Package: map (via r-universe)

October 6, 2024

Type Package

Title Defines a meta class of geographical objects, the 'map' class, and associated tools

Description The map class is a collection of map objects ('sp', 'raster', 'sf'), with a number of metadata additions to enable powerful methods, e.g., for leaflet, reproducible GIS etc.

URL http://map.predictiveecology.org, https://github.com/PredictiveEcology/map

Date 2023-11-06

Version 0.0.5.9003

Depends R (>= 4.1)

Imports backports, data.table (>= 1.10.4), fastdigest, fasterize, methods, parallelly, pemisc (>= 0.0.4.9008), quickPlot, reproducible (>= 1.2.6.9019), raster (>= 2.8.4), rgeos, sf, sp, stats, terra (>= 1.7-0), tiler (>= 0.2.6), utils

Suggests covr, gdalUtils, knitr, rgdal, rmarkdown, SpaDES.tools, testthat, usethis

Remotes ropensci/tiler, gearslaboratory/gdalUtils, PredictiveEcology/pemisc@development, PredictiveEcology/reproducible@development, cran/rgdal, cran/rgeos, PredictiveEcology/SpaDES.tools@development

Encoding UTF-8

Language en-CA

License GPL-3

VignetteBuilder knitr, rmarkdown

BugReports https://github.com/PredictiveEcology/map/issues

ByteCompile yes

Collate 'GIS.R' 'buildMetadata.R' 'makeTiles.R' 'map-analysis.R' 'map-class.R' 'map-package.R' 'map.R' 'rbindlistAG.R' 'zzz.R'

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Repository https://predictiveecology.r-universe.dev

RemoteUrl https://github.com/PredictiveEcology/map

RemoteRef development

RemoteSha 16e0a88694c1023bfc8416369582bb94a4ca42e6

Contents

`.rasterToMemory` `.rasterToMemory`

Description

`.rasterToMemory`

Usage

```
.rasterToMemory(x, ...)
```

Arguments

`x`	A `Raster*` object
`...`	Additional arguments passed to `raster`

`area,map-method` *Calculate area of (named) objects the* `map` *obj*

Description

Calculate area of (named) objects the `map` obj

Usage

```
## S4 method for signature 'map'
area(x)
```

Arguments

`x`	Raster* or SpatialPolygons object

See Also

Other mapMethods: `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `rasters()`, `studyAreaName()`, `studyArea()`

`areaAndPolyValue` `areaAndPolyValue`

Description

Determine the area of each zone in a raster. TODO: improve description

Usage

```
areaAndPolyValue(ras)
```

Arguments

`ras`	A `Raster*` object

Value

list containing: `sizeInHa`, the area; and `polyID`, the polygon ID.

buildMetadata *Build* `map` *obj metadata table*

Description

Build `map` obj metadata table

Usage

```
buildMetadata(
  metadata,
  isStudyArea,
  isRasterToMatch,
  layerName,
  obj,
  columnNameForLabels,
  objHash,
  leaflet,
  envir,
  ...
)
```

Arguments

`metadata` TODO: description needed

`isStudyArea` TODO: description needed

`isRasterToMatch`
Logical. Is this(these) layer(s) the `rasterToMatch` layers. If `TRUE`, then this layer can be accessed by `rasterToMatch(map)`

`layerName` TODO: description needed

`obj` TODO: description needed

`columnNameForLabels`
TODO: description needed

`objHash` TODO: description needed

`leaflet` Logical or Character vector of path(s) to write tiles. If `TRUE` or a character vector, then this layer will be added to a leaflet map. For `RasterLayer` object, this will trigger a call to `gdal2tiles`, making tiles. If path is not specified, it will be the current path. The tile base file path will be `paste0(layerName, "_", rndstr(1, 6))`.

`envir` TODO: description needed

`...` Additional arguments.

`crs,map-method` *Extract the crs of a* `map`

Description

Extract the crs of a `map`

Usage

```
## S4 method for signature 'map'
crs(x, ...)
```

Arguments

`x` Raster* or Spatial object

`...` additional arguments. None implemented

See Also

Other mapMethods: `area,map-method`, `mapRm()`, `rasterToMatch,map-method`, `rasters()`, `studyAreaName()`, `studyArea()`

`fasterize2` *Fasterize with crop & spTransform first*

Description

Fasterize with crop & spTransform first

Usage

```
fasterize2(emptyRaster, polygonToFasterize, field)
```

Arguments

`emptyRaster` An empty raster with res, crs, extent all correct for to pass to `fasterize`

`polygonToFasterize` passed to `fasterize`, but it will be cropped first if `extent(emptyRaster) < extent(polygonToFasterize)`

`field` passed to `fasterize`

`gdal_polygonizeR` *Polygonize with GDAL*

Description

Based on https://johnbaumgartner.wordpress.com/2012/07/26/getting-rasters-into-shape-from-r/.

Usage

```
gdal_polygonizeR(
  x,
  outshape = NULL,
  gdalformat = "ESRI Shapefile",
  pypath = NULL,
  readpoly = TRUE,
  quiet = TRUE
)
```

Arguments

`x`	TODO: description needed
`outshape`	TODO: description needed
`gdalformat`	TODO: description needed
`pypath`	TODO: description needed
`readpoly`	TODO: description needed
`quiet`	TODO: description needed

`leafletTiles` *Extract leaflet tile paths from a `map` obj*

Description

Extract leaflet tile paths from a `map` obj

Usage

```
leafletTiles(map)
```

Arguments

`map`	A `map` class obj

Value

A vector of paths indicating the relative paths. Any layers that don't have leaflet tiles will return NA.

makeTiles *Make tiles (pyramids) using* `gdal2tiles`

Description

Make tiles (pyramids) using `gdal2tiles`

Usage

```
makeTiles(tilePath, obj, overwrite = FALSE, ...)
```

Arguments

`tilePath`	A director to write tiles
`obj`	A raster objects with or without file-backing
`overwrite`	Logical. If `FALSE`, and the director exists, then it will not overwrite any files.
`...`	Passed to `reproducible::projectInputs` e.g., `useGDAL`

map-class *The* `map` *class*

Description

Contains a common system for organzing vector and raster layers, principally for use with **leaflet** and **shiny**.

Slots

`metadata` `data.table` with columns describing metadata of map objects in `maps` slot.

`.xData` Named environment of map-type objects (e.g., `sf`, `Raster*`, `Spatial*`. Each entry may also be simply an environment, which indicates where to find the object, i.e., via `get(layerName, envir = environment)`.

`CRS` The common crs of all layers

`paths` File paths. A named list of paths. The default is a list of length 2, `dataPath` and `tilePath`

`analyses` A `data.table` or `data.frame` of the types of analyses to perform.

`analysesData` A `data.table` or `data.frame` of the results of the analyses.

mapAdd *Append a spatial object to map*

Description

If `isStudyArea` = `TRUE`, then several things will be triggered:

1. This layer will be added to metadata with `studyArea` set to `max(studyArea(map)) + 1`.
2. update CRS slot to be the CRS of the study area.

Usage

```
mapAdd(obj, map, layerName, overwrite = getOption("map.overwrite", FALSE), ...)

## Default S3 method:
mapAdd(
  obj = NULL,
  map = new("map"),
  layerName = NULL,
  overwrite = getOption("map.overwrite"),
  columnNameForLabels = 1,
  leaflet = FALSE,
  isStudyArea = FALSE,
  isRasterToMatch = FALSE,
  envir = NULL,
  useCache = TRUE,
  useParallel = getOption("map.useParallel", FALSE),
  ...
)
```

Arguments

`obj` Optional spatial object, currently `RasterLayer`, `SpatialPolygons`.

`map` Optional map object. If not provided, then one will be created. If provided, then the present `object` or options passed to `prepInputs` e.g., `url`, will be appended to this `map`.

`layerName` Required. A label for this map layer. This can be the same as the object name.

`overwrite` Logical. If `TRUE` and this `layerName` exists in the `map`, then it will replace the existing object. Default is `getOption("map.overwrite")`

`...` Additonal arguments passed to `reproducible::postProcess()`, `reproducible::projectInputs()`, `reproducible::fixErrors()`, and `reproducible::prepInputs()`.

`columnNameForLabels`
A character string indicating which column to use for labels. This is currently only used if the object is a `SpatialPolygonsDataFrame`.

`leaflet` Logical or Character vector of path(s) to write tiles. If `TRUE` or a character vector, then this layer will be added to a leaflet map. For `RasterLayer` object, this will trigger a call to `gdal2tiles`, making tiles. If path is not specified, it will be the current path. The tile base file path will be `paste0(layerName, "_", rndstr(1, 6))`.

`isStudyArea` Logical. If `TRUE`, this will be assigned the label, "StudyArea", and will be passed into `prepInputs` for any future layers added.

`isRasterToMatch` Logical indicating ... TODO: need description

`envir` An optional environment. If supplied, then the obj will not be placed "into" the maps slot, rather the environment label will be placed into the maps slot. Upon re

`useCache` Logical. If `TRUE`, then internal calls to `Cache` will be used. Default is `TRUE`

`useParallel` Logical. If `TRUE`, then if there is more than one calculation to do at any stage, it will create and use a parallel cluster via `makeOptimalCluster`. If running analyses in parallel, it may be useful to pass a function (via `.clInit`) to be run on each of the nodes immediately upon cluster creation (e.g., to set options).

Examples

```
## Not run:
library(sp)
library(raster)
library(reproducible)
cwd <- getwd()
setwd(tempdir())
coords <- structure(c(-122.98, -116.1, -99.2, -106, -122.98,
                      59.9, 65.73, 63.58, 54.79, 59.9),
                    .Dim = c(5L, 2L))
Sr1 <- Polygon(coords)
Srs1 <- Polygons(list(Sr1), "s1")
StudyArea <- SpatialPolygons(list(Srs1), 1L)
crs(StudyArea) <- paste("+init=epsg:4326 +proj=longlat +datum=WGS84",
                        "+no_defs +ellps=WGS84 +towgs84=0,0,0")
StudyArea <- SpatialPolygonsDataFrame(StudyArea,
                           data = data.frame(ID = 1, shinyLabel = "zone2"),
                           match.ID = FALSE)

ml <- mapAdd(StudyArea, isStudyArea = TRUE, layerName = "Small Study Area",
             poly = TRUE, analysisGroup2 = "Small Study Area")

if (require("SpaDES.tools", quietly = TRUE)) {
  options(map.useParallel = FALSE)
  smallStudyArea <- randomPolygon(studyArea(ml), 1e5)
  smallStudyArea <- SpatialPolygonsDataFrame(smallStudyArea,
                           data = data.frame(ID = 1, shinyLabel = "zone1"),
                           match.ID = FALSE)
  ml <- mapAdd(smallStudyArea, ml, isStudyArea = TRUE, filename2 = NULL,
               analysisGroup2 = "Smaller Study Area",
               poly = TRUE,
               layerName = "Smaller Study Area") # adds a second studyArea within 1st

  rasTemplate <- raster(extent(studyArea(ml)), res = 0.001)
  tsf <- randomPolygons(rasTemplate, numTypes = 8)*30
  crs(tsf) <- crs(ml)
  vtm <- randomPolygons(tsf, numTypes = 4)
  levels(vtm) <- data.frame(ID = sort(unique(vtm[])),
                            Factor = c("black spruce", "white spruce", "aspen", "fir"))
  crs(vtm) <- crs(ml)
  ml <- mapAdd(tsf, ml, layerName = "tsf1",
               filename2 = "tsf1.tif", # to postProcess
               # to map object
               tsf = "tsf1.tif", # to column in map@metadata
               analysisGroup1 = "tsf1_vtm1",  # this is the label for analysisGroup1
             leaflet = TRUE, # to column in map@metadata; used for visualizing in leaflet
               overwrite = TRUE)
  ml <- mapAdd(vtm, ml, filename2 = "vtm1.grd",
               layerName = "vtm1",
               vtm = "vtm1.grd",
               analysisGroup1 = "tsf1_vtm1", leaflet = TRUE, overwrite = TRUE)

  ageClasses <- c("Young", "Immature", "Mature", "Old")
  ageClassCutOffs <- c(0, 40, 80, 120)

  # add an analysis -- this will trigger analyses because there are already objects in the map
  #    This will trigger 2 analyses:
  #    LeadingVegTypeByAgeClass on each raster x polygon combo (only 1 currently)
  #    so there is 1 raster group, 2 polygon groups, 1 analyses - Total 2, 2 run now
  ml <- mapAddAnalysis(ml, functionName ="LeadingVegTypeByAgeClass",
                       ageClasses = ageClasses, ageClassCutOffs = ageClassCutOffs)
  # add an analysis -- this will trigger analyses because there are already objects in the map
  #    This will trigger 2 more analyses:
  #    largePatches on each raster x polygon combo (only 1 currently)
  #    so there is 1 raster group, 2 polygon groups, 2 analyses - Total 4, only 2 run now
  ml <- mapAddAnalysis(ml, functionName = "LargePatches", ageClasses = ageClasses,
                       id = "1", labelColumn = "shinyLabel",
                       ageClassCutOffs = ageClassCutOffs)

  # Add a second polygon, trigger
  smallStudyArea2 <- randomPolygon(studyArea(ml), 1e5)
  smallStudyArea2 <- SpatialPolygonsDataFrame(smallStudyArea2,
                           data = data.frame(ID = 1, shinyLabel = "zone1"),
                           match.ID = FALSE)
  # add a new layer -- this will trigger analyses because there are already analyese in the map
  #   This will trigger 2 more analyses ... largePatches on each *new* raster x polygon combo
  #   (now there are 2) -- so there is 1 raster group, 3 polygon groups, 2 analyses - Total 6
  ml <- mapAdd(smallStudyArea2, ml, isStudyArea = FALSE, filename2 = NULL, overwrite = TRUE,
               analysisGroup2 = "Smaller Study Area 2",
               poly = TRUE,
               layerName = "Smaller Study Area 2") # adds a second studyArea within 1st

  # Add a *different* second polygon, via overwrite. This should trigger new analyses
  smallStudyArea2 <- randomPolygon(studyArea(ml), 1e5)
```

```
  smallStudyArea2 <- SpatialPolygonsDataFrame(smallStudyArea2,
                           data = data.frame(ID = 1, shinyLabel = "zone1"),
                           match.ID = FALSE)
  # add a new layer -- this will trigger analyses because there are already analyses in the map
  #   This will trigger 2 more analyses ... largePatches on each *new* raster x polygon combo
  #   (now there are 2) -- so there is 1 raster group, 3 polygon groups, 2 analyses - Total 6
  ml <- mapAdd(smallStudyArea2, ml, isStudyArea = FALSE, filename2 = NULL, overwrite = TRUE,
               analysisGroup2 = "Smaller Study Area 2",
               poly = TRUE,
               layerName = "Smaller Study Area 2") # adds a second studyArea within 1st

  # Add a 2nd pair of rasters
  rasTemplate <- raster(extent(studyArea(ml)), res = 0.001)
  tsf2 <- randomPolygons(rasTemplate, numTypes = 8)*30
  crs(tsf2) <- crs(ml)
  vtm2 <- randomPolygons(tsf2, numTypes = 4)
  levels(vtm2) <- data.frame(ID = sort(unique(vtm2[])),
                            Factor = c("black spruce", "white spruce", "aspen", "fir"))
  crs(vtm2) <- crs(ml)
  ml <- mapAdd(tsf2, ml, filename2 = "tsf2.tif", layerName = "tsf2",
               tsf = "tsf2.tif",
               analysisGroup1 = "tsf2_vtm2", leaflet = TRUE, overwrite = TRUE)
  ml <- mapAdd(vtm2, ml, filename2 = "vtm2.grd", layerName = "vtm2",
               vtm = "vtm2.grd",
               analysisGroup1 = "tsf2_vtm2", leaflet = TRUE, overwrite = TRUE)

  # post hoc analysis of data
  #  use or create a specialized function that can handle the analysesData slot
  ml <- mapAddPostHocAnalysis(map = ml, functionName = "rbindlistAG",
                              postHocAnalysisGroups = "analysisGroup2",
                              postHocAnalyses = "all")
}

## cleanup
setwd(cwd)
unlink(tempdir(), recursive = TRUE)

## End(Not run)
```

`mapAddAnalysis` *Add an analysis to a* `map` *object*

Description

TODO: description needed

Usage

```
mapAddAnalysis(
  map,
  functionName,
  useParallel = getOption("map.useParallel", FALSE),
  ...
)
```

Arguments

map A `map` object

functionName The name of the analysis function to add

useParallel Logical indicating whether to use multiple threads. Defaults to `getOption("map.useParallel", FALSE)`.

... Additional arguments passed to `functionName`.

`mapAddPostHocAnalysis` *Add a post hoc analysis function to a* `map` *object*

Description

Add a post hoc analysis function to a `map` object

Usage

```
mapAddPostHocAnalysis(
  map,
  functionName,
  postHocAnalysisGroups = NULL,
  postHocAnalyses = "all",
  useParallel = getOption("map.useParallel", FALSE),
  ...
)
```

Arguments

map Optional map object. If not provided, then one will be created. If provided, then the present `object` or options passed to `prepInputs` e.g., `url`, will be appended to this `map`.

functionName A function that is designed for post hoc analysis of map class objects, e.g., `rbindlistAG`.

postHocAnalysisGroups
Character string with one `analysisGroups`, i.e., `"analysisGroup1"` or `"analysisGroup2"`.

postHocAnalyses
Character vector with `"all"`, (which will do all `analysisGroups`; default), or 1 or more of the the `functionNames` that are in the analyses slot.

useParallel — Logical. If TRUE, then if there is more than one calculation to do at any stage, it will create and use a parallel cluster via makeOptimalCluster. If running analyses in parallel, it may be useful to pass a function (via .clInit) to be run on each of the nodes immediately upon cluster creation (e.g., to set options).

... — Optional arguments to pass into functionName

mapAnalysis *Generic analysis for map objects*

Description

This is the workhorse function that runs any analyses described in map@analyses. It uses hashing, and will not rerun any analysis that already ran on identical inputs.

Usage

```
mapAnalysis(
  map,
  functionName = NULL,
  purgeAnalyses = NULL,
  useParallel = getOption("map.useParallel", FALSE),
  ...
)
```

Arguments

map — Optional map object. If not provided, then one will be created. If provided, then the present object or options passed to prepInputs e.g., url, will be appended to this map.

functionName — A function name that will be run on combinations of inputs in the map object. See details.

purgeAnalyses — A character string indicating which analysis group combination or part thereof (e.g., the name entered into the row under analysisGroup2 column of the map@metadata or a functionName.

useParallel — Logical. If TRUE, then if there is more than one calculation to do at any stage, it will create and use a parallel cluster via makeOptimalCluster. If running analyses in parallel, it may be useful to pass a function (via .clInit) to be run on each of the nodes immediately upon cluster creation (e.g., to set options).

... — Additonal arguments passed to reproducible::postProcess(), reproducible::projectInputs(), reproducible::fixErrors(), and reproducible::prepInputs().

Details

This function will do a sequence of things. First, it will run `expand.grid` on any columns whose names start with `analysisGroup`, creating a factorial set of analyses as described by these columns. It will assess the combinations against the arguments used by the `functionName`. For any `analysisGroup` that does not provide the correct arguments for the `functionName`, these `analysisGroups` will be omitted for that particular function. For efficiency, the function will then assess if any of these has already been run. For those that have not been run, it will then run the `functionName` on arguments that it finds in the `metadata` slot of the map object, as well as any arguments passed in here in the `...`. In general, the arguments being passed in here should be fixed across all analyses, while any that vary by analysis should be entered into the metadata table at the time of adding the layer to the map, via `mapAdd`.

Value

TODO

`mapRm` *Remove objects from a* `map`

Description

Remove objects from a `map`

Usage

```
mapRm(map, layer, ask = TRUE, ...)

## Default S3 method:
mapRm(map = NULL, layer = NULL, ask = TRUE, ...)
```

Arguments

`map`	TODO: document this
`layer`	TODO: document this
`ask`	TODO: document this
`...`	TODO: document this

See Also

Other mapMethods: `area,map-method`, `crs,map-method`, `rasterToMatch,map-method`, `rasters()`, `studyAreaName()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `rasterToMatch,map-method`, `rasters()`, `studyAreaName()`, `studyArea()`

Examples

```
if (require("SpaDES.tools", quietly = TRUE)) {
  p <- terra::vect(cbind(-120, 60), crs = "epsg:4326") |>
    SpaDES.tools::randomPolygon(area = 1e5) |>
    sf::st_as_sf() |>
    sf::as_Spatial()
  m <- mapAdd(p, layerName = "p")
  m

  m <- mapRm(m, "p")
  m
}
```

`metadata` *Extract the metadata obj*

Description

Methods for specific classes exist.

Usage

```
metadata(x)

## S3 method for class 'Raster'
metadata(x)

## S3 method for class 'map'
metadata(x)
```

Arguments

`x`	TODO: description needed

`rasters` *Extract rasters in the* `map` *object*

Description

This will extract all objects in or pointed to within the `map`.

Usage

```
rasters(map)

## S3 method for class 'map'
rasters(map)

sp(map)

## S3 method for class 'map'
sp(map)

sf(map)

## S3 method for class 'map'
sf(map)

spatialPolygons(map)

spatialPoints(map)

maps(map, class = NULL, layerName = NULL)
```

Arguments

`map`	A `map` class obj
`class`	If supplied, this will be the class of objects returned. Default is `NULL` which is "all", meaning all objects in the `map` object.
`layerName`	TODO: description needed

Value

A list of maps (i.e., sp, raster, or sf objects) of class `class`

See Also

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `studyAreaName()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `studyAreaName()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `studyAreaName()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `studyAreaName()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `studyAreaName()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `studyAreaName()`, `studyArea()`

`rasterToMatch,map-method` *Extract the rasterToMatch(s) from a* `x`

Description

If `layer` is not provided and there is more than one `studyArea`, then this will extract the last one added.

Usage

```
## S4 method for signature 'map'
rasterToMatch(x, layer = 1)
```

Arguments

`x`	TODO: describe this
`layer`	TODO: describe this

See Also

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasters()`, `studyAreaName()`, `studyArea()`

`rbindlistAG` *Utility functions for grouping analyses in a* `map` *object*

Description

Utility functions for grouping analyses in a `map` object

Usage

```
rbindlistAG(map, functionName, analysisGroups)
```

Arguments

`map`	Optional map object. If not provided, then one will be created. If provided, then the present `object` or options passed to `prepInputs` e.g., `url`, will be appended to this `map`.
`functionName`	TODO: description needed
`analysisGroups`	A character (length 1 currently), indicating which analysis group (e.g., "analysisGroup1") should be used to `rbindlist`. Can also specify `"all"` which will `rbindlist` all outputs.

Value

A list of `data.tables`.

`runMapAnalyses` runMapAnalyses

Description

TODO: description needed

Usage

```
runMapAnalyses(
  map,
  purgeAnalyses = NULL,
  useParallel = getOption("map.useParallel", FALSE),
  ...
)
```

Arguments

`map`	TODO
`purgeAnalyses`	TODO
`useParallel`	TODO
`...`	TODO

Value

TODO

`show,map-method` *Show method for map class objects*

Description

Show method for map class objects

Usage

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

Arguments

`object`	TODO: describe this

studyArea *Extract the studyArea(s) from a* map

Description

If layer is not provided and there is more than one studyArea, then this will extract the last one added.

Usage

```
studyArea(map, layer = NA, sorted = FALSE)

## S4 method for signature 'ANY'
studyArea(map, layer = NA, sorted = FALSE)

## S4 method for signature 'map'
studyArea(map, layer = NA, sorted = FALSE)

studyArea(map, layer = NA) <- value

## S4 replacement method for signature 'map'
studyArea(map, layer = NA) <- value
```

Arguments

map	TODO: document this
layer	TODO: document this
sorted	Logical. Should the numeric layer be referring to geographic area of the area or the order that the studyArea were placed into map object
value	The value to assign to the object.

See Also

Other mapMethods: area,map-method, crs,map-method, mapRm(), rasterToMatch,map-method, rasters(), studyAreaName()

Other mapMethods: area,map-method, crs,map-method, mapRm(), rasterToMatch,map-method, rasters(), studyAreaName()

Other mapMethods: area,map-method, crs,map-method, mapRm(), rasterToMatch,map-method, rasters(), studyAreaName()

Other mapMethods: area,map-method, crs,map-method, mapRm(), rasterToMatch,map-method, rasters(), studyAreaName()

studyAreaName *Map class methods*

Description

Tools for getting objects and metadata in and out of a `map` class.

Usage

```
studyAreaName(x, layer)

## S3 method for class 'map'
studyAreaName(x, layer = 1)

## S3 method for class 'data.table'
studyAreaName(x, layer = 1)
```

Arguments

`x`	TODO: document this
`layer`	TODO: document this

See Also

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `rasters()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `rasters()`, `studyArea()`

Other mapMethods: `area,map-method`, `crs,map-method`, `mapRm()`, `rasterToMatch,map-method`, `rasters()`, `studyArea()`

Index