks if the columns entered by the user match the columns in the data source*

Description

Checks if the columns entered by the user match the columns in the data source

Usage

```
checkColnameCase(userColNames, dataColNames)
```

Arguments

`userColNames`	a vector of column names entered by the user
`dataColNames`	a vector names presents in a dataset

Value

returns an error if on or several colum names in userColNames are not present in dataColNames, except if they are present but in lower case, then the correspondent lower case name is substituted

Examples

```
userColNames<-c("ID", "LCZ_PRIMARY")
dataColNames<-c("ID", "LCZ_PRIMARY")
checkColnameCase(userColNames, dataColNames)
dataColNames<-c("ID", "lcz_primary")
```

checkDirSlash *Simply adds a slash at then end of a directory path if there is none*

Description

Simply adds a slash at then end of a directory path if there is none

Usage

```
checkDirSlash(dirPath)
```

Arguments

dirPath	is a directory pas as a string

Value

the same string if it ended with a slash, else the string with a slash at its end

Examples

```
dirPathTest<-"test"
checkDirSlash(dirPathTest)
```

checkWorkflowName *Checks if a character string can be used as a workflow name and corrects it if it contains spaces or underscores*

Description

Checks if a character string can be used as a workflow name and corrects it if it contains spaces or underscores

Usage

```
checkWorkflowName(wfName)
```

Arguments

wfName	contains the character string to

Value

the character string if suitable, or a modified one to comply to what is expected for a workflow name

Examples

```
checkWorkflowName("BDT UTRF_test")
```

CohenKappa *Compute Cohen's Kappa Coefficient*

Description

Compute Cohen's Kappa Coefficient

Usage

```
CohenKappa(
  x,
  y = NULL,
  weights = c("Unweighted", "Equal-Spacing", "Fleiss-Cohen"),
  conf.level = NA,
  ...
)
```

Arguments

x	can either be a numeric vector or a confusion matrix. In the latter case x must be a square matrix, in lczexplore, will take the matrix of agreement weighted by area for all intersected geometries
y	not used here, but allows to compute cross table of x and y as an entry
weights	either one out of "Unweighted" (default), "Equal-Spacing", "Fleiss-Cohen", which will calculate the weights accordingly, or a user-specified matrix having the same dimensions as x containing the weights for each cell.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.
...	further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Taken from descTools who based it on Kappa from library(vcd) author: David Meyer see also: kappa in library(psych) Integrated here to reduce dependecy to descTools as it is the only function from DescTools we used Computes the agreement rates Cohen's kappa and weighted kappa and their confidence intervals.

Cohen's kappa is the diagonal sum of the (possibly weighted) relative frequencies, corrected for expected values and standardized by its maximum value.
The equal-spacing weights (see Cicchetti and Allison 1971) are defined by

$$1 - \frac{|i-j|}{r-1}$$

r being the number of columns/rows, and the Fleiss-Cohen weights by

$$1 - \frac{(i-j)^2}{(r-1)^2}$$

The latter attaches greater importance to closer disagreements

Value

the value of this pseudoKappa

Author(s)

David Meyer david.meyer@r-project.org, some changes and tweaks Andri Signorell andri@signorell.net and integrated for areas by Matthieu Gousseff

References

Cohen, J. (1960) A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20, 37-46. Everitt, B.S. (1968), Moments of statistics kappa and weighted kappa. *The British Journal of Mathematical and Statistical Psychology*, 21, 97-103. Fleiss, J.L., Cohen, J., and Everitt, B.S. (1969), Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, 72, 332-327. Cicchetti, D.V., Allison, T. (1971) A New Procedure for Assessing Reliability of Scoring EEG Sleep Recordings *American Journal of EEG Technology*, 11, 101-109.

Examples

```
# from Bortz et. al (1990) Verteilungsfreie Methoden in der Biostatistik, Springer, pp. 459
m <- matrix(c(53,  5, 2,
              11, 14, 5,
              1,  6, 3), nrow=3, byrow=TRUE,
            dimnames = list(rater1 = c("V","N","P"), rater2 = c("V","N","P")) )
# confusion matrix interface
CohenKappa(m, weight="Unweighted")
```

`compareLCZ` *Compares two sets of geographical classifications, especially Local Climate Zones classifications*

Description

Compares two sets of geographical classifications, especially Local Climate Zones classifications

Usage

```
compareLCZ(
  sf1,
  geomID1 = "",
  column1 = "LCZ_PRIMARY",
  confid1 = "",
  wf1 = NULL,
  sf2,
  column2 = "LCZ_PRIMARY",
  geomID2 = "",
  confid2 = "",
  wf2 = NULL,
```

```
  ref = 1,
  repr = "standard",
  saveG = "",
  exwrite = FALSE,
  outDir = getwd(),
  location = "Your Place",
  plotNow = TRUE,
  confPlot = "matrix",
  tryGroup = FALSE,
  minZeroArea = 0,
  ...
)
```

Arguments

sf1	is the sf object that contains the first (LCZ) classification
geomID1	is the name of the optionnal column storing the ID of the geoms in sf1 By defautl it is set to an empty string and no ID is loaded.
column1	is the column of sf1 that contains the LCZ classification for each geom of sf1.
confid1	is an optionnal column that contains an indicator of confidence of the values in column1, e.g. a uniqueness value, or a probability of belonging to the class... By defautl it is set to an empty string and no confidence indicator is loaded.
wf1	is the workflow used to produce the first LCZ classification. It is used ti create titles or legends. When GeoClimate was used with BD_TOPO V2 data as input, use "bdtopo_2_2". When GeoClimate was used with Open Street Map data as input, use "osm". When the LCZ come from the wudapt Europe tiff, use "wudapt".
sf2	is the sf object that contains the second (LCZ) classification
column2	is the column of sf2 that contains the LCZ classification for each geom of sf2
geomID2	is the name of the column storing the ID of the geoms in sf2
confid2	is a column that contains an indicator of confidence
wf2	is the workflow used to produce the second LCZ classification.
ref	: If the coordinate reference systems (CRS) of sf1 and sf2 differ, ref indicates which CRS to choose for both files (1 or 2)
repr	"standard" means that standard values of LCZ are expected, "alter" means other values are expected, like grouped values of LCZ or other qualitative variable. In the latter case, the ... arguments can contain the expected levels and a color vector.
saveG	: when an empty character string, "", the plots are not saved. Else, the saveG string is used to produce the name of the saved png file.
exwrite	: when TRUE, the values of the LCZ on the intersected geoms are written down in a csv file
outDir	: when exwrite equals TRUE, outDir is the path to the folder where one wants to write the csv file containing the values of the LCZ on the intersected geoms

location : the name of the study area, as chosen as the name of the directory on the GeoClimate team cloud. If the area you wish to analyse is not uploaded yet, please contact the GeoClimate Team.

plotNow : when FALSE none of the graphics are plotted or saved

confPlot default is "matrix", to plot the confusion matrix. "sankey" allows to replaace it with a sankey diagram.

tryGroup : when TRUE, one can group and compare on-the-fly : if the specified level names don't match the data, but the specified levels do, a call to the groupLCZ function will be tried, and if it works, the resulting grouping columns wille be compared

minZeroArea allows you to specify a minimal area under which intersected spatial units will be discarded. It avoids numerical error and can save computing time.

... allow to pass optionnal arguments if repr is set to alter. The expected arguments are the name of each level of the variables contained in column1 and column2, and also a vector called colors.

Details

compareLCZ produces a map for each classification, a map of their agreement (and a pseudo Kappa coefficent), and a confusion matrix between them. All are stored in a list, easily reusable

Value

returns graphics of comparison and an object called matConfOut which contains : matConfLong, a confusion matrix in a longer form, matConfPlot is a ggplot2 object showing the confusion matrix. percAgg is the general agreement between the two sets of LCZ, expressed as a percentage of the total area of the study zone pseudoK is a heuristic estimate of a Cohen's kappa coefficient of agreement between classifications If saveG is not an empty string, graphics are saved under "saveG.png"

Examples

```
comparisonBDT_OSM<-compareLCZ(sf1=redonBDT, column1="LCZ_PRIMARY", geomID1 = "ID_RSU",
confid1="LCZ_UNIQUENESS_VALUE", wf1="bdtopo_2_2",
sf2=redonOSM, column2="LCZ_PRIMARY", geomID2 = "ID_RSU",
confid2="LCZ_UNIQUENESS_VALUE", wf2="osm",
repr="standard", saveG="", exwrite=FALSE, location="Redon", plotNow = TRUE)
# To get the summed area of each LCZ levels for both dataset :
comparisonBDT_OSM$areas
# The plots of each dataset can be produced with the showLCZ function.
# To get the value of the general percentage of agreement call :
comparisonBDT_OSM$percAgg
# The values of that confusion matrix is available in a wide form:
comparisonBDT_OSM$matConfLarge
# or in a long form :
comparisonBDT_OSM$matConf
 # Examples for non LCZ variables are available in the importQualVar function examples.
```

`compareMultipleLCZ` *Compares several sets of geographical classifications, especially Local Climate Zones classifications*

Description

Compares several sets of geographical classifications, especially Local Climate Zones classifications

Usage

```
compareMultipleLCZ(
  sfInt,
  columns,
  workflowNames = NULL,
  trimPerc = 0,
  labelMatch = NULL,
  ...
)
```

Arguments

`sfInt`	an sf objects with intersected geometries and the LCZ columns for each workflow LCZ
`columns`	a vector which contains, the name of the columns of the classification to compare
`workflowNames`	a vector of strings which contains the names of the workflows used to produce the sf objects
`trimPerc`	this parameters indicates which percentile to drop out of the smallest geometries resulting It allows to account for numeric precision errors and to speed up computations at the cost of not considering the smallest geometries.
`labelMatch`	feed matching between levels of the data and some labels from the intersection of the original sf geometries intersection.
`...`	areguments to be passed for groupint, in the form of groupLevel = levels to group to this group level, and a possible groupColors named vectors, which names are the grouped levels and values are the desired colors.

Value

returns graphics of comparison and an object called matConfOut which contains : matConfLong, a confusion matrix in a long form (each line has the percentage of LCZ type i from workflow j which is classified in LCZ type i' by workflow j'), matConfPlot is a ggplot2 object showing the confusion matrix. percAgg is the general agreement between the two sets of LCZ, expressed as a percentage of the total area of the study zone If saveG is not an empty string, graphics are saved under "saveG.png" agreements a dtaframe with the pairs of workflows, areas on which they agree, disagree, and the percentage of agreement

Examples

```
sfList<-importMultipleLCZvect(dirPath =
paste0(system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
workflowNames = c("osm","bdt","wudapt"), location = "Arville")
ArvilleIntersect <- createIntersect(
 sfList = sfList, columns = rep("lcz_primary", 4),
 workflowNames = c("osm","bdt","wudapt"))
ArvilleMultipleComparison<-compareMultipleLCZ(
 sfInt = ArvilleIntersect,
 columns = c("osm","bdt","wudapt"),
 trimPerc = 0.5)
```

`completeDT` *creates a complete data.table from the cross cases of the specified columns*

Description

creates a complete data.table from the cross cases of the specified columns

Usage

```
completeDT(DT, cols, defs = NULL)
```

Arguments

`DT`	is the data table you want to complete
`cols`	are the columns whose values will be crossed to complete the data.table
`defs`	a function name to apply to the crossed value

Value

the completed data table

Examples

```
# No example as this function is not to be called by the user, only needed for the package.
```

computeAgreements *compute agreements between several LCZ maps*

Description

compute agreements between several LCZ maps

Usage

```
computeAgreements(intersec_sf, nbWfs)
```

Arguments

intersec_sf	is a special sf or data.frame object whose lines correspond to intersected spatial units, the same for all workflows, and whose first columns contain LCZ values for the studied workflows, followed by a column containing the area of the spatial units
nbWfs	is the number of studied workflows (and therefore LCZ columns)

computeConsensus *Compute an indicator of consensus among workflows for each LCZ type*

Description

Compute an indicator of consensus among workflows for each LCZ type

Usage

```
computeConsensus(inDf, wfNames)
```

Arguments

inDf	a data frame, a data table or even an sf object, typically the output of the function createMultipleIntersect. It must contain values of LCZ types for different workflows in columns names after wf names, a column area which contains the area of each spatial unit
wfNames	a vector of strings containing the name of the workflows. The columns containing the LCZ type for each workflow bare the same names.

Value

Returns the values of consensus indicator per LCZ types

Examples

```
dirList <- list.dirs(paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs"), recursive = FALSE)
allLocIntersected<-concatIntersectedLocations(
dirList = dirList, locations = c("Arville", "Redon"))
consensus <- computeConsensus(inDf = allLocIntersected,
wfNames = c("bdt","osm", "wudapt"= "wud"))
```

`concatAllLocsWorkflows` *importConcatMultipleLocsLCZvect is recommended instead as it loadsand concatenate input files and can even fill missing geometries with "unclassified" type. But if maps are already loaded in a List, concatAllLocsWorkflows is the equivalent of concatenateAlocationWorkflows, but for several Locations Take a list of locations, each list contains a list of sf files with an lcz_primary column, and concatenates them into a single sf object, adding a column for location and workflow names*

Description

importConcatMultipleLocsLCZvect is recommended instead as it loadsand concatenate input files and can even fill missing geometries with "unclassified" type. But if maps are already loaded in a List, concatAllLocsWorkflows is the equivalent of concatenateAlocationWorkflows, but for several Locations Take a list of locations, each list contains a list of sf files with an lcz_primary column, and concatenates them into a single sf object, adding a column for location and workflow names

Usage

```
concatAllLocsWorkflows(sfList, refCrs = 1)
```

Arguments

`sfList`	contains a list of locations containing list of sf for each workflow
`refCrs`	a number telling which sf of the sfList will be the reference in termes of Coordinate Reference System

Value

returns a single sf object containing all geometries for all locations and workflows, and the the lcz types in the column lcz_primary

Examples

```
sfListAll<-importMultipleLocsLCZvect(dirPath = paste0(
 system.file("extdata", package = "lczexplore"),"/multipleWfs/"),
                              workflowNames = c("osm","bdt","wudapt"),
                              location = c("Arville", "Redon"))
allLocsAllWfs <-  concatAllLocsWorkflows(
 sfList = sfListAll)
```

`concatAlocationWorkflows` *Take a list of sf files with lcz_primary and wf columns, and concatenates them in a single sf object, adding a column for location and workflow names*

Description

Take a list of sf files with lcz_primary and wf columns, and concatenates them in a single sf object, adding a column for location and workflow names

Usage

```
concatAlocationWorkflows(
  sfList,
  location = NA,
  refCrs = 1,
  columns = "lcz_primary"
)
```

Arguments

`sfList`	the list of LCZ sf objects
`location`	the name of the location at which all LCZ are created
`refCrs`	a number telling which sf of the sfList will be the reference in termes of Coordinate Reference System
`columns`	contains the names of the columns where LCZ types are stored. If it contains only one string, it will be repeated for all workflows

Value

returns an sf with values of LCZ for all workflows for a given location

Examples

```
sfList<-importMultipleLCZvect(dirPath = paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
workflowNames = c("osm","bdt","wudapt"), location = "Arville"  )
zoneSf <- sf::read_sf(
paste0(system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville/zone.fgb")
)
ArvilleAllWfs <-  concatAlocationWorkflows(
sfList = sfList, location = "Arville")
```

`concatIntersectedLocations`

In a given directory (or a list of directories) the function looks for LCZ datafiles, intersects them and return a dataset with intersected geometries and LCZ values for each workflow

---

Description

In a given directory (or a list of directories) the function looks for LCZ datafiles, intersects them and return a dataset with intersected geometries and LCZ values for each workflow

Usage

```
concatIntersectedLocations(
  dirList,
  workflowNames = c("osm", "bdt", "wudapt"),
  locations,
  columns = "lcz_primary"
)
```

Arguments

`dirList`	the list of directories for which the different LCZ files will be intersected
`workflowNames`	sets the names of workflows and define the name of the files which will be loaded and intersected
`locations`	: for each diretory from dirList, a location name must be fed to the function
`columns`	the name (string) of the column containing LCZ types. If the different workflows do no use the same column names, a vector of names is passed

Value

returns an sf object with all intersections

Examples

```
dirList<-list.dirs(paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs"), recursive = FALSE)
allLocIntersected<-concatIntersectedLocations(
 dirList = dirList, locations = c("Arville", "Redon"), columns = "lcz_primary")
```

`confidSensib` *Computes the agreement between geographical (LCZ) classifications on a range of values of an indicator of confidence granted to each LCZ classification. The input file or dataset must have been produced by compareLCZ function, or at least the columns must be in the same order*

Description

Computes the agreement between geographical (LCZ) classifications on a range of values of an indicator of confidence granted to each LCZ classification. The input file or dataset must have been produced by compareLCZ function, or at least the columns must be in the same order

Usage

```
confidSensib(
  inputDf = "",
  filePath = "",
  nPoints = 5,
  wf1 = "bdtopo_2_2",
  wf2 = "osm",
  geomID1 = "ID_RSU",
  column1 = "LCZ_PRIMARY",
  confid1 = "LCZ_UNIQUENESS_VALUE",
  geomID2 = "ID_RSU.1",
  column2 = "LCZ_PRIMARY.1",
  confid2 = "LCZ_UNIQUENESS_VALUE.1",
  sep = ";",
  repr = "standard",
  plot = TRUE,
  saveG = ""
)
```

Arguments

`inputDf`	is an R file with geom IDs, (LCZ) classifications and a confidence value granted for the (LCZ) classifications values of each geom. Ignored if filePath is not empty.
`filePath`	is the path to a csv file containing geom IDs, LCZ classifications and a confidence value granted for the LCZ value of each geom. Set it to "" if inputDf is specified (recommanded)

nPoints	is the number of points (quantiles) of confidence for which the average agreement between classifications will be computed
wf1	is the name of the workflow used to produce the first LCZ
wf2	is the name of the workflow used to produce the second LCZ
geomID1	is the name of the column that contains the geom ID associated to the first workflow
column1	is the name of the column storing the first LCZ classification values
confid1	is the name of the column storing the first LCZ classification confidence value
geomID2	is the name of the column that contains the geom ID associated to the second workflow
column2	is the name of the column storing the second LCZ classification values
confid2	is the name of the column storing the second LCZ classification confidence value
sep	the separator used if filePath is not empty
repr	is standard when original LCZ values are expected, alter otherwise
plot	if TRUE the graph is plotted
saveG	if not an empty string, specifies where to save graphs

Value

returns an object called output, which contains the values of the thresholds for the confidence value and the agreement between classifications for the LCZ levels presents in the dataset

Examples

```
# creation of the comparison data on which to perform the analysis
redonCompare<-compareLCZ(sf1=redonBDT, wf1="bdt", geomID1 = "ID_RSU", column1 ="LCZ_PRIMARY",
                         confid1 = "LCZ_UNIQUENESS_VALUE",
                       sf2=redonOSM, wf2="osm",geomID2 = "ID_RSU", column2="LCZ_PRIMARY",
                     confid2 ="LCZ_UNIQUENESS_VALUE", exwrite=FALSE, plot=FALSE, saveG="")
confidSensib(inputDf=redonCompare$data,
nPoints=5, wf1="bdtopo_2_2", wf2="osm",
geomID1="ID_RSU", column1="bdt", confid1="LCZ_UNIQUENESS_VALUE",
geomID2="ID_RSU.1",column2="osm", confid2="LCZ_UNIQUENESS_VALUE.1",
sep=";", repr="standard", plot=TRUE, saveG="")
```

---

createIntersect *Intersects multiple sf files in which Local Climate Zones are set for polygon geometries*

Description

Intersects multiple sf files in which Local Climate Zones are set for polygon geometries

Usage

```
createIntersect(
  sfList,
  columns,
  refCrs = NULL,
  workflowNames = NULL,
  minZeroArea = 0
)
```

Arguments

sfList	a list which contains the classifications to compare, as sf objects
columns	a vector which contains, for each sf of sfList, the name of the columns of the classification to compare
refCrs	a number which indicates which sf object from sfList will provide the CRS in which all the sf objects will be projected before comparison By defautl the first sf object CRs is applied to all sf objects.
workflowNames	a vector of strings which contains the names of the workflows used to produce the sf objects
minZeroArea	all geometries smaller than this value are discarded (avoids numeric precision problems)

Details

The input SfList can contain two levels, the first level being the locations, the second being the workflow. In this case, the output will still be a single sf object, concatenating the workflows and locations as columns

Value

a single sf file with values of LCZ from all the input are assigned to geometries resulting from intersection of all input geometries

Examples

```
sfList<-importMultipleLCZvect(
dirPath = paste0(
system.file("extdata", package = "lczexplore"),
"/multipleWfs/Arville"),
workflowNames = c("osm","bdt","wudapt"), location = "Arville")
ArvilleIntersect <- createIntersect(
 sfList = sfList, columns = rep("lcz_primary", 4),
 workflowNames = c("osm","bdt","wudapt"))
# Two Locations
sfList2<-importMultipleLocsLCZvect(
 dirPath = paste0(
     system.file("extdata", package = "lczexplore"),
     "/multipleWfs"),
 workflowNames = c("osm","bdt","wudapt"))
```

```
TwoLocsIntersect <- createIntersect(
 sfList = sfList2, columns = rep("lcz_primary", 3),
 workflowNames = c("osm","bdt","wudapt"))
```

`createWeightedFlux` *Takes an sf object with several workflow values, and creates all pairwise confusion matrices, then conatenates them in long form, usable to plot chord diagrams*

Description

Takes an sf object with several workflow values, and creates all pairwise confusion matrices, then conatenates them in long form, usable to plot chord diagrams

Usage

```
createWeightedFlux(
  intersectSfWide,
  columns = NULL,
  wfNamesIn = NULL,
  typeLevelsDefaultIn = .lczenv$typeLevelsDefault
)
```

Arguments

`intersectSfWide`	is an sf object, containing the different lcz classification on already intersected geometries in wide format with a column wf for the workflow names, a column lcz_primary for the LCZ types.
`columns`	a vector containing the names of the LCZ type columns
`wfNamesIn`	the name of the input compared workflows. They will be the output columns for the LCZ types
`typeLevelsDefaultIn`	is a named vector of strings containing the LCZ levels. By default inherited from lczexplore

Value

a dataframe with columns orig, dest and weightedFlux, weighted flux the percentage of area from a given LCZ type of a given workflow (orig) to another LCZ type of another workflow (dest).

Examples

```
twoLocsDir<-paste0(
 system.file("extdata", package = "lczexplore"),"/multipleWfs")
twoLocsSfList<-importMultipleLocsLCZvect(
 dirPath = twoLocsDir, workflowNames = c("osm","bdt","wudapt"),
 location = c("Arville", "Redon"))

twoLocsSfIntersected <- createIntersect(
 sfList = twoLocsSfList, columns = rep("lcz_primary", 4),
 refCrs=NULL, workflowNames=c("osm", "bdt", "wudapt"), minZeroArea=0.001)

twoLocsWeightedFlux<-createWeightedFlux(
twoLocsSfIntersected, wfNamesIn = c("osm","bdt","wudapt"))
```

`drawChordDiagram` *Draws how LCZ types from several workflows break up into LCZ types of one another*

Description

Draws how LCZ types from several workflows break up into LCZ types of one another

Usage

```
drawChordDiagram(weightedFluxIn, colorMapIn = NULL, labelMatch = NULL, ...)
```

Arguments

`weightedFluxIn`	is typically the output of the function createWeightedFlux, and is expected to contain the following columns orig, dest and weightedFlux, whose names are quite self explanatory : orig is the origin LCZ type, dest is the destination LCZ type and weightedFlux is the percentage of area transfered from orig to dest
`colorMapIn`	is a named vector whose names are the unique values of orig and dest columns of weightedFluxIn and values are the associated colors. In cas of grouping, these values are overwritten by the groupCols argument, see the ... parameters
`labelMatch`	allox to match a level of dest or orig from weightedFluxIn and a label. It also allows to change the order in which they are visualized (default is alphabetical order). It is then possible to prefixe values of Levels by a letter to force this order and use LabelMatch to use original names. When using grouping vectors, they are visualized in the order they are specified.
`...`	allows the user to do on-the-fly grouping. These must be passed as groupName = groupValues where groupName is the name of a resulting group and groupValues a vector of the initial values it will regroup.

Value

a chord diagram of how LCZ types of a workflow break up into types of the other workflows

Examples

```
twoLocsDir<-paste0(
 system.file("extdata", package = "lczexplore"),"/multipleWfs")
twoLocsSfList<-importMultipleLocsLCZvect(
 dirPath = twoLocsDir, workflowNames = c("osm","bdt","wudapt"),
 location = c("Arville", "Redon"))
twoLocsSfIntersected <- createIntersect(
 sfList = twoLocsSfList, columns = rep("lcz_primary", 4),
 refCrs=NULL, workflowNames=c("osm", "bdt", "wudapt"), minZeroArea=0.001)
twoLocsWeightedFlux<-createWeightedFlux(
 twoLocsSfIntersected, wfNamesIn = c("osm","bdt","wudapt"))
# Subsetting and grouping allow further exploration.
# Playing on grouping names and alphabetical order allows
# to choose the order of the sectors
twoLocsWeightedFluxNo104no101<-subset(twoLocsWeightedFlux,
     !grepl("101", twoLocsWeightedFlux$orig) &
     !grepl("104", twoLocsWeightedFlux$orig) &
     !grepl("101", twoLocsWeightedFlux$dest) &
     !grepl("104", twoLocsWeightedFlux$dest))
aggregMatch<-c(
    "acompact"="Compact",
     "blessCompact" = "Less Compact",
    "cfewToNoBuild" = "Few to No Buildings",
    "dunclass" = "Unclassified")
drawChordDiagram(twoLocsWeightedFluxNo104no101, labelMatch = aggregMatch,
    acompact = c("1", "2", "3"),
    blessCompact = c("4", "5", "6", "7", "8", "10"),
    cfewToNoBuild = c("101", "102", "103", "104", "105", "106", "107", "9"),
    dunclass = "Unclassified",
    groupColors = c(
      "acompact" = "#8b0101",
      "blessCompact" = "#ff9856",
       "cfewToNoBuild" = "#bbdb7a","dunclass" = "grey"))
```

`drawChordDiagramWithGrouping` *Draws how LCZ types from several workflows break up into LCZ types of one another when some grouping is precised in the arguments*

Description

Draws how LCZ types from several workflows break up into LCZ types of one another when some grouping is precised in the arguments

Usage

```
drawChordDiagramWithGrouping(
  weightedFluxIn,
  colorMapIn = NULL,
```

```
  labelMatch = NULL,
  ...
)
```

Arguments

weightedFluxIn	is typically the output of the function createWeightedFlux, and is expected to contain the following columns orig, dest and weightedFlux, whose names are quite self explanatory : orig is the origin LCZ type, dest is the destination LCZ type and weightedFlux is the percentage of area transfered from orig to dest
colorMapIn	is a named vector whose names are the unique values of orig and dest columns of weightedFluxIn and values are the associated colors. In cas of grouping, these values are overwritten by the groupCols argument, see the ... parameters
labelMatch	allow to match a level of dest or orig from weightedFluxIn and a label. Remember to match labels with the grouping names you chose in the named grouping vectors passed to ... When using grouping vectors, they are visualized in the order they are specified.
...	allows the user to do on-the-fly grouping. These must be passed as groupName = groupValues where groupName is the name of a resulting group and groupValues a vector of the initial values it will regroup.

Value

a chord diagram after LCZ types are grouped according to ... arguments

Examples

```
twoLocsDir<-paste0(system.file("extdata", package = "lczexplore"),"/multipleWfs")
twoLocsSfList<-importMultipleLocsLCZvect(
dirPath = twoLocsDir, workflowNames = c("osm","bdt","wudapt"),
 location = c("Arville", "Redon"))
twoLocsSfIntersected <- createIntersect(
 sfList = twoLocsSfList, columns = rep("lcz_primary", 4),
 refCrs=NULL, workflowNames=c("osm", "bdt", "wudapt"), minZeroArea=0.001)
twoLocsWeightedFlux<-createWeightedFlux(
 twoLocsSfIntersected, wfNamesIn = c("osm","bdt","wudapt"))
# Subsetting and grouping allow further exploration.
# Playing on grouping names and alphabetical order allows
# to choose the order of the sectors
# Subsetting and grouping allow further exploration.
# Playing on grouping names and alphabetical order allows
# to choose the order of the sectors
twoLocsWeightedFluxNo104no101<-subset(twoLocsWeightedFlux,
     !grepl("101", twoLocsWeightedFlux$orig) &
     !grepl("104", twoLocsWeightedFlux$orig) &
     !grepl("101", twoLocsWeightedFlux$dest) &
     !grepl("104", twoLocsWeightedFlux$dest))
aggregMatch<-c("acompact"="Compact", "blessCompact" = "Less Compact",
"cfewToNoBuild" = "Few to No Buildings at All",
    "dunclass" = "Unclassified")
```

```
drawChordDiagram(twoLocsWeightedFluxNo104no101, labelMatch = aggregMatch,
    acompact = c("1", "2", "3"),
    blessCompact = c("4", "5", "6", "7", "8", "10"),
    cfewToNoBuild = c("101", "102", "103", "104", "105", "106", "107", "9"),
    dunclass = "Unclassified",
    groupColors = c(
      "acompact" = "#8b0101",
      "blessCompact" = "#ff9856",
       "cfewToNoBuild" = "#bbdb7a","dunclass" = "grey"))
```

`drawSectors` *Draws sectors and groups inside the drawChordDiagram function*

Description

Draws sectors and groups inside the drawChordDiagram function

Usage

```
drawSectors(sectorID, sectorsIn = sectors, colorMapIn = colorMap, textMatch)
```

Arguments

`sectorID`	ar the identifiers of sectors as returned by the makeSectorsAndGroups function
`sectorsIn`	are the values of a sectors as returned by the makeSectorsAndGroups function
`colorMapIn`	is a named vector whose names must contain the values of sectorsIn and whose values are the desired colors see drawChorDiagram function for details
`textMatch`	is fed by drawChordDiagram to provide labels for sectors

Value

a list containing vectors and groups for a chord diagram

`fetchLCZ` *Deprecated DO NOT USE, kept in case some users might need it Fetches LCZ classification data, by default from the Geomanum cloud, for locations already uploaded there by the Geoclimate team*

Description

Deprecated DO NOT USE, kept in case some users might need it Fetches LCZ classification data, by default from the Geomanum cloud, for locations already uploaded there by the Geoclimate team

Usage

```
fetchLCZ(
  location,
  outDir,
  wf = "bdtopo_2_2",
  refYear = "2022",
 baseURL = "https://cloud.geomanum.org/index.php/s/geoclimate/download?path=%2F",
  ...
)
```

Arguments

location	is the name of the place for which you want to get the GeoClimate data
outDir	is the path to the directory in which fetchLCZ will unpack the data
wf	workflow can be bdtopo_2_2, when GeoClimate used the BDTOPO data base, or osm when it used the open street map data.
refYear	this parameters is integrated in the name given to the data, in order to allow to re-run the function according to the reference year. At the moment only 2022 data have been tried
baseURL	is the adress of the server where the function fetches the data. The data must be organised in a bdtopo_2_2 and an osm/year directories. This function is intended to be used by Paendora project members, so the default is the Geomanum Foundation cloud.
...	allows to pass arguments from the produce analysis function to the groupLCZ and compareLCZ functions

Value

returns no R object. The data are uploaded from the GeoClimate cloud and locally unpacked in the outDir directory

Examples

```
# not run as the access is not open
# fetchLCZ(location="Allaire",
# outDir=system.file("extdata", package = "lczexplore"),
# wf="bdtopo_2_2",refYear="2022")
```

---

geoClimateCall *Calls GeoClimate jar file and feeds it a configuration file by building a command an using system (only tested on linux)*

Description

Calls GeoClimate jar file and feeds it a configuration file by building a command an using system (only tested on linux)

Usage

```
geoClimateCall(jarFilePath, configFilePath, wf = "OSM")
```

Arguments

jarFilePath	tells where the geoclimate jar file is, default points to the embedded jar file, i.e. the latest snapshot version when the package was built. Versions can be downloaded from https://github.com/orbisgis/geoclimate/wiki/Download
configFilePath	points to the configuration JSON file for GeoClimate, typically a file created with the geoClimateConfigFile function
wf	is the workflow to use with GeoClimate, the default is OSM for OpenStreetMap. The other possible value is "BDTOPO_V2". Other values will be added (e.g. for BDTOPO_V3) when tested.

Value

returns nothing but executes GeoClimate in the folder specified in the JSON configuration file.

Examples

```
# not run as they write files
# test <- geoClimateConfigFile(
#     outConfigFile = "",
#     wf = "OSM", outFolder = "", locations = "Redon",
#     rsuIndics = c("LCZ", "TEB", "UTRF"),
#     gridIndics = c(
#         "BUILDING_FRACTION", "BUILDING_HEIGHT", "WATER_FRACTION",
#         "VEGETATION_FRACTION", "ROAD_FRACTION",
#         "IMPERVIOUS_FRACTION", "LCZ_PRIMARY", "LCZ_FRACTION", "UTRF"),
#        writeNow = FALSE)
# not run as geoclimate jar source needed
# geoClimateCall(
# jarFilePath = "path/to/Geoclimate/geoclimate-0.0.2-SNAPSHOT.jar",
# configFilePath = "/tmp/RedonOSM2022.json", wf = "OSM")
# rsuIndics = c("LCZ","TEB","UTRF"),
# gridIndics = c(
#        "BUILDING_FRACTION","BUILDING_HEIGHT",
#        "WATER_FRACTION","VEGETATION_FRACTION","ROAD_FRACTION",
#        "IMPERVIOUS_FRACTION","LCZ_PRIMARY","LCZ_FRACTION","UTRF"))
```

---

geoClimateConfigFile *Builds a JSON configuration file for GeoClimate workflow.*

Description

Builds a JSON configuration file for GeoClimate workflow.

Usage

```
geoClimateConfigFile(
  wf,
  locations,
  forceSRID = FALSE,
  outFolder = "/tmp",
  date = "2022-01-01T12:00:00Z",
  rsuIndics = c("LCZ", "TEB", "UTRF"),
  svfSimplified = TRUE,
  estimatedHeight = TRUE,
  grid_x_size = 0,
  grid_y_size = 0,
  rowCol = FALSE,
  outputType = "geojson",
  gridIndics = "",
  outConfigDir = "/tmp",
  outConfigFile = "configFile",
  BDTinFolder = "",
  writeNow = FALSE
)
```

Arguments

wf	is the workflow used by GeoClimate. For now, only OSM" is available, for OpenStreetMap, but "BDT" for BDTOPO of IGN should be added when an online database is available.
locations	is either the town or the coordinates of the bounding box of the area on which GeoClimate will run. If a town name, it will be fed to the Nominoe API, through the overpass API of OpenStreetMap.
forceSRID	some BD TOPO input file may not have an srid, this forces srid to be 2154
outFolder	indicates where the results of GeoClimate will be put
date	is the date of the data we extract from OpenStreetMap. The format is "yyyy:mm:ddThh:mm:ssZ" (for BDTOPO the version depends from the input data folder)
rsuIndics	is a vector with the indicators one wants to compute at the RSU scale. The default is c("LCZ","TEB","UTRF"),
svfSimplified	uses the simplified method to calculate skyview factor, default = TRUE
estimatedHeight	uses an algorithm to estimate the missing building height, default = TRUE
grid_x_size	is the x size for the grid if some grid indicators are to be computed, default=100
grid_y_size	is the x size for the grid if some grid indicators are to be computed, default=100
rowCol	if grid_x_size and grid_y_size are not set, one cal set the number of rows and cols throug rowCol, but the recommended and default is FALSE
outputType	is the format of GeoClimate outputs, default="geojson"
gridIndics	is a vector containing the indicators to compute at the grid scale. Default is c("BUILDING_FRACTION", "BUILDING_HEIGHT", "WATER_FRACTION", "VEGETATION_FRACTION", "ROAD_FRACTION", "IMPERVIOUS_FRACTION","LCZ_PRIMARY

outConfigDir	is the folder were the resulting JSON file will be put, the folder where GeoClimate will read it from (different from out outFolder, where GeoClimates will put its geoJSON ouputs), default is "/tmp"
outConfigFile	is the name of your configuration file, if and empty string, a name will be created from location and workflow parameters.
BDTinFolder	the path of the folder where are the BDTOPO needed by geoclimate
writeNow	if TRUE, the resulting config file is written using the outConfigDir and outConfigFile parameters

Details

GeoClimate is not included in this package. The json configuration call can be fed to GeoClimate with the geoClimateCall function, provided that user has downloaded a version of the software from the official repository https://github.com/orbisgis/geoclimate

Value

Exports a json configuration file to be read by GeoClimate software

returns a JSON configuration file to be fed to GeoClimate

Examples

```
# not run as they write files
# test <- geoClimateConfigFile(
#     outConfigFile = "",
#     wf = "OSM", outFolder = "", locations = "Redon",
#     rsuIndics = c("LCZ", "TEB", "UTRF"),
#     gridIndics = c(
#         "BUILDING_FRACTION", "BUILDING_HEIGHT", "WATER_FRACTION",
#         "VEGETATION_FRACTION", "ROAD_FRACTION",
#         "IMPERVIOUS_FRACTION", "LCZ_PRIMARY", "LCZ_FRACTION", "UTRF"),
#        writeNow = FALSE)
# not run as geoclimate jar source needed
# geoClimateCall(
# jarFilePath = "path/to/Geoclimate/geoclimate-0.0.2-SNAPSHOT.jar",
# configFilePath = "/tmp/RedonOSM2022.json", wf = "OSM")
# rsuIndics = c("LCZ","TEB","UTRF"),
# gridIndics = c("BUILDING_FRACTION","BUILDING_HEIGHT",
# "WATER_FRACTION","VEGETATION_FRACTION","ROAD_FRACTION",
# "IMPERVIOUS_FRACTION","LCZ_PRIMARY","LCZ_FRACTION","UTRF"))
```

---

groupLCZ *Allows to group (Local Climate Zone) geographical classification levels into broader categories to explore classification agreements*

Description

Allows to group (Local Climate Zone) geographical classification levels into broader categories to explore classification agreements

Usage

```
groupLCZ(sf, column, outCol = "grouped", ...)
```

Arguments

sf	is the input files. It must be an sf file and contain an LCZ column (levels 1 to 10 and 101 to 107). It must contain the geom column.
column	is the name of the column containing the LCZ to be grouped
outCol	is the name of the colum in which the grouped LCZ will be put
...	the names of the groups followed by the levels the regrouped for instance urban=c("1","2","3","4","5","6","7","8","9")

Value

a file containing the original geom and lcz, and a new grouped column

Examples

```
redonBDTgrouped<-groupLCZ(redonBDT,column="LCZ_PRIMARY",
urban=c("1","2","3","4","5","6","7","8","9"),
industry="10", vegetation=c("101","102","103","104"),
impervious="105",pervious="106",water="107",
colors=c("red","black","green","grey","burlywood","blue"))

showLCZ(redonBDT,column="LCZ_PRIMARY", repr="standard")
showLCZ(redonBDTgrouped,column="grouped",repr="alter",
LCZlevels=c("urban","industry","vegetation","impervious","pervious","water"),
colors=c("red","black","green","grey","burlywood","blue"),wf="BD TOPO")
```

---

groupLCZsuffix *Regroups levels of origin and destination of a multiple confusion matrix in long form.*

Description

Regroups levels of origin and destination of a multiple confusion matrix in long form.

Usage

```
groupLCZsuffix(weightedFluxIn, ...)
```

Arguments

weightedFluxIn	is typically the output of the function createWeightedFlux, and is expected to contain the following columns orig, dest and weightedFlux, whose names are quite self explanatory : orig is the origin LCZ type, dest is the destination LCZ type and weightedFlux is the percentage of area transfered from orig to dest
...	a set of argument passed as groupName = groupValues where groupName is the name of a resulting group and groupValues a vector of the initial values it will regroup.

Value

a list containing vectors and groups for a chord diagram

---

importConcatMultipleLocsLCZvect

In a given list of directories the function looks for LCZ datafiles, return a datasets LCZ values for each geom and workflow

---

Description

In a given list of directories the function looks for LCZ datafiles, return a datasets LCZ values for each geom and workflow

Usage

```
importConcatMultipleLocsLCZvect(
  dirPath,
  locations = NA,
  workflowNames = c("osm", "bdt", "wud"),
  missingGeomsWf = "osm",
  refWf = NULL,
  refLCZ = NA,
  residualLCZvalue = NA,
  column = "lcz_primary"
)
```

Arguments

dirPath	is the path to the root directory where are the subdirectories containing different location LCZ files
locations	for each diretory from dirList, a location name must be fed to the function
workflowNames	sets the names of workflows and define the name of the files which will be loaded and intersected
missingGeomsWf	the name of the workflow where some areas were not classified

refWf — the name of the workflow one uses as a reference to complete the incomplete workflow. if NULL then the geometries added will get the level specified in residualLCZvalue, else, the level in refLCZ

refLCZ — the reference level the completed geometries will receive

residualLCZvalue — the value the completed geometries where the reference workflow does not classify the geometry with refLCZ level

column — a parameter to feed addMissingRSUs function

Value

an sf object conaining the LCZ types of all workflows for all locations

Examples

```
dirPath<-paste0(
system.file("extdata", package = "lczexplore"),
"/multipleWfs")
allLocAllWfs<-importConcatMultipleLocsLCZvect(
 dirPath = dirPath, locations = c("Redon", "Arville"),
workflowNames = c("osm","bdt","wudapt"),
 missingGeomsWf= "osm",
 refWf = NULL,
 refLCZ = "Unclassified",
 residualLCZvalue = "Unclassified",
 column = "lcz_primary"
)
```

---

importLCZgc

Imports the wf_lcz fgb file produced by GeoClimate. Not recommended. Use the more generic importLCZvect function

---

Description

Imports the wf_lcz fgb file produced by GeoClimate. Not recommended. Use the more generic importLCZvect function

Usage

```
importLCZgc(dirPath, output = "sfFile", file = "rsu_lcz.fgb")
```

Arguments

dirPath — : the path where the rsu_lcz.geojson file is.

output — : if sfFile the rsu_lcz.geojson is imported as an sf file, if bBox, a bounding box of the area is returned

file — : name of the file to load. Default is rsu_lcz.fgb, as it is the output of GeoClimate when the the rsu indicators are specified (as opposed to grid indicators).

Value

: if output is set to sfFile, the function returns an sf objects containing all the colums of the rsu_lcz.geojson found in the dirPath directory. If output is bBox a bounding box (seef sf package) of the area contained in the rsu_lcz file in the dirPath directory is returned. This is useful to select geoms analysed in other sf objects.

Examples

```
# not run
# bdtTest <- importLCZgc(dirPath = system.file("extdata/multipleWfs/Redon", package = "lczexplore"),
# file = "bdt_lcz.fgb")
```

`importLCZraster` *Imports the LCZ classifications produced on raster maps. For now, the european tif map produced by WUDAP is used as a reference example. Users can DOWNLOAD the WUDAPT tif file at the following url : https://figshare.com/articles/dataset/European_LCZ_map/13322450*

Description

Imports the LCZ classifications produced on raster maps. For now, the european tif map produced by WUDAP is used as a reference example. Users can DOWNLOAD the WUDAPT tif file at the following url : https://figshare.com/articles/dataset/European_LCZ_map/13322450

Usage

```
importLCZraster(
  dirPath,
  bBox,
  fileName = "EU_LCZ_map.tif",
  LCZband = 1,
  LCZcolumn = "EU_LCZ_map",
  confidenceBand = "",
  confidenceColumn = "confidence",
 typeLevels = c(`1` = "1", `2` = "2", `3` = "3", `4` = "4", `5` = "5", `6` = "6", `7` =
  "7", `8` = "8", `9` = "9", `10` = "10", `101` = "11", `102` = "12", `103` = "13",
    `104` = "14", `105` = "15", `106` = "16", `107` = "17")
)
```

Arguments

`dirPath`	is the path to the directory where the
`bBox`	bBox is a mask to crop the raster file. It can be a bounding box or any single geometry which contains the desired zone. It can be produced by the importLCZvect function. It can either be of class bBox or of class sfc, like the geometry column of an sf object.
`fileName`	is the name of the raster file (tif or geotif), by default \'EU_LCZ_map.tif\'.

LCZband	defines the band of the raster file which contains the LCZ types. It can be a number or a band name (default = 1)
LCZcolumn	the name of the column of the output sf File in which the LCZ types will be stored (default = "LCZ")
confidenceBand	the band of the raster files which contains a measure of confidence. Can be a number or a band name (default = "", that is no measure of confidence provided)
confidenceColumn	the name of the column of the output sf File in which the confidence values will be stored (default = "", no confidence value specified)
typeLevels	indicates a named vector of the unique values contained in LCZcolumn,

Value

an sf file containing the geom and LCZ levels from theraster tiff within the bBox bounding box

Examples

```
redonBbox<-importLCZvect(dirPath=paste0(system.file("extdata", package = "lczexplore"),
"/multipleWfs/Redon"), file="bdt_lcz.fgb", column="LCZ_PRIMARY", output="bBox")

redonWudapt<-importLCZraster(system.file("extdata", package = "lczexplore"),
fileName="redonWudapt.tif",bBox=redonBbox)

# Another way to get the bounding box when one explores a given city would be the use of the
# getbb() function from the osmdata package.
# This example requires the osmdata package and therefore is not executed here
# redonBbox<-osmdata::getbb("Redon")
# redonWudapt<-importLCZraster(system.file("extdata", package = "lczexplore"),
# fileName="redonWudapt.tif",bBox=redonBbox)

# another way to get the bounding box when one doesn't want
#to compare to a vector map is to enter it's coordinates
# and feed them to st_bbox() of the sf package.

# imporLCZraster also allows to choose which band of a raster file to import
redonWudapt<-importLCZraster(system.file("extdata", package = "lczexplore"),
fileName="redonWudapt.tif",bBox=redonBbox, LCZband=1, LCZcolumn='EU_LCZ_map')
```

---

importLCZvect *Imports Local Climate Zone classifications from a vector source, a geographical file or an sf object from the current R session*

Description

Imports Local Climate Zone classifications from a vector source, a geographical file or an sf object from the current R session

Usage

```
importLCZvect(
  dirPath,
  file = "bdt_lcz.fgb",
  output = "sfFile",
  column = "LCZ_PRIMARY",
  geomID = "",
  confid = "",
  typeLevels = .lczenv$typeLevelsConvert2,
  drop = TRUE,
  verbose = FALSE,
  sfIn = NULL,
  naAsUnclassified = TRUE
)
```

Arguments

dirPath	is the path of the directory of the file
file	is the name of the file from which the LCZ are imported
output	: if sfFile, the function returns an sfFile with the specified columns, if bBox, returns a bounding box one can use to crop a raster file or to intersect another sf file
column	indicates the name of the column containing LCZ values. LCZ values are expected to be of a standard LCZ format (1 to 17, or 1 to 10 and 101 to 107 or 1 to G), else, use the importQualVar function
geomID	is the name of the column containing the ID of each geom to load. If an empty string, no column is loaded.
confid	is the name of the column containing a confidence indicator to filter geoms, for instance the uniqueness of the LCZ level of each geom
typeLevels	the levels of the imported LCZ classification
drop	: the default is TRUE, which means all the column are dropped excepted those specified in previous parameters
verbose	if TRUE show the discrepancies between specified levels of LCZ and levels actually present in column
sfIn	allows you to import from an existing object of your session, not from an external file
naAsUnclassified	if TRUE allows NAs in the input file and assigns them the value "Unclassified"

Value

returns an sf object containing at least the geoms and the LCZ values, and if specified, columns for the IDs of the geoms and the confidence value of the LCZ levels.

Examples

```
redonBDTex<-importLCZvect(dirPath=paste0(system.file("extdata", package = "lczexplore"),
 "/multipleWfs/Redon"), file="bdt_lcz.fgb", column="LCZ_PRIMARY",
geomID="ID_RSU",confid="LCZ_UNIQUENESS_VALUE")
showLCZ(redonBDTex)
```

`importLCZvectFromFile` *Imports Local Climate Zone classifications from a standard geographical file (tested : geojson, shp, more to come)*

Description

Imports Local Climate Zone classifications from a standard geographical file (tested : geojson, shp, more to come)

Usage

```
importLCZvectFromFile(
  dirPath,
  file = "bdt_lcz.fgb",
  column,
  geomID = "",
  confid = "",
  verbose = TRUE,
  drop = TRUE
)
```

Arguments

`dirPath`	is the path of the directory of the file
`file`	is the name of the file from which the LCZ are imported
`column`	indicates the name of the column containing LCZ values. LCZ values are expected to be of a standard LCZ format (1 to 17, or 1 to 10 and 101 to 107 or 1 to G), else, use the importQualVar function
`geomID`	is the name of the column containing the ID of each geom to load. If an empty string, no column is loaded.
`confid`	is the name of the column containing a confidence indicator to filter geoms, for instance the uniqueness of the LCZ level of each geom
`verbose`	if TRUE show the discrepancies between specified levels of LCZ and levels actually present in column
`drop`	: the default is TRUE, which means all the column are dropped excepted those specified in previous parameters

Value

returns an sf object containing at least the geoms and the LCZ values, and if specified, columns for the IDs of the geoms and the confidence value of the LCZ levels.

Examples

```
redonBDTex<-importLCZvectFromFile(dirPath=paste0(system.file("extdata", package = "lczexplore"),
"/multipleWfs/Redon"), file="bdt_lcz.fgb", column="LCZ_PRIMARY",
geomID="ID_RSU",confid="LCZ_UNIQUENESS_VALUE")
```

`importLCZvectFromSf` *Imports Local Climate Zone classifications from an existing sf object*

Description

Imports Local Climate Zone classifications from an existing sf object

Usage

```
importLCZvectFromSf(sfIn, column, geomID = "", confid = "")
```

Arguments

`sfIn`	is the sf object containing the LCZ map
`column`	indicates the name of the column containing LCZ values. LCZ values are expected to be of a standard LCZ format (1 to 17, or 1 to 10 and 101 to 107 or 1 to G), else, use the importQualVar function
`geomID`	is the name of the column containing the ID of each geom to load. If an empty string, no column is loaded.
`confid`	is the name of the column containing a confidence indicator to filter geoms, for instance the uniqueness of the LCZ level of each geom

Value

returns an sf object containing at least the geoms and the LCZ values, and if specified, columns for the IDs of the geoms and the confidence value of the LCZ levels.

Examples

```
redonBDTex<-importLCZvect(dirPath=paste0(system.file("extdata", package = "lczexplore"),
"/multipleWfs/Redon"), file="bdt_lcz.fgb", column="LCZ_PRIMARY",
geomID="ID_RSU",confid="LCZ_UNIQUENESS_VALUE")
redonBDTex2<-importLCZvectFromSf(sfIn = redonBDTex , column="LCZ_PRIMARY",
                                 geomID="ID_RSU",confid="LCZ_UNIQUENESS_VALUE")
```

`importMultipleLCZvect` *In a given directory (or a list of directories) the function looks for LCZ datafiles and load them in a list. In each directory, files must have names built as follow : wf_lcz.fileExtension, where wf are the values specified in workflowNames parameter and fileExtension is a file extension known by sf drivers, like fgb, geojson...*

Description

In a given directory (or a list of directories) the function looks for LCZ datafiles and load them in a list. In each directory, files must have names built as follow : wf_lcz.fileExtension, where wf are the values specified in workflowNames parameter and fileExtension is a file extension known by sf drivers, like fgb, geojson...

Usage

```
importMultipleLCZvect(
  dirPath,
  workflowNames = c("osm", "bdt", "wudapt"),
  location = NA,
  fileExtension = ".fgb",
  columns = "lcz_primary",
  typeLevels = .lczenv$typeLevelsConvert2
)
```

Arguments

`dirPath`	is the place where the files are
`workflowNames`	sets the names of workflows
`location`	is the name of the location at which all LCZ are created
`fileExtension`	is the extensions of the files to load (.fgb is the recommended format)
`columns`	the name (string) of the column containing LCZ types. If the different workflows do no use the same column names, a vector of names is passed
`typeLevels`	allows to pass the expected levels of the lcz columns

Value

returns a list containing one sf by workflow for a given locations

Examples

```
sfList<-importMultipleLCZvect(dirPath = paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
workflowNames = c("osm","bdt","wudapt"), location = "Arville", columns = "lcz_primary")
```

`importMultipleLCZvectFromSession` *Take a list of sf objects within the session and format them to allow the use of createIntersect and multipleCompare functions.*

Description

Take a list of sf objects within the session and format them to allow the use of createIntersect and multipleCompare functions.

Usage

```
importMultipleLCZvectFromSession(
  sfList,
  workflowNames,
  columns,
  location,
  refCRS = 1
)
```

Arguments

`sfList`	the list of LCZ sf objects
`workflowNames`	a vector containing the workflow names of each sf object of sfList
`columns`	contain the names of the columns where the sf objects of sfList are stored
`location`	the name of the location, all files from sfList must regard the same location
`refCRS`	a number telling which sf of the sfList will be the reference in termes of Coordinate Reference System

Value

returns a list of sf objects, each of them has columns lcz_primary, wf and location

Examples

```
osm<-importLCZvect(dirPath = paste0(
     system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
     file = "osm_lcz.fgb")
bdt<-importLCZvect(dirPath = paste0(
     system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
 file = "bdt_lcz.fgb")
wudapt<-importLCZvect(dirPath = paste0(
     system.file("extdata", package = "lczexplore"),"/multipleWfs/Arville"),
 file = "wudapt_lcz.fgb", column = "lcz_primary")
sfList<-list(osm = osm, bdt = bdt, wudapt = wudapt)
sfListFormatted<- importMultipleLCZvectFromSession(sfList = sfList,
workflowNames = c("osm", "bdt", "wudapt"),location = "Arville",
```

```
columns = c("LCZ_PRIMARY", "LCZ_PRIMARY", "lcz_primary" ))
intersected<-createIntersect(sfList = sfListFormatted, columns = rep("lcz_primary", 3),
workflowNames = c("osm", "bdt", "wudapt"))
multicompare_test<-compareMultipleLCZ(intersected,
columns = c("osm","bdt","wudapt"),trimPerc = 0.5)
```

---

importMultipleLocsLCZvect

In a given directory the function looks different locations as subdirectories and LCZ datafiles in each subdirectory, and load them in a list In each directory, files must have names built as follow : wf_lcz.fileExtension, where wf are the values specified in workflowNames parameter and fileExtension is a file extension known by sf drivers, like fgb, geojson...

Description

In a given directory the function looks different locations as subdirectories and LCZ datafiles in each subdirectory, and load them in a list In each directory, files must have names built as follow : wf_lcz.fileExtension, where wf are the values specified in workflowNames parameter and fileExtension is a file extension known by sf drivers, like fgb, geojson...

Usage

```
importMultipleLocsLCZvect(
 dirPath = paste0(system.file("extdata", package = "lczexplore"), "/multipleWfs/"),
  workflowNames = c("osm", "bdt", "wudapt"),
  locations = c("Arville", "Redon"),
  fileExtension = ".fgb",
  columns = NULL
)
```

Arguments

dirPath	is the place where the files are
workflowNames	sets the names of workflows
locations	is the name of the location at which all LCZ are created
fileExtension	is the extensions of the files to load (.fgb is the recommended format)
columns	contains the names of the columns containing the LCZ types

Value

a list of sf, containing one element per location, each of which containing one sf per werkflow

Examples

```
sfList<-importMultipleLocsLCZvect(dirPath = paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs/"),
workflowNames = c("osm","bdt","wudapt"), location = c("Arville", "Redon"))
```

`importQualVar` *Imports a qualitative variable associated with polygons from a geographical dataset (tested : geojson, shp, more to come)*

Description

Imports a qualitative variable associated with polygons from a geographical dataset (tested : geojson, shp, more to come)

Usage

```
importQualVar(
  dirPath = paste0(system.file("extdata", package = "lczexplore"), "/utrfFiles"),
  file = "bdt_utrf_area.fgb",
  output = "sfFile",
  column = "TYPO_MAJ",
  geomID = "ID_RSU",
  confid = "UNIQUENESS_VALUE",
  typeLevels = "",
  drop = TRUE,
  verbose = TRUE
)
```

Arguments

`dirPath`	is the path of the directory of the file
`file`	is the name of the file from which the variable is imported
`output`	: if sfFile, the function returns an sfFile with the qualitative variable, if bBox, returns a bounding box one can use to crop a raster file or to intersect another sf file
`column`	indicates the name of the column containing the qualitative variable is imported
`geomID`	is the name of the column containing the ID of each geom to load. If an empty string, no column is loaded.
`confid`	is the name of the column containing a confidence indicator, for instance the uniqueness of the LCZ level of each geom, in order to filter geoms,
`typeLevels`	the levels of the imported qualitative variable
`drop`	: the default is TRUE, which means all the column are dropped excepted those specified in previous parameters
`verbose`	allows to hide some messages when set to false.

Value

returns an sf object containing at least the geoms and the qualitative variable, and if specified, columns for the IDs of the geoms and the confidence value of thevalues of the variable.

Examples

```
utrfRedonBDT<-importQualVar(dirPath=paste0(
system.file("extdata", package = "lczexplore"), "/utrfFiles"),
file="bdt_utrf_area.fgb", column="TYPO_MAJ")
showLCZ(sf=utrfRedonBDT, column="TYPO_MAJ",repr="alter")
utrfRedonOSM<-importQualVar(dirPath=
paste0(system.file("extdata", package = "lczexplore"),"/utrfFiles"),
file="osm_utrf_area.fgb", column="TYPO_MAJ",geomID="ID_RSU",confid="UNIQUENESS_VALUE")
# One can now compare these 2 classifications and store the result in an object...
utrfComparison<-compareLCZ(
 sf1=utrfRedonBDT, column1="TYPO_MAJ", wf1="UTRF.BDT",
 sf2=utrfRedonOSM, column2="TYPO_MAJ", wf2 = "UTRF.OSM",
location = " Redon",exwrite=FALSE,repr="alter")
 # ... then plot the confusion matrix of these two classifications
print(utrfComparison$matConfPlot)
```

---

intersectAlocation *DEPRECATED: use concatIntersectedLocations with the proper arguments Intersects several sf files for a given location identified by a given directory path and name*

Description

DEPRECATED: use concatIntersectedLocations with the proper arguments Intersects several sf files for a given location identified by a given directory path and name

Usage

```
intersectAlocation(
  dirPath,
  workflowNames = c("osm", "bdt", "wudapt"),
  location,
  addMissingRSUs = TRUE,
  missingGeomsWf = "osm",
  refWf = NULL,
  refLCZ = "Unclassified",
  residualLCZvalue = "Unclassified",
  column = "lcz_primary"
)
```

Arguments

dirPath	is the directory where the original data are
workflowNames	are the names of the workflows : they are used to identify the files
location	is the name of the location (string)
addMissingRSUs	if TRUE calls the addMissingRSUs function to avoid areas with no geometries
missingGeomsWf	is passed to addMissingRSU function
refWf	is passed to addMissingRSU function
refLCZ	is passed to addMissingRSU function
residualLCZvalue	is passed to addMissingRSU function
column	the column name of the LCZ classification

Details

This function is not generic, it expects the data files to be named wf_rsu_lcz; wf varying among the values of workflownames, and the LCZ columns are expected to be lcz_primary (but lower and upper cases are accepted)

Value

an sf file with values of LCZ from all the input assigned to geometries resulting from intersection of all input geometries

Examples

```
# deprecated : use concatIntersectedLocations with the proper arguments
```

LCZareas *Computes the area of each level of a LCZ*

Description

Computes the area of each level of a LCZ

Usage

```
LCZareas(sf, column, LCZlevels)
```

Arguments

sf	is the sf dataset containing the LCZ
column	allows to specify in which column are the LCZ
LCZlevels	is the vector of expected levels in column

Value

The percentage of the total area covered by each LCZ

Examples

```
#LCZareas is not to be used directly by user.
```

`LCZlevelToOrderedString` *To make LCZ types appear in the proper order, it can be useful ton convert them into strings with added zeros*

Description

To make LCZ types appear in the proper order, it can be useful ton convert them into strings with added zeros

Usage

```
LCZlevelToOrderedString(LCZlevels)
```

Arguments

`LCZlevels`	a vector containing LCZlevels from 1 to 10 and 101 to 107

Value

the input with leading zeros for standard LCZ types, in order to respect alphabetical order in graphs

---

`levCol` *Manages the levels and the colors of the LCZ columns*

Description

Manages the levels and the colors of the LCZ columns

Usage

```
levCol(sf, column, drop = FALSE, useStandCol = FALSE, ...)
```

Arguments

sf	is the input sf file
column	is the column that contains the data
drop	is set to TRUE if one wants to drop unused levels, in case column is a factor with unused levels
useStandCol	is set to TRUE if one wants recognized standard levels to be associated with standard colors. Default is FALSE
...	other parameters specified, expected a vector of levels and a vector of colors whose name must begin with colors. Other cases are handled to enhance usability.

Value

output is a list containing levelsColors, a named vector, which names are the levels present in the data and which values are the associated colors, and case, a string specifying what case was encountered when producing the levels and colors.

Examples

```
# levCol2 is not to be used directly by user. It deals with levels and colors provided by the user
```

---

makeSectorsAndGroups *Creates sectors and groups for the drawChordDiagram function*

Description

Creates sectors and groups for the drawChordDiagram function

Usage

```
makeSectorsAndGroups(weightedFluxIn, groupOrder = NULL)
```

Arguments

weightedFluxIn	is typically the output of the function createWeightedFlux, and is expected to contain the following columns orig, dest and weightedFlux, whose names are quite self explanatory : orig is the origin LCZ type, dest is the destination LCZ type and weightedFlux is the percentage of area transfered from orig to dest
groupOrder	allows to specify in which order groups are visualized, usually the ones specified in labelMAtch argument of drawChordDiagram function

Value

a list containing vectors and groups for a chord diagram

`matConfLCZ` *Produces a confusion matrix between two sets of lcz on the same area (geoms do not need to be the same)*

Description

Produces a confusion matrix between two sets of lcz on the same area (geoms do not need to be the same)

Usage

```
matConfLCZ(
  sf1,
  column1,
  sf2,
  column2,
  typeLevels = .lczenv$typeLevelsDefault,
  plotNow = FALSE,
  wf1 = "Reference",
  wf2 = "Alternative",
  sfInt = NULL,
  drop = FALSE,
  ...
)
```

Arguments

`sf1`	is the sf dataset containing the first lcz classification
`column1`	is the column of the first data set containing the lcz to be compared
`sf2`	is the sf dataset containing the second lcz classification
`column2`	is the column of the second data set containing the lcz to be compared
`typeLevels`	by default the levels of lcz incoded from 1 to 10 and 101 to 107.
`plotNow`	shows the graphical representation of the confusion matrix
`wf1`	the name of the workflow used to produce data from sf1
`wf2`	the name of the workflow used to produce data from sf2
`sfInt`	allows to produce the confusion matrix directly on the intersected sf object if it was produced before
`drop`	if TRUE the empty LCZ types are dropped When comparing othe variable, like grouped LCZ, the grouped levels have to be specified.
`...`	a set of unspecified arguments, for instance when the produceAnalysis function calls other functions

Details

Most of the time this function will not be called directly by the user but by the compareLCZ function

Value

returns an object called matConfOut which contains matConfLong, a confusion matrix in a longer form, which can be written in a file by the compareLCZ function and is used by the geom_tile function of the ggplot2 package. matConfPlot is a ggplot2 object showing the confusion matrix. If plot=TRUE, it is also directly plotted marginAreas contains the sums of each LCZ area percAgg is the general agreement between the two sets of LCZ, expressed as a percentage of the total area of the study zone

Examples

```
matConfRedonBDTOSM <- matConfLCZ(sf1=redonBDT, column1 = 'LCZ_PRIMARY',
sf2 = redonOSM, column2 = 'LCZ_PRIMARY', plotNow = TRUE)
```

---

matConfLCZGlob *Compares two LCZ classifications on several locations*

Description

Compares two LCZ classifications on several locations

Usage

```
matConfLCZGlob(
  filePath = "",
  inputDf,
  wf1,
  wf2,
  geomID1 = "",
  column1,
  confid1 = "",
  geomID2 = "",
  column2,
  confid2 = "",
  sep = ";",
  typeLevels = "",
  plot = TRUE,
  ...
)
```

Arguments

filePath	: the full path to the inputDfto import
inputDf	: if filePath is an empty string inputDf is the name of an R dataframe containing the data
wf1	: a string indicating the origin of the LCZ classification 1
wf2	: a string indicating the origin of the LCZ classification 1

geomID1	is the name of the column containing the ID of each spatial units from the first type of datasets in \'inputDf\', it allows to post-treat some of the units
column1	is the name of the column containing the Local Climatic Zone of the first type of datasets in the \'file\' dump file be analyzed
confid1	is the name of the column measuring the confidence given to the LCZ level in column1
geomID2	is the name of the column containing the ID of each spatial units from the second type of datasets in \'inputDf\'
column2	see column1 but for the second type of datasets
confid2	is the name of the column measuring the confidence given to the LCZ level in column2
sep	: the seperator used in the csv file
typeLevels	: by default levels of the LCZ classification are built reading the data, but one can input a character string vector of levels, for instance to produce several graphs with the same levels or levels in a specific order
plot	if TRUE, then confusion matrix plot is shown, else it is just returned in the matConfOut object
...	other parameters that may be passed from a function calling matConfLCZGlob

Value

matConfOut is a list containing : matconf, the confusion matrix in its longer form ; matConfPlot, a ggplot2 plot of said matrix ; areas, the summed area per levels for both LCZ classification ; and pourcAcc, the general agreement between classifications expressed in percent of areas

Examples

```
matConfLCZGlob(filePath= paste0(
system.file("extdata", package = "lczexplore"),"/bdtopo_2_2_osm.csv"),
file="bdtopo_2_2_osm.csv", wf1="bdt", wf2="osm",
geomID1="ID_RSU", column1="LCZ_PRIMARY", confid1="LCZ_UNIQUENESS_VALUE",
geomID2="ID_RSU.1", column2="LCZ_PRIMARY.1", confid2="LCZ_UNIQUENESS_VALUE.1",
sep=";",
typeLevels="", plot=TRUE)
testSource<-read.csv(paste0(
system.file("extdata", package = "lczexplore"),
"/bdtopo_2_2_osm.csv"), sep=";",header=TRUE)
matConfLCZGlob(filePath="",
inputDf = testSource, wf1="bdt", wf2="osm",
geomID1="ID_RSU", column1="LCZ_PRIMARY", confid1="LCZ_UNIQUENESS_VALUE",
geomID2="ID_RSU.1", column2="LCZ_PRIMARY.1", confid2="LCZ_UNIQUENESS_VALUE.1",
sep=";",
typeLevels="", plot=TRUE)
```

`matConfPlot` *Plots a tile graph of a confusion matrix between two sets of lcz on the same area*

Description

Plots a tile graph of a confusion matrix between two sets of lcz on the same area

Usage

```
matConfPlot(
  matConfLong,
  column1 = "lcz_primary",
  column2 = "lcz_primary.1",
  marginAreas = NULL,
  agreeColumn = "agreePercArea",
  wf1 = "reference",
  wf2 = "alternative"
)
```

Arguments

`matConfLong`	is a dataframe for the matrice of confusion/agreement, in the longform : first column is the LCZ level of the first workflow, second the level of the second workflow and the third column the percentage of surface of the level of the first classification classified in the level of the second classification
`column1`	is the name of the first classification
`column2`	is the name of the second classification
`marginAreas`	are the percentage of areas for LCZ types for both workflows (an output of matConfLCZ)
`agreeColumn`	is the name where the agreement between classification is stored
`wf1`	is the name of the first workflow (for labels)
`wf2`	is the name of the second workflow (for labels)

Details

Most of the time this function will not be called directly by the user but by the compareLCZ function

Value

returns a plot of a matrice of confusion importFrom forcats fct_recode

Examples

```
matConfRedonBDTOSM <- matConfLCZ(sf1 = redonBDT, column1 = 'LCZ_PRIMARY',
                                 sf2 = redonOSM, column2 = 'LCZ_PRIMARY', plot = TRUE)
matConfLong <- matConfRedonBDTOSM$matConf
marginAreas <- matConfRedonBDTOSM$areas
matConfPlot(matConfLong,
        column1 = "LCZ_PRIMARY", column2 = "LCZ_PRIMARY.1", agreeColumn = "agreePercArea",
           wf1 = "reference", wf2 = "alternative", marginAreas = marginAreas)
```

`plotSankeyfiedLCZ` *Plots a sankey graph between two LCZ workflows*

Description

Plots a sankey graph between two LCZ workflows

Usage

```
plotSankeyfiedLCZ(
  sankeyfied,
  plotNow = TRUE,
  colorMap = NULL,
  v_space = "auto"
)
```

Arguments

`sankeyfied`	is the data as produced by the prepareSankeyLCZ function
`plotNow`	is set to TRUE by default, if set to FAULT the graph is not plotted.
`colorMap`	is a vector of colors whose names cover the valus of the nodes in sankeyfied
`v_space`	allows a separation between LCZ type bars

Value

returns a sankey plot of LCZ workflows on the same areas

Examples

```
dirList<-list.dirs(paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs"))[-1]
allLocIntersected<-concatIntersectedLocations(
dirList = dirList, locations = c("Redon", "Arville"))
testSankey<-prepareSankeyLCZ(intersectedDf = allLocIntersected
 , wf1 = "wudapt", wf2 = "osm")
testSankeyPlot<-plotSankeyfiedLCZ(
sankeyfied = testSankey, plotNow=TRUE)
```

plotSummarisedRSUs *For a given LCZ sf object, plots the number of geometries and an average area indicator per level of LCZ*

Description

For a given LCZ sf object, plots the number of geometries and an average area indicator per level of LCZ

Usage

```
plotSummarisedRSUs(
  summarisedSfIn,
  workflowNames = c("wudapt", "osm", "bdt"),
  plotNow = TRUE,
  graphPath = ""
)
```

Arguments

summarisedSfIn	the output of summariseRSU function
workflowNames	contain the names of the workflows
plotNow	: if TRUE the plot will be displayed in the session
graphPath	: a valid directory path where th plot will be saved (for now an empty string to avoid saving in the working directory)

Value

the number of geometries (Reference Spatial units or RSUs) and their mean area per level of LCZ, and the same after agregatting geometries with same level of LCZ which touch each other

Examples

```
dirPath<-paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs")
allLocAllWfs<-importConcatMultipleLocsLCZvect(
 dirPath = dirPath,
    locations = c("Redon", "Arville"),
    workflowNames = c("osm","bdt","wudapt"),
 missingGeomsWf= "osm",
 refWf = NULL,
 refLCZ = "Unclassified",
 residualLCZvalue = "Unclassified",
 column = "lcz_primary"
 )
summarisedRSUs<-summariseRSUs(
     allLocAllWfs,
     aggregatingColumns = c("wf", "lcz_primary"))
```

```
plotSummarisedRSUs(
     summarisedSfIn = summarisedRSUs,
     workflowNames = c("wudapt"= "wud", "osm", "bdt"))
```

`prepareSankeyLCZ` *Prepares the data to produce a sankey graph between two LCZ workflows*

Description

Prepares the data to produce a sankey graph between two LCZ workflows

Usage

```
prepareSankeyLCZ(intersectedDf, wf1, wf2, colorMap)
```

Arguments

`intersectedDf`	is an sf object or a data frame which contains lcz values for at least two workflows on the same geometries, and the area of the geometries
`wf1`	is the column name where the LCZ value of the first workflow are stored
`wf2`	wf1 is the column name where the LCZ value of the first workflow are stored
`colorMap`	a vector containing colors and whose names are the levels of the data to plot

Value

an object to feed plotSankeyLCZ

Examples

```
dirPath<-paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs")
allLocConcatenated<-importMultipleLocsLCZvect(
  dirPath = dirPath, locations = c("Arville", "Redon"))
allLocIntersected<-createIntersect(allLocConcatenated, columns = rep("lcz_primary", 4),
workflowNames = c("osm","bdt","wudapt"))
testSankey<-prepareSankeyLCZ(intersectedDf = allLocIntersected
 , wf1 = "wudapt", wf2 = "osm")
```

redonBDT *RedonBDT*

Description

LCZ for Redon as produced by GeoClimate on BDTOPO data #'

Usage

```
redonBDT
```

Format

An sf object with 3 variables and a geometry :

`ID_RSU` index identifying the Reference spatial unit

`LCZ_PRIMARY` Local Climate Zone encoded according to Stewad and Oke approach, 1 to 10 then 101 to 107

`LCZ_UNIQUENESS_VALUE` indicates how the LCZ_PRIMARY value was likely to be the only one suitable for this RSU, similar to a confidence value

`geometry` containes the geometry of the RSU for all the spatial operations allowed by the sf class

Source

"https://github.com/orbisgis/geoclimate"

redonOSM *RedonOSM*

Description

RedonOSM

Usage

```
redonOSM
```

Format

LCZ for Redon as produced by GeoClimate on OpenStreetMap data

`ID_RSU` index identifying the Reference spatial unit

`LCZ_PRIMARY` Local Climate Zone encoded according to Stewad and Oke approach, 1 to 10 then 101 to 107

`LCZ_UNIQUENESS_VALUE` indicates how the LCZ_PRIMARY value was likely to be the only one suitable for this RSU, similar to a confidence value

`geometry` containes the geometry of the RSU for all the spatial operations allowed by the sf class

Source

"https://github.com/orbisgis/geoclimate"

`showAlterLCZ` *Produces a simple representation of the LCZ contained in an sf file with non standard LCZ type encoding*

Description

Produces a simple representation of the LCZ contained in an sf file with non standard LCZ type encoding

Usage

```
showAlterLCZ(
  sf,
  title = "",
  wf = "",
  column,
  drop = FALSE,
  useStandCol = FALSE,
  tryGroup = TRUE,
  plotNow = TRUE,
  labelType = "long",
  repr = "alter",
  naAsUnclassified = TRUE,
  noPercAlter = FALSE,
  addBorders = FALSE,
  ...
)
```

Arguments

`sf`	is the sf file which contains the LCZ column to be plotted.
`title`	allows the user to set the title of the plot
`wf`	is the workflow used to produced the LCZ. "bdt" and "osm" indicate that LCZ were produced by GeoClimate using the BD_TOPO_V2 or the Open Street Map data as input, respectively. If the LCZ were produced by WUDAPT, use "wudapt".
`column`	is the column that contains the LCZ.
`drop`	indicates if you want to show the levels present in no geometry.
`useStandCol`	is set to TRUE implies that any levels detected as a standard LCZ level will receive the standard associated color
`tryGroup`	is set to TRUE when one wants to group and plot on the fly

plotNow	is inherited from showLCZ, needs to be specified so it is not considered as part of ... arguments which contain grouping and color arguments
labelType	is inherited from showLCZ, same as previous argument
repr	is inherited from showLCZ, same as previous argument
naAsUnclassified	if TRUE, affects Unclassified value to NAs in the LCZ column
noPercAlter	allows to not show the percentage area in the alter representation
addBorders	if TRUE the borders of the geometries are visible
...	these dynamic dots allow you to pass arguments to specify levels expected in your dataset and colors associated to these levels when not in the standard representation. You can pas your levels through a vector and you colors through another vector called colors. For more details about this, read the "lcz_explore_alter" vignette.

Examples

```
# On grouped data, use the alter value for the repr argument.
redonBDTgrouped<-groupLCZ(redonBDT,column="LCZ_PRIMARY",
urban=c("1","2","3","4","5","6","7","8","9"),
industry="10", vegetation=c("101","102","103","104"),
impervious="105",pervious="106",water="107")
# For repr="alter", you can specify colors and levels this way :
showLCZ(redonBDTgrouped,column="grouped",repr="alter",
LCZlevels=c("urban","industry","vegetation","impervious","pervious","water"),
colors=c("red","black","green","grey","burlywood","blue"),wf="BD TOPO")
```

---

showLCZ *Produces a simple representation of the LCZ contained in an sf file*

Description

Produces a simple representation of the LCZ contained in an sf file

Usage

```
showLCZ(
  sf,
  title = "",
  wf = "",
  column = "LCZ_PRIMARY",
  repr = "standard",
  drop = FALSE,
  useStandCol = FALSE,
  tryGroup = TRUE,
  naAsUnclassified = TRUE,
```

```
  noPercAlter = FALSE,
  plotNow = TRUE,
  addBorders = FALSE,
  labelType = "long",
  ...
)
```

Arguments

sf	is the sf file which contains the LCZ column to be plotted.
title	allows the user to set the title of the plot
wf	is the workflow used to produced the LCZ. "bdt" and "osm" indicate that LCZ were produced by GeoClimate using the BD_TOPO_V2 or the Open Street Map data as input, respectively. If the LCZ were produced by WUDAPT, use "wudapt".
column	is the column that contains the LCZ.
repr	indicates if the sf dataset contains standarde LCZ levels or grouped LCZ. If "standard" then an optimal set of colors is used to produce the plotted map. Else, colors can be specified with the colors argument.
drop	indicates if you want to show the levels present in no geometry.
useStandCol	is set to TRUE implies that any levels detected as a standard LCZ level will receive the standard associated color
tryGroup	is set to TRUE when one wants to group and plot on the fly
naAsUnclassified	if TRUE, affects Unclassified value to NAs in the LCZ column
noPercAlter	allows to not show the percentage area in the alter representation
plotNow	if FALSE the plot is generated but not shown (needs to be stored in an object)
addBorders	if TRUE the borders of the geometries are visible
labelType	in the standard representation, this value can take the values "long" (number, name and percentage area for each LCZ types), "short" (value and percentage), "very short" (only number and percentage), "no perc" (no area percentage).
...	these dynamic dots allow you to pass arguments to specify levels expected in your dataset and colors associated to these levels when not in the standard representation. You can pas your levels through a vector and you colors through another vector called colors. For more details about this, read the "lcz_explore_alter" vignette.

Value

return the plot of the LCZ levels

Examples

```
# On original LCZ levels, use the standard value for the repr argument.
showLCZ(redonBDT, column="LCZ_PRIMARY", repr="standard")
```

```
# On grouped data, use the alter value for the repr argument.
redonBDTgrouped<-groupLCZ(redonBDT,column="LCZ_PRIMARY",
urban=c("1","2","3","4","5","6","7","8","9"),
industry="10", vegetation=c("101","102","103","104"),
impervious="105",pervious="106",water="107")
# For repr="alter", you can specify colors and levels this way :
showLCZ(sf = redonBDTgrouped, column = "grouped", repr = "alter",
LCZlevels = c("urban","industry","vegetation","impervious","pervious","water"),
colors = c("red","black","green","grey","burlywood","blue"), wf="BD TOPO")
```

`showStandardLCZ` *Produces a simple representation of the LCZ contained in an sf file when standard LCZ type format is used*

Description

Produces a simple representation of the LCZ contained in an sf file when standard LCZ type format is used

Usage

```
showStandardLCZ(
  sf,
  title = "",
  wf = "",
  column = "LCZ_PRIMARY",
  repr = "standard",
  drop = FALSE,
  naAsUnclassified = TRUE,
  addBorders = FALSE,
  labelType = "long",
  ...
)
```

Arguments

`sf`	is the sf file which contains the LCZ column to be plotted.
`title`	allows the user to set the title of the plot
`wf`	is the workflow used to produced the LCZ. "bdt" and "osm" indicate that LCZ were produced by GeoClimate using the BD_TOPO_V2 or the Open Street Map data as input, respectively. If the LCZ were produced by WUDAPT, use "wudapt".
`column`	is the column that contains the LCZ.
`repr`	indicates if the sf dataset contains standarde LCZ levels or grouped LCZ. If "standard" then an optimal set of colors is used to produce the plotted map. Else, colors can be specified with the colors argument.

drop	indicates if you want to show the levels present in no geometry.
naAsUnclassified	if TRUE, affects Unclassified value to NAs in the LCZ column
addBorders	if TRUE the borders of the geometries are visible
labelType	in the standard representation, this value can take the values "long" (number, name and percentage area for each LCZ types), "short" (value and percentage), "very short" (only number and percentage), "no perc" (no area percentage).
...	these dynamic dots allow you to pass arguments to specify levels expected in your dataset and colors associated to these levels when not in the standard representation. You can pas your levels through a vector and you colors through another vector called colors. For more details about this, read the "lcz_explore_alter" vignette.

Value

return the plot of the LCZ levels

Examples

```
# On original LCZ levels, use the standard value for the repr argument.
showLCZ(redonBDT, column="LCZ_PRIMARY", repr="standard")
```

---

standLevCol *Detects standard LCZ levels and associate standard colors to them*

Description

Detects standard LCZ levels and associate standard colors to them

Usage

```
standLevCol(levels, colors = "", useStandCol = FALSE)
```

Arguments

levels	is the vector of the LCZ levels present in a dataset
colors	is the vector of colors one intends to use to visualize the LCZ of the dataset
useStandCol	allows to say if standard colors should replace user specified colors when standard levels are detected.

Value

output is a list containing levelColors, a named vector, which names are the levels present in the data and which values are the associated colors, and case, a string spcifying what case was encountered when producing the levels and colors.

Examples

```
# standLevCol is not to be used directly by user.
# It deals with levels and colors provided by the user and
# tries to replace user color by standard color
# when a standard level of LCZ is recognized
```

---

summariseRSUs *For a given LCZ sf object, returns the number of geometries and their mean area per LCZ type*

Description

For a given LCZ sf object, returns the number of geometries and their mean area per LCZ type

Usage

```
summariseRSUs(sfIn, aggregatingColumns = "lcz_primary", trimValue = 0)
```

Arguments

sfIn	an sf objects that contains the geometry and LCZ levels
aggregatingColumns	should be the name of the LCZ types column, but can be any factor variable names
trimValue	a trim parameter to feed the mean function in order to avoid letting artificially small or big geometries to influence the average area results

Value

the number of geometries (Reference Spatial units or RSUs) and their mean area per level of LCZ, and the same after agregatting geometries with same level of LCZ which touch each other

Examples

```
dirPath<-paste0(
system.file("extdata", package = "lczexplore"),"/multipleWfs")
allLocAllWfs<-importConcatMultipleLocsLCZvect(
 dirPath = dirPath, locations = c("Redon", "Arville"),
workflowNames = c("osm","bdt","wudapt"),
 missingGeomsWf= "osm",
 refWf = NULL,
 refLCZ = "Unclassified",
 residualLCZvalue = "Unclassified",
 column = "lcz_primary"
)
summarisedRSUs<-summariseRSUs(allLocAllWfs, aggregatingColumns = c("wf", "lcz_primary"))

# After aggregating adjacent RSU of same LCZ type
```

```
ASUallLocAllWfs <- aggregateRSUsByLCZ(
     allLocAllWfs,
     column = "lcz_primary", wfColumn = "wf",
     locationColumn = "location", aggregateBufferSize = 0.5)
summarisedASUs<-summariseRSUs(ASUallLocAllWfs, aggregatingColumns = c("wf", "lcz_primary"))

plotSummarisedRSUs(
     summarisedSfIn = summarisedASUs,
     workflowNames = c("wud" = "wudapt", "osm", "bdt"))
```

`workflowAgreeAreas` *From the output of compare multiple, cmputes which workflows agree the most regarding the area of agreement*

Description

From the output of compare multiple, cmputes which workflows agree the most regarding the area of agreement

Usage

```
workflowAgreeAreas(sfMultiCompLong)
```

Arguments

`sfMultiCompLong` the sfLong output of compareMultipleLCZ function

Value

the pairwise agreement between workflows, sorted by decreasing agreeing areas

Examples

```
sfList<-importMultipleLCZvect(dirPath = paste0(
system.file("extdata", package = "lczexplore"),
"/multipleWfs/Arville"),
workflowNames = c("osm","bdt","wudapt"), location = "Arville")
ArvilleIntersect <- createIntersect(
 sfList = sfList, columns = rep("lcz_primary", 4),
 workflowNames = c("osm","bdt","wudapt"))
ArvilleMultipleComparison<-compareMultipleLCZ(
 sfInt = ArvilleIntersect,
 columns = c("osm","bdt","wudapt"),
 trimPerc = 0.5)
ArvilleWorkflowAgreement<-workflowAgreeAreas(ArvilleMultipleComparison$sfIntLong)
```

Index

∗ **datasets**
redonBDT, 55
redonOSM, 55

addMissingRSUs, 3
aggregateRSUsByLCZ, 4
areColors, 5

barplotLCZaLocation, 6
barplotLCZfromIntersect, 7

checkColnameCase, 8
checkDirSlash, 9
checkWorkflowName, 9
CohenKappa, 10
compareLCZ, 11
compareMultipleLCZ, 14
completeDT, 15
computeAgreements, 16
computeConsensus, 16
concatAllLocsWorkflows, 17
concatAlocationWorkflows, 18
concatIntersectedLocations, 19
confidSensib, 20
createIntersect, 21
createWeightedFlux, 23

drawChordDiagram, 24
drawChordDiagramWithGrouping, 25
drawSectors, 27

fetchLCZ, 27

geoClimateCall, 28
geoClimateConfigFile, 29
groupLCZ, 31
groupLCZsuffix, 32

importConcatMultipleLocsLCZvect, 33
importLCZgc, 34
importLCZraster, 35
importLCZvect, 36
importLCZvectFromFile, 38
importLCZvectFromSf, 39
importMultipleLCZvect, 40
importMultipleLCZvectFromSession, 41
importMultipleLocsLCZvect, 42
importQualVar, 43
intersectAlocation, 44

LCZareas, 45
LCZlevelToOrderedString, 46
levCol, 46

makeSectorsAndGroups, 47
matConfLCZ, 48
matConfLCZGlob, 49
matConfPlot, 51

plotSankeyfiedLCZ, 52
plotSummarisedRSUs, 53
prepareSankeyLCZ, 54

redonBDT, 55
redonOSM, 55

showAlterLCZ, 56
showLCZ, 57
showStandardLCZ, 59
standLevCol, 60
summariseRSUs, 61

table, *10*

workflowAgreeAreas, 62