# Package 'sp'

April 14, 2016

Version 1.2-3

Title Classes and Methods for Spatial Data

**Depends** R (>= 3.0.0), methods

Imports utils, stats, graphics, grDevices, lattice, grid

**Suggests** RColorBrewer, rgdal (>= 0.8-7), rgeos (>= 0.3-13), gstat, maptools, deldir

**Description** Classes and methods for spatial

data; the classes document where the spatial location information resides, for 2D or 3D data. Utility functions are provided, e.g. for plotting data as maps, spatial selection, as well as methods for retrieving coordinates, for subsetting, print, summary, etc.

**License** GPL (>= 2)

URL https://github.com/edzer/sp/ https://edzer.github.io/sp/

BugReports https://github.com/edzer/sp/issues

Collate bpy.colors.R AAA.R Class-CRS.R CRS-methods.R Class-Spatial.R

Spatial-methods.R projected.R Class-SpatialPoints.R

SpatialPoints-methods.R Class-SpatialPointsDataFrame.R

SpatialPointsDataFrame-methods.R Class-SpatialMultiPoints.R

SpatialMultiPoints-methods.R

Class-SpatialMultiPointsDataFrame.R

SpatialMultiPointsDataFrame-methods.R Class-GridTopology.R

Class-SpatialGrid.R Class-SpatialGridDataFrame.R

Class-SpatialLines.R SpatialLines-methods.R

Class-SpatialLinesDataFrame.R SpatialLinesDataFrame-methods.R

Class-SpatialPolygons.R Class-SpatialPolygonsDataFrame.R

SpatialPolygons-methods.R SpatialPolygonsDataFrame-methods.R

GridTopology-methods.R SpatialGrid-methods.R

SpatialGridDataFrame-methods.R SpatialPolygons-internals.R point.in.polygon.R SpatialPolygons-displayMethods.R zerodist.R image.R stack.R bubble.R mapasp.R select.spatial.R gridded.R asciigrid.R spplot.R over.R spsample.R recenter.R dms.R gridlines.R spdists.R rbind.R flipSGDF.R chfids.R loadmeuse.R compassRose.R surfaceArea.R spOptions.R subset.R disaggregate.R sp\_spat1.R merge.R aggregate.R

NeedsCompilation yes	<b>ation</b> yes	Com	<b>leeds</b>	N
----------------------	------------------	-----	--------------	---

Author Edzer Pebesma [aut, cre],
Roger Bivand [aut],
Barry Rowlingson [ctb],
Virgilio Gomez-Rubio [ctb],
Robert Hijmans [ctb],
Michael Sumner [ctb],
Don MacQueen [ctb],
Jim Lemon [ctb],
Josh O'Brien [ctb]

Maintainer Edzer Pebesma <edzer.pebesma@uni-muenster.de>

Repository CRAN

**Date/Publication** 2016-04-14 16:58:02

# **R** topics documented:

addAttrToGeom-methods
aggregate
as.SpatialPolygons.GridTopology
as.SpatialPolygons.PolygonsList
bbox-methods
bpy.colors
bubble
char2dms
compassRose
coordinates
coordinates-methods
coordnames-methods
CRS-class
degAxis
dimensions-methods
disaggregate-methods
DMS-class
flip
geometry-methods
gridded-methods
gridIndex2nb
gridlines
GridTopology-class
image.SpatialGridDataFrame
is.projected
Line
Line-class
Lines-class
loadMeuse
mapasp

merge
meuse
meuse.grid
meuse.grid_ll
meuse.riv
over-methods
panel.spplot
point.in.polygon
Polygon-class
polygons
Polygons-class
polygons-methods
read.asciigrid
recenter-methods
Rlogo
select.spatial
sp
sp-deprecated
Spatial-class
SpatialGrid-class
SpatialGridDataFrame-class
SpatialLines
SpatialLines-class
SpatialLinesDataFrame-class
SpatialMultiPoints
SpatialMultiPoints-class
SpatialMultiPointsDataFrame-class
SpatialPixels
SpatialPixels-class
SpatialPixelsDataFrame
SpatialPixelsDataFrame-class
SpatialPoints
SpatialPoints-class
SpatialPointsDataFrame-class
SpatialPolygons
SpatialPolygons-class
SpatialPolygonsDataFrame-class
spChFIDs-methods
spDistsN1
spplot
spsample
spTransform
stack
surfaceArea
zerodist
200000000000000000000000000000000000000

4 addAttrToGeom-methods

addAttrToGeom-methods constructs SpatialXxxDataFrame from geometry and attributes

## **Description**

constructs SpatialXxxDataFrame from geometry and attributes

## Usage

```
addAttrToGeom(x, y, match.ID, ...)
```

## **Arguments**

x geometry (locations) of the queries
y data.frame object with attributes
match.ID logical; if TRUE, the IDs of the geometry and of the data.frame are matched (possibly swapping records), and an error occurs when some IDs do not match
... (optional) arguments passed to the constructor functions

#### Value

an object of class XxxDataFrame, where Xxx is the class of x

#### Methods

```
x = "SpatialPoints", y = "data.frame"
x = "SpatialPixels", y = "data.frame"
x = "SpatialGrid", y = "data.frame"
x = "SpatialLines", y = "data.frame"
x = "SpatialPolygons", y = "data.frame"
```

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

# See Also

over

aggregate 5

aggregate	aggregation of spatial objects	

## **Description**

spatial aggregation of thematic information in spatial objects

#### Usage

```
## S3 method for class 'Spatial'
aggregate(x, by = list(ID = rep(1, length(x))),
FUN, ..., dissolve = TRUE, areaWeighted = FALSE)
```

#### **Arguments**

X	object deriving from Spatial, with attributes
by	aggregation predicate; if by is a Spatial object, the geometry by which attributes in x are aggregated; if by is a list, aggregation by attribute(s), see aggregate.data.frame
FUN	aggregation function, e.g. mean; see details
•••	arguments passed on to function FUN, unless minDimension is specified, which is passed on to function over
dissolve	logical; should, when aggregating based on attributes, the resulting geometries be dissolved? Note that if x has class SpatialPointsDataFrame, this returns an object of class SpatialMultiPointsDataFrame
areaWeighted	logical; should the aggregation of x be weighted by the areas it intersects with each feature of by? See value.

## Details

FUN should be a function that takes as first argument a vector, and that returns a single number. The canonical examples are mean and sum. Counting features is obtained when summing an attribute variable that has the value 1 everywhere.

#### Value

The aggregation of attribute values of x either over the geometry of by by using over for spatial matching, or by attribute values, using aggregation function FUN.

If areaWeighted is TRUE, FUN is ignored and the area weighted mean is computed for numerical variables, or if all attributes are factors, the area dominant factor level (area mode) is returned. This will compute the gIntersection of x and by; see examples below.

If by is missing, aggregates over all features.

## Note

uses over to find spatial match if by is a Spatial object

6 aggregate

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

```
data("meuse")
coordinates(meuse) <- ~x+y</pre>
data("meuse.grid")
coordinates(meuse.grid) <- ~x+y</pre>
gridded(meuse.grid) <- TRUE</pre>
i = cut(meuse.grid\$dist, c(0,.25,.5,.75,1), include.lowest = TRUE)
j = sample(1:2, 3103,replace=TRUE)
## Not run:
if (require(rgeos)) {
# aggregation by spatial object:
ab = gUnaryUnion(as(meuse.grid, "SpatialPolygons"), meuse.grid$part.a)
x = aggregate(meuse["zinc"], ab, mean)
spplot(x)
# aggregation of multiple variables
x = aggregate(meuse[c("zinc", "copper")], ab, mean)
spplot(x)
# aggregation by attribute, then dissolve to polygon:
x = aggregate(meuse.grid["dist"], list(i=i), mean)
spplot(x["i"])
x = aggregate(meuse.grid["dist"], list(i=i,j=j), mean)
spplot(x["dist"], col.regions=bpy.colors())
spplot(x["i"], col.regions=bpy.colors(4))
spplot(x["j"], col.regions=bpy.colors())
}
## End(Not run)
x = aggregate(meuse.grid["dist"], list(i=i,j=j), mean, dissolve = FALSE)
spplot(x["j"], col.regions=bpy.colors())
if (require(gstat) && require(rgeos)) {
x = idw(log(zinc)^{-1}, meuse, meuse.grid, debug.level=0)[1]
spplot(x[1],col.regions=bpy.colors())
i = cut(x$var1.pred, seq(4, 7.5, by=.5),
include.lowest = TRUE)
xa = aggregate(x["var1.pred"], list(i=i), mean)
spplot(xa[1],col.regions=bpy.colors(8))
}
if (require(rgeos)) {
# Area-weighted example, using two partly overlapping grids:
 gt1 = SpatialGrid(GridTopology(c(0,0), c(1,1), c(4,4)))
 gt2 = SpatialGrid(GridTopology(c(-1.25,-1.25), c(1,1), c(4,4)))
 # convert both to polygons; give p1 attributes to aggregate
 p1 = SpatialPolygonsDataFrame(as(gt1, "SpatialPolygons"),
```

```
data.frame(v = 1:16, w=5:20, x=factor(1:16)), match.ID = FALSE)
 p2 = as(gt2, "SpatialPolygons")
 # plot the scene:
 plot(p1, xlim = c(-2,4), ylim = c(-2,4))
 plot(p2, add = TRUE, border = 'red')
 i = gIntersection(p1, p2, byid = TRUE)
 plot(i, add=TRUE, density = 5, col = 'blue')
 # plot IDs p2:
 ids.p2 = sapply(p2@polygons, function(x) slot(x, name = "ID"))
 text(coordinates(p2), ids.p2)
 # plot IDs i:
 ids.i = sapply(i@polygons, function(x) slot(x, name = "ID"))
 text(coordinates(i), ids.i, cex = .8, col = 'blue')
 # compute & plot area-weighted average; will warn for the factor
 ret = aggregate(p1, p2, areaWeighted = TRUE)
 spplot(ret)
 # all-factor attributes: compute area-dominant factor level:
 ret = aggregate(p1["x"], p2, areaWeighted = TRUE)
 spplot(ret)
}
```

as.SpatialPolygons.GridTopology

Make SpatialPolygons object from GridTopology object

# **Description**

Converts grids of regular rectangles into a SpatialPolygons object, which can be transformed to a different projection or datum with spTransform in package rgdal. The function is not suitable for high-resolution grids. The ordering of the grid cells is as in coordinates() of the same object, and is reported by IDvaluesGridTopology.

#### Usage

```
as.SpatialPolygons.GridTopology(grd, proj4string = CRS(as.character(NA)))
IDvaluesGridTopology(obj)
as.SpatialPolygons.SpatialPixels(obj)
IDvaluesSpatialPixels(obj)
HexPoints2SpatialPolygons(hex, dx)
```

#### **Arguments**

```
grd GridTopology object
proj4string object of class CRS-class
obj SpatialPixels object
```

hex SpatialPoints object with points that are generated by hexagonal sampling; see spsample

dx spacing of two horizontally adjacent points; if missing, this will be computed from the points

#### Value

as.SpatialPolygons.GridTopology and as.SpatialPolygons.SpatialPixels return a SpatialPolygons object; IDvaluesGridTopology and IDvaluesSpatialPixels return a character vector with the object grid indices.

#### See Also

GridTopology, SpatialPixels, SpatialPolygons spTransform in package rgdal

#### **Examples**

```
library(lattice)
grd <- GridTopology(cellcentre.offset=c(-175,55), cellsize=c(10,10), cells.dim=c(4,4))</pre>
SpP_grd <- as.SpatialPolygons.GridTopology(grd)</pre>
plot(SpP_grd)
text(coordinates(SpP_grd), sapply(slot(SpP_grd, "polygons"), function(i) slot(i, "ID")), cex=0.5)
trdata \leftarrow data.frame(A=rep(c(1,2,3,4), 4), B=rep(c(1,2,3,4), each=4),
row.names=sapply(slot(SpP_grd, "polygons"), function(i) slot(i, "ID")))
SpPDF <- SpatialPolygonsDataFrame(SpP_grd, trdata)</pre>
spplot(SpPDF)
data(meuse.grid)
gridded(meuse.grid)=~x+y
xx = spsample(meuse.grid, type="hexagonal", cellsize=200)
xxpl = HexPoints2SpatialPolygons(xx)
image(meuse.grid["dist"])
plot(xxpl, add = TRUE)
points(xx, cex = .5)
## Not run:
spplot(aggregate(meuse.grid[,1:3], xxpl), main = "aggregated meuse.grid")
## End(Not run)
```

```
as.SpatialPolygons.PolygonsList

Making SpatialPolygons objects
```

# Description

This function is used in making SpatialPolygons objects from other formats.

# Usage

```
as. Spatial Polygons. Polygons List (Srl, proj4string = CRS (as. character(NA))) \\
```

# Arguments

Srl A list of Polygons objects

proj4string Object of class "CRS"; holding a valid proj4 string

# Value

The functions return a SpatialPolygons object

# Author(s)

Roger Bivand

10 bbox-methods

```
grd <- GridTopology(c(1,1), c(1,1), c(10,10))
polys <- as.SpatialPolygons.GridTopology(grd)
plot(polys)
text(coordinates(polys), labels=sapply(slot(polys, "polygons"), function(i) slot(i, "ID")), cex=0.6)</pre>
```

bbox-methods

retrieve bbox from spatial data

#### **Description**

retrieves spatial bounding box from spatial data

#### **Usage**

bbox(obj)

# **Arguments**

obj

object deriving from class "Spatial", or one of classes: "Line", "Lines", "Polygon" or "Polygons", or ANY, which requires obj to be an array with at least two columns

#### Value

two-column matrix; the first column has the minimum, the second the maximum values; rows represent the spatial dimensions

## Methods

```
obj = "Spatial" object deriving from class "Spatial"
obj = "ANY" an array with at least two columns
obj = "Line" object deriving from class "Line"
obj = "Lines" object deriving from class "Lines"
obj = "Polygon" object deriving from class "Polygon"
obj = "Polygons" object deriving from class "Polygons"
```

bpy.colors 11

## **Examples**

```
# just 9 points on a grid:
x <- c(1,1,1,2,2,2,3,3,3)
y <- c(1,2,3,1,2,3,1,2,3)
xy <- cbind(x,y)
S <- SpatialPoints(xy)
bbox(S)

# data.frame
data(meuse.grid)
coordinates(meuse.grid) <- ~x+y
gridded(meuse.grid) <- TRUE
bbox(meuse.grid)</pre>
```

bpy.colors

blue-pink-yellow color scheme, which also prints well on black/white printers

# **Description**

Create a vector of 'n' "contiguous" colors.

#### Usage

```
bpy.colors(n = 100, cutoff.tails = 0.1, alpha = 1.0)
```

#### **Arguments**

n number of colors ( $\geq 1$ ) to be in the palette

cutoff.tails tail fraction to be cut off on each side. If 0, this palette runs from black to white;

by cutting off the tails, it runs from blue to yellow, which looks nicer.

alpha numeric; alpha transparency, 0 is fully transparent, 1 is opaque.

#### Value

A character vector, 'cv', of color names. This can be used either to create a user-defined color palette for subsequent graphics by 'palette(cv)', a 'col=' specification in graphics functions or in 'par'.

## Note

This color map prints well on black-and-white printers.

#### Author(s)

unknown; the pallette was posted to gnuplot-info a few decades ago; R implementation Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

12 bubble

#### See Also

rainbow, cm.colors

#### **Examples**

```
bpy.colors(10)
p <- expand.grid(x=1:30,y=1:30)
p$z <- p$x + p$y
coordinates(p) <- c("x", "y")
gridded(p) <- TRUE
image(p, col = bpy.colors(100), asp = 1)
# require(lattice)
# trellis.par.set("regions", list(col=bpy.colors())) # make this default pallette</pre>
```

bubble

Create a bubble plot of spatial data

## **Description**

Create a bubble plot of spatial data, with options for bicolour residual plots (xyplot wrapper)

## Usage

```
bubble(obj, zcol = 1, ..., fill = TRUE, maxsize = 3, do.sqrt = TRUE, pch,
col = c("#d01c8b", "#4dac26"), key.entries = quantile(data[,zcol]), main,
identify = FALSE, labels = row.names(data.frame(obj)), key.space = "right",
scales = list(draw = FALSE), xlab = NULL, ylab = NULL, panel = panel.bubble,
sp.layout = NULL,
xlim = bbexpand(bbox(obj)[1,], 0.04),
ylim = bbexpand(bbox(obj)[2,], 0.04))
```

## **Arguments**

obj	object of, or extending, class SpatialPointsDataFrame or SpatialGridDataFrame, see coordinates or SpatialPointsDataFrame; the object knows about its spatial coordinates
zcol	z-variable column name, or column number after removing spatial coordinates from x@data: 1 refers to the first non-coordinate column
fill	logical; if TRUE, filled circles are plotted (pch = 16), else open circles (pch = 1); the pch argument overrides this
maxsize	cex value for largest circle
do.sqrt	logical; if TRUE the plotting symbol area (sqrt(diameter)) is proportional to the value of the z-variable; if FALSE, the symbol size (diameter) is proportional to the z-variable
pch	plotting character

bubble 13

col colours to be used; numeric vector of size two: first value is for negative values,

second for positive values. Default colors: 5-class PiYG from colorbrewer.org.

key.entries the values that will be plotted in the key; by default the five quantiles min, q.25,

median q.75, max

main main plotting title

identify logical; if true, regular plot is called instead of xyplot, and followed by a call

to identify().

labels labels argument passed to plot if identify is TRUE

arguments, passed to xyplot, or plot if identification is required.

key.space location of the key

scales scales argument as passed to xyplot

xlab x-axis label ylab y-axis label

panel panel function used

sp.layout possible layout items; see spplot

xlim x axis limit ylim y axis limit

## Value

returns (or plots) the bubble plot; if identify is TRUE, returns the indexes (row numbers) of identified points.

## Author(s)

Edzer Pebesma

## See Also

```
xyplot, mapasp, identify
```

```
data(meuse)
coordinates(meuse) <- c("x", "y") # promote to SpatialPointsDataFrame
bubble(meuse, "cadmium", maxsize = 2.5, main = "cadmium concentrations (ppm)",
   key.entries = 2^(-1:4))
bubble(meuse, "zinc", main = "zinc concentrations (ppm)",
   key.entries = 100 * 2^(0:4))</pre>
```

14 char2dms

_	h a	∽າ	dm:	,

Convert character vector to DMS-class object

## Description

These two helper functions convert character vectors and decimal degree vectors to the DMS-class representation of degrees, minutes, and decimal seconds. "DMS" objects cannot contain NAs.

## Usage

```
char2dms(from, chd = "d", chm = "'", chs = "\"")
dd2dms(dd, NS = FALSE)
```

# Arguments

from	character vector of degree, minute, decimal second data
chd	degree character terminator
chm	minute character terminator
chs	second character terminator
dd	numeric vector of decimal degrees
NS	logical, TRUE for north/south decimal degrees, FALSE for east/west decimal degrees

#### Details

In char2dms, the input data vector should use a regular format, such as that used in the PROJ.4 library, with a trailing capital (NSWE) indicating compass direction.

#### Value

Both functions return a "DMS" object.

#### Methods

## Author(s)

Roger Bivand < Roger . Bivand@nhh . no>

## See Also

```
DMS-class
```

compassRose 15

## **Examples**

```
data(state)
str(state.center$y)
stateN <- dd2dms(state.center$y, NS=TRUE)
str(attributes(stateN))
ch.stateN <- as.character(stateN)
str(ch.stateN)
stateNa <- char2dms(ch.stateN)
str(attributes(stateNa))
ch.stateN <- as(stateN, "character")
str(ch.stateN)
stateNa <- char2dms(ch.stateN)
stateNa <- char2dms(ch.stateN)
str(attributes(stateNa))</pre>
```

compassRose

Display a compass rose.

## **Description**

Displays a basic compass rose, usually to orient a map.

## Usage

```
compassRose(x,y,rot=0,cex=1)
```

# Arguments

x,y The position of the center of the compass rose in user units.

Rotation for the compass rose in degrees. See Details.

cex The character expansion to use in the display.

#### **Details**

'compassRose' displays a conventional compass rose at the position requested. The size of the compass rose is determined by the character expansion, as the central "rose" is calculated relative to the character size. Rotation is in degrees counterclockwise.

## Value

nil

# Author(s)

Jim Lemon

16 coordinates

coordinates	set spatial coordinates to create a Spatial object, or retrieve spatial
	coordinates from a Spatial object

#### **Description**

set spatial coordinates to create a Spatial object, or retrieve spatial coordinates from a Spatial object

## Usage

```
coordinates(obj, ...)
coordinates(object) <- value</pre>
```

## **Arguments**

obj object deriving from class "Spatial"

object of class "data.frame"

value spatial coordinates; either a matrix, list, or data frame with numeric data, or col-

umn names, column number or a reference: a formula (in the form of e.g. ~x+y), column numbers (e.g. c(1,2)) or column names (e.g. c("x","y")) specifying which columns in object are the spatial coordinates. If the coordinates are part of object, giving the reference does not duplicate them, giving their value does

duplicate them in the resulting structure.

... additional arguments that may be used by particular methods

# Value

usually an object of class SpatialPointsDataFrame; if the coordinates set cover the full set of variables in object, an object of class SpatialPoints is returned

```
# data.frame
data(meuse.grid)
coordinates(meuse.grid) <- ~x+y
gridded(meuse.grid) <- TRUE
class(meuse.grid)
bbox(meuse.grid)

data(meuse)
meuse.xy = meuse[c("x", "y")]
coordinates(meuse.xy) <- ~x+y
class(meuse.xy)</pre>
```

coordinates-methods 17

coordinates-methods

retrieve (or set) spatial coordinates

## **Description**

retrieve (or set) spatial coordinates from (for) spatial data

#### Methods

```
obj = "list" list with (at least) two numeric components of equal length
```

**obj = "data.frame"** data.frame with at least two numeric components

obj = "matrix" numeric matrix with at least two columns

**obj = "SpatialPoints"** object of, or deriving from, SpatialPoints

**obj = "SpatialPointsDataFrame"** object of, or deriving from, SpatialPointsDataFrame

**obj = "SpatialPolygons"** object of, or deriving from, SpatialPolygons

obj = "SpatialPolygonsDataFrame" object of, or deriving from, SpatialPolygonsDataFrame

**obj = "Line"** object of class Line; returned value is matrix

**obj = "Lines"** object of class Lines; returned value is list of matrices

**obj = "SpatialLines"** object of, or deriving from, SpatialLines; returned value is list of lists of matrices

**obj = "GridTopology"** object of, or deriving from, GridTopology

**obj = "GridTopology"** object of, or deriving from, GridTopology

**obj = "SpatialPixels"** object of, or deriving from, SpatialPixels

**obj = "SpatialPixelsDataFrame"** object of, or deriving from, SpatialPixelsDataFrame

obj = "SpatialGrid" object of, or deriving from, SpatialGrid

obj = "SpatialGridDataFrame" object of, or deriving from, SpatialGridDataFrame

# Methods for "coordinates<-"

object = "data.frame", value="ANY" promote data.frame to object of class SpatialPointsDataFrameclass, by specifying coordinates; see coordinates 18 CRS-class

coordnames-methods

retrieve or assign coordinate names for classes in sp

## Description

retrieve or assign coordinate names for classes in sp

#### Methods for coordnames

```
x = "SpatialPoints" retrieves coordinate names
```

**x** = "SpatialLines" retrieves coordinate names

**x** = "Lines" retrieves coordinate names

x = "Line" retrieves coordinate names

x = "SpatialPolygons" retrieves coordinate names

**x** = "**Polygons**" retrieves coordinate names

x = "Polygon" retrieves coordinate names

#### Methods for "coordnames<-"

```
x = "SpatialPoints", value = "character" replace coordinate names
```

x = "SpatialLines", value = "character" replace coordinate names

x = "Lines", value = "character" replace coordinate names

x = "Line", value = "character" replace coordinate names

x = "SpatialPolygons", value = "character" replace coordinate names

x = "GridTopology", value = "character" replace coordinate names

x = "SpatialGrid", value = "character" replace coordinate names

x = "SpatialPixels", value = "character" replace coordinate names

CRS-class

Class "CRS" of coordinate reference system arguments

## **Description**

Interface class to the PROJ.4 projection system. The class is defined as an empty stub accepting value NA in the sp package. If the rgdal package is available, then the class will permit spatial data to be associated with coordinate reference systems. The arguments must be entered exactly as in the PROJ.4 documentation, in particular there cannot be any white space in +<arg>=<value> strings, and successive such strings can only be separated by blanks. Note that only "+proj=longlat +ellps=WGS84" is accepted for geographical coordinates, which must be ordered (eastings, northings); the "+ellps=" definition must be given (or expanded internally from a given "+datum=" value) for recent versions of the PROJ.4 library, and should be set to an appropriate value.

CRS-class 19

#### Usage

```
CRS(projargs, doCheckCRSArgs=TRUE)
identicalCRS(x,y)
```

#### **Arguments**

projargs	A character string of projection arguments; the arguments must be entered exactly as in the PROJ.4 documentation; if the projection is unknown, use as . character(NA), it may be missing or an empty string of zero length and will then set to the missing value.
doCheckCRSArgs	default TRUE, must be set to FALSE by package developers including CRS in an S4 class definition to avoid uncontrolable loading of the <b>rgdal</b> namespace
х	object having a proj4string method, or if y is missing, list with objects that have a proj4string method
у	object of class Spatial, or having a proj4string method

#### Value

CRS returns on success an object of class CRS. identicalCRS returns a logical, indicating whether x and y have identical CRS, or if y is missing whether all objects in list x have identical CRS.

## **Objects from the Class**

Objects can be created by calls of the form CRS("projargs"), where "projargs" is a valid string of PROJ.4 arguments. The initiation function calls the PROJ.4 library to verify the argument set against those known in the library, returning error messages where necessary. The function CRSargs() can be used to show the expanded argument list used by the PROJ.4 library.

## **Slots**

projargs: Object of class "character": projection arguments; the arguments must be entered exactly as in the PROJ.4 documentation, in particular there cannot be any white space in +<arg>=<value> strings, and successive such strings can only be separated by blanks.

## Methods

```
show signature(object = "CRS"): print projection arguments in object
```

## Note

Lists of projections may be seen by using the programs installed with the PROJ.4 library, in particular proj and cs2cs; with the latter, -lp lists projections, -le ellipsoids, -lu units, and -ld datum(s) known to the installed software (available in **rgdal** using projInfo). These are added to in successive releases, so tracking the website or compiling and installing the most recent revisions will give the greatest choice. Finding the very important datum transformation parameters to be given with the +towgs84 tag is a further challenge, and is essential when the datums used in data to be used together differ. Tracing projection arguments is easier now than before the mass ownership of GPS receivers raised the issue of matching coordinates from different argument sets (GPS output

20 degAxis

and paper map, for example). See GridsDatums, make\_EPSG and showEPSG for help in finding CRS definitions.

The 4.9.1 release of PROJ.4 omitted a small file of defaults, leading to reports of "major axis or radius = 0 or not given" errors. From 0.9-3, rgdal checks for the presence of this file (proj\_def.dat), and if not found, and under similar conditions to those used by PROJ.4, adds "+ellps=WGS84" to the input string being checked by checkCRSArgs The "+no\_defs" tag ignores the file of defaults, and the default work-around implemented to get around this problem; strings including "init" and "datum" tags also trigger the avoidance of the work-around. Now messages are issued when a candidate CRS is checked; they may be suppressed using suppressMessages.

## Author(s)

Roger Bivand < Roger . Bivand@nhh . no>

## References

```
https://github.com/OSGeo/proj.4
```

## **Examples**

```
CRS()
CRS("")
CRS(as.character(NA))
CRS("+proj=longlat +datum=WGS84")
if (require(rgdal)) {
    print(CRSargs(CRS("+proj=longlat +datum=NAD27")))
    print(CRSargs(CRS("+init=epsg:4267")))
    print(CRSargs(CRS("+init=epsg:26978")))
    print(CRSargs(CRS("+init=epsg:26978")))
    print(CRSargs(CRS(paste("+proj=sterea +lat_0=52.15616055555555",
    "+lon_0=5.3876388888889 +k=0.999908 +x_0=155000 +y_0=463000 +ellps=bessel",
    " +towgs84=565.237,50.0087,465.658,-0.406857,0.350733,-1.87035,4.0812 +units=m"))))
    print(CRSargs(CRS("+init=epsg:28992")))
}
# see http://trac.osgeo.org/gdal/ticket/1987
```

degAxis

axis with degrees

# **Description**

draw axes on a plot using degree symbols in numbers

## Usage

```
degAxis(side, at, labels, ...)
```

dimensions-methods 21

# Arguments

side integer; see axis

at numeric; if missing, axTicks is called for nice values; see axis

labels character; if omitted labels are constructed with degree symbols, ending in N/S/E/W; in case of negative degrees, sign is reversed and S or W is added; see axis

... passed to the actual axis call

#### Value

axis is plotted on current graph

#### Note

decimal degrees are used if variation is small, instead of minutes and seconds

## **Examples**

dimensions-methods

retrieve spatial dimensions from spatial data

## **Description**

retrieves spatial dimensions box from spatial data

## Usage

```
dimensions(obj)
```

#### **Arguments**

obj object deriving from class "Spatial"

## Value

two-column matrix; the first column has the minimum, the second the maximum values; rows represent the spatial dimensions

## Methods

```
obj = "Spatial" object deriviving from class "Spatial"
```

22 disaggregate-methods

#### **Examples**

```
# just 9 points on a grid:
x <- c(1,1,1,2,2,2,3,3,3)
y <- c(1,2,3,1,2,3,1,2,3)
xy <- cbind(x,y)
S <- SpatialPoints(xy)
dimensions(S)

# data.frame
data(meuse.grid)
coordinates(meuse.grid) <- ~x+y
gridded(meuse.grid) <- TRUE
dimensions(meuse.grid)</pre>
```

disaggregate-methods

disaggregate SpatialLines, SpatialLinesDataFrame, SpatialPolygons, or SpatialPolygonsDataFrame objects

# **Description**

 $dis aggregate\ Spatial Lines, Spatial Lines Data Frame,\ Spatial Polygons,\ or\ Spatial Polygons Data Frame\ objects$ 

#### Usage

```
disaggregate(x, ...)
```

## **Arguments**

```
x object of class SpatialLines or SpatialPolygons... ignored
```

#### Value

object of class SpatialLines or SpatialPolygons, where groups of Line or Polygon are disaggregated to one Line per Lines, or one Polygon per Polygons, respectively.

## Author(s)

Robert Hijmans, Edzer Pebesma

DMS-class 23

#### **Examples**

```
Sr1 = Polygon(cbind(c(2,4,4,1,2),c(2,3,5,4,2)), hole = FALSE)
Sr2 = Polygon(cbind(c(5,4,2,5),c(2,3,2,2)), hole = FALSE)
Sr3 = Polygon(cbind(c(4,4,5,10,4),c(5,3,2,5,5)), hole = FALSE)
Sr4 = Polygon(cbind(c(5,6,6,5,5),c(4,4,3,3,4)), hole = TRUE)
Srs1 = Polygons(list(Sr1, Sr2), "s1/2")
Srs3 = Polygons(list(Sr3, Sr4), "s3/4")
sp = SpatialPolygons(list(Srs1,Srs3), 1:2)
length(sp) ## [1] 2
length(disaggregate(sp)) ## [1] 3
11 = cbind(c(1,2,3),c(3,2,2))
11a = cbind(11[,1]+.05,11[,2]+.05)
12 = cbind(c(1,2,3),c(1,1.5,1))
Sl1 = Line(l1)
Sl1a = Line(l1a)
S12 = Line(12)
S1 = Lines(list(Sl1, Sl1a), ID="a")
S2 = Lines(list(S12), ID="b")
sl = SpatialLines(list(S1,S2))
length(sl)
length(disaggregate(sl))
```

DMS-class

Class "DMS" for degree, minute, decimal second values

## **Description**

The class provides a container for coordinates stored as degree, minute, decimal second values.

## **Objects from the Class**

Objects can be created by calls of the form new("DMS", ...), converted from decimal degrees using dd2dms(), or converted from character strings using char2dms().

#### **Slots**

```
WS: Object of class "logical" TRUE if input value negative deg: Object of class "numeric" degrees min: Object of class "numeric" minutes sec: Object of class "numeric" decimal seconds

NS: Object of class "logical" TRUE if input value is a Northing
```

#### Methods

```
coerce signature(from = "DMS", to = "numeric"): convert to decimal degrees
show signature(object = "DMS"): print data values
```

24 flip

#### Author(s)

Roger Bivand < Roger . Bivand@nhh . no>

## See Also

```
char2dms, dd2dms
```

## **Examples**

```
data(state)
dd2dms(state.center$x)
dd2dms(state.center$y, NS=TRUE)
as.numeric(dd2dms(state.center$y))
as(dd2dms(state.center$y, NS=TRUE), "numeric")
as.numeric.DMS(dd2dms(state.center$y))
state.center$y
```

flip

rearrange data in SpatialPointsDataFrame or SpatialGridDataFrame for plotting with spplot (levelplot/xyplot wrapper)

## **Description**

rearrange SpatialPointsDataFrame for plotting with spplot or levelplot

#### Usage

```
flipHorizontal(x)
flipVertical(x)
```

# Arguments

Х

object of class SpatialGridDataFrame

#### Value

object of class SpatialGridDataFrame, with pixels flipped horizontally or vertically. Note that the spatial structure is destroyed (or at least: drastically changed).

## Author(s)

Michael Sumner

geometry-methods 25

## **Examples**

```
data(meuse.grid) # data frame
gridded(meuse.grid) = c("x", "y") # promotes to
fullgrid(meuse.grid) = TRUE
d = meuse.grid["dist"]
image(d, axes=TRUE)
image(flipHorizontal(d), axes=TRUE)
image(flipVertical(d), axes=TRUE)
```

geometry-methods

Methods for retrieving the geometry from a composite (geometry + attributes) object

# Description

geometry retrieves the SpatialXxx object from a SpatialXxxDataFrame object, with Xxx Lines, Points, Polygons, Grid, or Pixels. geometry<- converts a data.frame into a Spatial object.

#### Usage

```
geometry(obj)
geometry(obj) <- value</pre>
```

## Arguments

obj in case of assignment, a data.frame, else an object of class Spatial value object of class Spatial

#### Methods

```
obj = "Spatial"
obj = "SpatialPointsDataFrame"
obj = "SpatialMultiPointsDataFrame"
obj = "SpatialPolygonsDataFrame"
obj = "SpatialPixelsDataFrame"
obj = "SpatialGridDataFrame"
obj = "SpatialLinesDataFrame"
```

## Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

26 gridded-methods

#### **Examples**

```
data(meuse)
m = meuse
coordinates(m) = meuse[, c("x", "y")]
pts = geometry(m)
class(pts)
geometry(meuse) = pts
class(meuse)
identical(m, meuse) # TRUE
```

gridded-methods

specify spatial data as being gridded, or find out whether they are

#### **Description**

returns logical (TRUE or FALSE) telling whether the object is gridded or not; in assignment promotes a non-gridded structure to a gridded one, or demotes a gridded structure back to a non-structured one.

## Usage

```
gridded(obj)
gridded(obj) <- value
fullgrid(obj)
fullgrid(obj) <- value
gridparameters(obj)</pre>
```

#### Arguments

obj object deriviving from class "Spatial" (for gridded), or object of class SpatialGridDataFrame-

class (for fullgrid and gridparameters)

value logical replacement values, TRUE or FALSE

#### Value

if obj derives from class Spatial, gridded(object) will tell whether it is has topology on a regular grid; if assigned TRUE, if the object derives from SpatialPoints and has gridded topology, grid topology will be added to object, and the class of the object will be promoted to SpatialGrid-class or SpatialGridDataFrame-class

fullgrid returns a logical, telling whether the grid is full and ordered (i.e., in full matrix form), or whether it is not full or unordered (i.e. a list of points that happen to lie on a grid. If assigned, the way the points are stored may be changed. Changing a set of points to full matrix form and back may change the original order of the points, and will remove duplicate points if they were present.

gridparameters returns, if obj inherits from SpatialGridDataFrame its grid parameters, else it returns numeric(0). The returned value is a data.frame with three columns, named cellcentre.offset ("lower left cell centre coordinates"), cellsize, and cells.dim (cell dimension); the rows correspond to the spatial dimensions.

gridIndex2nb 27

## Methods

```
obj = "Spatial" object deriviving from class "Spatial"
```

## **Examples**

```
# just 9 points on a grid:
x \leftarrow c(1,1,1,2,2,2,3,3,3)
y \leftarrow c(1,2,3,1,2,3,1,2,3)
xy \leftarrow cbind(x,y)
S <- SpatialPoints(xy)</pre>
class(S)
plot(S)
gridded(S) <- TRUE</pre>
gridded(S)
class(S)
summary(S)
plot(S)
gridded(S) <- FALSE</pre>
gridded(S)
class(S)
# data.frame
data(meuse.grid)
coordinates(meuse.grid) <- \sim x+y
gridded(meuse.grid) <- TRUE</pre>
plot(meuse.grid) # not much good
summary(meuse.grid)
```

gridIndex2nb

create neighbourhood (nb) object from grid geometry

# Description

create neighbourhood (nb) object from grid geometry

# Usage

```
gridIndex2nb(obj, maxdist = sqrt(2), fullMat = TRUE, ...)
```

#### **Arguments**

obj	object of class SpatialGrid or SpatialPixels
maxdist	maximum distance to be considered (inclusive), expressed in number of grid cell (sqrt(2) results in queen neighbours)
fullMat	use dist to compute distances from grid (row/col) indices; FALSE avoids forming the full distance matrix, at a large performance cost
	arguments passed on to dist

28 gridlines

#### Value

Object of class nb, which is a list.

The nb object follows the convention of nb objects in package spdep; it is a list with each list element corresponding to a grid cell or pixel; the list element contains the indices of neighbours defined as cells less than maxdist away, measured in cell unit (N/S/E/W neighbour has distance 1).

#### Note

Unequal grid cell size is ignored; grid cell row/col indices are taken to be the coordinates from which distances are computed.

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

plot.nb in package spdep

gridlines

Create N-S and E-W grid lines over a geographic region

#### **Description**

Create N-S and E-W grid lines over a geographic region; create and plot corresponding labels

#### Usage

```
gridlines(x, easts = pretty(bbox(x)[1,]), norths = pretty(bbox(x)[2,]),
ndiscr = 100)
gridat(x, easts = pretty(bbox(x)[1,]), norths = pretty(bbox(x)[2,]),
offset = 0.5, side = "WS")
## S3 method for class 'SpatialLines'
labels(object, labelCRS, side = 1:2, ...)
## S3 method for class 'SpatialPointsDataFrame'
text(x, ...)
```

## Arguments

X	object deriving from class Spatial-class
easts	numeric; east-west values for vertical lines
norths	numeric; north-south values for horizontal lines
ndiscr	integer; number of points used to discretize the line, could be set to 2, unless the grid is (re)projected
offset	offset value to be returned, see text

gridlines 29

object	SpatialLines-class object, as returned by gridlines
labelCRS	the CRS in which the grid lines were drawn and labels should be printed; if missing, the CRS from object is taken
side	for labels: integer, indicating side(s) at which gridlines labels will be drawn: 1=below (S), 2=left (W), 3=above (N), and 4=right (E); for gridat: default "WS", if "EN" labels placed on the top and right borders
	for labels: ignored; for text: arguments passed on to text, see below for example use of adj

#### Value

gridlines returns an object of class SpatialLines-class, with lines as specified; the return object inherits the projection information of x; gridat returns a SpatialPointsDataFrame with points at the west and south ends of the grid lines created by gridlines, with degree labels.

The labels method for SpatialLines objects returns a SpatialPointsDataFrame-class object with the parameters needed to print labels below and left of the gridlines. The locations for the labels are those of proj4string(object) the labels also unless labelCRS is given, in which case they are in that CRS. This object is prepared to be plotted with text:

The text method for SpatialPointsDataFrame puts text labels on its coordinates, and takes care of attributes pos, labels, srt and offset; see text.

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>, using example code of Roger Bivand.

## See Also

spTransform; llgridlines in rgdal (recent versions) for plotting long-lat grid over projected data

```
data(meuse)
coordinates(meuse) = ~x+y
plot(meuse)
plot(gridlines(meuse), add = TRUE)
text(labels(gridlines(meuse)))
title("default gridlines within Meuse bounding box")
proj4string(meuse) <- CRS("+init=epsg:28992")</pre>
crs.longlat <- CRS("+init=epsg:4326")</pre>
meuse_ll <- spTransform(meuse, crs.longlat)</pre>
grd <- gridlines(meuse_ll)</pre>
grd_x <- spTransform(grd, CRS("+init=epsg:28992"))</pre>
# labels South and West:
plot(meuse)
plot(grd_x, add=TRUE, lty=2)
grdat_ll <- gridat(meuse_ll)</pre>
grdat_x <- spTransform(grdat_ll, CRS("+init=epsg:28992"))</pre>
```

30 GridTopology-class

```
text(grdat_x)
# labels North and East:
plot(meuse)
plot(grd_x, add=TRUE, lty=2)
grdat_ll <- gridat(meuse_ll, side="EN")</pre>
grdat_x <- spTransform(grdat_ll, CRS("+init=epsg:28992"))</pre>
text(grdat_x)
# now using labels:
plot(meuse)
plot(grd_x, add=TRUE, lty=2)
text(labels(grd_x, crs.longlat))
# demonstrate axis labels with angle, both sides:
sp = SpatialPoints(rbind(c(-101,9), c(-101,55), c(-19,9), c(-19,55)), crs.longlat)
laea = CRS("+proj=laea +lat_0=30 +lon_0=-40")
sp.l = spTransform(sp, laea)
plot(sp.1, expandBB = c(0, 0.05, 0, .05))
gl = spTransform(gridlines(sp), laea)
plot(gl, add = TRUE)
text(labels(gl, crs.longlat))
text(labels(gl, crs.longlat, side = 3:4), col = 'red')
title("curved text label demo")
# polar:
pts=SpatialPoints(rbind(c(-180,-70),c(0,-70),c(180,-89),c(180,-70)), crs.longlat)
polar = CRS("+init=epsg:3031")
gl = spTransform(gridlines(pts, easts = seq(-180,180,20), ndiscr = 100), polar)
plot(spTransform(pts, polar), expandBB = c(.05,0,.05,0))
lines(gl)
1 = labels(gl, crs.longlat, side = 3)
1$pos = NULL # pos is too simple, use adj:
text(1, adj = c(0.5, -0.5))
1 = labels(gl, crs.longlat, side = 4)
1$srt = 0 # otherwise they end up upside-down
text(1)
title("grid line labels on polar projection, epsg 3031")
## Not run:
demo(polar) # adds the map of the antarctic
## End(Not run)
```

GridTopology-class

Class "GridTopology"

# **Description**

class for defining a rectangular grid of arbitrary dimension

GridTopology-class 31

## **Objects from the Class**

```
Objects are created by using e.g. GridTopology(c(0,0), c(1,1), c(5,5)) see SpatialGrid
```

#### **Slots**

```
cellcentre.offset: numeric; vector with the smallest coordinates for each dimension; coordinates refer to the cell centre
cellsize: numeric; vector with the cell size in each dimension
cells.dim: integer; vector with number of cells in each dimension
```

#### Methods

```
coordinates signature(x = "SpatialGrid"): calculates coordinates for each point on the grid
summary signature(object = "SpatialGrid"): summarize object
coerce signature(from = "GridTopology", to = "data.frame"): convert to data.frame with
    columns cellcentre.offset, cellsize and cells.dim
```

## Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

```
{\tt SpatialGridDataFrame-class}, {\tt SpatialGrid-class}
```

```
x = GridTopology(c(0,0), c(1,1), c(5,5))
class(x)
x
summary(x)
coordinates(x)
y = SpatialGrid(grid = x)
class(y)
y
```

```
image.SpatialGridDataFrame
```

Image or contour method for gridded spatial data; convert to and from image data structure

## Description

Create image for gridded data in SpatialGridDataFrame or SpatialPixelsDataFrame objects.

## Usage

```
## S3 method for class 'SpatialGridDataFrame'
image(x, attr = 1, xcol = 1, ycol = 2,
col = heat.colors(12), red=NULL, green=NULL, blue=NULL,
       axes = FALSE, xlim = NULL,
ylim = NULL, add = FALSE, ..., asp = NA, setParUsrBB=FALSE,
        interpolate = FALSE, angle = 0,
useRasterImage = (!.isSDI() && missing(breaks)), breaks,
zlim = range(as.numeric(x[[attr]])[is.finite(x[[attr]])]))
## S3 method for class 'SpatialPixelsDataFrame'
image(x, ...)
## S3 method for class 'SpatialPixels'
image(x, ...)
## S3 method for class 'SpatialGridDataFrame'
contour(x, attr = 1, xcol = 1, ycol = 2,
col = 1, add = FALSE, xlim = NULL, ylim = NULL, axes = FALSE,
         ..., setParUsrBB = FALSE)
## S3 method for class 'SpatialPixelsDataFrame'
contour(x, ...)
as.image.SpatialGridDataFrame(x, xcol = 1, ycol = 2, attr = 1)
image2Grid(im, p4 = as.character(NA), digits=10)
```

# Arguments

Χ	object of class SpatialGridDataFrame
attr	column of attribute variable; this may be the column name in the data.frame of data (as.data.frame(data)), or a column number
xcol	column number of x-coordinate, in the coordinate matrix
ycol	column number of y-coordinate, in the coordinate matrix
col	a vector of colors
red,green,blue	columns names or numbers given instead of the attr argument when the data represent an image encoded in three colour bands on the 0-255 integer scale; all three columns must be given in this case, and the attribute values will be constructed using function rgb
axes	logical; should coordinate axes be drawn?

xlim	x-axis limits
ylim	y-axis limits
zlim	data limits for plotting the (raster, attribute) values
add	logical; if FALSE, the image is added to the plot layout setup by plot(as(x, "Spatial"), axes=axes,xl which sets up axes and plotting region; if TRUE, the image is added to the existing plot.
	arguments passed to image, see examples
asp	aspect ratio to be used for plot
setParUsrBB	default FALSE, see Spatial-class for further details
useRasterImage	default !.isSDI() as a workaround for a problem with repeated use in Windows SDI installations; if TRUE, use rasterImage to render the image if available; for legacy rendering set FALSE
breaks	class breaks for coloured values
interpolate	default FALSE, a logical vector (or scalar) indicating whether to apply linear interpolation to the image when drawing, see rasterImage
angle	default 0, angle of rotation (in degrees, anti-clockwise from positive x-axis, about the bottom-left corner), see rasterImage
im	list with components named x, y, and z, as used for image
p4	CRS object, proj4 string

#### Value

digits

spacing

as.image.SpatialGridDataFrame returns the list with elements x and y, containing the coordinates of the cell centres of a matrix z, containing the attribute values in matrix form as needed by image.

default 10, number of significant digits to use for checking equal row/column

#### Note

Providing xcol and ycol attributes seems obsolete, and it is for 2D data, but it may provide opportunities for plotting certain slices in 3D data. I haven't given this much thought yet.

filled.contour seems to misinterpret the coordinate values, if we take the image.default manual page as the reference.

# Author(s)

Edzer Pebesma

## See Also

image.default, SpatialGridDataFrame-class, levelplot in package lattice. Function image.plot in package fields can be used to make a legend for an image, see an example in https://stat.ethz.ch/pipermail/r-sig-geo/2007-June/002143.html

34 is.projected

#### **Examples**

```
data(meuse.grid)
coordinates(meuse.grid) = c("x", "y") # promote to SpatialPointsDataFrame
gridded(meuse.grid) = TRUE
                                      # promote to SpatialGridDataFrame
data(meuse)
coordinates(meuse) = c("x", "y")
image(meuse.grid["dist"], main = "Distance to river Meuse")
points(coordinates(meuse), pch = "+")
image(meuse.grid["dist"], main = "Distance to river Meuse",
useRasterImage=TRUE)
points(coordinates(meuse), pch = "+")
# color scale:
layout(cbind(1,2), c(4,1),1)
image(meuse.grid["dist"])
imageScale(meuse.grid$dist, axis.pos=4, add.axis=FALSE)
axis(4,at=c(0,.2,.4,.8), las=2)
data(Rlogo)
d = dim(Rlogo)
cellsize = abs(c(gt[2],gt[6]))
cells.dim = c(d[1], d[2]) # c(d[2],d[1])
cellcentre.offset = c(x = gt[1] + 0.5 * cellsize[1], y = gt[4] - (d[2] - 0.5) * abs(cellsize[2]))
grid = GridTopology(cellcentre.offset, cellsize, cells.dim)
df = as.vector(Rlogo[,,1])
for (band in 2:d[3]) df = cbind(df, as.vector(Rlogo[,,band]))
df = as.data.frame(df)
names(df) = paste("band", 1:d[3], sep="")
Rlogo <- SpatialGridDataFrame(grid = grid, data = df)</pre>
summary(Rlogo)
image(Rlogo, red="band1", green="band2", blue="band3")
image(Rlogo, red="band1", green="band2", blue="band3",
useRasterImage=FALSE)
is.na(Rlogo$band1) <- Rlogo$band1 == 255
is.na(Rlogo$band2) <- Rlogo$band2 == 255
is.na(Rlogo$band3) <- Rlogo$band3 == 255
Rlogo$i7 <- 7
image(Rlogo, "i7")
image(Rlogo, red="band1", green="band2", blue="band3", add=TRUE)
```

is.projected

Sets or retrieves projection attributes on classes extending Spatial-Data

#### **Description**

Sets or retrieves projection attributes on classes extending SpatialData; set or retrieve option value for error or warning on exceedance of geographical coordinate range, set or retrieve option value for exceedance tolerance of geographical coordinate range. Note that only "+proj=longlat +ellps=WGS84" is accepted for geographical coordinates, which must be ordered (eastings, northings); the "+ellps="

is.projected 35

definition must be given (or expanded internally from a given "+datum=" value) for recent versions of the PROJ.4 library, and should be set to an appropriate value.

#### Usage

```
is.projected(obj)
proj4string(obj)
proj4string(obj) <- value
get_ll_warn()
get_ll_TOL()
get_ReplCRS_warn()
set_ll_warn(value)
set_ll_TOL(value)
set_ReplCRS_warn(value)</pre>
```

# Arguments

obj

An object of class or extending Spatial-class

value

For proj4string CRS object, containing a valid proj4 string; attempts to assign an object containing "longlat" to data extending beyond longitude [-180, 360] or lattitude [-90, 90] will be stopped. For set\_ll\_warn a single logical value, if FALSE (default) error on range exceedance, if TRUE, warning. For set\_ll\_TOL the value of the power of .Machine\$double.eps (default 0.25) to use as tolerance in testing range exceedance. set\_ReplCRS\_warn may be used to turn off warnings issued when changing object CRS with the proj4string replacement method (by setting value=FALSE).

#### **Details**

proj4 strings are operative through CRAN package rgdal. For strings defined as "longlat", the minimum longitude should be -180, the maximum longitude 360, the minimum latitude -90, and the maximum latitude 90. Note that the proj4string replacement method does not project spatial data - for this use spTransform methods in the rgdal package.

#### Value

is.projected returns a logical that may be NA; proj4string returns a character vector of length 1.

## Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

CRS

36 Line

Line

create objects of class Line or Lines

## **Description**

create objects of class Line or Lines from coordinates

## Usage

```
Line(coords)
Lines(slinelist, ID)
```

## **Arguments**

coords 2-column numeric matrix with coordinates for a single line

slinelist list with elements of class Line-class

ID a single word unique character identifier, character vector of length one

## Value

Line returns an object of class Line-class; Lines returns an object of class Lines-class

## See Also

SpatialLines-class

```
# from the sp vignette:
11 = cbind(c(1,2,3),c(3,2,2))
11a = cbind(l1[,1]+.05,l1[,2]+.05)
12 = cbind(c(1,2,3),c(1,1.5,1))
S11 = Line(l1)
S11a = Line(l1a)
S12 = Line(l2)
S1 = Lines(list(S11, S11a), ID="a")
S2 = Lines(list(S12), ID="b")
```

Line-class 37

Line-class

Class "Line"

# Description

class for line objects

## **Objects from the Class**

Objects can be created by calls of the form new("Line", ...), or (preferred) by calls to the function Line

#### **Slots**

coords: Object of class "matrix", containing the line coordinates

### Methods

```
coordinates signature(obj = "Line"): retrieve coordinates from line
lines signature(x = "Line"): add lines to a plot
```

## Author(s)

Roger Bivand, Edzer Pebesma

#### See Also

Lines-class, SpatialLines-class

Lines-class

Class "Lines"

## **Description**

class for sets of line objects

# Arguments

SL, Lines

an Lines object

# **Objects from the Class**

Objects can be created by calls to the function Line

38 loadMeuse

## **Slots**

```
Lines: Object of class "list", containing elements of class Line-class ID: "character" vector of lenght one, with unique identifier string
```

### Methods

```
coordinates signature(obj = "Line"): retrieve coordinates from lines; returns list with matri-
ces
lines signature(x = "Line"): add lines to a plot
```

# Author(s)

Roger Bivand, Edzer Pebesma

### See Also

Lines-class, SpatialLines-class

loadMeuse

deprecated function to load the Meuse data set

# Description

deprecated function to load the Meuse data set

# Usage

```
loadMeuse()
```

## Value

none; it prints a warning to run demo(meuse)

## See Also

meuse, meuse.grid

# **Examples**

demo(meuse)

mapasp 39

mapasp	Calculate aspect ratio for plotting geographic maps; create nice degree axis labels

# Description

Calculate aspect ratio for plotting geographic maps; create nice degree axis labels

# Usage

```
mapasp(data, xlim, ylim)
degreeLabelsEW(x)
degreeLabelsNS(x)
```

## **Arguments**

data	object of class or extending Spatial
xlim	the xlim argument passed (or derived from bounding box)
ylim	the ylim argument passed (or derived from bounding box)
X	numeric; values at which tics and marks will be generated

## Value

```
mapasp is used for the aspect argument in lattice plots and spplot; let x = dy/dx, with dy and dx the y- and x-size of the map. let s = 1/\cos((My * pi)/180) with My the y coordinate of the middle of the map (the mean of ylim) for latlong (longlat) data, mapasp returns s * x. for other data, mapasp returns "iso".
```

# Note

the values for x are typically obtained from axTicks

# See Also

levelplot in package lattice

40 merge

merge

Merge a Spatial\* object having attributes with a data.frame

# Description

Merge a Spatial object having a data.frame (i.e. merging of non-spatial attributes).

### Usage

```
## S4 method for signature 'Spatial,data.frame'
merge(x, y, by = intersect(names(x), names(y)),
  by.x = by, by.y = by, all.x = TRUE, suffixes = c(".x",".y"),
  incomparables = NULL, duplicateGeoms = FALSE, ...)
```

# Arguments

x	object deriving from Spatial
У	object of class $\mathtt{data}$ . frame, or any other class that can be coerced to a $\mathtt{data}$ .frame with $\mathtt{as}$ . $\mathtt{data}$ . frame
by, by.x, by.y	specifications of the common columns. See 'Details' in (base) merge.
all.x	logical; if TRUE, then the returned object will have all rows of $x$ , even those that has no matching row in $y$ . These rows will have NAs in those columns that are usually filled with values from $y$
suffixes	character(2) specifying the suffixes to be used for making non-by names() unique.
incomparables	values which cannot be matched. See match.
duplicateGeoms	logical; if TRUE geometries in $\boldsymbol{x}$ are duplicated if there are multiple matches between records in $\boldsymbol{x}$ and $\boldsymbol{y}$
	arguments to be passed to or from methods.

### Value

```
a Spatial* object
```

# Author(s)

Robert J. Hijmans

# See Also

merge

meuse 41

meuse

Meuse river data set

## **Description**

This data set gives locations and topsoil heavy metal concentrations, along with a number of soil and landscape variablesat the observation locations, collected in a flood plain of the river Meuse, near the village of Stein (NL). Heavy metal concentrations are from composite samples of an area of approximately 15 m x 15 m.

#### **Usage**

data(meuse)

#### **Format**

This data frame contains the following columns:

x a numeric vector; Easting (m) in Rijksdriehoek (RDH) (Netherlands topographical) map coordinates

y a numeric vector; Northing (m) in RDH coordinates

**cadmium** topsoil cadmium concentration, mg kg-1 soil ("ppm"); zero cadmium values in the original data set have been shifted to 0.2 (half the lowest non-zero value)

**copper** topsoil copper concentration, mg kg-1 soil ("ppm")

**lead** topsoil lead concentration, mg kg-1 soil ("ppm")

**zinc** topsoil zinc concentration, mg kg-1 soil ("ppm")

**elev** relative elevation above local river bed, m

**dist** distance to the Meuse; obtained from the nearest cell in meuse.grid, which in turn was derived by a spread (spatial distance) GIS operation, horizontal precision 20 metres; then normalized to \$[0,1]\$

om organic matter, kg (100 kg)-1 soil (percent)

**ffreq** flooding frequency class: 1 = once in two years; 2 = once in ten years; 3 = one in 50 years

soil soil type according to the 1:50 000 soil map of the Netherlands. 1 = Rd10A (Calcareous weakly-developed meadow soils, light sandy clay); 2 = Rd90C/VII (Non-calcareous weakly-developed meadow soils, heavy sandy clay to light clay); 3 = Bkd26/VII (Red Brick soil, fine-sandy, silty light clay)

**lime** lime class: 0 = absent, 1 = present by field test with 5% HCl

landuse landuse class: Aa Agriculture/unspecified = , Ab = Agr/sugar beetsm, Ag = Agr/small grains, Ah = Agr/??, Am = Agr/maize, B = woods, Bw = trees in pasture, DEN = ??, Fh = tall fruit trees, Fl = low fruit trees; Fw = fruit trees in pasture, Ga = home gardens, SPO = sport field, STA = stable yard, Tv = ??, W = pasture

dist.m distance to river Meuse in metres, as obtained during the field survey

42 meuse.grid

#### Note

row.names refer to the original sample number.

Soil units were mapped with a minimum delination width of 150 m, and so somewhat generalize the landscape.

Approximate equivalent World Reference Base 2002 for Soil Resources names are: Rd10A Gleyic Fluvisols; Rd90C Haplic Fluvisols; Bkd26 Haplic Luvisols. Units Rd90C and Bkd26 have winter groundwater > 80cm, summer > 120cm depth.

#### Author(s)

Field data were collected by Ruud van Rijn and Mathieu Rikken; compiled for R by Edzer Pebesma; description extended by David Rossiter

#### References

M G J Rikken and R P G Van Rijn, 1993. Soil pollution with heavy metals - an inquiry into spatial variation, cost of mapping and the risk evaluation of copper, cadmium, lead and zinc in the floodplains of the Meuse west of Stein, the Netherlands. Doctoraalveldwerkverslag, Dept. of Physical Geography, Utrecht University

P.A. Burrough, R.A. McDonnell, 1998. Principles of Geographical Information Systems. Oxford University Press.

Stichting voor Bodemkartering (STIBOKA), 1970. Bodemkaart van Nederland : Blad 59 Peer, Blad 60 West en 60 Oost Sittard: schaal 1 : 50 000. Wageningen, STIBOKA.

```
http://www.gstat.org/
```

### **Examples**

```
data(meuse)
summary(meuse)
coordinates(meuse) <- ~x+y
proj4string(meuse) <- CRS("+init=epsg:28992")</pre>
```

meuse.grid

Prediction Grid for Meuse Data Set

# Description

The meuse.grid data frame has 3103 rows and 7 columns; a grid with 40 m x 40 m spacing that covers the Meuse study area (see meuse)

## Usage

```
data(meuse.grid)
```

meuse.grid\_ll 43

### **Format**

This data frame contains the following columns:

```
x a numeric vector; x-coordinate (see meuse)y a numeric vector; y-coordinate (see meuse)
```

**dist** distance to the Meuse river; obtained by a spread (spatial distance) GIS operation, from border of river; normalized to \$[0,1]\$

**ffreq** flooding frequency class, for definitions see this item in meuse; it is not known how this map was generated

part.a arbitrary division of the area in two areas, a and b

```
part.b see part.a
```

**soil** soil type, for definitions see this item in meuse; it is questionable whether these data come from a real soil map, they do not match the published 1:50 000 map

#### **Details**

x and y are in RD New, the Dutch topographical map coordinate system. Roger Bivand projected this to UTM in the R-Grass interface package.

### **Source**

```
http://www.gstat.org/
```

### References

See the meuse documentation

## **Examples**

```
data(meuse.grid)
coordinates(meuse.grid) = ~x+y
proj4string(meuse.grid) <- CRS("+init=epsg:28992")
gridded(meuse.grid) = TRUE
spplot(meuse.grid)</pre>
```

meuse.grid\_ll

Prediction Grid for Meuse Data Set, geographical coordinates

## **Description**

The object contains the meuse.grid data as a SpatialPointsDataFrame after transformation to WGS84 and geographical coordinates.

#### Usage

```
data(meuse.grid_ll)
```

44 meuse.riv

# **Format**

The format is: Formal class 'SpatialPointsDataFrame' [package "sp"].

## Source

See the meuse documentation

# **Examples**

```
data(meuse.grid_ll)
```

meuse.riv

River Meuse outline

## **Description**

The meuse.riv data consists of an outline of the Meuse river in the area a few kilometers around the meuse data set.

The meuse.area polygon has an outline of meuse.grid. See example below how it can be created from meuse.grid.

## Usage

```
data(meuse.riv)
data(meuse.area)
```

## **Format**

```
meuse.riv: two-column data.frame containing 176 coordinates.
meuse.area: two-column matrix with coordinates of outline.
```

### **Details**

x and y are in RDM, the Dutch topographical map coordinate system. See examples of spTransform in the rgdal package for projection parameters.

## References

See the meuse documentation

over-methods 45

### **Examples**

```
data(meuse.riv)
plot(meuse.riv, type = "l", asp = 1)
data(meuse.grid)
coordinates(meuse.grid) = c("x", "y")
gridded(meuse.grid) = TRUE
image(meuse.grid, "dist", add = TRUE)
data(meuse)
coordinates(meuse) = c("x", "y")
meuse.sr = SpatialPolygons(list(Polygons(list(Polygon(meuse.riv))), "meuse.riv")))
spplot(meuse.grid, col.regions=bpy.colors(), main = "meuse.grid",
  sp.layout=list(
list("sp.polygons", meuse.sr),
list("sp.points", meuse, pch="+", col="black")
)
spplot(meuse, "zinc", col.regions=bpy.colors(), main = "zinc, ppm",
  cuts = c(100, 200, 400, 700, 1200, 2000), key.space = "right",
  sp.layout= list("sp.polygons", meuse.sr, fill = "lightblue")
)
# creating meuse.area from meuse.grid:
if (require(rgeos)) {
    meuse.area = gUnaryUnion(as(meuse.grid, "SpatialPolygons"))
plot(meuse.area)
}
```

over-methods

consistent spatial overlay for points, grids and polygons

# Description

consistent spatial overlay for points, grids and polygons: at the spatial locations of object x retrieves the indexes or attributes from spatial object y

## Usage

```
over(x, y, returnList = FALSE, fn = NULL, ...)
x %over% y
```

### **Arguments**

```
    x geometry (locations) of the queries
    y layer from which the geometries or attributes are queried
    returnList logical; see value
    fn (optional) a function; see value
    arguments passed on to function fn
```

46 over-methods

#### Value

If y is only geometry an object of length length(x). If returnList is FALSE, a vector with the (first) index of y for each geometry (point, grid cell centre, polygon or lines) matching x. if returnList is TRUE, a list of length length(x), with list element i the vector of all indices of the geometries in y that correspond to the \$i\$-th geometry in x.

If y has attribute data, attribute data are returned. returnList is FALSE, a data.frame with number of rows equal to length(x) is returned, if it is TRUE a list with length(x) elements is returned, with a list element the data.frame elements of all geometries in y that correspond to that element of x.

In case the rgeos over methods are used, matching is done by gRelate, which uses DE-9IM (https://en.wikipedia.org/wiki/DE-9IM). From the string returned, characters 1, 2, 4 and 5 are used, indicating the dimension of the overlap of the inner and boundary of each x geometry with the inner and boundary of each y geometry. The order in which matched y geometries are returned is determined by the dimension of the overlap (2: area overlap, 1: line in common, 0: point in common), and then by the position in the string (1, 2, 4, 5, meaning points in polygons are prefered over points on polygon boundaries).

### Methods

- x = "SpatialPoints", y = "SpatialPolygons" returns a numeric vector of length equal to the number of points; the number is the index (number) of the polygon of y in which a point falls; NA denotes the point does not fall in a polygon; if a point falls in multiple polygons, the last polygon is recorded.
- x = "SpatialPointsDataFrame", y = "SpatialPolygons" equal to the previous method, except that an argument fn=xxx is allowed, e.g. fn = mean which will then report a data.frame with the mean attribute values of the x points falling in each polygon (set) of y
- x = "SpatialPoints", y = "SpatialPolygonsDataFrame" returns a data.frame of the second argument with row entries corresponding to the first argument
- x = "SpatialPolygons", y = "SpatialPoints" returns the polygon index of points in y; if x is a SpatialPolygonsDataFrame, a data.frame with rows from x corresponding to points in y is returned.
- x = "SpatialGridDataFrame", y = "SpatialPoints" returns object of class SpatialPointsDataFrame with grid attribute values x at spatial point locations y; NA for NA grid cells or points outside grid, and NA values on NA grid cells.
- x = "SpatialGrid", y = "SpatialPoints" returns grid values x at spatial point locations y; NA for NA grid cells or points outside the grid
- x = "SpatialPixelsDataFrame", y = "SpatialPoints" returns grid values x at spatial point locations y; NA for NA grid cells or points outside the grid
- x = "SpatialPixels", y = "SpatialPoints" returns grid values x at spatial point locations y; NA for NA grid cells or points outside the grid
- x = "SpatialPoints", y = "SpatialGrid" xx
- x = "SpatialPoints", y = "SpatialGridDataFrame" xx
- x = "SpatialPoints", y = "SpatialPixels" xx
- x = "SpatialPoints", y = "SpatialPixelsDataFrame" xx
- x = "SpatialPolygons", y = "SpatialGridDataFrame" xx

over-methods 47

#### Note

over can be seen as a left outer join in SQL; the match is a spatial intersection.

points on a polygon boundary and points corresponding to a polygon vertex are considered to be inside the polygon.

These methods assume that pixels and grid cells are never overlapping; for objects of class SpatialPixels this is not guaranteed.

over methods that involve SpatialLines objects, or pairs of SpatialPolygons require package rgeos, and use gIntersects.

### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

vignette("over") for examples and figures; point.in.polygon, package gIntersects

```
r1 = cbind(c(180114, 180553, 181127, 181477, 181294, 181007, 180409,
180162, 180114), c(332349, 332057, 332342, 333250, 333558, 333676,
332618, 332413, 332349))
r2 = cbind(c(180042, 180545, 180553, 180314, 179955, 179142, 179437,
179524, 179979, 180042), c(332373, 332026, 331426, 330889, 330683,
331133, 331623, 332152, 332357, 332373))
r3 = cbind(c(179110, 179907, 180433, 180712, 180752, 180329, 179875,
179668, 179572, 179269, 178879, 178600, 178544, 179046, 179110),
c(331086, 330620, 330494, 330265, 330075, 330233, 330336, 330004,
329783, 329665, 329720, 329933, 330478, 331062, 331086))
r4 = cbind(c(180304, 180403, 179632, 179420, 180304),
c(332791, 333204, 333635, 333058, 332791))
sr1=Polygons(list(Polygon(r1)), "r1")
sr2=Polygons(list(Polygon(r2)), "r2")
sr3=Polygons(list(Polygon(r3)),"r3")
sr4=Polygons(list(Polygon(r4)),"r4")
sr=SpatialPolygons(list(sr1,sr2,sr3,sr4))
srdf=SpatialPolygonsDataFrame(sr, data.frame(cbind(1:4,5:2),
row.names=c("r1","r2","r3","r4")))
data(meuse)
coordinates(meuse) = ~x+y
# plot(meuse)
polygon(r1)
polygon(r2)
polygon(r3)
polygon(r4)
# retrieve mean heavy metal concentrations per polygon:
over(sr, meuse[,1:4], fn = mean)
```

48 panel.spplot

```
# return the number of points in each polygon:
sapply(over(sr, geometry(meuse), returnList = TRUE), length)
data(meuse.grid)
coordinates(meuse.grid) = \sim x+y
gridded(meuse.grid) = TRUE
over(sr, geometry(meuse))
over(sr, meuse)
over(sr, geometry(meuse), returnList = TRUE)
over(sr, meuse, returnList = TRUE)
over(meuse, sr)
over(meuse, srdf)
# same thing, with grid:
over(sr, meuse.grid)
over(sr, meuse.grid, fn = mean)
over(sr, meuse.grid, returnList = TRUE)
over(meuse.grid, sr)
over(meuse.grid, srdf, fn = mean)
over(as(meuse.grid, "SpatialPoints"), sr)
over(as(meuse.grid, "SpatialPoints"), srdf)
```

panel.spplot

panel and panel utility functions for spplot

## **Description**

panel functions for spplot functions, and functions that can be useful within these panel functions

## Usage

```
spplot.key(sp.layout, rows = 1, cols = 1)
SpatialPolygonsRescale(obj, offset, scale = 1, fill = "black", col = "black",
plot.grid = TRUE, ...)
sp.lines(obj, col = 1, ...)
sp.points(obj, pch = 3, ...)
sp.polygons(obj, col = 1, fill = "transparent", ...)
sp.grid(obj, col = 1, alpha = 1,..., at = pretty(obj[[1]]), col.regions = col)
sp.text(loc, txt, ...)
sp.panel.layout(lst, p.number, ...)
bbexpand(x, fraction)
```

panel.spplot 49

#### **Arguments**

sp.layout list; see spplot for definition

rows integer; panel row(s) for which the layout should be drawn cols integer; panel column(s) for which the layout should be drawn

object of class SpatialPolygons-class for SpatialPolygonsRescale; of class

SpatialLines-class, Lines-class or Line-class for sp.lines of a class that has a coordinates-methods for sp.points; of class SpatialPolygons-class for sp.polygons. When obj is character, the actual object is retrieved by get(obj) before its class

is evaluated.

offset offset for shifting a Polygons object

scale scale for rescaling

fill fill color col line color

plot.grid logical; plot through grid functions (TRUE), or through traditional graphics

functions (FALSE)

pch plotting character

at numeric; values at which colour breaks should occur

col.regions colours to fill the grid cells, defaults to col

loc numeric vector of two elements

txt text to be plotted

alpha alpha (transparency) level

1st sp.layout argument, see spplot

p.number panel number; in a panel, panel.number() should be passed to this argument

x length two numeric vector, containing a range

fraction fraction to expand the range by

... arguments passed to the underlying panel, lattice or grid functions

#### Note

The panel functions of spplot, panel.gridplot for grids, panel.pointsplot for points, or panel.polygonsplot for lines or polygons can be called with arguments (x,y,...). Customizing spplot plots can be done by extending the panel function, or by supplying an sp.layout argument; see the documentation for spplot. Inside these panel functions, sp.panel.layout is called to deal with plotting the items in a sp.layout object.

SpatialPolygonsRescale scales and shifts an object of class SpatialPolygons-class; this is useful e.g. for scale bars, or other layout items.

sp.lines, sp.points, sp.polygons and sp.text plot lines, points, polygons or text in a panel. spplot.key draws the sp.layout object at given rows/cols.

sp.pagefn can be passed as a page argument, and will call function spplot.key for the last panel drawn on a page.

50 point.in.polygon

### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### References

http://rspatial.r-forge.r-project.org/gallery/ has a graph gallery with examples with R code.

#### See Also

```
spplot, spplot-methods
```

point.in.polygon

do point(s) fall in a given polygon?

## **Description**

verifies for one or more points whether they fall in a given polygon

### Usage

```
point.in.polygon(point.x, point.y, pol.x, pol.y, mode.checked=FALSE)
```

## **Arguments**

point.x	numerical array of x-coordinates of points
point.y	numerical array of y-coordinates of points
pol.x	numerical array of x-coordinates of polygon
pol.y	numerical array of y-coordinates of polygon
mode.checked	default FALSE, used internally to save time when all the other argument are
	known to be of storage mode double

#### Value

integer array; values are: 0: point is strictly exterior to pol; 1: point is strictly interior to pol; 2: point lies on the relative interior of an edge of pol; 3: point is a vertex of pol.

## References

Uses the C function InPoly(). InPoly is Copyright (c) 1998 by Joseph O'Rourke. It may be freely redistributed in its entirety provided that this copyright notice is not removed.

```
# open polygon:
point.in.polygon(1:10,1:10,c(3,5,5,3),c(3,3,5,5))
# closed polygon:
point.in.polygon(1:10,rep(4,10),c(3,5,5,3,3),c(3,3,5,5,3))
```

Polygon-class 51

Polygon-class

Class "Polygon"

# Description

class for spatial polygon

## **Objects from the Class**

Objects can be created by calls to the function Polygon

#### **Slots**

ringDir: Object of class "integer"; the ring direction of the ring (polygon) coordinates, holes are expected to be anti-clockwise

labpt: Object of class "numeric"; an x, y coordinate pair forming the label point of the polygon

area: Object of class "numeric"; the planar area of the polygon, does not respect projection as objects of this class have no projection defined

hole: Object of class "logical"; does the polygon seem to be a hole

coords: Object of class "matrix"; coordinates of the polygon; first point should equal the last point

#### **Extends**

Class "Line", directly.

### Methods

No methods defined with class "Polygon" in the signature.

## Author(s)

Roger Bivand

### See Also

Polygons-class, SpatialPolygons-class

Polygons-class

polygons	sets spatial coordinates to create spatial data, or retrieves spatial
	coordinates

### **Description**

sets spatial coordinates to create spatial data, or retrieves spatial coordinates

## Usage

```
polygons(obj)
polygons(object) <- value</pre>
```

## Arguments

```
obj object of class "SpatialPolygons" or "SpatialPolygonsDataFrame"
object of class "data.frame"
value object of class "SpatialPolygons"
```

### Value

polygons returns the SpatialPolygons of obj; polygons<- promotes a data.frame to a SpatialPolygonsDataFrame object

# **Examples**

```
grd <- GridTopology(c(1,1), c(1,1), c(10,10))
polys <- as.SpatialPolygons.GridTopology(grd)
centroids <- coordinates(polys)
x <- centroids[,1]
y <- centroids[,2]
z <- 1.4 + 0.1*x + 0.2*y + 0.002*x*x
df <- data.frame(x=x, y=y, z=z, row.names=row.names(polys))
polygons(df) <- polys
class(df)
summary(df)</pre>
```

Polygons-class

Class "Polygons"

## **Description**

Collection of objects of class "Polygon"

## **Objects from the Class**

Objects can be created by calls to the function Polygons

polygons-methods 53

#### Slots

Polygons: Object of class "list"; list with objects of class Polygon-class

plot0rder: Object of class "integer"; order in which the Polygon objects should be plotted, currently by order of decreasing size

labpt: Object of class "numeric"; pair of x, y coordinates giving a label point, the label point of the largest polygon component

ID: Object of class "character"; unique identifier string

area: Object of class "numeric"; the gross total planar area of the Polygon list but not double-counting holes (changed from 0.9-58 - islands are summed, holes are ignored rather than subtracted); these values are used to make sure that polygons of a smaller area are plotted after polygons of a larger area, does not respect projection as objects of this class have no projection defined

#### Methods

No methods defined with class "Polygons" in the signature.

#### Note

By default, single polygons (where Polygons is a list of length one) are not expected to be holes, but in multiple polygons, hole definitions for member polygons can be set. Polygon objects belonging to an Polygons object should either not overlap one-other, or should be fully included (as lakes or islands in lakes). They should not be self-intersecting. Checking of hole FALSE/TRUE status for Polygons objects is included in the maptools package using functions in the rgeos package, function checkPolygonsHoles().

## Author(s)

Roger Bivand

polygons-methods

Retrieve polygons from SpatialPolygonsDataFrame object

## **Description**

Retrieve polygons from SpatialPolygonsDataFrame object

# Methods for polygons

```
obj = "SpatialPolygons" object of, or deriving from, SpatialPolygons
```

obj = "SpatialPolygonsDataFrame" object of, or deriving from, SpatialPolygonsDataFrame

## Methods for "polygons<-"

object = "data.frame", value="SpatialPolygons" promote data.frame to object of class SpatialPolygonsDataFrameclass, by specifying polygons 54 read.asciigrid

read.	•	•		•	-1
raan	2661	7	$\sigma r$	· Т	$\alpha$

read/write to/from (ESRI) asciigrid format

## **Description**

read/write to/from ESRI asciigrid format

## Usage

```
read.asciigrid(fname, as.image = FALSE, plot.image = FALSE, colname = fname,
proj4string = CRS(as.character(NA)))
write.asciigrid(x, fname, attr = 1, na.value = -9999, ...)
```

## **Arguments**

fname	file name
as.image	logical; if FALSE, a list is returned, ready to be shown with the image command; if FALSE an object of class SpatialGridDataFrame-class is returned
plot.image	logical; if TRUE, an image of the map is plotted
colname	alternative name for data column if not file name
proj4string	A CRS object setting the projection arguments of the Spatial Grid returned
Х	object of class SpatialGridDataFrame
attr	attribute column; if missing, the first column is taken; a name or a column number may be given
na.value	numeric; value given to missing valued cells in the resulting map
	arguments passed to write.table, which is used to write the numeric data

## Value

read.asciigrid returns the grid map read; either as an object of class SpatialGridDataFrame-class or, if as.image is TRUE, as list with components x, y and z.

## Author(s)

Edzer Pebesma

## See Also

```
as.image.SpatialGridDataFrame, image
```

```
x <- read.asciigrid(system.file("external/test.ag", package="sp")[1])
class(x)
image(x)</pre>
```

recenter-methods 55

recenter-methods

Methods for Function recenter in Package 'sp'

## **Description**

Methods for function recenter in package **sp** to shift or re-center geographical coordinates for a Pacific view. All longitudes < 0 are added to 360, to avoid for instance parts of Alaska being represented on the far left and right of a plot because they have values straddling 180 degrees. In general, using a projected coordinate reference system is to be prefered, but this method permits a geographical coordinate reference system to be used. This idea was suggested by Greg Snow, and corresponds to the two world representations in the **maps** package.

#### Methods

```
obj = "SpatialPolygons" recenter a SpatialPolygons object
obj = "Polygons" recenter a Polygons object
obj = "Polygon" recenter an Polygon object
obj = "SpatialLines" recenter a SpatialLines object
obj = "Lines" recenter a Lines object
obj = "Line" recenter an Line object
```

```
crds <- matrix(c(179, -179, -179, 179, 50, 50, 52, 52), ncol=2)
SL <- SpatialLines(list(Lines(list(Line(crds)), "1")),</pre>
CRS("+proj=longlat +ellps=WGS84"))
bbox(SL)
SLr <- recenter(SL)</pre>
bbox(SLr)
rcrds <- rbind(crds, crds[1,])</pre>
SpP <- SpatialPolygons(list(Polygons(list(Polygon(rcrds)), ID="r1")),</pre>
proj4string=CRS("+proj=longlat +ellps=WGS84"))
bbox(SpP)
SpPr <- recenter(SpP)</pre>
bbox(SpPr)
opar <- par(mfrow=c(1,2))</pre>
plot(SpP)
plot(SpPr)
par(opar)
crds \leftarrow matrix(c(-1, 1, 1, -1, 50, 50, 52, 52), ncol=2)
SL <- SpatialLines(list(Lines(list(Line(crds)), "1")),</pre>
CRS("+proj=longlat +ellps=WGS84"))
bbox(SL)
SLr <- recenter(SL)</pre>
bbox(SLr)
rcrds <- rbind(crds, crds[1,])</pre>
SpP <- SpatialPolygons(list(Polygons(list(Polygon(rcrds)), ID="r1")),</pre>
```

S6 Rlogo

```
proj4string=CRS("+proj=longlat +ellps=WGS84"))
bbox(SpP)
SpPr <- recenter(SpP)
bbox(SpPr)
opar <- par(mfrow=c(1,2))
plot(SpP)
plot(SpPr)
par(opar)</pre>
```

Rlogo

Rlogo jpeg image

### **Description**

Rlogo jpeg image data as imported by getRasterData in the rgdal package

### Usage

```
data(Rlogo)
```

### **Format**

```
## Not run:
library(rgdal)
logo <- system.file("pictures/Rlogo.jpg", package="rgdal")[1]</pre>
x <- GDAL.open(logo)</pre>
gt = .Call('RGDAL_GetGeoTransform', x, PACKAGE="rgdal")
data <- getRasterData(x)</pre>
GDAL.close(x)
## End(Not run)
data(Rlogo)
d = dim(Rlogo)
cellsize = abs(c(gt[2],gt[6]))
cells.dim = c(d[1], d[2]) # c(d[2], d[1])
cellcentre.offset = c(x = gt[1] + 0.5 * cellsize[1], y = gt[4] - (d[2] - 0.5) * abs(cellsize[2]))
grid = GridTopology(cellcentre.offset, cellsize, cells.dim)
df = as.vector(Rlogo[,,1])
for (band in 2:d[3]) df = cbind(df, as.vector(Rlogo[,,band]))
df = as.data.frame(df)
names(df) = paste("band", 1:d[3], sep="")
Rlogo <- SpatialGridDataFrame(grid = grid, data = df)</pre>
summary(Rlogo)
spplot(Rlogo, zcol=1:3, names.attr=c("red", "green", "blue"),
col.regions=grey(0:100/100),
main="example of three-layer (RGB) raster image", as.table=TRUE)
```

select.spatial 57

select.spatial	select points spatially	
----------------	-------------------------	--

## **Description**

select a number of points by digitizing the area they fall in

# Usage

```
select.spatial(data, digitize = TRUE, pch = "+", rownames = FALSE)
```

### **Arguments**

data data object of class, or extending SpatialPoints; this object knows about its x

and y coordinate

digitize logical; if TRUE, points in a digitized polygon are selected; if FALSE, points

identified by mouse clicks are selected

pch plotting character used for points

rownames logical; if FALSE, row (coordinate) numbers are returned; if TRUE and data

contains a data.frame part, row.names for selected points in the data.frame are

returned.

#### Value

if rownames == FALSE, array with either indexes (row numbers) of points inside the digitized polygon; if rownames == TRUE, character array with corresponding row names in the data.frame part

#### See Also

point.in.polygon, locator, SpatialPoints-class, SpatialPointsDataFrame-class

```
data(meuse)
## the following command requires user interaction: left mouse
## selects points, right mouse ends digitizing
data(meuse)
coordinates(meuse) = c("x", "y")
# select.spatial(meuse)
```

sp A package providing classes and methods for spatial data: points, lines, polygons and grids

### **Description**

This package provides S4 classes for importing, manipulating and exporting spatial data in R, and for methods including print/show, plot, subset, [, [[, \$, names, dim, summary, and a number of methods specific to spatial data handling.

#### Introduction

Several spatial statistical packages have been around for a long while, but no organized set of classes for spatial data has yet been devised. Many of the spatial packages make their own assumptions, or use their own class definitions for spatial data, making it inconvenient to move from one package to another. This package tries to provide a solid set of classes for many different types of spatial data. The idea is that spatial statistical packages will either support these classes (i.e., directly read and write them) or will provide conversion to them, so that we have a base class set with which any package can exchange. This way, many-to-many conversions can be replace with one-to-many conversions, provided either in this package or the spatial packages. Wherever possible conversion (coercion) functions are automatic, or provided by sp.

External packages that depend on sp will provide importing and exporting from and to external GIS formats, e.g. through GDAL, OGR or shapelib.

In addition, this package tries to provide convenient methods to print, summarize and plot such spatial data.

## **Dimensions**

In principal, geographical data are two-dimensional, on a flat surface (a map) or on a sphere (the earth). This package provides space for dealing with higher dimensional data where possible; this is e.g. very simple for points and grids, but hard to do for polygons. Plotting functions are devised primarily for two-dimensional data, or two-dimensional projections of higher dimensional data.

#### Coordinate reference systems

Central to spatial data is that they have a coordinate reference system, which is coded in object of CRS class. Central to operations on different spatial data sets is that their coordinate reference system is compatible (i.e., identical).

This CRS can be a character string describing a reference system in a way understood by the PROJ.4 projection library, or a (character) missing value. An interface to the PROJ.4 library is available only if the R package rgdal is present.

## Class structure

All spatial classes derive from a basic class Spatial, which only provides a bounding box and a CRS. This class has no useful instances, but useful derived classes.

sp-deprecated 59

SpatialPoints extends Spatial and has coordinates. The method coordinates extracts the numeric matrix with coordinates from an object of class SpatialPoints, or from other (possibly derived) classes that have points.

Objects of class SpatialGrid points on a regular grid. Either a full grid is stored or a partial grid (i.e., only the non-missing valued cells); calling coordinates on them will give the coordinates for the grid cells.

SpatialPoints, SpatialPixels and SpatialGrid can be of arbitray dimension, although most of the effort is in making them work for two dimensional data.

SpatialLines provides lines, and SpatialPolygons provides polygons, i.e., lines that end where they start and do not intersect with itself. SpatialLines and SpatialPolygons only have two-dimensional data

SpatialPointsDataFrame extends SpatialPoints with a data slot, having a data.frame with attribute data. Similarly, SpatialPixelsDataFrame, SpatialLinesDataFrame, SpatialPolygonsDataFrame extend the primary spatial information with attribute data.

#### References

```
PROJ.4: https://github.com/OSGeo/proj.4
GDAL and OGR: http://www.gdal.org/.
```

### Authors

sp is a collaborative effort of Edzer Pebesma, Roger Bivand, Barry Rowlingson and Virgilo G\'omez-Rubio.

sp-deprecated

Deprecated functions in sp

# Description

Deprecated functions is sp: getSpP\*, getPolygon\*, getLines\* getSL\*

### Note

For overlay the new implementation is found in the over method; this works slightly different and more consistent.

60 Spatial-class

## **Description**

An abstract class from which useful spatial classes are derived

# Usage

```
Spatial(bbox, proj4string = CRS(as.character(NA)))
## S3 method for class 'Spatial'
subset(x, subset, select, drop = FALSE, ...)
```

## **Arguments**

bbox a bounding box matrix
proj4string a CRS object

x object of class Spatial
subset see subset.data.frame
select see subset.data.frame

drop see subset.data.frame

... passed through

## **Objects from the Class**

are never to be generated; only derived classes can be meaningful

### **Slots**

bbox: Object of class "matrix"; 2-column matrix holding the minimum in first and maximum in second column for the x-coordinate (first row), y-coordinate (second row) and optionally, for points and grids only, further coordinates. The constructed Spatial object will be invalid if any bbox values are NA or infinite. The column names must be c("min", "max")

proj4string: Object of class "CRS"; holding a valid proj4 string, which can be used for unprojecting or reprojecting coordinates; it is initialised to NA. Other strings are checked for validity in the rgdal package, but attempts to assign a string containing "longlat" to data extending beyond longitude [-180, 360] or lattitude [-90, 90] will be stopped or warned, use set\_ll\_warn to warn rather than stop, and set\_ll\_TOL to change the default tolerance for the range exceedance tests.

Spatial-class 61

#### Methods

```
bbox signature(obj = "Spatial"): retrieves the bbox element
dimensions signature(obj = "Spatial"): retrieves the number of spatial dimensions spanned
gridded signature(obj = "Spatial"): logical, tells whether the data is on a regular spatial grid
plot signature(x = "Spatial", y = "missing"): plot method for spatial objects; does nothing
     but setting up a plotting region choosing a suitable aspect if not given(see below), colouring
     the plot background using either a bg= argument or par("bg"), and possibly drawing axes.
summary signature(object = "Spatial"): summarize object
```

\$ retrieves attribute column

\$<- sets or replaces attribute column, or promote a geometry-only object to an object having an

### plot method arguments

The plot method for "Spatial" objects takes the following arguments:

x object of class Spatial

**xlim** default NULL; the x limits (x1, x2) of the plot

ylim default NULL; the y limits of the plot

asp default NA; the y/x aspect ratio

axes default FALSE; a logical value indicating whether both axes should be drawn

bg default par ("bg"); colour to be used for the background of the device region

**xaxs** The style of axis interval calculation to be used for the x-axis

yaxs The style of axis interval calculation to be used for the y-axis

lab A numerical vector of the form c(x, y, len) which modifies the default way that axes are annotated

setParUsrBB default FALSE; set the par "usr" bounding box; see below

**bgMap** object of class ggmap, or returned by function RgoogleMaps::GetMap

**expandBB** numeric; factor to expand the plotting region default: bbox(x) with on each side (1=below, 2=left, 3=above and 4=right); defaults to c(0,0,0,0); setting xlim or ylim overrides this.

... passed through

## Warning

this class is not useful in itself, but all spatial classes in this package derive from it

## Note

The default aspect for map plots is 1; if however data are not projected (coordinates are longlat), the aspect is by default set to 1/cos(My \* pi)/180) with My the y coordinate of the middle of the map (the mean of ylim, which defaults to the y range of bounding box).

62 SpatialGrid-class

The argument setParUsrBB may be used to pass the logical value TRUE to functions within plot. Spatial. When set to TRUE, par("usr") will be overwritten with c(xlim, ylim), which defaults to the bounding box of the spatial object. This is only needed in the particular context of graphic output to a specified device with given width and height, to be matched to the spatial object, when using par("xaxs") and par("yaxs") in addition to par(mar=c(0,0,0,0)).

#### Author(s)

r-spatial team; Edzer Pebesma, <edzer.pebesma@uni-muenster.de> Roger Bivand, Barry Rowlingson, Virgilio G\'omez-Rubio

#### See Also

```
SpatialPoints-class, SpatialGrid-class, SpatialPointsDataFrame-class, SpatialGridDataFrame-class
```

SpatialGrid-class

Class "SpatialGrid"

### Description

class for defining a full, rectangular grid of arbitrary dimension

# **Objects from the Class**

```
Objects are created by using e.g.

SpatialGrid(grid)

with grid of class GridTopology-class
```

#### Slots

```
grid object of class <a href="GridTopology-class">GridTopology-class</a>, defining the grid topology (offset, cellsize, dim) bbox: Object of class "matrix"; bounding box proj4string: Object of class "CRS"; projection
```

## Extends

```
Class "SpatialPoints" directly; Class "Spatial", by class "SpatialPoints".
```

### Methods

```
coordinates signature(x = "SpatialGrid"): calculates coordinates for each point on the grid;
    coordinates are not stored in objects of class SpatialGrid
summary signature(object = "SpatialGrid"): summarize object
plot signature(x = "SpatialGrid"): plots cell centers
"[" signature(x = "SpatialGrid"): select rows and columns
```

### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

```
SpatialGridDataFrame-class, SpatialGrid
```

## **Examples**

```
x = GridTopology(c(0,0), c(1,1), c(5,5))
class(x)
x
summary(x)
coordinates(x)
y = SpatialGrid(grid = x)
class(y)
y
```

```
SpatialGridDataFrame-class
```

Class "SpatialGridDataFrame"

## **Description**

Class for spatial attributes that have spatial locations on a (full) regular grid.

## **Objects from the Class**

Objects can be created by calls of the form as(x, "SpatialGridDataFrame"), where x is of class SpatialPixelsDataFrame-class, or by importing through rgdal. Ordered full grids are stored instead or unordered non-NA cells:

## **Slots**

```
grid: see GridTopology-class; grid parameters
bbox: Object of class "matrix"; bounding box
proj4string: Object of class "CRS"; projection
data: Object of class data.frame, containing attribute data
```

## Extends

```
Class "SpatialGrid", directly. Class "Spatial", by class "SpatialGrid".
```

#### Methods

```
coordinates signature(x = "SpatialGridDataFrame"): retrieves (and calculates!) coordinates
    [ signature(x = "SpatialGridDataFrame"): selects rows, columns, and attributes; returns an
         object of class SpatialGridDataFrame
    as.matrix signature(x = "SpatialGridDataFrame"): coerce to matrix; increasing col index
         corresponds to decreasing y coordinate, row index increases with coordinate index
    as.array signature(x = "SpatialGridDataFrame"): coerce to array; increasing array index for
         the second dimension corresponds to decreasing coordinates, all other coordinate dimensions
         increase with array index
    cbind signature(...): if arguments have identical topology, combine their attribute values
Plot method arguments
    The plot methods for "SpatialPixelsDataFrame" or "SpatialGridDataFrame" objects take the fol-
    lowing arguments:
    x object of class SpatialPixelsDataFrame or SpatialGridDataFrame
    ... arguments passed on to image.SpatialGridDataFrame
    attr integer or character, indicating the attribute variable to be plotted; default 1
    col color ramp to be used; default bpy.colors(100) for continuous, or RColorBrewer::brewer.pal(nlevels(x[[1]]),
         for factor variables
    breaks for continous attributes: values at which color breaks should take place
    zlim for continuous attributes: numeric of length 2, specifying the range of attribute values to be
         plotted; default to data range range(as.numeric(x[[attr]])[is.finite(x[[attr]])])
    axes logical: draw x and y axes? default FALSE
    xaxs character, default "i", see par
    yaxs character, default equal to xaxs, see par
    at numeric or NULL, values at which axis tics and labes should be drawn; default NULL (use
         pretty)
    border color, to be used for drawing grid lines; default NA (don't draw grid lines)
    axis.pos integer, 1-4; default 4, see axis
    add.axis logical: draw axis along scale? default TRUE
    what what to draw: "image", "scale", or "both"; default "both"
    scale.size size for the scale bar; use lcm to specify in absolute size, or a numeric value such as 1/6
         to specify relative size; default 1cm(2.8)
```

**scale.n** for categorical attributes: integer, indicating how many scale categories should fill a complete width; default 15

**scale.shrink** non-negative numeric indicating the amount to shrink the scale length, default 0 **scale.frac** for categorical attributes: numeric between 0 and 1, indicating the scale width, default

### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

SpatialGrid-class, which does not contain the attribute data, and SpatialPixelsDataFrame-class which holds possi grids

Plotting gridded data with sp: http://r-spatial.org/r/2016/03/08/plotting-spatial-grids.html

```
data(meuse.grid) # only the non-missing valued cells
coordinates(meuse.grid) = c("x", "y") # promote to SpatialPointsDataFrame
gridded(meuse.grid) <- TRUE # promote to SpatialPixelsDataFrame</pre>
x = as(meuse.grid, "SpatialGridDataFrame") # creates the full grid
x[["idist"]] = 1 - x[["dist"]] # assigns new attribute
image(x["idist"]) # note the single [ for attribute selection
# toy example:
df = data.frame(z = c(1:6,NA,8,9),
   xc = c(1,1,1,2,2,2,3,3,3),
   yc = c(rep(c(0, 1.5, 3),3)))
coordinates(df) = \sim xc + yc
gridded(df) = TRUE
df = as(df, "SpatialGridDataFrame") # to full grid
image(df["z"])
# draw labels to verify:
cc = coordinates(df)
z=df[["z"]]
zc=as.character(z)
zc[is.na(zc)]="NA"
text(cc[,1],cc[,2],zc)
# the following is weird, but illustrates the concept of row/col selection:
fullgrid(meuse.grid) = TRUE
image(meuse.grid)
image(meuse.grid[20:70, 10:70, "dist"], add = TRUE, col = bpy.colors())
# as.matrix, as.array
sgdim = c(3,4)
SG = SpatialGrid(GridTopology(rep(0,2), rep(10,2), sgdim))
SGDF = SpatialGridDataFrame(SG, data.frame(val = 1:12))
as.array(SGDF)
as.matrix(SGDF)
as(SGDF, "array")
```

SpatialLines

SpatialLines	create objects of class SpatialLines or SpatialLinesDataFrame

## **Description**

create objects of class SpatialLines or SpatialLinesDataFrame from lists of Lines objects and data.frames; extract list od Lines from a SpatialLines object

## Usage

```
SpatialLines(LinesList, proj4string = CRS(as.character(NA)))
SpatialLinesDataFrame(sl, data, match.ID = TRUE)
as.SpatialLines.SLDF(SLDF)
getSpatialLinesMidPoints(SL)
LineLength(cc, longlat = FALSE, sum = TRUE)
LinesLength(Ls, longlat = FALSE)
SpatialLinesLengths(SL, longlat)
```

## **Arguments**

LinesList proj4string	list with objects of class Lines-class Object of class "CRS"; holding a valid proj4 string
sl, SL	object of class SpatialLines-class
data	object of class data.frame; the number of rows in data should equal the number of Lines elements in sl
match.ID	logical: (default TRUE): match SpatialLines member Lines ID slot values with data.frame row names, and re-order the data frame rows if necessary; if character: indicates the column in data with Lines IDs to match
SLDF	SpatialLinesDataFrame object
Ls	Object of class Lines
СС	Object of class Line, or two-column matrix with points
longlat	if FALSE, Euclidean distance, if TRUE Great Circle distance in kilometers
sum	logical; if TRUE return scalar length of sum of segments in Line, if FALSE return vector with segment lengths $$

#### Value

SpatialLines returns object of class SpatialLines; SpatialLinesDataFrame returns object of class SpatialLinesDataFrame getSpatialLinesMidPoints returns an object of class SpatialPoints, each point containing the (weighted) mean of the lines elements; weighted in the sense that mean is called twice.

### See Also

SpatialLines-class

SpatialLines-class 67

SpatialLines-class a class for spatial lines

## **Description**

a class that holds spatial lines

### **Objects from the Class**

hold a list of Lines objects; each Lines object holds a list of Line (line) objects.

#### **Slots**

```
lines: Object of class "list"; list members are all of class Lines-class bbox: Object of class "matrix"; see Spatial-class proj4string: Object of class "CRS"; see CRS-class
```

#### **Extends**

Class "Spatial", directly.

#### Methods

## plot method arguments

The plot method for "SpatialLines" objects takes the following arguments:

```
x object of class SpatialLines
xlim default NULL; the x limits (x1, x2) of the plot
ylim default NULL; the y limits of the plot
col default 1; default plotting color
lwd default 1; line width
lty default 1; line type
add default FALSE; add to existing plot
axes default FALSE; a logical value indicating whether both axes should be drawn
```

```
lend default 0; line end style
ljoin default 0; line join style
lmitre default 10; line mitre limit
... passed through
setParUsrBB set the par "usr" bounding box, see note in Spatial-class
```

### Note

rbind calls the function SpatialLines, where it is checked that all IDs are unique. If rbind-ing SpatialLines without unique IDs, it is possible to set the argument makeUniqueIDs = TRUE, although it is preferred to change these explicitly with spChFIDs.

### Author(s)

Roger Bivand, Edzer Pebesma

### See Also

Line-class. Lines-class

### **Examples**

```
# from the sp vignette:
11 = cbind(c(1,2,3),c(3,2,2))
rownames(l1) = letters[1:3]
11a = cbind(l1[,1]+.05,l1[,2]+.05)
rownames(l1a) = letters[1:3]
12 = cbind(c(1,2,3),c(1,1.5,1))
rownames(l2) = letters[1:3]
S11 = Line(l1)
S11a = Line(l1a)
S12 = Line(l2)
S1 = Lines(list(S11, S11a), ID="a")
S2 = Lines(list(S12), ID="b")
S1 = SpatialLines(list(S1,S2))
summary(S1)
plot(S1, col = c("red", "blue"))
```

SpatialLinesDataFrame-class

a class for spatial lines with attributes

### **Description**

this class holds data consisting of (sets of lines), where each set of lines relates to an attribute row in a data.frame

### **Objects from the Class**

can be created by the function SpatialLinesDataFrame

#### **Slots**

```
data: Object of class data.frame containing the attribute table lines: Object of class "list"; see SpatialLines-class bbox: Object of class "matrix"; see Spatial-class proj4string: Object of class "CRS"; see CRS-class
```

### **Extends**

```
Class "SpatialLines", directly. Class "Spatial", by class "SpatialLines".
```

#### Methods

Methods defined with class "SpatialLinesDataFrame" in the signature:

```
[ signature(x = "SpatialLinesDataFrame"): subset rows or columns; in case of row subset-
ting, the line sets are also subsetted; NAs are not permitted in the row index

coordinates signature(obj = "SpatialLinesDataFrame"): retrieves a list with lists of coordinate matrices

show signature(object = "SpatialLinesDataFrame"): print method

plot signature(x = "SpatialLinesDataFrame"): plot points

lines signature(object = "SpatialLinesDataFrame"): add lines to plot

rbind signature(object = "SpatialLinesDataFrame"): rbind-like method
```

#### Note

rbind for SpatialLinesDataFrame is only possible for objects with unique IDs. If you want to rbind objects with duplicated IDs, seespChFIDs.

### Author(s)

```
Roger Bivand; Edzer Pebesma
```

### See Also

SpatialLines-class

70 SpatialMultiPoints

SpatialMultiPoints	create objects of class SpatialMultiPoints or SpatialMultiPoints- DataFrame
--------------------	--------------------------------------------------------------------------------

### **Description**

create objects of class SpatialMultiPoints-class or SpatialMultiPointsDataFrame-class from coordinates, and from coordinates and data.frames

## Usage

### **Arguments**

coords list with in each element a numeric matrix or data.frame with coordinates (each

row representing a point); in case of SpatialMultiPointsDataFrame an object of

class SpatialMultiPoints-class is also allowed

proj4string projection string of class CRS-class

bbox bounding box matrix, usually NULL and constructed from the data, but may be

passed through for coercion purposes if clearly needed

data object of class data. frame; the number of rows in data should equal the num-

ber of points in the coords object

match.ID logical or character; if missing, and coords and data both have row names,

and their order does not correspond, matching is done by these row names and a warning is issued; this warning can be suppressed by setting match.ID to TRUE. If TRUE AND coords has non-automatic rownames (i.e., coerced to a matrix by as.matrix, dimnames(coords)[[1]] is not NULL), AND data has row.names (i.e. is a data.frame), then the SpatialMultiPointsDataFrame object is formed by matching the row names of both components, leaving the order of the coordinates in tact. Checks are done to see whether both row names are sufficiently unique, and all data are matched. If FALSE, coordinates and data are simply "glued" together, ignoring row names. If character: indicates the column in data with coordinates IDs to use for matching records. See examples below.

### Value

SpatialMultiPoints returns an object of class SpatialMultiPoints; SpatialMultiPointsDataFrame returns an object of class SpatialMultiPointsDataFrame;

#### See Also

coordinates, SpatialMultiPoints-class, SpatialMultiPointsDataFrame-class

SpatialMultiPoints-class

71

## **Examples**

```
cl1 = cbind(rnorm(3, 10), rnorm(3, 10))
cl2 = cbind(rnorm(5, 10), rnorm(5, 0))
cl3 = cbind(rnorm(7, 0), rnorm(7, 10))

mp = SpatialMultiPoints(list(cl1, cl2, cl3))
mpx = rbind(mp, mp) # rbind method
plot(mp, col = 2, cex = 1, pch = 1:3)
mp
mp[1:2]

print(mp, asWKT=TRUE, digits=3)

mpdf = SpatialMultiPointsDataFrame(list(cl1, cl2, cl3), data.frame(a = 1:3))
mpdf
mpdfx = rbind(mpdf, mpdf) # rbind method

plot(mpdf, col = mpdf$a, cex = 1:3)
as(mpdf, "data.frame")
mpdf[1:2,]
```

SpatialMultiPoints-class

Class "SpatialMultiPoints"

### **Description**

Class for (irregularly spaced) MultiPoints

## **Objects from the Class**

Objects can be created by calls of the form SpatialPoints(x).

#### **Slots**

```
coords: Object of class "list", containing the coordinates of point sets (each list element is a matrix)bbox: Object of class "matrix", with bounding boxproj4string: Object of class "CRS", projection string
```

## **Extends**

```
Class "Spatial", directly.
```

#### Methods

```
[ signature(x = "SpatialMultiPoints"): subsets point sets
coerce signature(from = "SpatialPoints", to = "data.frame"): coerce to data.frame
coordinates signature(obj = "SpatialMultiPoints"): retrieves all the coordinates, as one single matrix
plot signature(x = "SpatialPoints", y = "missing"): plot points
summary signature(object = "SpatialPoints"): summarize object
points signature(x = "SpatialPoints"): add point symbols to plot
show signature(object = "SpatialPoints"): prints coordinates
rbind signature(object = "SpatialPoints"): rbind-like method
```

### plot method arguments

The plot method for "SpatialPoints" objects takes the following arguments:

```
x object of class SpatialPoints
```

**pch** default 3; either an integer specifying a symbol or a single character to be used as the default in plotting points

axes default FALSE; a logical value indicating whether both axes should be drawn

add default FALSE; add to existing plot

**xlim** default NULL; the x limits (x1, x2) of the plot

ylim default NULL; the y limits of the plot

... passed through

setParUsrBB default FALSE; set the par "usr" bounding box, see note in Spatial-class

**cex** default 1; numerical value giving the amount by which plotting text and symbols should be magnified relative to the default

col default 1; default plotting color

lwd default 1; line width

bg default 1; colour to be used for the background of the device region

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

## See Also

SpatialMultiPointsDataFrame-class SpatialPoints-class

## **Examples**

```
cl1 = cbind(rnorm(3, 10), rnorm(3, 10))
cl2 = cbind(rnorm(5, 10), rnorm(5, 0))
cl3 = cbind(rnorm(7, 0), rnorm(7, 10))

mp = SpatialMultiPoints(list(cl1, cl2, cl3))
plot(mp, col = 2, cex = 1, pch = 1:3)
mp
mp[1:2]

print(mp, asWKT=TRUE, digits=3)
```

 ${\tt Spatial MultiPoints Data Frame-class}$ 

 ${\it Class~"Spatial MultiPoints Data Frame"}$ 

## **Description**

Class for spatial attributes that correspond to point sets

## Usage

```
## S4 method for signature 'SpatialMultiPointsDataFrame'
x[i, j, ..., drop = TRUE]
  ## S4 method for signature 'SpatialMultiPointsDataFrame,data.frame'
coerce(from, to, strict=TRUE)
  ## S4 method for signature 'SpatialMultiPointsDataFrame'
coordinates(obj)
  ## S4 method for signature 'SpatialMultiPointsDataFrame'
show(object)
  ## S4 method for signature 'SpatialMultiPointsDataFrame'
points(x)
```

# Arguments

```
x,from,obj,object
SpatialMultiPointsDataFrame object
to class to which to coerce
strict see as
i row indices
j column indices
drop see Extract
... indices passed through
```

74 SpatialPixels

## **Slots**

```
data: Object of class data.frame containing the attribute data (may or may not contain the coordinates in its columns)
coords: Object of class "list"; the list with coordinates matrices; points are rows in the matrix, the list length equals the number of rows in the data slot
bbox: Object of class "matrix"; bounding box
proj4string: Object of class "CRS"; projection string
```

## **Extends**

Class "SpatialMultiPoints", directly. Class "Spatial", by class "SpatialMultiPoints".

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

coordinates, SpatialMultiPoints-class

# **Examples**

```
# create three sets of points:
cl1 = cbind(rnorm(3, 10), rnorm(3, 10))
cl2 = cbind(rnorm(5, 10), rnorm(5, 0))
cl3 = cbind(rnorm(7, 0), rnorm(7, 10))

mpdf = SpatialMultiPointsDataFrame(list(cl1, cl2, cl3), data.frame(a = 1:3))
mpdf

plot(mpdf, col = mpdf$a, cex = 1:3)
as(mpdf, "data.frame")
mpdf[1:2,]
```

SpatialPixels

define spatial grid

# Description

defines spatial grid by offset, cell size and dimensions

SpatialPixels 75

## Usage

```
GridTopology(cellcentre.offset, cellsize, cells.dim)
SpatialPixels(points, tolerance = sqrt(.Machine$double.eps),
proj4string = CRS(as.character(NA)), round = NULL, grid = NULL)
SpatialGrid(grid, proj4string = CRS(as.character(NA)))
coordinatevalues(obj)
points2grid(points, tolerance = sqrt(.Machine$double.eps), round=NULL)
getGridIndex(cc, grid, all.inside = TRUE)
getGridTopology(obj)
areaSpatialGrid(obj)
```

## Arguments

cellcentre.offset

numeric; vector with the smallest coordinates for each dimension

cellsize numeric; vector with the cell size in each dimension cells.dim integer; vector with number of cells in each dimension

points coordinates, object of class SpatialPoints-class

grid grid topology; object of class GridTopology-class; for calls to SpatialPixels,

a value of NULL implies that this will be derived from the point coordinates

tolerance precision, used to which extent points are exactly on a grid

round default NULL, otherwise a value passed to as the digits argument to round for

setting cell size

proj4string object of class CRS-class

obj object of class or deriving from SpatialGrid-class

cc numeric matrix with coordinates

all.inside logical; if TRUE and cc points fall outside the grid area, an error message is

generated; if FALSE, NA values are generated for such points

#### Value

GridTopology returns a value of class GridTopology-class; SpatialGrid returns an object of class SpatialGrid-class

coordinatevalues returns a list with the unique x-coordinates, the unique y-coordinate, etc. instead of the coordinates of all grid cells

SpatialGrid returns an object of class SpatialGrid-class.

points2grid returns the GridTopology-class from a set of points.

getGridIndex finds the index of a set of point coordinates in a given grid topology, and depending on all.inside setting, generates NA or an error message if points are outside the grid domain.

getGridTopology returns the slot of class GridTopology-class from obj.

areaSpatialGrid returns the spatial area of (the non-missing valued cells of) the grid. For objects of class SpatialGridDataFrame-class the area refers to cells where any (one or more) of the attribute columns are non-missing valued.

76 SpatialPixels

#### Note

SpatialGrid stores grid topology and may or may not store the coordinates of the actual points, which may form a subset of the full grid. To find out or change this, see fullgrid.

points2grid tries to figure out the grid topology from points. It succees only if points on a grid line have constant y column, and points on a grid column have constant x coordinate, etc. In other cases, use signif on the raw coordinate matrices to make sure this is the case.

## Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

SpatialGrid-class, SpatialGridDataFrame-class,

## **Examples**

```
x = GridTopology(c(0,0), c(1,1), c(5,4))
class(x)
summary(x)
coordinates(x)
coordinates(GridTopology(c(0,0), c(1,1), c(5,4)))
coordinatevalues(x)
data(meuse.grid)
coordinates(meuse.grid) <- c("x", "y")</pre>
points2grid(meuse.grid)
data(meuse.grid)
set.seed(1)
meuse.grid$x <- meuse.grid$x + rnorm(length(meuse.grid$x), 0, 0.002)</pre>
meuse.grid$y <- meuse.grid$y + rnorm(length(meuse.grid$y), 0, 0.002)</pre>
coordinates(meuse.grid) <- c("x", "y")</pre>
# points2grid(meuse.grid, tolerance=0.76, round=1)
data(meuse.grid)
a <- which(meuse.gridx == 180140)
b \leftarrow which(meuse.grid$x == 180180)
c \leftarrow which(meuse.grid$x == 179260)
d \leftarrow which(meuse.grid\$y == 332460)
e \leftarrow which(meuse.grid$y == 332420)
f \leftarrow which(meuse.grid$y == 330740)
meuse.grid <- meuse.grid[-c(a, b, c, d, e, f),]</pre>
coordinates(meuse.grid) <- c("x", "y")</pre>
points2grid(meuse.grid)
data(meuse.grid)
set.seed(1)
meuse.grid$x <- meuse.grid$x + rnorm(length(meuse.grid$x), 0, 0.002)</pre>
meuse.grid$y <- meuse.grid$y + rnorm(length(meuse.grid$y), 0, 0.002)</pre>
meuse.grid <- meuse.grid[-c(a, b, c, d, e, f),]</pre>
coordinates(meuse.grid) <- c("x", "y")</pre>
# EJP
```

SpatialPixels-class 77

```
# points2grid(meuse.grid, tolerance=0.69, round=1)
```

```
SpatialPixels-class Class "SpatialPixels"
```

## **Description**

class for defining a pixels, forming a possibly incomplete rectangular grid of arbitrary dimension

## **Objects from the Class**

```
Objects are created by using e.g.
SpatialPixels(points)
with points of class SpatialPoints-class
```

#### Slots

```
grid object of class GridTopology-class, defining the grid topology (offset, cellsize, dim) grid.index integer; index of points in full grid coords coordinates of points, or bbox of grid bbox: Object of class "matrix"; bounding box proj4string: Object of class "CRS"; projection
```

## **Extends**

Class "SpatialPoints" directly; Class "Spatial", by class "SpatialPoints".

## Methods

```
coordinates signature(x = "SpatialPixels"): calculates coordinates for each point on the
    grid; coordinates are not stored in objects of class SpatialGrid
summary signature(object = "SpatialPixels"): summarize object
plot signature(x = "SpatialPixels"): plots cell centers
"[" signature(x = "SpatialPixels"): select pixel cells; the argument drop=FALSE (default)
    does not recalculate grid topology for the selection, if drop=TRUE the grid topology is recomputed, and might change.

rbind signature(x = "SpatialPixels"): rbind-like method
```

## Author(s)

```
Edzer Pebesma, <edzer.pebesma@uni-muenster.de>
```

#### See Also

```
SpatialPixelsDataFrame-class, SpatialGrid-class
```

## **Examples**

```
data(meuse.grid)
pts = meuse.grid[c("x", "y")]
y = SpatialPixels(SpatialPoints(pts))
class(y)
y
summary(y)
plot(y) # plots grid
plot(y, grid = FALSE) # plots points
```

 ${\tt SpatialPixelsDataFrame}$ 

define spatial grid with attribute data

# Description

defines spatial grid by offset, cell size and dimensions

# Usage

```
SpatialPixelsDataFrame(points, data, tolerance = sqrt(.Machine$double.eps),
proj4string = CRS(as.character(NA)), round = NULL, grid = NULL)
SpatialGridDataFrame(grid, data, proj4string = CRS(as.character(NA)))
```

# Arguments

points	coordinates, either as numeric matrix or as object of class SpatialPoints-class
grid	grid topology; object of class $GridTopology$ -class; for calls to SpatialPixelsDataFrame a value of NULL implies that this will be derived from the point coordinates
data	data.frame; contains the attribute (actual grid) data
tolerance	precision up to which extent points should be exactly on a grid
round	default NULL, otherwise a value passed to as the digits argument to round for setting cell size
proj4string	object of class CRS-class in the first form only used when points does not inherit from Spatial-class

#### Value

SpatialPixelsDataFrame returns an object of class SpatialPixelsDataFrame-class; SpatialGridDataFrame returns an object of class SpatialGridDataFrame-class.

#### Note

SpatialPixels stores grid topology and coordinates of the actual points, which may be in the form of a subset (set of pixels) of a full grid. To find out or change this, see fullgrid and SpatialGrid-class.

## Author(s)

Edzer Pebesma

#### See Also

```
gridded, gridded<-, SpatialGrid, SpatialGrid-class
```

# **Examples**

```
data(meuse.grid)
m = SpatialPixelsDataFrame(points = meuse.grid[c("x", "y")], data = meuse.grid)
class(m)
summary(m)
```

```
\label{lem:class} Class \ \ "Spatial Pixels Data Frame"
```

# Description

Class for spatial attributes that have spatial locations on a regular grid.

# Objects from the Class

Objects can be created by calls of the form as(x, "SpatialPixelsDataFrame"), where x is of class SpatialPointsDataFrame-class, or by importing through rgdal. Ordered full grids are stored instead or unordered non-NA cells;

# Slots

```
bbox: Object of class "matrix"; bounding box
proj4string: Object of class "CRS"; projection
coords: see SpatialPoints; points slot
coords.nrs see SpatialPointsDataFrame
grid: see GridTopology-class; grid parameters
grid.index: integer; index of points in the list to points in the full (ordered) grid. x cycles fastest;
all coordinates increase from low to hight except y, which decreases from high to low
data: Object of class data.frame, containing the attribute data
```

# **Extends**

```
Class "SpatialPixels", directly. Class "Spatial", by class "SpatialPixels".
```

## Methods

## Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### See Also

SpatialPixels-class, which does not contain the attribute data

## **Examples**

```
data(meuse.grid) # only the non-missing valued cells
coordinates(meuse.grid) = c("x", "y") # promote to SpatialPointsDataFrame
gridded(meuse.grid) <- TRUE # promote to SpatialPixelsDataFrame</pre>
meuse.grid[["idist"]] = 1 - meuse.grid[["dist"]] # assigns new attribute
image(meuse.grid["idist"]) # note the single [
# toy example:
df = data.frame(z = c(1:6,NA,8,9),
   xc = c(1,1,1,2,2,2,3,3,3),
    yc = c(rep(c(0, 1.5, 3), 3)))
coordinates(df) = \sim xc + yc
gridded(df) = TRUE
image(df["z"])
# draw labels to verify:
cc = coordinates(df)
z=df[["z"]]
zc=as.character(z)
zc[is.na(zc)]="NA"
text(cc[,1],cc[,2],zc)
```

SpatialPoints 81

SpatialPoints create objects of class SpatialPoints or SpatialPointsDataFrame
-------------------------------------------------------------------------------

# Description

create objects of class SpatialPoints-class or SpatialPointsDataFrame-class from coordinates, and from coordinates and data.frames

# Usage

# **Arguments**

coords	numeric matrix or data.frame with coordinates (each row is a point); in case of SpatialPointsDataFrame an object of class SpatialPoints-class is also allowed
proj4string	projection string of class CRS-class
bbox	bounding box matrix, usually NULL and constructed from the data, but may be passed through for coercion purposes if clearly needed
data	object of class data.frame; the number of rows in data should equal the number of points in the coords object
coords.nrs	numeric; if present, records the column positions where in data the coordinates were taken from (used by coordinates<-)
match.ID	logical or character; if missing, and coords and data both have row names, and their order does not correspond, matching is done by these row names and a warning is issued; this warning can be suppressed by setting match.ID to TRUE. If TRUE AND coords has non-automatic rownames (i.e., coerced to a matrix by as.matrix, dimnames(coords)[[1]] is not NULL), AND data has row.names (i.e. is a data.frame), then the SpatialPointsDataFrame object is formed by matching the row names of both components, leaving the order of the coordinates in tact. Checks are done to see whether both row names are sufficiently unique, and all data are matched. If FALSE, coordinates and data are simply "glued" together, ignoring row names. If character: indicates the column in data with coordinates IDs to use for matching records. See examples below.

# Value

 ${\tt SpatialPoints\ returns\ an\ object\ of\ class\ SpatialPoints};\ {\tt SpatialPointsDataFrame\ returns\ an\ object\ of\ class\ SpatialPointsDataFrame};$ 

82 SpatialPoints-class

## See Also

coordinates, SpatialPoints-class, SpatialPointsDataFrame-class

## **Examples**

```
set.seed(1331)
pts = cbind(1:5, 1:5)
dimnames(pts)[[1]] = letters[1:5]
df = data.frame(a = 1:5)
row.names(df) = letters[5:1]

library(sp)
options(warn=1) # show warnings where they occur
SpatialPointsDataFrame(pts, df) # warn
SpatialPointsDataFrame(pts, df, match.ID = TRUE) # don't warn
SpatialPointsDataFrame(pts, df, match.ID = FALSE) # don't warn
df$m = letters[5:1]
SpatialPointsDataFrame(pts, df, match.ID = "m") # don't warn
dimnames(pts)[[1]] = letters[5:1]
SpatialPointsDataFrame(pts, df) # don't warn: ID matching doesn't reorder
```

SpatialPoints-class Class "SpatialPoints"

## **Description**

Class for (irregularly spaced) points

# **Objects from the Class**

Objects can be created by calls of the form SpatialPoints(x).

## **Slots**

```
coords: Object of class "matrix", containing the coordinates (each row is a point) bbox: Object of class "matrix", with bounding box proj4string: Object of class "CRS", projection string
```

# **Extends**

```
Class "Spatial", directly.
```

SpatialPoints-class 83

## Methods

```
[ signature(x = "SpatialPoints"): subsets the points; only rows (points) can be subsetted
coerce signature(from = "SpatialPoints", to = "data.frame"): retrieves the data part
coerce signature(from = "SpatialPoints", to = "SpatialPixels"): equivalent to assign-
    ing gridded TRUE for a copy of the object
coerce signature(from = "SpatialPointsDataFrame", to = "SpatialPixelsDataFrame"):
     equivalent to assigning gridded TRUE for a copy of the object
coerce signature(from = "data.frame", to = "SpatialPoints"): sets coordinates, which
     may be in a data frame
coerce signature(from = "matrix", to = "SpatialPoints"): set coordinates, which may
    be in a matrix
coordinates signature(obj = "SpatialPoints"): retrieves the coordinates, as matrix
plot signature(x = "SpatialPoints", y = "missing"): plot points
summary signature(object = "SpatialPoints"): summarize object
points signature(x = "SpatialPoints"): add point symbols to plot
show signature(object = "SpatialPoints"): prints coordinates
rbind signature(object = "SpatialPoints"): rbind-like method
```

#### plot method arguments

The plot method for "SpatialPoints" objects takes the following arguments:

```
x object of class SpatialPoints
```

**pch** default 3; either an integer specifying a symbol or a single character to be used as the default in plotting points

axes default FALSE; a logical value indicating whether both axes should be drawn

add default FALSE; add to existing plot

**xlim** default NULL; the x limits (x1, x2) of the plot

ylim default NULL; the y limits of the plot

... passed through

setParUsrBB default FALSE; set the par "usr" bounding box, see note in Spatial-class

**cex** default 1; numerical value giving the amount by which plotting text and symbols should be magnified relative to the default

col default 1; default plotting color

lwd default 1; line width

**bg** default 1; colour to be used for the background of the device region

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

## See Also

SpatialPointsDataFrame-class

## **Examples**

```
x = c(1,2,3,4,5)
y = c(3,2,5,1,4)
S <- SpatialPoints(cbind(x,y))
S <- SpatialPoints(list(x,y))
S <- SpatialPoints(data.frame(x,y))
S
plot(S)</pre>
```

SpatialPointsDataFrame-class

Class "SpatialPointsDataFrame"

## **Description**

Class for spatial attributes that have spatial point locations

## Usage

```
## S4 method for signature 'SpatialPointsDataFrame'
x[i, j, ..., drop = TRUE]
  ## S4 method for signature 'SpatialPointsDataFrame, SpatialPoints'
coerce(from, to, strict=TRUE)
  ## S4 method for signature 'SpatialPointsDataFrame, data.frame'
coerce(from, to, strict=TRUE)
  ## S4 method for signature 'SpatialPointsDataFrame'
coordinates(obj)
  ## S4 method for signature 'SpatialPointsDataFrame'
show(object)
  ## S4 method for signature 'SpatialPointsDataFrame'
points(x)
  ## S3 method for class 'SpatialPointsDataFrame'
rbind(...)
```

## **Arguments**

```
x,from,obj,object
SpatialPointsDataFrame object
to class to which to coerce
strict see as
i row indices
j column indices
drop see Extract
... indices passed through
```

## **Objects from the Class**

Objects can be created by calls of the form coordinates(x) = c("x", "y") . or of the form coordinates(x) = xy; see coordinates.

#### Slots

data: Object of class data.frame containing the attribute data (may or may not contain the coordinates in its columns)

coords: Object of class "matrix"; the coordinates matrix (points are rows in the matrix)

coords.nrs Object of class logical; if TRUE, when the object was created the coordinates were retrieved from the data.frame, and hence stripped from it; after coercion to data.frame, e.g. by as.data.frame(x), coordinates will again be added (as first few columns) to the data.frame

```
bbox: Object of class "matrix"; bounding box proj4string: Object of class "CRS"; projection string
```

#### **Extends**

```
Class "SpatialPoints", directly. Class "Spatial", by class "SpatialPoints".
```

## Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

## See Also

```
coordinates, SpatialPoints-class
```

# **Examples**

```
data(meuse)
xy = meuse[c("x", "y")] # retrieve coordinates as data.frame
class(meuse)
data(meuse) # reload data.frame
coordinates(meuse) = c("x", "y") # specify column names
class(meuse)
data(meuse) # reload data.frame
coordinates(meuse) = c(1, 2) # specify column names
class(meuse)
data(meuse) # reload data.frame
coordinates(meuse) = ~x+y # formula
class(meuse)
data(meuse) # reload data.frame
coordinates(meuse) = xy # as data frame
class(meuse)
data(meuse) # reload data.frame
coordinates(meuse) = as.matrix(xy) # as matrix
meuse$log.zn = log(meuse$zinc)
class(meuse)
dim(meuse)
```

SpatialPolygons

SpatialPolygons	$create\ objects\ of\ class\ Spatial Polygons\ or\ Spatial Polygons Data Frame$

# Description

create objects of class SpatialPolygons or SpatialPolygonsDataFrame from lists of Polygons objects and data.frames

# Usage

```
Polygon(coords, hole=as.logical(NA))
Polygons(srl, ID)
SpatialPolygons(Srl, p0, proj4string=CRS(as.character(NA)))
SpatialPolygonsDataFrame(Sr, data, match.ID = TRUE)
getSpatialPolygonsLabelPoints(SP)
```

# **Arguments**

coords	2-column numeric matrix with coordinates; first point (row) should equal last coordinates (row); if the hole argument is not given, the status of the polygon as a hole or an island will be taken from the ring direction, with clockwise meaning island, and counter-clockwise meaning hole
hole	logical value for setting polygon as hole or not; if the hole argument is not given, the status of the polygon as a hole or an island will be taken from the ring direction, with clockwise meaning island, and counter-clockwise meaning hole
proj4string	projection string of class CRS-class
srl	list with Polygon-class objects
ID	character vector of length one with identifier
Srl	list with objects of class Polygons-class
рО	integer vector; plotting order; if missing in reverse order of Polygons area
Sr	object of class SpatialPolygons-class
data	object of class data. frame; the number of rows in data should equal the number of $\underline{\text{Polygons-class}}$ objects in Sr
match.ID	logical: (default TRUE): match SpatialPolygons member Polygons ID slot values with data frame row names, and re-order the data frame rows if necessary. If character: indicates the column in data with Polygons IDs to match
SP	object of class SpatialPolygons-class

SpatialPolygons-class 87

#### **Details**

In Polygon, if the hole argument is not given, the status of the polygon as a hole or an island will be taken from the ring direction, with clockwise meaning island, and counter-clockwise meaning hole. In Polygons, if all of the member Polygon objects are holes, the largest by area will be converted to island status. Until 2010-04-17, version 0.9-61, the area of this converted object was erroneously left at its hole value of zero. Thanks to Patrick Giraudoux for spotting the bug.

The class definitions used for polygons in **sp** do not accord with those of the simple features specification of the Open Geospatial Consortium. The **rgeos** package, an interface to Geometry Engine – Open Source (GEOS), uses this specification, in which each hole (interior ring) must be associated with its containing exterior ring. In order to avoid introducing incompatible changes into the class definition of Polygons objects, a comment has been added as a single character string to each such object. Here we can trust the data source to assign the hole status correctly, and use the simple function **createSPComment** to add such comments to each Polygons member of the polygons slot of this SpatialPolygons object. Exterior rings are coded zero, while interior rings are coded with the 1-based index of the exterior ring to which they belong. SpatialPolygons objects created by reading using **read0GR** from **rgdal** have the comments set on input, as OGR also uses SFS.

#### Value

Polygon returns an object of class Polygon; Polygons returns an object of class Polygons; SpatialPolygons returns object of class SpatialPolygons; SpatialPolygonsDataFrame returns object of class SpatialPolygonsDataFrame getSpatialPolygonsLabelPoints returns an object of class SpatialPoints with label points.

## See Also

SpatialPolygons-class, SpatialPolygonsDataFrame-class

```
SpatialPolygons-class Class "SpatialPolygons"
```

## **Description**

class to hold polygon topology (without attributes)

## **Objects from the Class**

Objects can be created by calls to the function SpatialPolygons

#### **Slots**

```
polygons: Object of class "list"; list elements are all of class Polygons-class
plotOrder: Object of class "integer"; integer array giving the order in which objects should be
    plotted
bbox: Object of class "matrix"; see Spatial-class
proj4string: Object of class "CRS"; see CRS-class
```

#### **Extends**

```
Class "Spatial", directly.
```

#### Methods

Methods defined with class "SpatialPolygons" in the signature:

```
[ signature(obj = "SpatialPolygons"): select subset of (sets of) polygons; NAs are not per-
mitted in the row index

plot signature(x = "SpatialPolygons", y = "missing"): plot polygons in SpatialPolygons
    object

summary signature(object = "SpatialPolygons"): summarize object

rbind signature(object = "SpatialPolygons"): rbind-like method
```

## plot method arguments

The plot method for spatial polygons takes the following arguments:

```
x a SpatialPolygons object
col a vector of colour values
border default par("fg"); the colour to draw the border
add default FALSE; if TRUE, add to existing plot
xlim, ylim default NULL; ranges for the plotted 'x' and 'y' values
xpd default NULL; controls clipping, see par
density default NULL; the density of shading lines, in lines per inch, see polygon
angle default 45; the slope of shading lines, given as an angle in degrees (counter-clockwise), see
     polygon
pbg default NULL, set to par ("bg") by default "transparent"; the colour to paint holes
axes default FALSE; draw axes
lty default par("lty"); border line type
... other arguments passed through
setParUsrBB default FALSE; see Spatial-class for further details
usePolypath default NULL to set from option value; use polypath for hole-handling in plot
rule default NULL to set from option value; character value specifying the path fill mode, see
     polypath
```

The options for usePolypath and rule may be retrieved with get\_Polypath (default TRUE on package load) and get\_PolypathRule (default "winding" on package load), and set with set\_Polypath and set\_PolypathRule

The class definitions used for polygons in **sp** do not accord with those of the simple features specification of the Open Geospatial Consortium. The **rgeos** package, an interface to Geometry Engine – Open Source (GEOS), uses this specification, in which each hole (interior ring) must be associated with its containing exterior ring. In order to avoid introducing incompatible changes into the class definition of Polygons objects, a comment has been added as a single character string to each such

object. Here we can trust the data source to assign the hole status correctly, and use the simple function createSPComment to add such comments to each Polygons member of the polygons slot of this SpatialPolygons object. Exterior rings are coded zero, while interior rings are coded with the 1-based index of the exterior ring to which they belong. SpatialPolygons objects created by reading using readOGR from rgdal have the comments set on input, as OGR also uses SFS.

#### Note

rbind calls the function SpatialPolygons, where it is checked that all IDs are unique. If rbind-ing SpatialPolygons without unique IDs, it is possible to set the argument makeUniqueIDs = TRUE, although it is preferred to change these explicitly with spChFIDs.

## Author(s)

Roger Bivand

#### See Also

SpatialPolygons

## **Examples**

```
# simple example, from vignette("sp"):
Sr1 = Polygon(cbind(c(2,4,4,1,2),c(2,3,5,4,2)))
Sr2 = Polygon(cbind(c(5,4,2,5),c(2,3,2,2)))
Sr3 = Polygon(cbind(c(4,4,5,10,4),c(5,3,2,5,5)))
Sr4 = Polygon(cbind(c(5,6,6,5,5),c(4,4,3,3,4)), hole = TRUE)

Srs1 = Polygons(list(Sr1), "s1")
Srs2 = Polygons(list(Sr2), "s2")
Srs3 = Polygons(list(Sr3, Sr4), "s3/4")
SpP = SpatialPolygons(list(Srs1,Srs2,Srs3), 1:3)
plot(SpP, col = 1:3, pbg="white")
grd <- GridTopology(c(1,1), c(1,1), c(10,10))
polys <- as(grd, "SpatialPolygons")
plot(polys)
text(coordinates(polys), labels=row.names(polys))</pre>
```

SpatialPolygonsDataFrame-class

Class "SpatialPolygonsDataFrame"

# **Description**

class to hold polygons with attributes

#### **Objects from the Class**

Objects can be created by calls to the function SpatialPolygonsDataFrame

#### Slots

```
data: Object of class "data.frame"; attribute table polygons: Object of class "list"; see SpatialPolygons-class plotOrder: Object of class "integer"; see SpatialPolygons-class bbox: Object of class "matrix"; see Spatial-class proj4string: Object of class "CRS"; see CRS-class
```

#### **Extends**

Class "SpatialPolygons", directly. Class "Spatial", by class "SpatialPolygons".

#### Methods

Methods defined with class "SpatialPolygonsDataFrame" in the signature:

```
[ signature(x = "SpatialPolygonsDataFrame"): select subset of (sets of) polygons; NAs are not permitted in the row index
```

rbind signature(object = "SpatialPolygonsDataFrame"): rbind-like method, see notes below

#### Note

SpatialPolygonsDataFrame with default ID matching checks the data frame row names against the Polygons ID slots. They must then agree with each other, and be unique (no Polygons objects can share IDs); the data frame rows will be re-ordered if needed to match the Polygons IDs..

If you want to rbind objects with duplicated IDs, seespChFIDs.

#### Author(s)

Roger Bivand

## See Also

SpatialPolygons-class

# **Examples**

```
# simple example, from scratch:
Sr1 = Polygon(cbind(c(2,4,4,1,2),c(2,3,5,4,2)))
Sr2 = Polygon(cbind(c(5,4,2,5),c(2,3,2,2)))
Sr3 = Polygon(cbind(c(4,4,5,10,4),c(5,3,2,5,5)))
Sr4 = Polygon(cbind(c(5,6,6,5,5),c(4,4,3,3,4)), hole = TRUE)

Srs1 = Polygons(list(Sr1), "s1")
Srs2 = Polygons(list(Sr2), "s2")
Srs3 = Polygons(list(Sr3, Sr4), "s3/4")
SpP = SpatialPolygons(list(Srs1,Srs2,Srs3), 1:3)
plot(SpP, col = 1:3, pbg="white")
```

spChFIDs-methods 91

```
grd <- GridTopology(c(1,1), c(1,1), c(10,10))
polys <- as(grd, "SpatialPolygons")
centroids <- coordinates(polys)
x <- centroids[,1]
y <- centroids[,2]
z <- 1.4 + 0.1*x + 0.2*y + 0.002*x*x
ex_1.7 <- SpatialPolygonsDataFrame(polys,
    data=data.frame(x=x, y=y, z=z, row.names=row.names(polys)))
brks <- quantile(z, seq(0,1,1/7))
cols <- grey((length(brks):2)/length(brks))
dens <- (2:length(brks))*3
plot(ex_1.7, col=cols[findInterval(z, brks, all.inside=TRUE)])
plot(ex_1.7, density=dens[findInterval(z, brks, all.inside=TRUE)])</pre>
```

spChFIDs-methods

change feature IDs in spatial objects

## Description

When the feature IDs need to be changed in SpatialLines\* or SpatialPolygons\* objects, these methods may be used. The new IDs should be a character vector of unique IDs of the correct length.

# Methods

```
obj = "SpatialLines", x = "character" replace IDs in a SpatialLines object
```

**obj = "SpatialLinesDataFrame"**, **x = "character"** replace IDs in a SpatialLinesDataFrame object

obj = "SpatialPolygons", x = "character" replace IDs in a SpatialPolygons object

**obj = "SpatialPolygonsDataFrame"**, **x = "character"** replace IDs in a SpatialPolygonsDataFrame object

## Note

It is usually sensible to keep a copy of the original feature IDs in the object, but this should be done by the user.

## Author(s)

Roger Bivand

## See Also

spCbind-methods, spRbind-methods

92 spDistsN1

## **Examples**

```
## Not run:
require(maptools)
xx <- readShapePoly(system.file("shapes/sids.shp", package="maptools")[1],
    IDvar="FIPSNO", proj4string=CRS("+proj=longlat +ellps=clrk66"))
row.names(as(xx, "data.frame"))
xx1 <- spChFIDs(xx, as.character(xx$CNTY_ID))
row.names(as(xx1, "data.frame"))
## End(Not run)</pre>
```

spDistsN1

Euclidean or Great Circle distance between points

# Description

The function returns a vector of distances between a matrix of 2D points, first column longitude, second column latitude, and a single 2D point, using Euclidean or Great Circle distance (WGS84 ellipsoid) methods.

## Usage

```
spDistsN1(pts, pt, longlat = FALSE)
spDists(x, y = x, longlat = FALSE, segments = FALSE, diagonal = FALSE)
```

# Arguments

pts	A matrix of 2D points, first column x/longitude, second column y/latitude, or a SpatialPoints or SpatialPointsDataFrame object
pt	A single 2D point, first value x/longitude, second value y/latitude, or a Spatial-Points or SpatialPointsDataFrame object with one point only
x	A matrix of n-D points with row denoting points, first column x/longitude, second column y/latitude, or a Spatial object that has a coordinates method
У	A matrix of n-D points with row denoting points, first column x/longitude, second column y/latitude, or a Spatial object that has a coordinates method
longlat	logical; if FALSE, Euclidean distance, if TRUE Great Circle (WGS84 ellipsoid) distance; if x is a Spatial object, longlat should not be specified but will be derived from is.projected(x)
segments	logical; if TRUE, y must be missing; the vector of distances between consecutive points in x is returned.
diagonal	logical; if TRUE, y must be given and have the same number of points as $x$ ; the vector with distances between points with identical index is returned.

spDistsN1 93

## Value

spDistsN1 returns a numeric vector of distances in the metric of the points if longlat=FALSE, or in kilometers if longlat=TRUE.

spDists returns a full matrix of distances in the metric of the points if longlat=FALSE, or in kilometers if longlat=TRUE; it uses spDistsN1 in case points are two-dimensional. In case of spDists(x,x), it will compute all n x n distances, not the sufficient n x (n-1).

#### Note

The function can also be used to find a local kilometer equivalent to a plot scaled in decimal degrees in order to draw a scale bar.

#### Author(s)

Roger Bivand, Edzer Pebesma

#### References

http://www.abecedarical.com/javascript/script\_greatcircle.html

#### See Also

```
is.projected
```

## **Examples**

```
11 <- matrix(c(5, 6, 60, 60), ncol=2)</pre>
km <- spDistsN1(ll, ll[1,], longlat=TRUE)</pre>
zapsmall(km)
utm32 <- matrix(c(276.9799, 332.7052, 6658.1572, 6655.2055), ncol=2)
spDistsN1(utm32, utm32[1,])
dg <- spDistsN1(ll, ll[1,])</pre>
dg
dg[2]/km[2]
data(meuse)
coordinates(meuse) <- c("x", "y")</pre>
res <- spDistsN1(meuse, meuse[1,])</pre>
summary(res)
p1 = SpatialPoints(cbind(1:3, 1:3))
spDists(p1)
spDists(p1, p1)
spDists(p1, p1, diagonal = TRUE)
try(spDists(p1, p1, segments = TRUE))
spDists(p1, segments = TRUE)
p2 = SpatialPoints(cbind(5:2, 2:5))
spDists(p1, p2)
try(spDists(p1, p2, diagonal = TRUE)) # fails
try(spDists(p1, p2, segments = TRUE)) # fails
# longlat points:
```

```
proj4string(p1) = "+proj=longlat +ellps=WGS84"
proj4string(p2) = "+proj=longlat +ellps=WGS84"
is.projected(p1)
is.projected(p2)
spDists(p1)
spDists(p1, p1)
spDists(p1, p1, diagonal = TRUE)
spDists(p1, p2)
try(spDists(p1, p2, diagonal = TRUE)) # fails
spDists(p1, p2[1:length(p1),], diagonal = TRUE)
spDists(p1, segments = TRUE)
spDists(p1[0],p2[0],diagonal=TRUE)
spDists(p1[0])
p1 = SpatialPoints(cbind(1:3, 1:3, 1:3))
spDists(p1)
spDists(p1, p1)
try(spDists(p1, p1, diagonal = TRUE))
try(spDists(p1, p1, segments = TRUE))
try(spDists(p1, segments = TRUE))
p2 = SpatialPoints(cbind(5:2, 2:5, 3:6))
spDists(p1, p2)
try(spDists(p1, p2, diagonal = TRUE)) # fails
try(spDists(p1, p2, segments = TRUE)) # fails
```

spplot

Plot methods for spatial data with attributes

## **Description**

Lattice (trellis) plot methods for spatial data with attributes

## Usage

```
spplot(obj, ...)
spplot.grid(obj, zcol = names(obj), ..., names.attr, scales = list(draw = FALSE),
    xlab = NULL, ylab = NULL, aspect = mapasp(obj,xlim,ylim),
    panel = panel.gridplot, sp.layout = NULL, formula, xlim = bbox(obj)[1, ],
    ylim = bbox(obj)[2, ], checkEmptyRC = TRUE, col.regions = get_col_regions())
spplot.polygons(obj, zcol = names(obj), ..., names.attr, scales = list(draw = FALSE),
    xlab = NULL, ylab = NULL, aspect = mapasp(obj,xlim,ylim),
    panel = panel.polygonsplot, sp.layout = NULL, formula, xlim = bbox(obj)[1, ],
    ylim = bbox(obj)[2, ], col.regions = get_col_regions())
spplot.points(obj, zcol = names(obj), ..., names.attr, scales = list(draw = FALSE),
    xlab = NULL, ylab = NULL, aspect = mapasp(obj,xlim,ylim),
    panel = panel.pointsplot, sp.layout = NULL, identify = FALSE, formula,
    xlim = bbexpand(bbox(obj)[1, ], 0.04), ylim = bbexpand(bbox(obj)[2, ], 0.04),
    edge.col = "transparent", colorkey = FALSE, col.regions = get_col_regions())
```

```
mapLegendGrob(obj, widths = unit(1, "cm"), heights = unit(1, "cm"),
fill = "black", just = "right")
sp.theme(set = FALSE, regions = list(col = bpy.colors(100)), ...)
layout.north.arrow(type = 1)
layout.scale.bar(height = 0.05)
spplot.locator(n = 512, type = "n", ...)
set_col_regions(value)
get_col_regions()
```

# Arguments

obj	object of class extending Spatial-class
zcol	character; attribute name(s) or column number(s) in attribute table
names.attr	names to use in panel, if different from zcol names
scales	scales argument to be passed to Lattice plots; use list(draw = TRUE) to draw axes scales; see xyplot for full options
	other arguments passed to levelplot (grids, polygons) or xyplot (points)
xlab	label for x-axis
ylab	label for y-axis
aspect	aspect ratio for spatial axes; defaults to "iso" (one unit on the x-axis equals one unit on the y-axis) but may be set to more suitable values if the data are e.g. if coordinates are latitude/longitude
panel	depending on the class of obj, panel.polygonsplot (for polygons or lines), panel.gridplot (grids) or panel.pointsplot (points) is used; for further control custom panel functions can be supplied that call one of these panel functions, but do read below how the argument sp.layout may help
sp.layout	NULL or list; see notes below
identify	if not FALSE, identify plotted objects (currently only working for points plots). Labels for identification are the row.names of the attribute table row.names(as.data.frame(obj)). If TRUE, identify on panel (1,1); for identifying on panel i, j, pass the value c(i,j)
formula	optional; may be useful to plot a transformed value. Defaults to $z^x+y$ for single and $z^x+y$ name for multiple attributes; use e.g. $\exp(x)^x+y$ name to plot the exponent of the z-variable
xlim	numeric; x-axis limits
ylim	numeric; y-axis limits
edge.col	color of symbol edge
colorkey	if FALSE, use symbol key; if TRUE, use continuous, levelplot-like colorkey; if list, follow syntax of argument colorkey in levelplot (see below for an example)
widths	width of grob
heights	heights of grob
fill	fill color of grob
just	grob placement justification

set logical; if TRUE, trellis.par.set is called, else a list is returned that can be passed

to trellis.par.set()

regions color ramp for the theme

height height of scale bar; width is 1.0

n see locator type see locator

checkEmptyRC logical; if TRUE, a check is done to see if empty rows or columns are present,

and need to be taken care of. Setting to FALSE may improve speed.

col.regions vector with fill colours; in case the variable to be plotted is a factor, this vector

should have length equal to the number of factor levels

value vector with color values, default for col. regions

#### Value

spplot returns a lattice plot of class "trellis", if you fail to "see" it, explicitly call print(spplot(...)). If identify is TRUE, the plot is plotted and the return value is a vector with row names of the selected points.

spplot.locator returns a matrix with identified point locations; use trellis.focus first to focus on a given panel.

get\_col\_regions returns the default value for col.regions

#### Methods

obj = "SpatialPixelsDataFrame" see spplot

obj = "SpatialGridDataFrame" see spplot

obj = "SpatialPolygonsDataFrame" see spplot

obj = "SpatialLinesDataFrame" see spplot

obj = "SpatialPointsDataFrame" see spplot

## Note

Missing values in the attributes are (currently) not allowed.

spplot.grid, spplot.polygons and spplot.points are S4 methods for spplot; see spplot-methods.

Useful arguments that can be passed as . . . are:

layout integer; for the layout of panels (cols,rows)

pretty logical; choose colour breaks at pretty numbers?

at specify at which values colours change

as.table logical; start drawing panels upper-left instead of lower-left

page to add marks to each plotted page

for useful values see the appropriate documentation of xyplot (in case of points), and levelplot (otherwise).

If obj is of SpatialPointsDataFrame, the following options are useful to pass:

key.space character: "bottom", "right", "left" or "right" to denote key location, or list: see argument key in the help for xyplot what the options are

legendEntries character; array with key legend (text) entries; suitable defaults obtained from data cuts number of cuts, or, for objects of class SpatialPointsDataFrame only, the actual cuts to use

do.log logical; if TRUE use log-linear scale to divide range in equal cuts, else use a linear scale if cuts is only number of cuts

pch integer; plotting character to use; defaults to 16 if fill is TRUE, else 1

cex numeric; character expansion, proportional to default value of 1

fill logical; use filled circles?

layout.north.arrow and layout.scale.bar can be used to set a north arrow or scale bar.

The sp.layout argument is either a single layout item, or a list with one or more layout items. A layout item is one of

- a list with one or more Spatial\* objects, along with style arguments like col, lty, pch, fill etc.
- a list with its first argument the layout function or the name of the layout function to be called: sp.points for SpatialPoints, sp.polygons for SpatialPolygons object, sp.lines for a SpatialLines object, and sp.text for text to place. The second argument contains the object (or text) to be plotted; remaining arguments are passed to the corresponding panel.\* functions.

The order of items in sp.layout matters; objects are drawn in the order they appear. With respect to obj, default plot order and precedence of sp.layout items is as follows: for points and lines, sp.layout items are drawn over (after) obj; for grids and polygons, sp.layout items are drawn behind (before) obj. Transparency may further help making multiple things visible. Adding a first argument to a layout item overrides its default plotting order with respect to obj:

Special control elements of sp. layout items:

first logical; should the layout item be drawn before the obj (TRUE), or after (FALSE)? This overrides the default order (points and lines in front, polygons and grids behind).

which integer; controls to which panel a layout item should be added. If which is present in the main, top-level list it applies to all layout items; in sub-lists with layout items it denotes the (set of) panel(s) in which the layout item should be drawn. Without a which item, layout items are drawn in each panel.

sp. theme returns a lattice theme; use, after loading package lattice, the command trellis.par.set(sp.theme()) after a device is opened or changed to make this work. Currently, this only sets the colors to bpy.colors.

If the attributes to be plotted are of type factor, spplot tries to create a legend that reflects this. In this case, the color ramp passed needs to be of the same length as the number of factor levels. The factor levels are derived from the first map; subsequent factors with different factor levels result in an error.

#### Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

## References

```
http://rspatial.r-forge.r-project.org/gallery/
```

#### See Also

xyplot, levelplot, panel.identify to identify objects

# **Examples**

```
library(lattice)
trellis.par.set(sp.theme()) # sets bpy.colors() ramp
demo(meuse, ask = FALSE, echo = FALSE)
12 = list("SpatialPolygonsRescale", layout.north.arrow(), offset = c(181300,329800),
scale = 400)
13 = list("SpatialPolygonsRescale", layout.scale.bar(), offset = c(180500,329800),
scale = 500, fill=c("transparent", "black"))
14 = list("sp.text", c(180500,329900), "0")
15 = list("sp.text", c(181000,329900), "500 m")
spplot(meuse, c("ffreq"), sp.layout=list(12,13,14,15), col.regions= "black",
pch=c(1,2,3), key.space=list(x=0.1,y=.95,corner=c(0,1)))
spplot(meuse, c("zinc", "lead"), sp.layout=list(12,13,14,15, which = 2),
key.space=list(x=0.1,y=.95,corner=c(0,1)))
# plotting factors:
meuse$f = factor(sample(letters[6:10], 155, replace=TRUE),levels=letters[1:10])
meuse$g = factor(sample(letters[1:5], 155, replace=TRUE),levels=letters[1:10])
spplot(meuse, c("f", "g"), col.regions=bpy.colors(10))
if (require(RColorBrewer)) {
spplot(meuse, c("ffreq"), sp.layout=list(12,13,14,15),
col.regions=brewer.pal(3, "Set1"))
meuse.grid$g = factor(sample(letters[1:5], 3103, replace=TRUE),
levels=letters[1:10])
meuse.grid$f = factor(sample(letters[6:10], 3103, replace=TRUE),
levels=letters[1:10])
spplot(meuse.grid, c("f","g"), col.regions=bpy.colors(10))
# example modifying colorkey for points:
spplot(meuse["dist"], colorkey = list(
right = list( # see ?levelplot in package trellis, argument colorkey:
fun = draw.colorkey,
args = list(
key = list(
at = seq(0, 1, .1), # colour breaks
col = bpy.colors(11), # colours
labels = list(
at = c(0, .2, .4, .6, .8, 1),
labels = c("0x", "20x", "40x", "60x", "80x", "100x")
)
)
```

spsample 99

```
)
)
))
16 = list(meuse.grid["dist"], col = grey(seq(.5,.9,length.out=10)))
spplot(meuse, c("zinc", "lead"), sp.layout = 16)
spplot(meuse, c("zinc", "lead"),
sp.layout = list(meuse.grid, meuse.riv, col = 'grey'))
# Custom legend placement, taken from
# http://stackoverflow.com/questions/29344692/custom-placement-of-spplot-legend-in-the-map
s <- spplot(meuse.grid[,'dist'], colorkey = list(space = "left", height = 0.4))</pre>
args <- s$legend$left$args$key</pre>
## Prepare list of arguments needed by `legend=` argument (as described in ?xyplot)
library(lattice) # draw.colorkey
legendArgs <- list(fun = draw.colorkey,</pre>
                   args = list(key = args),
                   corner = c(0.05, .75))
## Call spplot() again, this time passing in to legend the arguments
## needed to print a color key
spplot(meuse.grid[,'dist'], colorkey = FALSE,
       legend = list(inside = legendArgs))
```

spsample

sample point locations in (or on) a spatial object

## Description

sample point locations within a square area, a grid, a polygon, or on a spatial line, using regular or random sampling methods; the methods used assume that the geometry used is not spherical, so objects should be in planar coordinates

# Usage

```
spsample(x, n, type, ...) makegrid(x, n = 10000, nsig = 2, cellsize, offset = rep(0.5, nrow(bb)), pretty = TRUE)
```

(approximate) sample size

## **Arguments**

n

x Spatial object; spsample(x,...) is a generic method for the existing sample. Xxx functions
 ... optional arguments, passed to the appropriate sample. Xxx functions; see NOTES for nclusters and iter

100 spsample

type character; "random" for completely spatial random; "regular" for regular (systematically aligned) sampling; "stratified" for stratified random (one single random location in each "cell"); "nonaligned" for nonaligned systematic sampling (nx random y coordinates, ny random x coordinates); "hexagonal" for sampling on a hexagonal lattice; "clustered" for clustered sampling; "Fibonacci"

for Fibonacci sampling on the sphere (see references).

bounding box of the sampled domain; setting this to a smaller value leads to

sub-region sampling

offset for square cell-based sampling types (regular, stratified, nonaligned): the offset

(position) of the regular grid; the default for spsample methods is a random location in the unit cell [0,1] x [0,1], leading to a different grid after each call; if this is set to c(0.5,0.5), the returned grid is not random (but, in Ripley's wording, "centric systematic"). For line objects, a single offset value is taken, where the value varies within the [0,1] interval, and 0 is the beginning of each

Line object, and 1 its end

cellsize if missing, a cell size is derived from the sample size n; otherwise, this cell size

is used for all sampling methods except "random"

nsig for "pretty" cell size; spsample does not result in pretty grids

pretty logical; if TRUE, choose pretty (rounded) coordinates

#### Value

bb

an object of class SpatialPoints-class. The number of points is only guaranteed to equal n when sampling is done in a square box, i.e. (sample.Spatial). Otherwise, the obtained number of points will have expected value n.

When x is of a class deriving from Spatial-class for which no spsample-methods exists, sampling is done in the bounding box of the object, using spsample. Spatial. An overlay using over may be necessary to select the features inside the geometry afterwards.

Sampling type "nonaligned" is not implemented for line objects.

Some methods may return NULL if no points could be successfully placed.

makegrid makes a regular grid that covers x; when cellsize is not given it derives one from the number of grid points requested (approximating the number of cells). It tries to choose pretty cell size and grid coordinates.

## Methods

x = "Spatial" sample in the bbox of x

x = "Line" sample on a line

x = "Polygon" sample in a Polygon

**x = "Polygons"** sample in a Polygons object, consisting of possibly multiple Polygon objects (holes must be correctly defined, use checkPolygonsHoles if need be)

**x** = "SpatialPolygons" sample in an SpatialPolygons object; sampling takes place over all Polygons objects present, use subsetting to vary sampling intensity (density); holes must be correctly defined, use checkPolygonsHoles if need be

x = "SpatialGrid" sample in an SpatialGrid object

x = "SpatialPixels" sample in an SpatialPixels object

spsample 101

#### Note

If an Polygon-class object has zero area (i.e. is a line), samples on this line element are returned. If the area is very close to zero, the algorithm taken here (generating points in a square area, selecting those inside the polygon) may be very resource intensive. When numbers of points per polygon are small and type="random", the number searched for is inflated to ensure hits, and the points returned sampled among these.

The following two arguments can be further specified:

nclusters Number of clusters (strata) to sample from.

iter(default = 4) number of times to try to place sample points in a polygon before giving up and returning NULL - this may occur when trying to hit a small and awkwardly shaped polygon in a large bounding box with a small number of points

## Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

#### References

Chapter 3 in B.D. Ripley, 1981. Spatial Statistics, Wiley

Fibonacci sampling: Alvaro Gonzalez, 2010. Measurement of Areas on a Sphere Using Fibonacci and Latitude-Longitude Lattices. Mathematical Geosciences 42(1), p. 49-64

#### See Also

```
over, point.in.polygon, sample
```

## **Examples**

```
data(meuse.riv)
meuse.sr = SpatialPolygons(list(Polygons(list(Polygon(meuse.riv)), "x")))

plot(meuse.sr)
points(spsample(meuse.sr, n = 1000, "regular"), pch = 3)

plot(meuse.sr)
points(spsample(meuse.sr, n = 1000, "random"), pch = 3)

plot(meuse.sr)
points(spsample(meuse.sr, n = 1000, "stratified"), pch = 3)

plot(meuse.sr)
points(spsample(meuse.sr, n = 1000, "nonaligned"), pch = 3)

plot(meuse.sr)
points(spsample(meuse.sr, n = 1000, "nonaligned"), pch = 3)

plot(meuse.sr)
points(spsample(meuse.sr@polygons[[1]], n = 100, "stratified"), pch = 3, cex=.5)

data(meuse.grid)
gridded(meuse.grid) = ~x+y
```

102 spTransform

```
image(meuse.grid)
points(spsample(meuse.grid,n=1000,type="random"), pch=3, cex=.5)
image(meuse.grid)
points(spsample(meuse.grid,n=1000,type="stratified"), pch=3, cex=.5)
image(meuse.grid)
points(spsample(meuse.grid,n=1000,type="regular"), pch=3, cex=.5)
image(meuse.grid)
points(spsample(meuse.grid,n=1000,type="nonaligned"), pch=3, cex=.5)
fullgrid(meuse.grid) = TRUE
image(meuse.grid)
points(spsample(meuse.grid,n=1000,type="stratified"), pch=3,cex=.5)
```

spTransform

spTransform for map projection and datum transformation

# Description

spTransform for map projection and datum transformation

# Usage

```
spTransform(x, CRSobj, ...)
```

# **Arguments**

x object to be transformed
 CRSobj object of class CRS, or of class character in which case it is converted to CRS
 ... further arguments (ignored)

## Value

object with coordinates transformed to the new coordinate reference system.

# Note

Package rgdal provides the methods doing actual transformation, see spTransform; when rgdal cannot be loaded, an error message follows.

stack 103

stack	rearrange data in SpatialPointsDataFrame or SpatialGridDataFrame
	for plotting with spplot (levelplot/xyplot wrapper)

# **Description**

rearrange SpatialPointsDataFrame for plotting with spplot or levelplot

# Usage

```
spmap.to.lev(data, zcol = 1:n, n = 2, names.attr)
## S3 method for class 'SpatialPointsDataFrame'
stack(x, select, ...)
## S3 method for class 'SpatialGridDataFrame'
stack(x, select, ...)
```

# Arguments

data	object of class (or extending) SpatialPointsDataFrame or SpatialGridDataFrame
zcol	z-coordinate column name(s), or a column number (range) (after removing the spatial coordinate columns: $1$ refers to the first non-coordinate column, etc. )
names.attr	names of the set of z-columns (these names will appear in the plot); if omitted, column names of $zcol$
n	number of columns to be stacked
x	same as data
select	same as zcol
	ignored

## Value

spmap.to.lev returns a data frame with the following elements:

```
    x x-coordinate for each row
    y y-coordinate for each row
    z column vector with each of the elements in columns zcol of data stacked
    name factor; name of each of the stacked z columns
```

stack is an S3 method: it return a data.frame with a column values that has the stacked coordinates and attributes, and a column ind that indicates the variable stacked; it also replicates the coordinates.

# See Also

```
spplot, levelplot in package lattice, and stack
```

104 surfaceArea

## **Examples**

```
library(lattice)
data(meuse.grid) # data frame
coordinates(meuse.grid) = c("x", "y") # promotes to SpatialPointsDataFrame
meuse.grid[["idist"]] = 1 - meuse.grid[["dist"]] # add variable
# the following is made much easier by spplot:
levelplot(z~x+y|name, spmap.to.lev(meuse.grid, z=c("dist","idist"), names.attr =
c("distance", "inverse of distance")), aspect = "iso")
levelplot(values~x+y|ind, as.data.frame(stack(meuse.grid)),aspect = "iso")
gridded(meuse.grid) = TRUE
levelplot(z~x+y|name, spmap.to.lev(meuse.grid, z=c("dist","idist"), names.attr =
c("distance", "inverse of distance")), aspect = "iso")
levelplot(values~x+y|ind, as.data.frame(stack(meuse.grid)), asp = "iso")
```

surfaceArea

Compute surface area of a digital elevation model.

## **Description**

It is often said that if Wales was flattened out it would have an area bigger than England. This function computes the surface area of a grid of heights taking into account the sloping nature of the surface.

## **Usage**

```
surfaceArea(m, ...)
surfaceArea.matrix(m, cellx = 1, celly = 1, byCell = FALSE)
```

## Arguments

m	a matrix of height values, or an object of class SpatialPixelsDataFrame or SpatialGridDataFrame.
cellx	the size of the grid cells in the x-direction, in the same units as the height values.
celly	the size of the grid cells in the y-direction, in the same units as the height values.
byCell	return single value or matrix of values
	ignored

## Value

Either a single value of the total area if byCell=FALSE, or a matrix the same shape as m of individual cell surface areas if byCell=TRUE. In this case, the sum of the returned matrix should be the same value as that which is returned if byCell=FALSE.

Missing values (NA) in the input matrix are allowed. They will produce an NA in the output matrix for byCell=TRUE, and contribute zero to the total area. They also have an effect on adjacent cells - see code comments for details.

zerodist 105

## Methods

- **obj = "matrix"** takes a matrix as input, requires cellx and celly to be set
- **obj = "SpatialGridDataFrame"** takes an object of class SpatialGridDataFrame as input, and retrieves cellx and celly from this
- **obj = "SpatialPixelsDataFrame"** takes an object of class SpatialPixelsDataFrame as input, and retrieves cellx and celly from this

## Author(s)

Barry Rowlingson <b.rowlingson@lancaster.ac.uk>, integration in sp Edzer Pebesma.

#### References

Calculating Landscape Surface Area from Digital Elevation Models, Jeff S. Jenness Wildlife Society Bulletin, Vol. 32, No. 3 (Autumn, 2004), pp. 829-839

## **Examples**

```
surfaceArea(volcano)
image(surfaceArea(volcano,byCell=TRUE))

data(meuse.grid)
gridded(meuse.grid) = ~x+y
image(surfaceArea(meuse.grid["dist"], byCell=TRUE))
surfaceArea(meuse.grid["dist"])
```

zerodist

find point pairs with equal spatial coordinates

# **Description**

find point pairs with equal spatial coordinates

## Usage

```
zerodist(obj, zero = 0.0, unique.ID = FALSE, memcmp = TRUE)
zerodist2(obj1, obj2, zero = 0.0, memcmp = TRUE)
remove.duplicates(obj, zero = 0.0, remove.second = TRUE, memcmp = TRUE)
```

#### **Arguments**

obj	object of, or extending, class SpatialPoints
obj1	object of, or extending, class SpatialPoints
obj2	object of, or extending, class SpatialPoints

106 zerodist

distance values less than or equal to this threshold value are considered to have zero distance (default 0.0); units are those of the coordinates for projected data or unknown projection, or km if coordinates are defined to be longitute/latitude unique.ID logical; if TRUE, return an ID (integer) for each point that is different only when two points do not share the same location

memcmp use memcmp to find exactly equal coordinates; see NOTE

remove.second logical; if TRUE, the second of each pair of duplicate points is removed, if

FALSE remove the first

#### Value

zerodist and zerodist2 return a two-column matrix with in each row pairs of row numbers with identical coordinates; a matrix with zero rows is returned if no such pairs are found. For zerodist, row number pairs refer to row pairs in obj. For zerodist2, row number pairs refer to rows in obj and obj2, respectively. remove.duplicates removes duplicate observations if present, and else returns obj.

#### Note

When using kriging, duplicate observations sharing identical spatial locations result in singular covariance matrices. This function may help identify and remove spatial duplices. The full matrix with all pair-wise distances is not stored; the double loop is done at the C level.

When unique. ID=TRUE is used, an integer index is returned. sp 1.0-14 returned the highest index, sp 1.0-15 and later return the lowest index.

When zero is 0.0 and memcmp is not FALSE, zerodist uses memcmp to evaluate exact equality of coordinates; there may be cases where this results in a different evaluation compared to doing the double arithmetic of computing distances.

## **Examples**

```
data(meuse)
summary(meuse)
# pick 10 rows
n <- 10
ran10 <- sample(nrow(meuse), size = n, replace = TRUE)
meusedup <- rbind(meuse, meuse[ran10, ])</pre>
coordinates(meusedup) <- c("x", "y")</pre>
zd <- zerodist(meusedup)</pre>
sum(abs(zd[1:n,1] - sort(ran10))) # 0!
# remove the duplicate rows:
meusedup2 <- meusedup[-zd[,2], ]</pre>
summary(meusedup2)
meusedup3 <- subset(meusedup, !(1:nrow(meusedup) %in% zd[,2]))</pre>
summary(meusedup3)
coordinates(meuse) <- c("x", "y")</pre>
zerodist2(meuse, meuse[c(10:33,1,10),])
```

# **Index**

*Topic <b>classes</b>	stack, 103
CRS-class, 18	zerodist, 105
DMS-class, 23	*Topic <b>manip</b>
GridTopology-class, 30	coordinates, 16
Line, 36	gridIndex2nb, 27
Line-class, 37	point.in.polygon, 50
Lines-class, 37	polygons, 52
Polygon-class, 51	sp-deprecated, 59
Polygons-class, 52	SpatialLines, 66
Spatial-class, 60	SpatialMultiPoints, 70
SpatialGrid-class, 62	SpatialPoints, 81
SpatialGridDataFrame-class, 63	SpatialPolygons, 86
SpatialLines-class, 67	spsample, 99
SpatialLinesDataFrame-class, 68	*Topic <b>methods</b>
SpatialMultiPoints-class, 71	addAttrToGeom-methods, 4
SpatialMultiPointsDataFrame-class,	aggregate, 5
73	bbox-methods, 10
SpatialPixels-class, 77	coordinates-methods, 17
SpatialPixelsDataFrame-class, 79	coordnames-methods, 18
SpatialPoints-class, 82	dimensions-methods, 21
SpatialPointsDataFrame-class, 84	disaggregate-methods, 22
SpatialPolygons-class, 87	geometry-methods, 25
SpatialPolygonsDataFrame-class, 89	gridded-methods, 26
*Topic <b>color</b>	merge, 40
bpy.colors, 11	over-methods, 45
*Topic datasets	polygons-methods, 53
meuse, 41	recenter-methods, 55
meuse.grid,42	spChFIDs-methods, 91
meuse.grid_11,43	spsample, 99
meuse.riv,44	spTransform, 102
Rlogo, 56	*Topic <b>misc</b>
*Topic <b>dplot</b>	compassRose, 15
bubble, 12	*Topic <b>models</b>
degAxis, 20	select.spatial, 57
flip, 24	*Topic <b>programming</b>
loadMeuse, 38	read.asciigrid,54
mapasp, 39	*Topic <b>spatial</b>
panel.spplot,48	as.SpatialPolygons.GridTopology,7
spplot, 94	as. $SpatialPolygons.PolygonsList, 8$

bbox-methods, 10	[[,Spatial,ANY,missing-method
char2dms, 14	(Spatial-class), 60
CRS-class, 18	<pre>[[&lt;-,Spatial,ANY,missing-method</pre>
DMS-class, 23	(Spatial-class), 60
gridded-methods, 26	\$,Spatial-method(Spatial-class),60
gridlines, 28	<pre>\$,SpatialMultiPoints-method</pre>
image.SpatialGridDataFrame, 32	(SpatialMultiPoints-class), 71
is.projected, 34	<pre>\$,SpatialPoints-method</pre>
merge, 40	(SpatialPoints-class), 82
polygons-methods, 53	<pre>\$&lt;-,Spatial-method (Spatial-class), 60</pre>
sp, 58	<pre>\$&lt;-,SpatialMultiPoints,character-method</pre>
SpatialPixels, 74	(SpatialMultiPoints-class), 71
SpatialPixelsDataFrame, 78	<pre>\$&lt;-,SpatialPoints,character-method</pre>
spChFIDs-methods, 91	(SpatialPoints-class), 82
spDistsN1, 92	%over% (over-methods), 45
spTransform, 102	
surfaceArea, 104	addAttrToGeom(addAttrToGeom-methods), 4
[,SpatialGrid-method	addAttrToGeom,SpatialGrid,data.frame-method
(SpatialGrid-class), 62	(addAttrToGeom-methods), 4
[,SpatialGridDataFrame-method	addAttrToGeom,SpatialLines,data.frame-method
(SpatialGridDataFrame-class),	(addAttrToGeom-methods), 4
63	${\tt addAttrToGeom,SpatialMultiPoints,data.frame-method}$
[,SpatialLines-method	(addAttrToGeom-methods), 4
(SpatialLines-class), 67	addAttrToGeom,SpatialPixels,data.frame-method
[,SpatialLinesDataFrame-method	<pre>(addAttrToGeom-methods), 4</pre>
(SpatialLinesDataFrame-class),	addAttrToGeom,SpatialPoints,data.frame-method
68	<pre>(addAttrToGeom-methods), 4</pre>
[,SpatialMultiPoints-method	addAttrToGeom,SpatialPolygons,data.frame-method
(SpatialMultiPoints-class), 71	<pre>(addAttrToGeom-methods), 4</pre>
[,SpatialMultiPointsDataFrame-method	addAttrToGeom-methods, 4
(SpatialMultiPointsDataFrame-class),	aggregate, 5
73	aggregate.data.frame, 5
[,SpatialPixels-method	areaSpatialGrid(SpatialPixels),74
(SpatialPixels-class), 77	as, 73, 84
[,SpatialPixelsDataFrame-method	as.array.SpatialGridDataFrame
(SpatialPixelsDataFrame-class),	(SpatialGridDataFrame-class),
79	63
[,SpatialPoints-method	as.character.DMS(char2dms),14
(SpatialPoints-class), 82	as.data.frame.SpatialGrid
[,SpatialPointsDataFrame-method	(SpatialGrid-class), 62
(SpatialPointsDataFrame-class),	as.data.frame.SpatialGridDataFrame
84	(SpatialGridDataFrame-class),
[,SpatialPolygons-method	63
(SpatialPolygons-class), 87	as.data.frame.SpatialMultiPoints
[,SpatialPolygonsDataFrame-method	(SpatialMultiPoints-class), 71
(SpatialPolygonsDataFrame-class),	as.data.frame.SpatialMultiPointsDataFrame
89	(SpatialMultiPointsDataFrame-class),
[<-,Spatial-method(Spatial-class),60	73

as.data.frame.SpatialPixels	cbind.SpatialGridDataFrame
(SpatialPixels-class), 77	(Spatial Grid Data Frame-class),
as.data.frame.SpatialPixelsDataFrame	63
(SpatialPixelsDataFrame-class),	char2dms, 14, 24
79	cm.colors, 12
as.data.frame.SpatialPoints	coerce,deldir,SpatialLines-method
(SpatialPoints-class), 82	(SpatialLines-class), 67
as.data.frame.SpatialPointsDataFrame	coerce,deldir,SpatialPolygons-method
$({\tt SpatialPointsDataFrame-class}),$	(SpatialPolygons-class), 87
84	<pre>coerce,DMS,character-method(char2dms),</pre>
as.data.frame.SpatialPolygons	14
(SpatialPolygons-class), 87	coerce, DMS, numeric-method (char2dms), 14
as.data.frame.SpatialPolygonsDataFrame	coerce, DMS-method (DMS-class), 23
(SpatialPolygonsDataFrame-class),	<pre>coerce,GridTopology,data.frame-method</pre>
89	(GridTopology-class), 30
as.double.DMS(DMS-class), 23	${\tt coerce,GridTopology,SpatialPolygons-method}$
as.image.SpatialGridDataFrame, 54	(as. Spatial Polygons. Grid Topology),
as.image.SpatialGridDataFrame	7
(image.SpatialGridDataFrame),	coerce,im,SpatialGridDataFrame-method
32	(Spatial Grid Data Frame-class),
as.numeric.DMS (DMS-class), 23	63
as.SpatialLines.SLDF (SpatialLines), 66	coerce,Lines,SpatialMultiPoints-method
as.SpatialPoints.SpatialPointsDataFrame	(SpatialLines-class), 67
(SpatialPointsDataFrame-class),	coerce,Lines,SpatialPoints-method
84	(SpatialLines-class), 67
as.SpatialPolygons.GridTopology,7	coerce, Polygons, Lines-method
as.SpatialPolygons.PolygonsList,8	(SpatialPolygons-class), 87
as.SpatialPolygons.SpatialPixels	coerce,ppp,SpatialGridDataFrame-method
<pre>(as.SpatialPolygons.GridTopology), 7</pre>	(SpatialGridDataFrame-class),
as.SpatialPolygonsDataFrame.SpatialPolygons	63
(SpatialPolygons-class), 87	coerce,ppp,SpatialPoints-method
axis, 21, 64	(SpatialPoints-class), 82
axTicks, 21, 39	coerce,ppp,SpatialPointsDataFrame-method
ax11cn3, 21, 39	(SpatialPointsDataFrame-class), 84
bbexpand (panel.spplot), 48	coerce, Spatial Grid, data. frame-method
bbox (bbox-methods), 10	(SpatialGrid-class), 62
bbox, ANY-method (bbox-methods), 10	<pre>coerce,SpatialGrid,GridTopology-method</pre>
bbox, Line-method (bbox-methods), 10	(GridTopology-class), 30
bbox, Lines-method (bbox-methods), 10	coerce, SpatialGrid, SpatialPixels-method
bbox, Polygon-method (bbox-methods), 10	(SpatialGrid-class), 62
bbox, Polygons-method (bbox-methods), 10	coerce, SpatialGrid, SpatialPoints-method
bbox, Spatial-method (bbox-methods), 10	(SpatialGrid-class), 62
bbox-methods, 10	<pre>coerce,SpatialGrid,SpatialPolygons-method</pre>
bpy.colors, 11, 97	(SpatialGrid-class), 62
bubble, 12	coerce,SpatialGridDataFrame,array-method
	(SpatialGridDataFrame-class),
cbind.Spatial (Spatial-class), 60	63

```
coerce, SpatialGridDataFrame, data.frame-methodcoerce, SpatialPixels, SpatialPolygons-method
                         (SpatialGridDataFrame-class),
                                                                                                                                                                           (as.SpatialPolygons.GridTopology),
coerce, Spatial Grid Data Frame, matrix-method
                                                                                                                                                  coerce, SpatialPixelsDataFrame, array-method
                         (SpatialGridDataFrame-class),
                                                                                                                                                                            (SpatialPixelsDataFrame-class),
coerce, SpatialGridDataFrame, SpatialPixelsDataFramee, Mata.frame-method
                         (SpatialGridDataFrame-class),
                                                                                                                                                                            (SpatialPixelsDataFrame-class),
coerce, Spatial Grid Data Frame, Spatial Points Data \textit{\it Evalue} e, \textit{\it Mathibal} al Pixels Data Frame, matrix-method
                         (SpatialGridDataFrame-class),
                                                                                                                                                                            (SpatialPixelsDataFrame-class),
                         63
coerce, Spatial Grid Data Frame, Spatial Polygons Data \textbf{\textit{Frame}} S \textbf{\textit{pratial}} P ixels Data Frame, Spatial Grid Data Frame-method
                         (SpatialGridDataFrame-class),
                                                                                                                                                                            (SpatialPixelsDataFrame-class),
coerce, Spatial Lines, Spatial MultiPoints-method coerce, Spatial Pixels Data Frame, Spatial Points Data Frame-method coerce, Spatial Points Data Frame-method
                         (SpatialLines-class), 67
                                                                                                                                                                            (SpatialPixelsDataFrame-class),
coerce, SpatialLines, SpatialPoints-method
                                                                                                                                                  coerce, SpatialPixelsDataFrame, SpatialPolygonsDataFrame-met
                         (SpatialLines-class), 67
                                                                                                                                                                            (SpatialPixelsDataFrame-class),
coerce, SpatialLines, SpatialPointsDataFrame-method
                         (SpatialLines-class), 67
coerce, Spatial Lines Data Frame, data. frame-metho \textbf{\textit{d}} o erce, Spatial Points, data. frame-method
                         (SpatialLinesDataFrame-class),
                                                                                                                                                                           (SpatialPoints-class), 82
                                                                                                                                                  coerce, Spatial Points, Line-method
coerce, SpatialLinesDataFrame, SpatialMultiPointsDataFramethedHoodints-class), 82
                         (SpatialLinesDataFrame-class),
                                                                                                                                                 coerce, Spatial Points, Lines-method
                                                                                                                                                                            (SpatialPoints-class), 82
coerce, Spatial Lines Data Frame, Spatial Points Data \textit{Gerame}, \textit{Spatial} Points, matrix-method and \textit{Gerame}, \textit{Spatial} Points, \textit{Constitution} Points Points
                         (SpatialLinesDataFrame-class),
                                                                                                                                                                            (SpatialPoints-class), 82
                                                                                                                                                  coerce, SpatialPoints, SpatialLines-method
coerce,SpatialMultiPoints,data.frame-method
                                                                                                                                                                           (SpatialPoints-class), 82
                         (SpatialMultiPoints-class), 71
                                                                                                                                                  coerce, SpatialPoints, SpatialPixels-method
coerce,SpatialMultiPoints,matrix-method
                                                                                                                                                                            (SpatialPoints-class), 82
                         (SpatialMultiPoints-class), 71
                                                                                                                                                  coerce, Spatial Points Data Frame, data. frame-method
                                                                                                                                                                            (SpatialPointsDataFrame-class),
coerce, Spatial MultiPoints, Spatial Points-method
                         (SpatialMultiPoints-class), 71
coerce, Spatial MultiPoints Data Frame, data. frame \textbf{comentbe} \\ dSpatial Points Data Frame, Spatial Pixels Data Frame-method Spatial Points Data Frame \\ data
                         (SpatialMultiPointsDataFrame-class),
                                                                                                                                                                           (SpatialPoints-class), 82
                                                                                                                                                  coerce, Spatial Points Data Frame, Spatial Points-method
coerce, Spatial MultiPoints Data Frame, Spatial Points Data Frame-class),
                         (SpatialMultiPointsDataFrame-class),
                                                                                                                                                                            84
                                                                                                                                                  coerce, SpatialPolygons, SpatialLines-method
coerce,SpatialPixels,data.frame-method
                                                                                                                                                                            (SpatialPolygons-class), 87
                         (SpatialPixels-class), 77
                                                                                                                                                  coerce, Spatial Polygons, Spatial Polygons Data Frame-method\\
coerce, SpatialPixels, GridTopology-method
                                                                                                                                                                            (SpatialPolygons-class), 87
                         (GridTopology-class), 30
                                                                                                                                                  coerce,SpatialPolygonsDataFrame,data.frame-method
coerce, SpatialPixels, SpatialGrid-method
                                                                                                                                                                            (SpatialPolygonsDataFrame-class),
                         (SpatialPixels-class), 77
                                                                                                                                                                            89
```

coerce, SpatialPolygonsDataFrame, SpatialLines	DataFrame(concentrobichates-methods), 17
$({\sf SpatialPolygonsDataFrame-class}),$	coordinates,SpatialPolygonsDataFrame-method
89	(coordinates-methods), 17
coerce, Spatial Polygons Data Frame, Spatia	o <b>osomethat</b> es-methods, 17, 49
(SpatialPolygonsDataFrame-class),	coordinates<-,81
89	coordinates<- (coordinates), 16
compassRose, 15	coordinates<-,data.frame-method
contour.SpatialGridDataFrame	(coordinates-methods), 17
<pre>(image.SpatialGridDataFrame),</pre>	coordinates<-,Spatial-method
32	(Spatial-class), 60
contour.SpatialPixelsDataFrame	coordinatevalues (SpatialPixels), 74
<pre>(image.SpatialGridDataFrame),</pre>	coordnames (coordnames-methods), 18
32	coordnames,Line-method
coordinates, 12, 16, 17, 59, 70, 74, 75, 82,	(coordnames-methods), 18
85, 92	coordnames,Lines-method
coordinates,data.frame-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames, Polygon-method
coordinates,GridTopology-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames, Polygons-method
coordinates,Line-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames, SpatialGrid-method
coordinates,Lines-method	(SpatialGrid-class), 62
(coordinates-methods), 17	coordnames, SpatialLines-method
coordinates, list-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames, Spatial MultiPoints-method
coordinates, matrix-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames, SpatialPoints-method
coordinates,SpatialGrid-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames, SpatialPolygons-method
coordinates,SpatialGridDataFrame-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames-methods, 18
coordinates,SpatialLines-method	coordnames<- (coordnames-methods), 18
(coordinates-methods), 17	<pre>coordnames&lt;-,GridTopology,character-method</pre>
coordinates,SpatialMultiPoints-method	(coordnames-methods), 18
(coordinates-methods), 17	coordnames<-,Line,character-method
coordinates, Spatial MultiPoints Data Frame-method	od (coordnames-methods), 18
(SpatialMultiPointsDataFrame-class),	coordnames<-,Lines,character-method
73	(coordnames-methods), 18
coordinates,SpatialPixels-method	coordnames<-,SpatialGrid,character-method
(coordinates-methods), 17	(coordnames-methods), 18
coordinates,SpatialPixelsDataFrame-method	coordnames<-,SpatialLines,character-method
(coordinates-methods), 17	(coordnames-methods), 18
coordinates,SpatialPoints-method	<pre>coordnames&lt;-,SpatialMultiPoints,character-method</pre>
(coordinates-methods), 17	(coordnames-methods), 18
coordinates,SpatialPointsDataFrame-method	coordnames<-,SpatialPixels,character-method
(SpatialPointsDataFrame-class),	(coordnames-methods), 18
84	coordnames<-,SpatialPoints,character-method
coordinates, Spatial Polygons-method	(coordnames-methods), 18

<pre>coordnames&lt;-,SpatialPolygons,character-metho</pre>	dExtract, 73, 84
(coordnames-methods), 18	
createSPComment, 87, 89	filled.contour, 33
CRS, 19, 35, 102	flip, 24
CRS (CRS-class), 18	flipHorizontal (flip), 24
CRS-class, 7, 18, 67, 69, 70, 75, 78, 81, 86,	flipVertical(flip), 24
87, 90	fullgrid, 76, 78
CRSargs (CRS-class), 18	fullgrid(gridded-methods), 26
	fullgrid,Spatial-method
data.frame, 69	(gridded-methods), 26
dd2dms, 24	fullgrid<- (gridded-methods), 26
dd2dms (char2dms), 14	fullgrid<-,Spatial,ANY-method
degAxis, 20	(gridded-methods), 26
degreeLabelsEW (mapasp), 39	fullgrid<-,SpatialGrid,logical-method
degreeLabelsNS (mapasp), 39	(gridded-methods), 26
dim.SpatialGridDataFrame	<pre>fullgrid&lt;-,SpatialGridDataFrame,logical-method</pre>
(SpatialGridDataFrame-class),	(gridded-methods), 26
63	fullgrid<-,SpatialPixels,logical-method
dim.SpatialLinesDataFrame	(gridded-methods), 26
(SpatialLinesDataFrame-class),	fullgrid<-,SpatialPixelsDataFrame,logical-method
68	(gridded-methods), 26
dim.SpatialMultiPointsDataFrame	//
(SpatialMultiPointsDataFrame-class),	geometry (geometry-methods), 25
73	geometry, Spatial-method
dim.SpatialPixelsDataFrame	(geometry-methods), 25
(SpatialPixelsDataFrame-class),	<pre>geometry,SpatialGridDataFrame-method</pre>
79	(geometry-methods), 25
dim.SpatialPointsDataFrame	<pre>geometry,SpatialLinesDataFrame-method</pre>
(SpatialPointsDataFrame-class),	(geometry-methods), 25
84	<pre>geometry,SpatialMultiPointsDataFrame-method</pre>
dim.SpatialPolygonsDataFrame	(geometry-methods), 25
(SpatialPolygonsDataFrame-class),	<pre>geometry,SpatialPixelsDataFrame-method</pre>
89	(geometry-methods), 25
dimensions (dimensions-methods), 21	<pre>geometry,SpatialPointsDataFrame-method</pre>
dimensions, Spatial-method	(geometry-methods), 25
(dimensions-methods), 21	<pre>geometry,SpatialPolygonsDataFrame-method</pre>
dimensions-methods, 21	(geometry-methods), 25
disaggregate (disaggregate-methods), 22	geometry-methods, 25
disaggregate, SpatialLines-method	<pre>geometry&lt;- (geometry-methods), 25</pre>
(disaggregate-methods), 22	<pre>geometry&lt;-,data.frame,Spatial-method</pre>
disaggregate, SpatialLinesDataFrame-method	(geometry-methods), 25
(disaggregate-methods), 22	get_col_regions (spplot), 94
disaggregate, SpatialPolygons-method	<pre>get_ll_TOL (is.projected), 34</pre>
(disaggregate-methods), 22	<pre>get_ll_warn(is.projected), 34</pre>
disaggregate, Spatial Polygons Data Frame-method	
(disaggregate-methods), 22	get_PolypathRule
disaggregate-methods, 22	(SpatialPolygons-class), 87
dist, 27	<pre>get_ReplCRS_warn(is.projected), 34</pre>
DMS-class, 23	getGridIndex (SpatialPixels), 74

<pre>getGridTopology (SpatialPixels), 74</pre>	(gridded-methods), 26
getLinesIDSlot (sp-deprecated), 59	<pre>gridded&lt;-,SpatialPixels,logical-method</pre>
getLinesLinesSlot (sp-deprecated), 59	(gridded-methods), 26
getParUsrBB (Spatial-class), 60	<pre>gridded&lt;-,SpatialPixelsDataFrame,logical-method</pre>
getPolygonAreaSlot (sp-deprecated), 59	(gridded-methods), 26
getPolygonCoordsSlot (sp-deprecated), 59	<pre>gridded&lt;-,SpatialPoints,list-method</pre>
getPolygonHoleSlot(sp-deprecated), 59	(gridded-methods), 26
getPolygonLabptSlot (sp-deprecated), 59	<pre>gridded&lt;-,SpatialPoints,logical-method</pre>
getPolygonsIDSlot (sp-deprecated), 59	(gridded-methods), 26
getPolygonsLabptSlot (sp-deprecated), 59	<pre>gridded&lt;-,SpatialPointsDataFrame,list-method</pre>
getPolygonsplotOrderSlot	(gridded-methods), 26
(sp-deprecated), 59	<pre>gridded&lt;-,SpatialPointsDataFrame,logical-method</pre>
getPolygonsPolygonsSlot	(gridded-methods), 26
(sp-deprecated), 59	gridIndex2nb, 27
getSLLinesIDSlots (sp-deprecated), 59	gridlines, 28
getSLlinesSlot (sp-deprecated), 59	gridparameters (gridded-methods), 26
getSpatialLinesMidPoints	GridsDatums, 20
(SpatialLines), 66	GridTopology, 8
getSpatialPolygonsLabelPoints	GridTopology (SpatialPixels), 74
(SpatialPolygons), 86	GridTopology-class, 30, 62, 63, 75, 77–79
getSpPnHoles (sp-deprecated), 59	gt (Rlogo), 56
getSpPnParts (sp-deprecated), 59	80 (1.2080), 00
getSpPplotOrderSlot (sp-deprecated), 59	head.Spatial(Spatial-class), 60
getSpPPolygonsIDSlots(sp-deprecated),	
59	HexPoints2SpatialPolygons
	<pre>(as.SpatialPolygons.GridTopology), 7</pre>
getSpPPolygonsLabptSlots	/
(sp-deprecated), 59	
getSpPpolygonsSlot(sp-deprecated),59	identicalCRS (CRS-class), 18
gIntersection, 5	identify, 13
gIntersects, 47	IDvaluesGridTopology
gRelate, 46	<pre>(as.SpatialPolygons.GridTopology),</pre>
gridat (gridlines), 28	7
gridded, 79	IDvaluesSpatialPixels
gridded (gridded-methods), 26	(as.SpatialPolygons.GridTopology),
gridded, Spatial-method	7
(gridded-methods), 26	im-class(Spatial-class), 60
gridded-methods, 26	image, $33, 54$
gridded<-, <i>7</i> 9	image.default, 33
<pre>gridded&lt;- (gridded-methods), 26</pre>	image.SpatialGridDataFrame, 32,64
gridded<-,data.frame,character-method	<pre>image.SpatialPixels</pre>
(gridded-methods), 26	<pre>(image.SpatialGridDataFrame),</pre>
gridded<-,data.frame,formula-method	32
(gridded-methods), 26	image.SpatialPixelsDataFrame
<pre>gridded&lt;-,data.frame,GridTopology-method</pre>	<pre>(image.SpatialGridDataFrame),</pre>
(gridded-methods), 26	32
gridded<-,SpatialGrid,logical-method	image2Grid
(gridded-methods), 26	<pre>(image.SpatialGridDataFrame),</pre>
<pre>gridded<spatialgriddataframe.logical-met< pre=""></spatialgriddataframe.logical-met<></pre>	hod 32

imageScale	over, SpatialGrid, SpatialPixelsDataFrame-method
<pre>(image.SpatialGridDataFrame),</pre>	(over-methods), 45
32	over, SpatialGrid, SpatialPoints-method
is.projected, 34, 92, 93	(over-methods), 45
is.projected,Spatial-method	over, SpatialGrid, SpatialPointsDataFrame-method
(is.projected), 34	(over-methods), 45
	over, SpatialGrid, SpatialPolygons-method
labels(gridlines),28	(over-methods), 45
layout.north.arrow(spplot),94	over, SpatialGrid, SpatialPolygonsDataFrame-method
layout.scale.bar(spplot),94	(over-methods), 45
lcm, <i>64</i>	over, SpatialGridDataFrame, SpatialPoints-method
levelplot, 33, 39, 95, 96, 98, 103	(over-methods), 45
Line, 22, 36, <i>37</i>	over, Spatial Grid Data Frame, Spatial Polygons Data Frame-method
Line-class, <i>36</i> , <i>37</i> , <i>38</i> , <i>49</i> , <i>68</i>	(over-methods), 45
LineLength (SpatialLines), 66	over, SpatialPixels, SpatialPoints-method
Lines, 22	(over-methods), 45
Lines (Line), 36	over, SpatialPixelsDataFrame, SpatialPoints-method
Lines-class, 36, 37, 37, 38, 49, 66–68	(over-methods), 45
LinesLength (SpatialLines), 66	over, SpatialPoints, SpatialGrid-method
loadMeuse, 38	(over-methods), 45
locator, 57	over, SpatialPoints, SpatialGridDataFrame-method
longlat.scales (panel.spplot), 48	(over-methods), 45
	over, SpatialPoints, SpatialPixels-method
make_EPSG, 20	(over-methods), 45
makegrid(spsample),99	over, SpatialPoints, SpatialPixelsDataFrame-method
mapasp, <i>13</i> , <i>39</i>	(over-methods), 45
mapLegendGrob(spplot),94	over, SpatialPoints, SpatialPoints-method
match, $40$	(over-methods), 45
mean, $5$	over, SpatialPoints, SpatialPointsDataFrame-method
merge, 40, 40	(over-methods), 45
merge, Spatial, ANY-method (merge), 40	over, SpatialPoints, SpatialPolygons-method
merge,Spatial,data.frame-method	(over-methods), 45
(merge), 40	over, SpatialPoints, SpatialPolygonsDataFrame-method
meuse, <i>38</i> , 41, <i>42–44</i>	(over-methods), 45
meuse.area(meuse.riv),44	over, SpatialPolygons, SpatialGrid-method
meuse.grid, <i>38</i> , <i>41</i> , 42, <i>44</i>	(over-methods), 45
meuse.grid_11,43	over, SpatialPolygons, SpatialGridDataFrame-method
meuse.riv,44	(over-methods), 45
	over, SpatialPolygons, SpatialPoints-method
over, 4, 5, 59, 100, 101	(over-methods), 45
over (over-methods), 45	over, SpatialPolygons, SpatialPointsDataFrame-method
over,Spatial,Spatial-method	(over-methods), 45
(over-methods), 45	over-methods, 45
over,SpatialGrid,SpatialGrid-method	overDF_for_rgeos (over-methods), 45
(over-methods), 45	owin-class (Spatial-class), 60
over, Spatial Grid, Spatial Grid Data Frame-method	
(over-methods), 45	panel.ggmap(spplot), 94
over,SpatialGrid,SpatialPixels-method	panel.gridplot, 95
(over-methods), 45	panel.gridplot(panel.spplot), 48

panel.identify, 98	Polygons (SpatialPolygons), 86
panel.pointsplot, 95	polygons, 52
panel.pointsplot(panel.spplot), 48	polygons,Spatial-method
panel.polygonsplot, 95	(polygons-methods), 53
panel.polygonsplot (panel.spplot), 48	polygons, SpatialPolygons-method
panel.RgoogleMaps(spplot), 94	(polygons-methods), 53
panel.spplot, 48	Polygons-class, <i>51</i> , <i>52</i> , <i>86</i> , <i>87</i>
par, 64, 88	polygons-methods, 53
plot, Spatial, missing-method	polygons<- (polygons), 52
(Spatial-class), 60	polygons<-,data.frame,SpatialPolygons-method
plot, SpatialGrid, missing-method	(polygons-methods), 53
(SpatialGrid-class), 62	polypath, 88
plot,SpatialGridDataFrame,missing-method	ppp-class (Spatial-class), 60
(SpatialGridDataFrame-class),	pretty, 64
63	print.CRS (CRS-class), 18
plot, SpatialLines, missing-method	print.DMS (DMS-class), 23
(SpatialLines-class), 67	print.SpatialMultiPoints
plot, Spatial MultiPoints, missing-method	(SpatialMultiPoints-class), 71
(SpatialMultiPoints-class), 71	print.SpatialMultiPointsDataFrame
plot, SpatialPixels, missing-method	(SpatialMultiPointsDataFrame-class),
(SpatialPixels-class), 77	73
plot, SpatialPixelsDataFrame, missing-method	print.SpatialPoints
(SpatialPixelsDataFrame-class),	(SpatialPoints-class), 82
79	print.SpatialPointsDataFrame
plot, SpatialPoints, missing-method	(SpatialPointsDataFrame-class),
(SpatialPoints-class), 82	84
plot, SpatialPolygons, missing-method	<pre>print.summary.GridTopology</pre>
(SpatialPolygons-class), 87	(GridTopology-class), 30
plot.SpatialGrid (SpatialPixels), 74	<pre>print.summary.Spatial(Spatial-class),</pre>
plot.SpatialGridDataFrame	60
(SpatialGridDataFrame-class),	print.summary.SpatialGrid
63	(SpatialGrid-class), 62
plot.SpatialPixelsDataFrame	print.summary.SpatialGridDataFrame
(SpatialPixelsDataFrame-class),	(SpatialGridDataFrame-class),
79	63
point.in.polygon, 47, 50, 57, 101	print.summary.SpatialPixels
points, Spatial MultiPoints Data Frame-method	(SpatialPixels-class), 77
(SpatialMultiPointsDataFrame-class),	print.summary.SpatialPixelsDataFrame
73	(SpatialPixelsDataFrame-class),
points,SpatialPointsDataFrame-method	79
(SpatialPointsDataFrame-class),	proj4string, 19
84	proj4string (is.projected), 34
points2grid (SpatialPixels), 74	proj4string,Spatial-method
Polygon, 22	(is.projected), 34
Polygon (SpatialPolygons), 86	proj4string<- (is.projected), 34
polygon, 88	<pre>proj4string&lt;-,Spatial,character-method</pre>
Polygon-class, 51, 53, 86, 101	(is.projected), 34
Polygons, 22	<pre>proj4string&lt;-,Spatial,CRS-method</pre>

	0 1
(is.projected), 34	row.names.SpatialLines
psp-class (Spatial-class), $60$	(SpatialLines-class), 67
	row.names.SpatialLinesDataFrame
rainbow, 12	(SpatialLinesDataFrame-class),
rasterImage, 33	68
rbind.SpatialLines	row.names.SpatialMultiPoints
(SpatialLines-class), 67	(SpatialMultiPoints-class), 71
rbind.SpatialLinesDataFrame	row.names.SpatialMultiPointsDataFrame
(SpatialLinesDataFrame-class),	(SpatialMultiPointsDataFrame-class), 73
68	row.names.SpatialPoints
rbind.SpatialMultiPoints	(SpatialPoints-class), 82
(SpatialMultiPoints-class), 71	row.names.SpatialPointsDataFrame
rbind.SpatialMultiPointsDataFrame	(SpatialPointsDataFrame-class),
(SpatialMultiPointsDataFrame-class),	84
73	row.names.SpatialPolygons
rbind.SpatialPixels	(SpatialPolygons-class), 87
(SpatialPixels-class), 77	row.names.SpatialPolygonsDataFrame
rbind.SpatialPixelsDataFrame (SpatialPixelsDataFrame-class),	(SpatialPolygonsDataFrame-class),
79	89
rbind.SpatialPoints	
(SpatialPoints-class), 82	sample, 101
rbind.SpatialPointsDataFrame	select.spatial, 57
(SpatialPointsDataFrame-class),	set_col_regions (spplot), 94
84	set_11_TOL, 60
rbind.SpatialPolygons	set_ll_TOL (is.projected), 34
(SpatialPolygons-class), 87	set_ll_warn, 60
rbind.SpatialPolygonsDataFrame	set_ll_warn (is.projected), 34
(SpatialPolygonsDataFrame-class),	set_Polypath (SpatialPolygons-class), 87
89	set_PolypathRule  (SpatialPolygans_class) 87
read.asciigrid,54	<pre>(SpatialPolygons-class), 87 set_ReplCRS_warn(is.projected), 34</pre>
read0GR, 87, 89	setParUsrBB (Spatial-class), 60
recenter (recenter-methods), 55	show, CRS-method (CRS-class), 18
recenter,Line-method	show, DMS-method (DMS-class), 23
(recenter-methods), 55	show, GridTopology-method
recenter,Lines-method	(GridTopology-class), 30
(recenter-methods), 55	show, SpatialGrid-method
recenter, Polygon-method	(SpatialGrid-class), 62
(recenter-methods), 55	show, SpatialGridDataFrame-method
recenter, Polygons-method	(SpatialGridDataFrame-class),
(recenter-methods), 55	63
recenter,SpatialLines-method	show, Spatial MultiPoints-method
(recenter-methods), 55	(SpatialMultiPoints-class), 71
recenter, Spatial Polygons-method	show, Spatial MultiPoints Data Frame-method
(recenter-methods), 55	(SpatialMultiPointsDataFrame-class),
recenter-methods, 55	73
remove.duplicates(zerodist), 105	show, SpatialPixels-method
Rlogo, 56	(SpatialPixels-class), 77

show,SpatialPixelsDataFrame-method	SpatialPixels, 8, 27, 74
(SpatialPixelsDataFrame-class),	SpatialPixels-class, 77
79	SpatialPixelsDataFrame, 64, 78, 104, 105
show, SpatialPoints-method	SpatialPixelsDataFrame-class, 63, 78, 79
(SpatialPoints-class), 82	SpatialPoints, 79, 81, 105
show,SpatialPointsDataFrame-method	SpatialPoints-class, <i>57</i> , <i>75</i> , <i>77</i> , <i>78</i> , <i>81</i> , <i>82</i> ,
$({\sf SpatialPointsDataFrame-class}),$	82, 100
84	SpatialPointsDataFrame, 12, 79, 97
show, summary. GridTopology-method	<pre>SpatialPointsDataFrame (SpatialPoints),</pre>
(GridTopology-class), 30	81
showEPSG, 20	SpatialPointsDataFrame-class, 17, 29, 57,
ShowSpatialPointsDataFrame	79, 81, 82, 84
(SpatialPointsDataFrame-class),	SpatialPolygons, <i>8</i> , <i>22</i> , <i>86</i> , <i>87</i> , <i>89</i>
84	SpatialPolygons-class, 49, 51, 86, 87, 87,
sp, 58	90
sp-deprecated, 59	SpatialPolygonsDataFrame, 89
sp.grid (panel.spplot), 48	SpatialPolygonsDataFrame
sp.lines (panel.spplot), 48	(SpatialPolygons), 86
sp.panel.layout (panel.spplot), 48	SpatialPolygonsDataFrame-class, 53, 87,
sp.points (panel.spplot), 48	89
sp.polygons (panel.spplot), 48	SpatialPolygonsRescale (panel.spplot),
sp.text (panel.spplot), 48	48
sp. theme (spplot), 94	spChFIDs, 68, 69, 89, 90
Spatial, 5, 19, 40	spChFIDs (spChFIDs-methods), 91
Spatial (Spatial-class), 60	spChFIDs,SpatialLines,character-method
Spatial-class, 28, 35, 60, 67–69, 72, 78, 83,	(spChFIDs-methods), 91
87, 90, 95, 100	spChFIDs,SpatialLinesDataFrame,character-method
SpatialGrid, 27, 31, 63, 79	(spChFIDs-methods), 91
SpatialGrid (SpatialPixels), 74	spChFIDs,SpatialPolygons,character-method
SpatialGrid-class, 26, 62, 75, 76, 78, 79	(spChFIDs-methods), 91
SpatialGridDataFrame, 32, 54, 64, 104, 105	spChFIDs,SpatialPolygonsDataFrame,character-method
SpatialGridDataFrame (SpatialBivelaDataFrame) 78	(spChFIDs-methods), 91
(SpatialPixelsDataFrame), 78	spChFIDs-methods, 91
SpatialGridDataFrame-class, 26, 33, 54,	<pre>spChFIDs&lt;- (spChFIDs-methods), 91 spChFIDs&lt;-, Spatial-method</pre>
63, 75, 76, 78, 80 Spatiallines, 22, 66, 68	(spChFIDs-methods), 91
SpatialLines, 22, 66, 68 SpatialLines-class, 29, 36–38, 49, 66, 67,	spDists (spDistsN1), 92
69	spDistsN1, 92
	spmap.to.lev(stack), 103
SpatialLinesDataFrame, 69 SpatialLinesDataFrame (SpatialLines), 66	sppanel (spplot), 94
SpatialLinesDataFrame-class, 68	sppanel, character-method (spplot), 94
SpatialLinesLengths (SpatialLines), 66	sppanel, Line-method (spplot), 94
SpatialMultiPoints, 70	sppanel, Lines-method (spplot), 94
SpatialMultiPoints-class, 70, 71	sppanel, list-method (spplot), 94
SpatialMultiPointsDataFrame	sppanel, NULL-method (spplot), 94 sppanel, NULL-method (spplot), 94
(SpatialMultiPoints), 70	sppanel, SpatialGrid-method (spplot), 94
SpatialMultiPointsDataFrame-class, 70,	sppanel, SpatialLines-method (spplot), 94
73	sppanel, SpatialPixels-method (spplot), 94
15	opposition, opacitati incitio metiloa (oppioc),

94	(spTransform), 102
<pre>sppanel,SpatialPoints-method(spplot),</pre>	spTransform-methods (spTransform), 102
94	stack, 103, 103
sppanel, SpatialPolygons-method	stack.SpatialGridDataFrame(stack), 103
(spplot), 94	<pre>stack.SpatialPointsDataFrame(stack),</pre>
sppanel-methods (spplot), 94	103
spplot, 13, 49, 50, 94, 96, 103	subset.data.frame, 60
spplot,SpatialGridDataFrame-method	subset.Spatial(Spatial-class), 60
(spplot), 94	sum, 5
<pre>spplot,SpatialLinesDataFrame-method</pre>	summary,GridTopology-method
(spplot), 94	(GridTopology-class), 30
<pre>spplot,SpatialMultiPointsDataFrame-method</pre>	<pre>summary,Spatial-method(Spatial-class),</pre>
(spplot), 94	60
<pre>spplot,SpatialPixelsDataFrame-method</pre>	summary, SpatialGrid-method
(spplot), 94	(SpatialGrid-class), 62
<pre>spplot,SpatialPointsDataFrame-method</pre>	summary, SpatialLines-method
(spplot), 94	(SpatialLines-class), 67
spplot,SpatialPolygonsDataFrame-method	summary, Spatial MultiPoints-method
(spplot), 94	(SpatialMultiPoints-class), 71
spplot-methods, 50, 96	summary, SpatialPixels-method
spplot-methods (spplot), 94	(SpatialPixels-class), 77
spplot.grid(spplot), 94	summary, Spatial Points - method
spplot.key(panel.spplot),48	(SpatialPoints-class), 82
spplot.locator(spplot), 94	<pre>summary,SpatialPolygons-method     (SpatialPolygons-class),87</pre>
spplot.points(spplot), 94	summary.SpatialMultiPoints
spplot.polygons (spplot), 94	(SpatialMultiPoints-class), 71
spsample, $8,99$	summary.SpatialPoints
spsample, Line-method (spsample), 99	(SpatialPoints-class), 82
spsample, Lines-method (spsample), 99	surfaceArea, 104
spsample, Polygon-method (spsample), 99	surfaceArea, matrix-method
spsample, Polygons-method (spsample), 99	(surfaceArea), 104
spsample, Spatial-method (spsample), 99	surfaceArea, SpatialGridDataFrame-method
<pre>spsample,SpatialGrid-method(spsample),</pre>	(surfaceArea), 104
99	surfaceArea,SpatialPixelsDataFrame-method
spsample,SpatialLines-method	(surfaceArea), 104
(spsample), 99	surfaceArea.matrix(surfaceArea), 104
spsample, SpatialPixels-method	
(spsample), 99	tail.Spatial(Spatial-class), 60
spsample, Spatial Polygons-method	text, 28, 29
(spsample), 99	text(gridlines), 28
spsample-methods, 100	
spsample-methods (spsample), 99	write.asciigrid(read.asciigrid), 54
spTransform, 29, 102, 102	write.table,54
spTransform, Spatial, ANY-method	xyplot, 13, 95–98
(spTransform), 102	луртос, 13, 73-70
spTransform, Spatial, character-method	zerodist, 105
(spTransform), 102	zerodist2 (zerodist), 105
spTransform,Spatial,CRS-method	* **