

Package ‘terra’

August 7, 2022

Type Package

Title Spatial Data Analysis

Version 1.6-7

Date 2022-08-07

Depends R (>= 3.5.0)

Suggests parallel, tinytest, ncdf4, sf (>= 0.9-8), deldir, XML,
leaflet

LinkingTo Rcpp

Imports methods, Rcpp

SystemRequirements C++11, GDAL (>= 2.2.3), GEOS (>= 3.4.0), PROJ (>= 4.9.3), sqlite3

Encoding UTF-8

Maintainer Robert J. Hijmans <r.hijmans@gmail.com>

Description Methods for spatial data analysis with raster and vector data. Raster methods allow for low-level data manipulation as well as high-level global, local, zonal, and focal computation. The predict and interpolate methods facilitate the use of regression type (interpolation, machine learning) models for spatial prediction, including with satellite remote sensing data. Processing of very large files is supported. See the manual and tutorials on <<https://rspatial.org/terra/>> to get started. 'terra' is very similar to the 'raster' package; but 'terra' can do more, is easier to use, and it is faster.

License GPL (>= 3)

URL <https://rspatial.org/terra/>

BugReports <https://github.com/rspatial/terra/issues/>

LazyLoad yes

NeedsCompilation yes

Author Robert J. Hijmans [cre, aut] (<<https://orcid.org/0000-0001-5872-2872>>),
Roger Bivand [ctb] (<<https://orcid.org/0000-0003-2392-6140>>),
Karl Forner [ctb],
Jeroen Ooms [ctb] (<<https://orcid.org/0000-0002-4035-0289>>),
Edzer Pebesma [ctb] (<<https://orcid.org/0000-0001-8049-7069>>),
Michael D. Sumner [ctb]

Repository CRAN

Date/Publication 2022-08-07 15:20:02 UTC

R topics documented:

terra-package	6
activeCat	18
add	19
adjacent	20
aggregate	21
align	23
all.equal	24
animate	25
app	26
approximate	28
Arith-methods	30
as.character	31
as.data.frame	31
as.list	32
as.raster	33
as.spatvector	34
atan2	36
autocorrelation	37
barplot	38
boundaries	39
boxplot	40
buffer	41
c	42
cartogram	43
catalyze	44
cells	45
cellSize	46
centroids	48
clamp	49
classify	50
click	52
coerce	53
colors	54
combineGeoms	55
Compare-methods	57
compareGeom	57
concat	58
contour	59
convHull	60
costDistance	61
cover	62
crds	64

crop	65
crosstab	66
crs	67
deepcopy	69
densify	70
density	71
deprecated	71
depth	72
describe	72
diff	73
dimensions	74
direction	76
disagg	77
distance	78
dots	80
draw	81
erase	82
expanse	83
ext	85
extend	86
extract	87
extremes	90
factors	91
fillHoles	93
fillTime	94
flip	95
focal	96
focal3D	98
focalCor	99
focalCpp	100
focalMat	102
focalReg	103
focalValues	104
freq	105
gaps	106
gdal	106
geom	107
geomtype	109
global	110
gridDistance	111
head and tail	112
hist	113
ifel	114
image	115
impose	116
initialize	116
inplace	117
inset	119

intersect	121
is.bool	122
is.lonlat	123
lapp	124
layerCor	126
linearUnits	127
lines	128
makeTiles	129
makeVRT	130
mask	132
match	133
Math-methods	134
mem	135
merge	136
mergeTime	137
modal	138
mosaic	139
na.omit	140
NAflag	141
names	142
nearest	144
north	145
not.na	146
options	147
origin	148
pairs	149
patches	150
perim	151
persp	152
plet	153
plot	155
plotRGB	159
predict	161
project	164
quantile	166
query	167
rapp	169
rast	170
rasterize	173
rasterizeGeom	175
read and write	176
rectify	177
relate	178
rep	180
replace	181
resample	181
rescale	183
RGB	184

rotate	185
sapp	186
sbar	187
scale	188
scatterplot	189
sds	190
segregate	191
sel	192
selectHighest	193
selectRange	194
serialize	195
setValues	196
shade	197
sharedPaths	198
shift	199
simplifyGeom	200
sort	201
sources	201
SpatExtent-class	202
Spatial interpolation	203
SpatRaster-class	206
spatSample	207
SpatVector-class	209
spin	209
split	210
sprc	211
stretch	212
subset	213
subst	214
summarize	215
summary	218
svc	219
syndif	220
tapp	220
terrain	222
text	225
tighten	226
time	226
tmpFiles	228
topology	229
transpose	230
trim	231
union	231
unique	233
units	234
valid	235
values	236
vect	237

vector-attributes	240
vector_layers	241
voronoi	241
vrt	242
weighted.mean	243
where	244
which.lyr	245
width	245
window	246
wrap	247
writeCDF	248
writeRaster	249
writeVector	251
xmin	252
xyRowColCell	254
zonal	256
zoom	258

Index	259
--------------	------------

terra-package	<i>Description of the methods in the terra package</i>
---------------	--

Description

terra provides methods to manipulate geographic (spatial) data in "raster" and "vector" form. Raster data divide space into rectangular cells (pixels) and they are commonly used to represent spatially continuous phenomena, such as elevation or the weather. Satellite images also have this data structure. In contrast, "vector" spatial data (points, lines, polygons) are typically used to represent discrete spatial entities, such as a road, country, or bus stop.

The package implements two main classes (data types): `SpatRaster` and `SpatVector`. `SpatRaster` supports handling large raster files that cannot be loaded into memory; local, focal, zonal, and global raster operations; polygon, line and point to raster conversion; integration with modeling methods to make spatial predictions; and more. `SpatVector` supports all types of geometric operations such as intersections.

Additional classes include `SpatExtent`, which is used to define a spatial extent (bounding box); `SpatRasterDataset`, which represents a collection of sub-datasets for the same area. Each sub-dataset is a `SpatRaster` with possibly many layers, and may, for example, represent different weather variables; and `SpatRasterCollection` and `SpatVectorCollection` that are equivalent to lists of `SpatRaster` or `SpatVector` objects.

These classes hold a C++ pointer to the data "reference class" and that creates some limitations. They cannot be recovered from a saved R session either or directly passed to nodes on a computer cluster. Generally, you should use `writeRaster` to save `SpatRaster` objects to disk (and pass a filename or cell values to cluster nodes). Also see `wrap`.

The terra package is conceived as a replacement of the raster package. terra has a very similar, but simpler, interface, and it is faster than raster. At the bottom of this page there is a table that shows differences in the methods between the two packages.

Below is a list of some of the most important methods grouped by theme.

SpatRaster

I. Creating, combining and sub-setting

<code>rast</code>	Create a SpatRaster from scratch, file, or another object
<code>c</code>	Combine SpatRasters (multiple layers)
<code>add<-</code>	Add a SpatRaster to another one
<code>subset</code> or <code>[[</code> , or <code>\$</code>	Select layers of a SpatRaster
<code>selectRange</code>	Select cell values from different layers using an index layer

II. Changing the spatial extent or resolution

Also see the methods in section VIII

<code>merge</code>	Combine SpatRasters with different extents (but same origin and resolution)
<code>mosaic</code>	Combine SpatRasters with different extents using a function for overlapping cells
<code>crop</code>	Select a geographic subset of a SpatRaster
<code>extend</code>	Add rows and/or columns to a SpatRaster
<code>trim</code>	Trim a SpatRaster by removing exterior rows and/or columns that only have NAs
<code>aggregate</code>	Combine cells of a SpatRaster to create larger cells
<code>disagg</code>	Subdivide cells
<code>resample</code>	Resample (warp) values to a SpatRaster with a different origin and/or resolution
<code>project</code>	Project (warp) values to a SpatRaster with a different coordinate reference system
<code>shift</code>	Adjust the location of SpatRaster
<code>flip</code>	Flip values horizontally or vertically
<code>rotate</code>	Rotate values around the date-line (for lon/lat data)
<code>t</code>	Transpose a SpatRaster

III. Local (cell based) methods

Apply-like methods:

<code>app</code>	Apply a function to all cells, across layers, typically to summarize (as in <code>base::apply</code>)
<code>tapp</code>	Apply a function to groups of layers (as in <code>base::tapply</code> and <code>stats::aggregate</code>)
<code>lapp</code>	Apply a function to using the layers of a SpatRaster as variables
<code>sapp</code>	Apply a function to each layer
<code>rapp</code>	Apply a function to a spatially variable range of layers

Arithmetic, logical, and standard math methods:

Arith-methods	Standard arithmetic methods (+, -, *, ^, %, %/, /)
Compare-methods	Comparison methods for SpatRaster (==, !=, >, <, <=, >=)
Logic-methods	Boolean methods (!, &,)
Math-methods	abs, sign, sqrt, ceiling, floor, trunc, cummax, cummin, cumprod, cumsum, log, log10, log2, log1p, acos, acosh, asin, asinh, atan, atanh, exp, expm1, cos, cosh, sin, sinh, tan, tanh, round, signif
Summary-methods	mean, max, min, median, sum, range, prod, any, all, stdev, which.min, which.max
as.bool	create a Boolean (logical) SpatRaster
as.int	create an integer (whole numbers) SpatRaster

Other methods:

approximate	Compute missing values for cells by interpolation across layers
cellSize	Compute the area of cells
classify	(Re-)classify values
subst	Substitute (replace) cell values
cover	First layer covers second layer except where the first layer is NA
init	Initialize cells with new values
mask	Replace values in a SpatRaster based on values in another SpatRaster
which.lyr	which is the first layer that is TRUE?
segregate	Make a 0/1 layer for each unique value

IV. Zonal and global methods

expand	Compute the summed area of cells
crossstab	Cross-tabulate two SpatRasters
freq	Frequency table of SpatRaster cell values
global	Summarize SpatRaster cell values with a function
quantile	Quantiles
layerCor	Correlation between layers
stretch	Stretch values
scale	Scale values
summary	Summary of the values of a SpatRaster (quantiles and mean)
unique	Get the unique values in a SpatRaster
zonal	Summarize a SpatRaster by zones in another SpatRaster

V. Situation (spatial context) based methods

<code>adjacent</code>	Identify cells that are adjacent to a set of cells of a <code>SpatRaster</code>
<code>boundaries</code>	Detection of boundaries (edges)
<code>distance</code>	Shortest distance to a cell that is not NA or to or from a vector object
<code>gridDistance</code>	Shortest distance through adjacent grid cells
<code>costDistance</code>	Shortest distance considering cell-varying friction
<code>direction</code>	Direction (azimuth) to or from cells that are not NA
<code>focal</code>	Focal (neighborhood; moving window) functions
<code>focalCpp</code>	Faster focal by using custom C++ functions
<code>focalReg</code>	Regression between layers for focal areas
<code>focalCor</code>	Correlation between layers for focal areas
<code>patches</code>	Find patches (clumps)
<code>terrain</code>	Compute slope, aspect and other terrain characteristics from elevation data
<code>shade</code>	Compute hill shade from slope and aspect layers
<code>autocor</code>	Compute global or local spatial autocorrelation

VI. Model predictions

<code>predict</code>	Predict a non-spatial model to a <code>SpatRaster</code>
<code>interpolate</code>	Predict a spatial model to a <code>SpatRaster</code>

VII. Accessing cell values

Apart from the function listed below, you can also use indexing with `[` with cell numbers, and row and/or column numbers

<code>values</code>	cell values (fails with very large rasters)
<code>values<-</code>	Set new values to the cells of a <code>SpatRaster</code>
<code>setValues</code>	Set new values to the cells of a <code>SpatRaster</code>
<code>as.matrix</code>	Get cell values as a matrix
<code>as.array</code>	Get cell values as an array
<code>extract</code>	Extract cell values from a <code>SpatRaster</code> (e.g., by cell, coordinates, polygon)
<code>spatSample</code>	Regular or random sample
<code>minmax</code>	Get the minimum and maximum value of the cells of a <code>SpatRaster</code> (if known)
<code>setMinMax</code>	Compute the minimum and maximum value of a <code>SpatRaster</code> if these are not known
<code>extract</code>	spatial queries of a <code>SpatRaster</code> with a <code>SpatVector</code>

VIII. Getting and setting dimensions

Get or set basic parameters of SpatRasters. If there are values associated with a SpatRaster object (either in memory or via a link to a file) these are lost when you change the number of columns or rows or the resolution. This is not the case when the extent is changed (as the number of columns and rows will not be affected). Similarly, with `crs` you can set the coordinate reference system, but this does not transform the data (see [project](#) for that).

<code>ncol</code>	The number of columns
<code>nrow</code>	The number of rows
<code>ncell</code>	The number of cells (can not be set directly, only via <code>ncol</code> or <code>nrow</code>)
<code>res</code>	The resolution (x and y)
<code>nlyr</code>	Get or set the number of layers
<code>names</code>	Get or set the layer names
<code>xres</code>	The x resolution (can be set with <code>res</code>)
<code>yres</code>	The y resolution (can be set with <code>res</code>)
<code>xmin</code>	The minimum x coordinate (or longitude)
<code>xmax</code>	The maximum x coordinate (or longitude)
<code>ymin</code>	The minimum y coordinate (or latitude)
<code>ymax</code>	The maximum y coordinate (or latitude)
<code>ext</code>	Get or set the extent (minimum and maximum x and y coordinates ("bounding box"))
<code>origin</code>	The origin of a SpatRaster
<code>crs</code>	The coordinate reference system (map projection)
<code>is.lonlat</code>	Test if an object has (or may have) a longitude/latitude coordinate reference system
<code>sources</code>	Get the filename(s) to which a SpatRaster is linked
<code>inMemory</code>	Are the data sources in memory (or on disk)?
<code>compareGeom</code>	Compare the geometry of SpatRasters
<code>NAflag</code>	Set the NA value (for reading from a file with insufficient metadata)

IX. Computing row, column, cell numbers and coordinates

Cell numbers start at 1 in the upper-left corner. They increase within rows, from left to right, and then row by row from top to bottom. Likewise, row numbers start at 1 at the top of the raster, and column numbers start at 1 at the left side of the raster.

<code>xFromCol</code>	x-coordinates from column numbers
<code>yFromRow</code>	y-coordinates from row numbers
<code>xFromCell</code>	x-coordinates from row numbers
<code>yFromCell</code>	y-coordinates from cell numbers
<code>xyFromCell</code>	x and y coordinates from cell numbers
<code>colFromX</code>	Column numbers from x-coordinates (or longitude)
<code>rowFromY</code>	Row numbers from y-coordinates (or latitude)
<code>rowColFromCell</code>	Row and column numbers from cell numbers
<code>cellFromXY</code>	Cell numbers from x and y coordinates

<code>cellFromRowCol</code>	Cell numbers from row and column numbers
<code>cellFromRowColCombine</code>	Cell numbers from all combinations of row and column numbers
<code>cells</code>	Cell numbers from an <code>SpatVector</code> or <code>SpatExtent</code>

X. Time related methods

<code>time</code>	Get or set time
<code>fillTime</code>	can add empty layers in between existing layers to assure that the time step between layers is constant
<code>mergeTime</code>	combine multiple rasters, perhaps partly overlapping in time, into a single time series

XI. Methods for categorical rasters

<code>is.factor</code>	Are there categorical layers?
<code>levels</code>	Get active categories, or set categories
<code>activeCat</code>	Get or set the active category
<code>cats</code>	Get categories (active and inactive)
<code>set.cats</code>	Set categories in place
<code>concats</code>	Combine <code>SpatRasters</code> with different categories
<code>catalyze</code>	Create a layer for each category
<code>as.numeric</code>	use the active category to create a non-categorical <code>SpatRaster</code>
<code>as.factor</code>	Make the layers of a <code>SpatRaster</code> categorical

XII. Writing `SpatRaster` files

Basic:

<code>writeRaster</code>	Write all values of <code>SpatRaster</code> to disk. You can set the filetype, datatype, compression.
<code>writeCDF</code>	Write <code>SpatRaster</code> data to a netCDF file

Advanced:

<code>readStart</code>	Open file connections for efficient multi-chunk reading
<code>readStop</code>	Close file connections
<code>writeStart</code>	Open a file for writing
<code>writeValues</code>	Write some values

<code>writeStop</code>	Close the file after writing
------------------------	------------------------------

XIII. Miscellaneous SpatRaster methods

<code>terraOptions</code>	Show, set, or get session options, mostly to control memory use and to set write options
<code>sources</code>	Show the data sources of a SpatRaster
<code>tmpFiles</code>	Show or remove temporary files
<code>mem_info</code>	memory needs and availability
<code>inMemory</code>	Are the cell values in memory?

SpatRasterDataSet

XIV. SpatRasterDataset

A SpatRasterDataset contains SpatRaster objects that are sub-datasets for the same area. They all have the same extent and resolution.

<code>sds</code>	Create a SpatRasterDataset from a file with subdatasets (ncdf or hdf) or from SpatRaster objects
<code>[or \$</code>	Extract a SpatRaster
<code>names</code>	Get the names of the sub-datasets

SpatVector

XV. Create SpatVector objects

<code>vect</code>	Create a SpatVector from a file (for example a "shapefile") or from another object
<code>vector_layers</code>	list or delete layers in a vector database such as GPKG
<code>rbind</code>	append SpatVectors of the same geometry type
<code>unique</code>	remove duplicates
<code>na.omit</code>	remove empty geometries and/or fields that are NA
<code>project</code>	Project a SpatVector to a different coordinate reference system
<code>writeVector</code>	Write SpatVector data to disk
<code>centroids</code>	Get the centroids of a SpatVector
<code>voronoi</code>	Voronoi diagram
<code>del aunay</code>	Delaunay triangles

<code>convHull</code>	Compute the convex hull of a <code>SpatVector</code>
<code>fillHoles</code>	Remove or extract holes from polygons

XVI. Properties of `SpatVector` objects

<code>geom</code>	returns the geometries as matrix or WKT
<code>crds</code>	returns the coordinates as a matrix
<code>linearUnits</code>	returns the linear units of the crs (in meter)
<code>ncol</code>	The number of columns (of the attributes)
<code>nrow</code>	The number of rows (of the geometries and attributes)
<code>names</code>	Get or set the layer names
<code>ext</code>	Get the extent (minimum and maximum x and y coordinates ("bounding box"))
<code>crs</code>	The coordinate reference system (map projection)
<code>is.lonlat</code>	Test if an object has (or may have) a longitude/latitude coordinate reference system

XVII. Geometric queries

<code>adjacent</code>	find adjacent polygons
<code>expanse</code>	computes the area covered by polygons
<code>nearby</code>	find nearby geometries
<code>nearest</code>	find the nearest geometries
<code>relate</code>	geometric relationships such as "intersects", "overlaps", and "touches"
<code>perim</code>	computes the length of the perimeter of polygons, and the length of lines

XVIII. Geometric operations

<code>erase</code> or "-"	erase (parts of) geometries
<code>intersect</code> or "*"	intersect geometries
<code>union</code> or "+"	Merge geometries
<code>cover</code>	update polygons
<code>symdif</code>	symmetrical difference of two polygons
<code>aggregate</code>	dissolve smaller polygons into larger ones
<code>buffer</code>	buffer geometries
<code>disagg</code>	split multi-geometries into separate geometries

<code>crop</code>	clip geometries using a rectangle (<code>SpatExtent</code>) or <code>SpatVector</code>
-------------------	--

XIX. SpatVector attributes

We use the term "attributes" for the tabular data (`data.frame`) associated with vector geometries.

<code>extract</code>	spatial queries between <code>SpatVector</code> and <code>SpatVector</code> (e.g. point in polygons)
<code>sel</code>	select - interactively select geometries
<code>click</code>	identify attributes by clicking on a map
<code>merge</code>	Join a table with a <code>SpatVector</code>
<code>as.data.frame</code>	get attributes as a <code>data.frame</code>
<code>as.list</code>	get attributes as a list
<code>values</code>	Get the attributes of a <code>SpatVector</code>
<code>values<-</code>	Set new attributes to the geometries of a <code>SpatRaster</code>

XX. Change geometries (for display, experimentation)

<code>shift</code>	change the position geometries by shifting their coordinates in horizontal and/or vertical direction
<code>spin</code>	rotate geometries around an origin
<code>rescale</code>	shrink (or expand) geometries, for example to make an inset map
<code>flip</code>	flip geometries vertically or horizontally
<code>t</code>	transpose geometries (switch x and y)

XXI. Geometry properties and topology

<code>width</code>	the minimum diameter of the geometries
<code>clearance</code>	the minimum clearance of the geometries
<code>sharedPaths</code>	shared paths (arcs) between line or polygon geometries
<code>simplifyGeom</code>	simplify geometries
<code>gaps</code>	find gaps between polygon geometries
<code>fillHoles</code>	get or remove the polygon holes
<code>makeNodes</code>	create nodes on lines
<code>mergeLines</code>	connect lines to form polygons
<code>removeDupNodes</code>	remove duplicate nodes in geometries and optionally rounds the coordinates
<code>is.valid</code>	check if geometries are valid
<code>makeValid</code>	attempt to repair invalid geometries
<code>snap</code>	make boundaries of geometries identical if they are very close to each other

<code>erase</code> (single argument)	remove parts of geometries that overlap
<code>union</code> (single argument)	create new polygons such that there are no overlapping polygons
<code>combineGeoms</code>	combine geometries that overlap, share a border, or are within a minimum distance of each other

Spat* Collections

XXII. Collections

A `SpatRasterCollection` is a vector of `SpatRaster` objects. Unlike for a `SpatRasterDataset`, there the extent and resolution of the `SpatRasters` do not need to match each other. A `SpatVectorCollection` is a vector of `SpatVector` objects.

<code>svc</code>	create a <code>SpatRasterCollection</code> from a set of <code>SpatRaster</code> objects
<code>length</code>	how many <code>SpatRasters</code> does the <code>SpatRasterCollection</code> have?
<code>[</code>	extract a <code>SpatRaster</code>

SpatExtent

XXIII. SpatExtent

<code>ext</code>	Create a <code>SpatExtent</code> object. For example to <code>crop</code> a Spatial dataset
<code>intersect</code>	Intersect two <code>SpatExtent</code> objects, same as <code>-</code>
<code>union</code>	Combine two <code>SpatExtent</code> objects, same as <code>+</code>
<code>Math-methods</code>	round/floor/ceiling of a <code>SpatExtent</code>
<code>align</code>	Align a <code>SpatExtent</code> with a <code>SpatRaster</code>
<code>draw</code>	Create a <code>SpatExtent</code> by drawing it on top of a map (plot)

General methods

XXIV. Conversion between spatial data objects from different packages

You can coerce `SpatRasters` to `Raster*` objects, after loading the raster package, with `as(object, "Raster")`, or `raster(object)` or `brick(object)` or `stack(object)`

<code>rast</code>	SpatRaster from matrix and other objects
<code>vect</code>	SpatVector from sf or Spatial* vector data
<code>sf::st_as_sf</code>	sf object from SpatVector
<code>rasterize</code>	Rasterizing points, lines or polygons
<code>rasterizeGeom</code>	Rasterize attributes of geometries such as "count", "area", or "length"
<code>as.points</code>	Create points from a SpatRaster or SpatVector
<code>as.lines</code>	Create points from a SpatRaster or SpatVector
<code>as.polygons</code>	Create polygons from a SpatRaster
<code>as.contour</code>	Contour lines from a SpatRaster

XXV. Plotting

Maps:

<code>plot</code>	Plot a SpatRaster or SpatVector. The main method to create a map
<code>points</code>	Add points to a map
<code>lines</code>	Add lines to a map
<code>polys</code>	Add polygons to a map
<code>text</code>	Add text (such as the values of a SpatRaster or SpatVector) to a map
<code>image</code>	Alternative to plot to make a map with a SpatRaster
<code>plotRGB</code>	Combine three layers (red, green, blue channels) into a single "real color" plot
<code>sbar</code>	Add a scalebar to a map
<code>north</code>	Add a north arrow to a map
<code>inset</code>	Add a small inset (overview) map
<code>dots</code>	Make a dot-density map
<code>cartogram</code>	Make a cartogram
<code>persp</code>	Perspective plot of a SpatRaster
<code>contour</code>	Contour plot or filled-contour plot of a SpatRaster
<code>colorize</code>	Combine three layers (red, green, blue channels) into a single layer with a color-table

Interacting with a map:

<code>zoom</code>	Zoom in to a part of a map by drawing a bounding box on it
<code>click</code>	Query values of SpatRaster or SpatVector by clicking on a map
<code>sel</code>	Select a spatial subset of a SpatRaster or SpatVector by drawing on a map
<code>draw</code>	Create a SpatExtent or SpatVector by drawing on a map

Other plots:

<code>plot</code>	x-y scatter plot of the values of (a sample of) the layers of two SpatRaster objects
<code>hist</code>	Histogram of SpatRaster values
<code>barplot</code>	Bar plot of a SpatRaster
<code>density</code>	Density plot of SpatRaster values

pairs	Pairs plot for layers in a SpatRaster
boxplot	Box plot of the values of a SpatRaster

Comparison with the raster package

XXVI. New method names

terra has a single class SpatRaster for which raster has three (RasterLayer, RasterStack, RasterBrick). Likewise there is a single class for vector data SpatVector that replaces six Spatial* classes. Most method names are the same, but note the following important differences in methods names with the raster package

raster package	terra package
raster, brick, stack	rast
rasterFromXYZ	rast(, type="xyz")
stack, addLayer	c
addLayer	add<-
area	cellSize or expand
approxNA	approximate
calc	app
cellFromLine, cellFromPolygon,	cells
cellsFromExtent	cells
cellStats	global
corLocal	focalCor
coordinates	crds
clump	patches
compareRaster	compareGeom
disaggregate	disagg
drawExtent, drawPoly, drawLine	draw
dropLayer	subset
extent	ext
distanceFromPoints	distance
isLonLat, isGlobalLonLat	is.lonlat
couldBeLonLat	is.lonlat
layerize	segregate
layerStats	layerCor
NAvalue	NAflag
nlayers	nlyr
overlay	lapp
projectRaster	project
rasterToPoints	as.points
rasterToPolygons	as.polygons
reclassify, subs, cut	classify
sampleRandom, sampleRegular	spatSample
shapefile	vect

stackApply	tapp
stackSelect	selectRange

XXVII. Changed behavior

Also note that even if function names are the same in terra and raster, their output can be different. In most cases this was done to get more consistency in the returned values (and thus fewer errors in the downstream code that uses them). In other cases it simply seemed better. Here are some examples:

<p>as.polygons quantile extract</p>	<p>By default, terra returns dissolved polygons computes by cell, across layers instead of the other way around</p>
<p>values Summary-methods</p>	<p>By default, terra returns a matrix, with the first column the sequential ID of the vectors. raster returns a list (for lines or polygons) or a matrix (for points, but without the ID column). You can use list=TRUE to get the results as a list terra always returns a matrix. raster returns a vector for a RasterLayer With raster, mean(x, y) and mean(stack(x, y)) return the same result, a single layer with the mean of all cell values. This is also what terra returns with mean(c(x, y)), but with mean(x, y) the parallel mean is returned – that is, the computation is done layer-wise, and the number of layers in the output is the same as that of x and y (or the larger of the two if they are not the same). This affects all summary functions (sum, mean, median, which.min, which.max, min, max, prod, any, all, stdev), except range, which is not implemented for this case (you can use min and max instead)</p>

Authors

Except where indicated otherwise, the methods and functions in this package were written by Robert Hijmans. The configuration scripts were written by Roger Bivand. Some of the C++ code for GDAL/GEOS was adapted from code by Edzer Pebesma for sf. The progress bar code is by Karl Forner (RcppProgress). Jeroen Ooms provided the compiled GDAL and GEOS libraries for installation on windows. Michael Sumner contributed various bits and pieces.

Acknowledgments

This package is an attempt to climb on the shoulders of giants (GDAL, PROJ, GEOS, NCDF, GeographicLib, Rcpp, R). Many people have contributed by asking questions or **raising issues**. Feedback and suggestions by Márcia Barbosa, Kendon Bell, Andrew Gene Brown, Jean-Luc Dupouey, Krzysztof Dyba, Alex Ilich, Gerald Nelson, Jakub Nowosad and Monika Tomaszewska have been especially helpful.

Description

Get or set the active category of a multi-categorical SpatRaster layer

Usage

```
## S4 method for signature 'SpatRaster'
activeCat(x, layer=1)
## S4 replacement method for signature 'SpatRaster'
activeCat(x, layer=1)<-value
```

Arguments

x	SpatRaster
layer	positive integer, the layer number or name
value	a data.frame (ID, category) or vector with category names

Value

integer

See Also

[catalyze](#), [cats](#)

Examples

```
set.seed(0)
r <- rast(nrows=10, ncols=10)
values(r) <- sample(3, ncell(r), replace=TRUE) + 10
d <- data.frame(id=11:13, cover=c("forest", "water", "urban"), letters=letters[1:3], value=10:12)
levels(r) <- d

activeCat(r)
activeCat(r) <- 3
activeCat(r)
```

add

Add (in place) a SpatRaster to another SpatRaster object

Description

Add (in place) a SpatRaster to another SpatRaster object. Comparable with [c](#), but without copying the object.

Usage

```
## S4 replacement method for signature 'SpatRaster,SpatRaster'
add(x)<-value
```

Arguments

x	SpatRaster
value	SpatRaster

Value

SpatRaster

See Also

[c](#)

Examples

```
r <- rast(nrows=5, ncols=9, vals=1:45)
x <- c(r, r*2)
add(x) <- r*3
x
```

adjacent

Adjacent cells

Description

Identify cells that are adjacent to a set of raster cells. Or identify adjacent polygons

Usage

```
## S4 method for signature 'SpatRaster'
adjacent(x, cells, directions="rook", pairs=FALSE, include=FALSE)
```

```
## S4 method for signature 'SpatVector'
adjacent(x, type="rook", pairs=TRUE, symmetrical=FALSE)
```

Arguments

x	SpatRaster
cells	vector of cell numbers for which adjacent cells should be found. Cell numbers start with 1 in the upper-left corner and increase from left to right and from top to bottom
directions	character or matrix to indicated the directions in which cells are considered connected. The following character values are allowed: "rook" or "4" for the horizontal and vertical neighbors; "bishop" to get the diagonal neighbors; "queen" or "8" to get the vertical, horizontal and diagonal neighbors; or "16" for knight and one-cell queen move neighbors. If directions is a matrix it should have odd dimensions and have logical (or 0, 1) values

<code>pairs</code>	logical. If TRUE, a two-column matrix of pairs of adjacent cells is returned. If <code>x</code> is a <code>SpatRaster</code> and <code>pairs</code> is FALSE, an <code>n*m</code> matrix is returned where the number of rows <code>n</code> is <code>length(cells)</code> and the number of columns <code>m</code> is the number of neighbors requested with <code>directions</code>
<code>include</code>	logical. Should the focal cells be included in the result? They are always included if <code>pairs=TRUE</code>
<code>type</code>	character. One of "rook", "queen", "touches", or "intersects". "queen" and "touches" are synonyms. "rook" exclude polygons that touch at a single node only. "intersects" includes polygons that touch or overlap
<code>symmetrical</code>	logical. If TRUE, an adjacent pair is only included once. For example, if polygon 1 is adjacent to polygon 3, the implied adjacency between 3 and 1 is not reported

Value

matrix

See Also

[relate](#), [nearby](#)

Examples

```
r <- rast(nrows=10, ncols=10)
adjacent(r, cells=c(1, 5, 55), directions="queen")
r <- rast(nrows=10, ncols=10, crs="+proj=utm +zone=1 +datum=WGS84")
adjacent(r, cells=11, directions="rook")

#same as
rk <- matrix(c(0,1,0,1,0,1,0,1,0), 3, 3)
adjacent(r, cells=11, directions=rk)

## note that with global lat/lon data the E and W connect
r <- rast(nrows=10, ncols=10, crs="+proj=longlat +datum=WGS84")
adjacent(r, cells=11, directions="rook")

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
a <- adjacent(v, symmetrical=TRUE)
head(a)
```

aggregate

Aggregate raster or vector data

Description

Aggregate a `SpatRaster` to create a new `SpatRaster` with a lower resolution (larger cells). Aggregation groups rectangular areas to create larger cells. The value for the resulting cells is computed with a user-specified function.

Or aggregate ("dissolve") a `SpatVector`.

Usage

```
## S4 method for signature 'SpatRaster'
aggregate(x, fact=2, fun="mean", ..., cores=1, filename="", overwrite=FALSE, wopt=list())

## S4 method for signature 'SpatVector'
aggregate(x, by=NULL, dissolve=TRUE, fun="mean", count=TRUE, ...)
```

Arguments

x	SpatRaster
fact	positive integer. Aggregation factor expressed as number of cells in each direction (horizontally and vertically). Or two integers (horizontal and vertical aggregation factor) or three integers (when also aggregating over layers)
fun	function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum" and "modal"
...	additional arguments passed to fun, such as na.rm=TRUE
cores	positive integer. If cores > 1, a 'parallel' package cluster with that many cores is created. Ignored for C++ level implemented functions "mean", "max", "min", "median", "sum" and "modal"
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in writeRaster
by	character. The variable used to aggregate the geometries
dissolve	logical. Should borders between aggregated geometries be dissolved?
count	logical. If TRUE and by is not NULL, a variable "agg_n" is included that shows the number of input geometries for each output geometry

Details

Aggregation starts at the upper-left end of a SpatRaster. If a division of the number of columns or rows with factor does not return an integer, the extent of the resulting SpatRaster will be somewhat larger than that of the original SpatRaster. For example, if an input SpatRaster has 100 columns, and fact=12, the output SpatRaster will have 9 columns and the maximum x coordinate of the output SpatRaster is also adjusted.

The function fun should take multiple numbers, and return a single number. For example mean, modal, min or max.

It should also accept a na.rm argument (or ignore it as one of the 'dots' arguments).

Value

SpatRaster

See Also

[disagg](#) to disaggregate

Examples

```

r <- rast()
# aggregated SpatRaster, no values
ra <- aggregate(r, fact=10)

values(r) <- runif(ncell(r))
# aggregated raster, max of the values
ra <- aggregate(r, fact=10, fun=max)

# multiple layers
s <- c(r, r*2)
x <- aggregate(s, 20)

## SpatVector
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
va <- aggregate(v, "ID_1")

plot(va, "NAME_1", lwd=5, plg=list(x="topright"), mar=rep(2,4))
lines(v, lwd=3, col="light gray")
lines(va)
text(v, "ID_1", halo=TRUE)

```

align

*Align a SpatExtent***Description**

Align an SpatExtent with a SpatRaster. This can be useful to create a new SpatRaster with the same origin and resolution as an existing SpatRaster. Do not use this to force data to match that really does not match (use e.g. [resample](#) or (dis)aggregate for this).

It is also possible to align a SpatExtent to a clean divisor.

Usage

```

## S4 method for signature 'SpatExtent,SpatRaster'
align(x, y, snap="near")

## S4 method for signature 'SpatExtent,numeric'
align(x, y)

```

Arguments

x	SpatExtent
y	SpatRaster or numeric
snap	Character. One of "near", "in", or "out", to determine in which direction the extent should be aligned. To the nearest border, inwards or outwards

Value

SpatExtent

See Also[ext](#), [draw](#)**Examples**

```

r <- rast()
e <- ext(-10.1, 9.9, -20.1, 19.9)
ea <- align(e, r)
e
ext(r)
ea

align(e, 0.5)

```

all.equal

*Compare two SpatRasters for equality***Description**

Compare two SpatRasters for (near) equality.

First the attributes of the objects are compared. If these are the same, a (perhaps small) sample of the raster cells is compared as well.

The sample size used can be increased with the `maxcell` argument. You can set it to `Inf`, but for large rasters your computer may not have sufficient memory. See the examples for a safe way to compare all values.

Usage

```

## S4 method for signature 'SpatRaster,SpatRaster'
all.equal(target, current, maxcell=10000, ...)

```

Arguments

<code>target</code>	SpatRaster
<code>current</code>	SpatRaster
<code>maxcell</code>	positive integer. The size of the regular sample used to compare cell values
<code>...</code>	additional arguments passed to <code>all.equal.numeric</code> to compare cell values

Value

Either TRUE or a character vector describing the differences between target and current.

See Also[compareGeom](#)**Examples**

```
x <- sqrt(1:100)
mat <- matrix(x, 10, 10)
r1 <- rast(nrows=10, ncols=10, xmin=0, vals = x)
r2 <- rast(nrows=10, ncols=10, xmin=0, vals = mat)

all.equal(r1, r2)
all.equal(r1, r1*1)
all.equal(rast(r1), rast(r2))

# compare geometries
compareGeom(r1, r2)

# Compare all cell values for near equality
# as floating point number imprecision can be a problem
m <- minmax(r1 - r2)
all(abs(m) < 1e-7)

# comparison of cell values to create new SpatRaster
e <- r1 == r2
```

animate

*Animate a SpatRaster***Description**

Animate (sequentially plot) the layers of a SpatRaster to create a movie.

This does not work with R-Studio.

Usage

```
## S4 method for signature 'SpatRaster'
animate(x, pause=0.25, main, range, maxcell=50000, n=1, ...)
```

Arguments

x	SpatRaster
pause	numeric. How long should be the pause be between layers?
main	title for each layer. If not supplied the z-value is used if available. Otherwise the names are used.
range	numeric vector of length 2. Range of values to plot
maxcell	integer > 0. Maximum number of cells to use for the plot. If maxcell < ncell(x), spatSample(type="regular") is used before plotting
n	integer > 0. Number of loops
...	Additional arguments passed to plot

Value

None

See Also[plot](#)**Examples**

```
s <- rast(system.file("ex/logo.tif", package="terra"))
animate(s, n=1)
```

app

*Apply a function to the cells of a SpatRaster***Description**

Apply a function to the values of each cell of a SpatRaster. Similar to [apply](#) – think of each layer in a SpatRaster as a column (or row) in a matrix.

This is generally used to summarize the values of multiple layers into one layer; but this is not required.

app calls function fun with the raster data as first argument. Depending on the function supplied, the raster data is represented as either a matrix in which each layer is a column, or a vector representing a cell. The function should return a vector or matrix that is divisible by ncell(x). Thus, both "sum" and "rowSums" can be used, but "colSums" cannot be used.

You can also apply a function fun across datasets by layer of a SpatRasterDataset. In that case, summarization is across SpatRasters, not across layers.

Usage

```
## S4 method for signature 'SpatRaster'
app(x, fun, ..., cores=1, filename="", overwrite=FALSE, wopt=list())

## S4 method for signature 'SpatRasterDataset'
app(x, fun, ..., cores=1, filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster or SpatRasterDataset
fun	a function that operates on a vector or matrix. This can be a function that is defined in base-R or in a package, or a function you write yourself (see examples). Functions that return complex output (e.g. a list) may need to be wrapped in your own function to simplify the output to a vector or matrix. The following functions have been re-implemented in C++ for speed: "sum", "mean", "median", "modal", "which", "which.min", "which.max", "min", "max", "prod", "any", "all", "sd", "std", "first". To use the base-R function for say, "min", you could use something like fun=function(i) min(i) or the equivalent fun = \ (i) min(i)

...	additional arguments for fun. These are typically numerical constants. They should <i>never</i> be another SpatRaster
cores	positive integer. If cores > 1, a 'parallel' package cluster with that many cores is created and used. You can also supply a cluster object. Ignored for functions that are implemented by terra in C++ (see under fun)
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in writeRaster

Details

To speed things up, parallelization is supported, but this is often not helpful, and it may actually be slower. There is only a speed gain if you have many cores (> 8) and/or a very complex (slow) function fun. If you write fun yourself, consider supplying a `cppFunction` made with the `Rcpp` package instead (or go have a cup of tea while the computer works for you).

Value

SpatRaster

See Also

[lapp](#), [tapp](#), [Math-methods](#)

Examples

```
r <- rast(ncols=10, nrows=10)
values(r) <- 1:ncell(r)
x <- c(r, sqrt(r), r+50)
s <- app(x, fun=sum)
s
# for a few generic functions like
# "sum", "mean", and "max" you can also do
sum(x)

## SpatRasterDataset
sd <- sds(x, x*2, x/3)
a <- app(sd, max)
a
# same as
max(x, x*2, x/3)

## also works for a single layer
f <- function(i) (i+1) * 2 * i + sqrt(i)
s <- app(r, f)
# same as above, but that is not memory-safe
# and has no filename argument
s <- f(r)

## Not run:
```

```
#### multiple cores
test0 <- app(x, sqrt)
test1 <- app(x, sqrt, cores=2)

testfun <- function(i) { 2 * sqrt(i) }
test2 <- app(x, fun=testfun, cores =2)

## this fails because testfun is not exported to the nodes
# test3 <- app(x, fun=function(i) testfun(i), cores=2)
## to export it, add it as argument to fun
test3 <- app(x, fun=function(i, ff) ff(i), cores =3, ff=testfun)

## End(Not run)
```

approximate	<i>Estimate values for cell values that are NA by interpolating between layers</i>
-------------	--

Description

approximate uses the stats function [approx](#) to estimate values for cells that are NA by interpolation across layers. Layers are considered equidistant, unless argument `z` is used, or `time(x)` returns values that are not NA, in which case these values are used to determine distance between layers.

For estimation based on neighboring cells see [focal](#)

Usage

```
## S4 method for signature 'SpatRaster'
approximate(x, method="linear", yleft, yright,
            rule=1, f=0, ties=mean, z=NULL, NArule=1,filename="", ...)
```

Arguments

<code>x</code>	SpatRaster
<code>method</code>	specifies the interpolation method to be used. Choices are "linear" or "constant" (step function; see the example in approx)
<code>yleft</code>	the value to be returned before a non-NA value is encountered. The default is defined by the value of <code>rule</code> given below
<code>yright</code>	the value to be returned after the last non-NA value is encountered. The default is defined by the value of <code>rule</code> given below
<code>rule</code>	an integer (of length 1 or 2) describing how interpolation is to take place at for the first and last cells (before or after any non-NA values are encountered). If rule is 1 then NAs are returned for such points and if it is 2, the value at the closest data extreme is used. Use, e.g., <code>rule = 2:1</code> , if the left and right side extrapolation should differ

<code>f</code>	for <code>method = "constant"</code> a number between 0 and 1 inclusive, indicating a compromise between left- and right-continuous step functions. If <code>y0</code> and <code>y1</code> are the values to the left and right of the point then the value is $y_0*(1-f)+y_1*f$ so that <code>f = 0</code> is right-continuous and <code>f = 1</code> is left-continuous
<code>ties</code>	Handling of tied 'z' values. Either a function with a single vector argument returning a single number result or the string "ordered"
<code>z</code>	numeric vector to indicate the distance between layers (e.g., depth). The default is <code>time(x)</code> if these are not NA or else <code>1:nlys(x)</code>
<code>NARule</code>	single integer used to determine what to do when only a single layer with a non-NA value is encountered (and linear interpolation is not possible). The default value of 1 indicates that all layers will get this value for that cell; all other values do not change the cell values
<code>filename</code>	character. Output filename
<code>...</code>	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[focal](#), [fillTime](#)**Examples**

```

r <- rast(ncols=5, nrows=5)
r1 <- setValues(r, runif(ncell(r)))
r2 <- setValues(r, runif(ncell(r)))
r3 <- setValues(r, runif(ncell(r)))
r4 <- setValues(r, runif(ncell(r)))
r5 <- setValues(r, NA)
r6 <- setValues(r, runif(ncell(r)))
r1[6:10] <- NA
r2[5:15] <- NA
r3[8:25] <- NA
s <- c(r1,r2,r3,r4,r5,r6)
s[1:5] <- NA
x1 <- approximate(s)
x2 <- approximate(s, rule=2)
x3 <- approximate(s, rule=2, z=c(1,2,3,5,14,15))

```

Description

Standard arithmetic operators for computations with `SpatRasters`. Computations are local (applied on a cell by cell basis). If multiple `SpatRaster` objects are used, these must have the same geometry (extent and resolution). These operators have been implemented:

`+`, `-`, `*`, `/`, `^`, `%%`, `%%/`

The following methods have been implemented for `SpatExtent`:

for (`SpatExtent`, `SpatExtent`): `+`, `-`, and for (`SpatExtent`, `numeric`): `+`, `-`, `*`, `/`, `%%`

Value

`SpatRaster` or `SpatExtent`

seealso

[ifel](#) to conveniently combine operations and [Math-methods](#) or [app](#) to use mathematical functions not implemented by the package.

Examples

```
r1 <- rast(ncols=10, nrows=10)
v <- runif(ncell(r1))
v[10:20] <- NA
values(r1) <- v
r2 <- rast(r1)
values(r2) <- 1:ncell(r2) / ncell(r2)
r3 <- r1 + r2
r2 <- r1 / 10
r3 <- r1 * (r2 - 1 / r2)

b <- c(r1, r2, r3)
b2 <- b * 10

### SpatExtent methods
x <- ext(0.1, 2.2, 0, 3)
y <- ext(-2, 1, -2,2)
# union
x + y
# intersection
x * y

e <- x
e
e * 2
e / 2
```

```
e + 1
e - 1
```

as.character	<i>Create a text representation of (the skeleton of) an object</i>
--------------	--

Description

Create a text representation of (the skeleton of) an object

Usage

```
## S4 method for signature 'SpatExtent'
as.character(x)

## S4 method for signature 'SpatRaster'
as.character(x)
```

Arguments

x SpatRaster

Value

character

Examples

```
r <- rast()
ext(r)
ext(c(0, 20, 0, 20))
```

as.data.frame	<i>SpatRaster or SpatVector to data.frame</i>
---------------	---

Description

Coerce a SpatRaster or SpatVector to a data.frame

Usage

```
## S4 method for signature 'SpatVector'
as.data.frame(x, row.names=NULL, optional=FALSE, geom=NULL, ...)

## S4 method for signature 'SpatRaster'
as.data.frame(x, row.names=NULL, optional=FALSE, xy=FALSE, cells=FALSE, na.rm=TRUE, ...)
```

Arguments

x	SpatRaster or SpatVector
geom	character or NULL. If not NULL, either "WKT" or "HEX", to get the geometry included in Well-Known-Text or hexadecimal notation. If x has point geometry, it can also be "XY" to add the coordinates of each point
xy	logical. If TRUE, the coordinates of each raster cell are included
cells	logical. If TRUE, the cell numbers of each raster cell are included
na.rm	logical. If TRUE, cells that have a NA value in at least one layer are removed
...	Additional arguments passed to the <code>data.frame</code>
row.names	This argument is ignored
optional	This argument is ignored

Value

data.frame

See Also

[as.list](#), [as.matrix](#). See [geom](#) to only extract the geometry of a SpatVector

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
as.data.frame(v)
```

as.list

SpatRaster or SpatVector to list*

Description

Coerce a SpatRaster, SpatRasterCollection, SpatRasterDataset, or SpatVector to a list. With a SpatRaster, each layer becomes a list element. With a SpatRasterCollection or SpatRasterDataset, each SpatRaster becomes a list element. With a SpatVector, each variable (attribute) becomes a list element.

Usage

```
## S4 method for signature 'SpatRaster'
as.list(x, ...)

## S4 method for signature 'SpatRasterCollection'
as.list(x, ...)

## S4 method for signature 'SpatVector'
as.list(x, geom=NULL, ...)
```


Arguments

x	SpatRaster, SpatRasterDataset, SpatRasterCollection, or SpatVector
geom	character or NULL. If not NULL, either "WKT" or "HEX", to get the geometry included in Well-Known-Text or hexadecimal notation. If x has point geometry, it can also be "XY" to add the coordinates of each point
...	Additional arguments. These are ignored

Value

list

See Also

see [coerce](#) for as.data.frame with a SpatRaster; and [geom](#) to only extract the geometry of a SpatVector

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
as.list(v)

s <- rast(system.file("ex/logo.tif", package="terra")) + 1
as.list(s)
```

as.raster	<i>Coerce to a "raster" object</i>
-----------	------------------------------------

Description

Implementation of the generic [as.raster](#) function to create a "raster" (small r) object. Such objects can be used for plotting with the [rasterImage](#) function. NOT TO BE CONFUSED with the Raster* (big R) objects defined by the 'raster' package!

Usage

```
## S4 method for signature 'SpatRaster'
as.raster(x, maxcell=500000, col)
```

Arguments

x	SpatRaster
maxcell	positive integer. Maximum number of cells to use for the plot
col	vector of colors. Default is col=rev(terrain.colors(255))

Value

'raster' object

Examples

```
r <- rast(ncols=3, nrows=3)
values(r) <- 1:ncell(r)
as.raster(r)
```

as.spatvector

Conversion to a SpatVector, or to another SpatVector type

Description

Conversion of a SpatRaster or SpatExtent to a SpatVector of points, lines, or polygons;
And conversion of a SpatVector to a another SpatVector type.

Usage

```
## S4 method for signature 'SpatRaster'
as.polygons(x, trunc=TRUE, dissolve=TRUE, values=TRUE,
na.rm=TRUE, na.all=FALSE, extent=FALSE)

## S4 method for signature 'SpatRaster'
as.lines(x)

## S4 method for signature 'SpatRaster'
as.points(x, values=TRUE, na.rm=TRUE, na.all=FALSE)

## S4 method for signature 'SpatVector'
as.polygons(x, extent=FALSE)

## S4 method for signature 'SpatVector'
as.lines(x)

## S4 method for signature 'SpatVector'
as.points(x, multi=FALSE, skiplast=TRUE)

## S4 method for signature 'SpatExtent'
as.polygons(x, crs="")

## S4 method for signature 'SpatExtent'
as.lines(x, crs="")

## S4 method for signature 'SpatExtent'
as.points(x, crs="")
```

Arguments

<code>x</code>	SpatRaster or SpatVector
<code>trunc</code>	logical; truncate values to integers. Cells with the same value are merged. Therefore, if <code>trunc=FALSE</code> the object returned can have many cells and can be very large
<code>dissolve</code>	logical; combine cells with the same values? If TRUE only the first layer in <code>x</code> is processed
<code>values</code>	logical; include cell values as attributes?
<code>multi</code>	logical. If TRUE a multi-point geometry is returned
<code>skiplast</code>	logical. If TRUE the last point of a polygon (which is the same as the first point) is not included
<code>extent</code>	logical. if TRUE, a polygon for the extent of the SpatRaster or SpatVector is returned. If <code>x</code> is a SpatRaster, the polygon has vertices for each row and column, not just the four corners of the raster. This can be useful for more precise projection. If that is not required, it is more efficient to get the extent represented by only the four corners with <code>as.polygons(ext(x), crs=crs(x))</code>
<code>na.rm</code>	logical. If TRUE cells that are NA are ignored
<code>na.all</code>	logical. If TRUE cells are only ignored if <code>na.rm=TRUE</code> and their value is NA for all layers instead of for any layer
<code>crs</code>	character. The coordinate reference system (see crs)

Value

SpatVector

Examples

```

r <- rast(ncols=2, nrows=2)
values(r) <- 1:ncell(r)

as.points(r)
as.lines(ext(r), crs=crs(r))

if (gdal() >= "3.0.0") {
  p <- as.polygons(r)
  p
  as.lines(p)
  as.points(p)
}

```

atan2	<i>Two argument arc-tangent</i>
-------	---------------------------------

Description

For `SpatRasters` `x` and `y`, `atan2(y, x)` returns the angle in radians for the tangent `y/x`, handling the case when `x` is zero. See [Trig](#)

See [Math-methods](#) for other trigonometric and mathematical functions that can be used with `SpatRasters`.

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'  
atan2(y, x)  
  
## S4 method for signature 'SpatRaster,SpatRaster'  
atan_2(y, x, filename, ...)
```

Arguments

<code>y</code>	<code>SpatRaster</code>
<code>x</code>	<code>SpatRaster</code>
<code>filename</code>	character. Output filename
<code>...</code>	additional arguments for writing files as in writeRaster

See Also

[Math-methods](#)

Examples

```
r1 <- rast(nrows=10, ncols=10)  
r2 <- rast(nrows=10, ncols=10)  
values(r1) <- (runif(ncell(r1))-0.5) * 10  
values(r2) <- (runif(ncell(r1))-0.5) * 10  
atan2(r1, r2)
```

autocorrelation	<i>Spatial autocorrelation</i>
-----------------	--------------------------------

Description

Compute spatial autocorrelation for a numeric vector or a SpatRaster. You can compute standard (global) Moran's I or Geary's C, or local indicators of spatial autocorrelation (Anselin, 1995).

Usage

```
## S4 method for signature 'numeric'
autocor(x, w, method="moran")

## S4 method for signature 'SpatRaster'
autocor(x, w=matrix(c(1,1,1,1,0,1,1,1,1),3), method="moran", global=TRUE)
```

Arguments

x	numeric or SpatRaster
w	Spatial weights defined by or a rectangular matrix. For a SpatRaster this matrix must the sides must have an odd length (3, 5, ...)
global	logical. If TRUE global autocorrelation is computed instead of local autocorrelation
method	character. If x is numeric or SpatRaster: "moran" for Moran's I and "geary" for Geary's C. If x is numeric also: "Gi", "Gi*" (the Getis-Ord statistics), locmor (local Moran's I) and "mean" (local mean)

Details

The default setting uses a 3x3 neighborhood to compute "Queen's case" indices. You can use a filter (weights matrix) to do other things, such as "Rook's case", or different lags.

Value

numeric or SpatRaster

References

Moran, P.A.P., 1950. Notes on continuous stochastic phenomena. *Biometrika* 37:17-23

Geary, R.C., 1954. The contiguity ratio and statistical mapping. *The Incorporated Statistician* 5: 115-145

Anselin, L., 1995. Local indicators of spatial association-LISA. *Geographical Analysis* 27:93-115
https://en.wikipedia.org/wiki/Indicators_of_spatial_association

See Also

The `spdep` package for additional and more general approaches for computing spatial autocorrelation

Examples

```
### raster
r <- rast(nrows=10, ncols=10, xmin=0)
values(r) <- 1:ncell(r)

autocor(r)

# rook's case neighbors
f <- matrix(c(0,1,0,1,0,1,0,1,0), nrow=3)
autocor(r, f)

# local
rc <- autocor(r, w=f, global=FALSE)

### numeric (for vector data)
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
w <- relate(v, relation="touches")

# global
autocor(v$AREA, w)

# local
v$Gi <- autocor(v$AREA, w, "Gi")
plot(v, "Gi")
```

barplot

Bar plot of a SpatRaster

Description

Create a barplot of the values of a the first layer of a `SpatRaster`. For large datasets a regular sample with a size of approximately `maxcells` is used.

Usage

```
## S4 method for signature 'SpatRaster'
barplot(height, maxcell=1000000, digits=0, breaks=NULL, col, ...)
```

Arguments

<code>height</code>	<code>SpatRaster</code>
<code>maxcell</code>	integer. To regularly subsample very large datasets

<code>digits</code>	integer used to determine how to round the values before tabulating. Set to NULL or to a large number if you do not want any rounding
<code>breaks</code>	breaks used to group the data as in cut
<code>col</code>	a color generating function such as rainbow (the default), or a vector of colors
<code>...</code>	additional arguments for plotting as in barplot

Value

A numeric vector (or matrix, when `beside = TRUE`) of the coordinates of the bar midpoints, useful for adding to the graph. See [barplot](#)

See Also

[hist](#), [boxplot](#)

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
barplot(r, digits=-1, las=2, ylab="Frequency")

op <- par(no.readonly = TRUE)
par(mai = c(1, 2, .5, .5))
barplot(r, breaks=10, col=c("red", "blue"), horiz=TRUE, digits=NULL, las=1)
par(op)
```

boundaries

Detect boundaries (edges)

Description

Detect boundaries (edges). Boundaries are cells that have more than one class in the 4 or 8 cells surrounding it, or, if `classes=FALSE`, cells with values and cells with NA.

Usage

```
## S4 method for signature 'SpatRaster'
boundaries(x, classes=FALSE, inner=TRUE,
           directions=8, falseval=0, filename="", ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>inner</code>	logical. If TRUE, "inner" boundaries are returned, else "outer" boundaries are returned

classes	character. Logical. If TRUE all different values are (after rounding) distinguished, as well as NA. If FALSE (the default) only edges between NA and non-NA cells are considered
directions	integer. Which cells are considered adjacent? Should be 8 (Queen's case) or 4 (Rook's case)
falseval	numeric. The value to use for cells that are not a boundary and not NA
filename	character. Output filename
...	options for writing files as in writeRaster

Value

SpatRaster. Cell values are either 1 (a border) or 0 (not a border), or NA

See Also

[focal](#), [patches](#)

Examples

```
r <- rast(nrows=18, ncols=36, xmin=0)
r[150:250] <- 1
r[251:450] <- 2
bi <- boundaries(r)
bo <- boundaries(r, inner=FALSE)
bc <- boundaries(r, classes=TRUE)
#plot(bc)
```

boxplot

Box plot of SpatRaster data

Description

Box plot of layers in a SpatRaster

Usage

```
## S4 method for signature 'SpatRaster'
boxplot(x, y=NULL, maxcell=100000, ...)
```

Arguments

x	SpatRaster
y	NULL or a SpatRaster. If x is a SpatRaster it used to group the values of x by "zone"
maxcell	Integer. Number of cells to sample from datasets
...	additional arguments passed to <code>graphics::boxplot</code>

Value

boxplot returns a list (invisibly) that can be used with [bxp](#)

See Also

[pairs](#), [hist](#)

Examples

```
r1 <- r2 <- r3 <- rast(ncols=10, nrows=10)
set.seed(409)
values(r1) <- rnorm(ncell(r1), 100, 40)
values(r2) <- rnorm(ncell(r1), 80, 10)
values(r3) <- rnorm(ncell(r1), 120, 30)
s <- c(r1, r2, r3)
names(s) <- c("Apple", "Pear", "Cherry")

boxplot(s, notch=TRUE, col=c("red", "blue", "orange"), main="Box plot", ylab="random", las=1)

op <- par(no.readonly = TRUE)
par(mar=c(4,6,2,2))
boxplot(s, horizontal=TRUE, col="lightskyblue", axes=FALSE)
axis(1)
axis(2, at=0:3, labels=c("", names(s)), las=1, cex.axis=.9, lty=0)
par(op)

## boxplot with 2 layers
v <- vect(system.file("ex/lux.shp", package="terra"))
r <- rast(system.file("ex/elev.tif", package="terra"))
y <- rasterize(v, r, "NAME_2")
b <- boxplot(r, y)
bxp(b)
```

buffer

Create a buffer around vector geometries or raster patches

Description

Calculate a buffer around all cells that are not NA in a `SpatRaster`, or around the geometries of a `SpatVector`

Note that the distance unit of the buffer width parameter is meters if the CRS is (+proj=longlat), and in map units (typically also meters) if not.

Usage

```
## S4 method for signature 'SpatRaster'
buffer(x, width, filename="", ...)

## S4 method for signature 'SpatVector'
buffer(x, width, quadsegs=10)
```

Arguments

x	SpatRaster or SpatVector
width	numeric. Unit is meter if x has a longitude/latitude CRS, or mapunits in other cases. Should be > 0 for SpatRaster
filename	character. Output filename
...	additional arguments for writing files as in writeRaster
quadsegs	positive integer. Number of line segments to use to draw a quart circle

Value

SpatRaster

See Also

[distance](#)

Examples

```
r <- rast(ncols=36, nrows=18)
v <- rep(NA, ncell(r))
v[500] <- 1
values(r) <- v
b <- buffer(r, width=5000000)
plot(b)

v <- vect(rbind(c(10,10), c(0,60)), crs="+proj=merc")
b <- buffer(v, 20)
plot(b)
points(v)

crs(v) <- "+proj=longlat"
b <- buffer(v, 1500000)
plot(b)
points(v)
```

c

Combine SpatRaster or SpatVector objects

Description

With `c` you can:

- Combine SpatRaster objects. They must have the same extent and resolution. However, if `x` is empty (has no cell values), its geometry is ignored with a warning. Two empty SpatRasters with the same geometry can also be combined (to get a summed number of layers). Also see [add<-](#)

- Add a SpatRaster to a SpatRasterDataset

- Add SpatVector objects to a new or existing SpatVectorCollection

To append SpatVectors, use `rbind`.

Usage

```
## S4 method for signature 'SpatRaster'
c(x, ..., warn=TRUE)

## S4 method for signature 'SpatRasterDataset'
c(x, ...)

## S4 method for signature 'SpatVector'
c(x, ...)

## S4 method for signature 'SpatVectorCollection'
c(x, ...)
```

Arguments

x	SpatRaster, SpatVector, SpatRasterDataset or SpatVectorCollection
warn	logical. If TRUE, a warning is emitted if x is an empty SpatRaster
...	as for x (you can only combine raster with raster data and vector with vector data)

Value

Same class as x

See Also

[add<-](#)

Examples

```
r <- rast(nrows=5, ncols=9)
values(r) <- 1:ncell(r)
x <- c(r, r*2, r*3)
```

cartogram

Cartogram

Description

Make a cartogram, that is, a map where the area of polygons is made proportional to another variable. This can be a good way to map raw count data (e.g. votes).

Usage

```
## S4 method for signature 'SpatVector'
cartogram(x, var, type)
```

Arguments

x	SpatVector
var	character. A variable name in x
type	character. Cartogram type, only "nc" (non-contiguous) is currently supported

Value

SpatVector

See Also

[plot](#), [rescale](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
v$value <- 1:12
p <- cartogram(v, "value", "nc")
plot(v, col="light gray", border="gray")
lines(p, col="red", lwd=2)
```

catalyze

Factors to numeric

Description

Change a categorical layer into one or more numerical layers. With `as.numeric` you can transfer the active category values to cell values in a non-categorical `SpatRaster`. `catalyze` creates new layers for each category.

Usage

```
## S4 method for signature 'SpatRaster'
as.numeric(x, index=NULL, filename="", ...)
```

```
## S4 method for signature 'SpatRaster'
catalyze(x, filename="", ...)
```

Arguments

x	SpatRaster
index	positive integer, indicating the column in data.frame value to be used as the category, skipping the first column with the ID. If NULL the active category is used
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[activeCat](#), [cats](#)**Examples**

```

set.seed(0)
r <- rast(nrows=10, ncols=10)
values(r) <- sample(3, ncell(r), replace=TRUE) + 10
d <- data.frame(id=11:13, cover=c("forest", "water", "urban"), letters=letters[1:3], value=10:12)
levels(r) <- d
catalyze(r)

activeCat(r) <- 3
as.numeric(r)

```

cells

Get cell numbers

Description

Get the cell numbers covered by a SpatVector or SpatExtent. Or that match values in a vector; or all non NA values.

Usage

```

## S4 method for signature 'SpatRaster,missing'
cells(x, y)

## S4 method for signature 'SpatRaster,numeric'
cells(x, y)

## S4 method for signature 'SpatRaster,SpatVector'
cells(x, y, method="simple", weights=FALSE, exact=FALSE, touches=is.lines(y))

## S4 method for signature 'SpatRaster,SpatExtent'
cells(x, y)

```

Arguments

x	SpatRaster
y	SpatVector, SpatExtent, 2-column matrix representing points, numeric representing values to match, or missing

method	character. Method for getting cell numbers for points. The default is "simple", the alternative is "bilinear". If it is "bilinear", the four nearest cells and their weights are returned
weights	logical. If TRUE and y has polygons, the approximate fraction of each cell that is covered is returned as well
exact	logical. If TRUE and y has polygons, the exact fraction of each cell that is covered is returned as well
touches	logical. If TRUE, values for all cells touched by lines or polygons are extracted, not just those on the line render path, or whose center point is within the polygon. Not relevant for points

Value

numeric vector or matrix

Examples

```
r <- rast(ncols=10, nrows=10)
values(r) <- 1:ncell(r)
r[c(1:25, 31:100)] <- NA
r <- ifel(r > 28, r + 10, r)

# all cell numbers of cells that are not NA
cells(r)

# cell numbers that match values
x <- cells(r, c(28,38))
x$lyr.1

# cells for points
m <- cbind(x=c(0,10,-30), y=c(40,-10,20))
cellFromXY(r, m)

v <- vect(m)
cells(r, v)
cells(r, v, method="bilinear")

# cells for polygons
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
r <- rast(v)
cv <- cells(r, v)
```

Description

Compute the area covered by individual raster cells. Computing the surface area of raster cells is particularly relevant for longitude/latitude rasters.

Note that for both angular (longitude/latitude) and for planar (projected) coordinate reference systems raster cells sizes are generally not constant, unless you are using an equal-area coordinate reference system.

For planar CRSs, the area is therefore not computed based on the linear units of the coordinate reference system, but on the *actual* area, correcting for distortion. If you do not want that, you can instead use `init(x, prod(res(x)))`

Usage

```
## S4 method for signature 'SpatRaster'
cellSize(x, mask=TRUE, unit="m", transform=TRUE, rcx=100, filename="", ...)
```

Arguments

x	SpatRaster
mask	logical. If TRUE, cells that are NA in x are also NA in the output
unit	character. One of "m", "km", or "ha"
transform	logical. If TRUE, planar CRS data are transformed to lon/lat for accuracy
rcx	positive integer. The maximum number of rows and columns to be used to compute area of planar data if transform=TRUE. If x has more rows and/or columns, the raster is aggregated to match this limit, and values for the original cells are estimated by bilinear interpolation (see <code>resample</code>). This can save a lot of time
filename	character. Output filename
...	additional arguments for writing files as in <code>writeRaster</code>

Value

numeric. The area of each cell, expressed in square meters, square kilometers, or hectares.

See Also

[expand](#)

Examples

```
# SpatRaster
r <- rast(nrows=18, ncols=36)
v <- 1:ncell(r)
v[200:400] <- NA
values(r) <- v

# size of each raster cell
```

```

a <- cellSize(r)

# illustration of distortion
r <- rast(ncols=90, nrows=45, ymin=-80, ymax=80)
m <- project(r, "+proj=merc")

bad <- init(m, prod(res(m)) / 1000000, names="naive")
good <- cellSize(m, unit="km", names="corrected")
plot(c(good, bad), nc=1, mar=c(2,2,1,6))

```

centroids

Centroids

Description

Get the centroids of polygons or lines, or centroid-like points that are guaranteed to be inside the polygons or on the lines.

Usage

```

## S4 method for signature 'SpatVector'
centroids(x, inside=FALSE)

```

Arguments

x	SpatVector
inside	logical. If TRUE the points returned are guaranteed to be inside the polygons or on the lines, but they are not the true centroids. True centroids may be outside a polygon, for example when a polygon is "bean shaped", and they are unlikely to be on their line

Value

SpatVector of points

Examples

```

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
x <- centroids(v)
y <- centroids(v, TRUE)

```

`clamp`*Clamp values*

Description

Clamp values to a minimum and maximum value. That is, all values below a lower threshold value and above the upper threshold value become either NA, or, if `values=TRUE`, become the threshold value

Usage

```
## S4 method for signature 'SpatRaster'  
clamp(x, lower=-Inf, upper=Inf, values=TRUE, filename="", ...)  
  
## S4 method for signature 'numeric'  
clamp(x, lower=-Inf, upper=Inf, values=TRUE, ...)
```

Arguments

<code>x</code>	SpatRaster
<code>lower</code>	numeric. lowest value
<code>upper</code>	numeric. highest value
<code>values</code>	logical. If FALSE values outside the clamping range become NA, if TRUE, they get the extreme values
<code>filename</code>	character. Output filename
<code>...</code>	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[classify](#)

Examples

```
r <- rast(ncols=10, nrows=10)  
values(r) <- 1:ncell(r)  
rc <- clamp(r, 25, 75)  
rc
```

classify	<i>Classify (or reclassify) cell values</i>
----------	---

Description

Classify values of a SpatRaster. The function (re-)classifies groups of values to other values.

The classification is done based on the argument `rc1`. You can classify ranges by specifying a three-column matrix "from-to-becomes" or change specific values by using a two-column matrix "is-becomes". You can also supply a vector with "cuts" or the "number of cuts".

With "from-to-becomes" or "is-becomes" classification is done in the row order of the matrix. Thus, if there are overlapping ranges or values, the first time a number is within a range determines the reclassification value.

With "cuts" the values are sorted, so that the order in which they are provided does not matter.

Usage

```
## S4 method for signature 'SpatRaster'
classify(x, rc1, include.lowest=FALSE, right=TRUE,
         others=NULL, brackets=TRUE, filename="", ...)
```

Arguments

<code>x</code>	SpatRaster
<code>rc1</code>	matrix for classification. This matrix must have 1, 2 or 3 columns. If there are three columns, the first two columns are "from" "to" of the input values, and the third column "becomes" has the new value for that range. The two column matrix ("is", "becomes") can be useful for classifying integer values. In that case, the arguments <code>right</code> and <code>include.lowest</code> are ignored. A single column matrix (or a vector) is interpreted as a set of cuts if there is more than one value. In that case the values are classified based on their location in-between the cut-values. If a single number is provided, that is used to make that number of cuts, at equal intervals between the lowest and highest values of the SpatRaster.
<code>include.lowest</code>	logical, indicating if a value equal to the lowest value in <code>rc1</code> (or highest value in the second column, for <code>right=FALSE</code>) should be included.
<code>right</code>	logical. If TRUE, the intervals are closed on the right (and open on the left). If FALSE they are open at the right and closed at the left. "open" means that the extreme value is <i>not</i> included in the interval. Thus, right-closed and left open is $(0, 1] = \{x \mid 0 < x \leq 1\}$. You can also close both sides with <code>right=NA</code> , that is only meaningful if you "from-to-becomes" classification with integers. For example to classify 1-5 -> 1, 6-10 -> 2, 11-15 -> 3. That may be easier to read and write than the equivalent 1-5 -> 1, 5-10 -> 2, 10-15 -> 3 with <code>right=TRUE</code> and <code>include.lowest=TRUE</code>
<code>others</code>	numeric. If not NULL all values that are not matched are set to this value. Otherwise they retain their original value.

brackets	logical. If TRUE, intervals are have parenthesis or brackets around them to indicate whether they are open or closed. Only applies if rcl is a vector (or single column matrix)
filename	character. Output filename
...	Additional arguments for writing files as in writeRaster

Value

SpatRaster

NoteFor model-based classification see [predict](#)**See Also**[subst](#) for simpler from-to replacement**Examples**

```

r <- rast(ncols=10, nrows=10)
values(r) <- (0:99)/99

## from-to-becomes
# classify the values into three groups
# all values >= 0 and <= 0.25 become 1, etc.
m <- c(0, 0.25, 1,
      0.25, 0.5, 2,
      0.5, 1, 3)
rclmat <- matrix(m, ncol=3, byrow=TRUE)
rc1 <- classify(r, rclmat, include.lowest=TRUE)

## cuts
# equivalent to the above, but now a categorical SpatRaster is returned
rc2 <- classify(r, c(0, 0.25, 0.5, 1), include.lowest=TRUE, brackets=TRUE)
freq(rc2)

## is-becomes
x <- round(r*3)
unique(x)
# replace 0 with NA
y <- classify(x, cbind(0, NA))
unique(y)

# multiple replacements
m <- rbind(c(2, 200), c(3, 300))
m

rcx1 <- classify(x, m)
unique(rcx1)

```

```
rcx2 <- classify(x, m, others=NA)
unique(rcx2)
```

click *Query by clicking on a map*

Description

Click on a map (plot) to get the coordinates or the values of a SpatRaster or SpatVector at that location. For a SpatRaster you can also get the coordinates and cell number of the location.

Usage

```
## S4 method for signature 'SpatRaster'
click(x, n=10, id=FALSE, xy=FALSE, cell=FALSE, type="p", show=TRUE, ...)

## S4 method for signature 'SpatVector'
click(x, n=10, id=FALSE, xy=FALSE, type="p", show=TRUE, ...)

## S4 method for signature 'missing'
click(x, n=10, id=FALSE, type="p", show=TRUE, ...)
```

Arguments

x	SpatRaster or SpatVector, or missing
n	number of clicks on the plot (map)
id	logical. If TRUE, a numeric ID is shown on the map that corresponds to the row number of the output
xy	logical. If TRUE, xy coordinates are included in the output
cell	logical. If TRUE, cell numbers are included in the output
type	one of "n", "p", "l" or "o". If "p" or "o" the points are plotted; if "l" or "o" they are joined by lines. See ?locator
show	logical. Print the values after each click?
...	additional graphics parameters used if type != "n" for plotting the locations. See ?locator

Value

The value(s) of x at the point(s) clicked on (or touched by the box drawn).

Note

The plot only provides the coordinates for a spatial query, the values are read from the SpatRaster that is passed as an argument. Thus you can extract values from an object that has not been plotted, as long as it spatially overlaps with with the extent of the plot.

Unless the process is terminated prematurely values at at most n positions are determined. The identification process can be terminated by hitting Esc, or by clicking the right mouse button and selecting "Stop" from the menu, or from the "Stop" menu on the graphics window.

See Also[draw](#)**Examples**

```
## Not run:
r <-rast(system.file("ex/elev.tif", package="terra"))
plot(r)
click(r, n=1)
## now click on the plot (map)

## End(Not run)
```

coerce

*Coercion of a SpatRaster to a vector, matrix or array***Description**

Coercion to other object types

Usage

```
## S4 method for signature 'SpatRaster'
as.vector(x, mode='any')

## S4 method for signature 'SpatRaster'
as.matrix(x, wide=FALSE, ...)

## S4 method for signature 'SpatRaster'
as.array(x)
```

Arguments

x	SpatRaster or SpatVector
wide	logical. If FALSE each layer in the SpatRaster becomes a column in the matrix and each cell in the SpatRaster becomes a row. If TRUE each row in the SpatRaster becomes a row in the matrix and each column in the SpatRaster becomes a column in the matrix
mode	this argument is ignored
...	additional arguments (none implemented)

Value

vector, matrix, or array

See Also[as.data.frame](#) and [as.polygons](#)

Examples

```

r <- rast(ncols=2, nrows=2)
values(r) <- 1:ncell(r)

as.vector(r)
as.matrix(r)
as.matrix(r, wide=TRUE)
as.data.frame(r, xy=TRUE)
as.array(r)

```

colors

Color table

Description

Get or set color table(s) associated with a SpatRaster. Color tables are used for associating colors with values, for use in mapping (plot).

Usage

```

## S4 method for signature 'SpatRaster'
coltab(x)

## S4 replacement method for signature 'SpatRaster'
coltab(x, layer=1)<-value

## S4 method for signature 'SpatRaster'
has.colors(x)

```

Arguments

x	SpatRaster
layer	positive integer, the layer number or name
value	a two-column data.frame (first column the cell value, second the color); a vector of colors (the first one is the color for value 0 and so on); or a three (red,green,blue) or four (alpha) column data.frame also from 0 to n; or NULL to remove the color table

Value

data.frame

Examples

```

r <- rast(ncols=3, nrows=2, vals=0:5)
coltb <- data.frame(t(col2rgb(rainbow(6, end=.9), alpha=TRUE)))
coltb

plot(r)

has.colors(r)
coltab(r) <- coltb
plot(r)
has.colors(r)

tb <- coltab(r)
class(tb)
dim(tb[[1]])

```

combineGeoms

Combine geometries

Description

Combine the geometries of one `SpatVector` with those of another. Geometries can be combined based on overlap, shared boundaries and distance (in that order of operation).

The typical use-case of this method is when you are editing geometries and you have a number of small polygons in one `SpatVector` that should be part of the geometries of the another `SpatVector`; perhaps because they were small holes inbetween the borders of two `SpatVectors`.

To append `SpatVectors` use 'rbind' and see methods like `intersect` and `union` for "normal" polygons combinations.

Usage

```

## S4 method for signature 'SpatVector,SpatVector'
combineGeoms(x, y, overlap=TRUE, boundary=TRUE, distance=TRUE,
append=TRUE, minover=0.1, maxdist=Inf, dissolve=TRUE, erase=TRUE)

```

Arguments

<code>x</code>	<code>SpatVector</code> of polygons
<code>y</code>	<code>SpatVector</code> of polygons geometries that are to be combined with <code>x</code>
<code>overlap</code>	logical. If <code>TRUE</code> , a geometry is combined with the geometry it has most overlap with, if the overlap is above <code>minover</code>
<code>boundary</code>	logical. If <code>TRUE</code> , a geometry is combined with the geometry it has most shared border with
<code>distance</code>	logical. If <code>TRUE</code> , a geometry is combined with the geometry it is nearest to
<code>append</code>	logical. Should remaining geometries be appended to the output? Not relevant if <code>distance=TRUE</code>

minover	numeric	The fraction of the geometry in codey that overlaps with a geometry in x. Below this threshold, geometries are not considered overlapping
maxdist	numeric	Geometries further away from each other than this distance (in meters) will not be combined
dissolve	logical	Should internal boundaries be dissolved?
erase	logical	If TRUE no new overlapping areas are created

Value

SpatVector

See Also

[union](#), [erase](#), [intersect](#)

[sharedPaths](#), [erase](#), [intersect](#)

Examples

```
x1 <- vect("POLYGON ((0 0, 8 0, 8 9, 0 9, 0 0))")
x2 <- vect("POLYGON ((10 4, 12 4, 12 7, 11 7, 11 6, 10 6, 10 4))")

y1 <- vect("POLYGON ((5 6, 15 6, 15 15, 5 15, 5 6))")
y2 <- vect("POLYGON ((8 2, 9 2, 9 3, 8 3, 8 2))")
y3 <- vect("POLYGON ((2 6, 3 6, 3 8, 2 8, 2 6))")
y4 <- vect("POLYGON ((2 12, 3 12, 3 13, 2 13, 2 12))")

x <- rbind(x1, x2)
values(x) <- data.frame(xid=1:2)
crs(x) <- "+proj=utm +zone=1"

y <- rbind(y1, y2, y3, y4)
values(y) <- data.frame(yid=letters[1:4])
crs(y) <- "+proj=utm +zone=1"

plot(rbind(x, y), border=c(rep("red",2), rep("blue", 4)), lwd=2)
text(x, "xid")
text(y, "yid")

v <- combineGeoms(x, y)
plot(v, col=c("red", "blue"))

v <- combineGeoms(x, y, boundary=FALSE, maxdist=1, minover=.05)
plot(v, col=rainbow(4))
```


Description

Standard comparison and logical operators for computations with SpatRasters. Computations are local (applied on a cell by cell basis). If multiple SpatRaster objects are used, these must have the same geometry (extent and resolution). These operators have been implemented:

Logical: `!`, `&`, `|`, `isTRUE`, `isFALSE`

Compare: `==`, `!=`, `>`, `<`, `<=`, `>=`, `is.na`, `is.nan`, `is.finite`, `is.infinite`

The terra package does not distinguish between NA (not available) and NaN (not a number). In most cases this state is represented by NaN.

The following method has been implemented for

(SpatExtent, SpatExtent): `==`

Value

SpatRaster or SpatExtent

seealso

[all.equal](#), [Arith-methods](#). See [ifel](#) to conveniently combine operations and [Math-methods](#) or [app](#) to apply any R function to a SpatRaster.

Examples

```
r1 <- rast(ncols=10, nrows=10)
values(r1) <- runif(ncell(r1))
r1[10:20] <- NA
r2 <- rast(r1)
values(r2) <- 1:ncell(r2) / ncell(r2)
```

```
x <- is.na(r1)
!x
r1 == r2
```

Description

Evaluate whether two SpatRaster objects have the same extent, number of rows and columns, projection, resolution, and origin (or a subset of these comparisons).

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
compareGeom(x, y, ..., lyrs=FALSE, crs=TRUE, warncrs=FALSE, ext=TRUE,
rowcol=TRUE, res=FALSE, stopOnError=TRUE, messages=FALSE)
```

Arguments

x	SpatRaster
y	SpatRaster
...	Additional SpatRasters
lyrs	logical. If TRUE, the number of layers is compared
crs	logical. If TRUE, coordinate reference systems are compared
warncrs	logical. If TRUE, a warning is given if the crs is different (instead of an error)
ext	logical. If TRUE, bounding boxes are compared
rowcol	logical. If TRUE, number of rows and columns of the objects are compared
res	logical. If TRUE, resolutions are compared (redundant when checking extent and rowcol)
stopOnError	logical. If TRUE, code execution stops if raster do not match
messages	logical. If TRUE, warning/error messages are printed even if stopOnError=FALSE

Examples

```
r1 <- rast()
r2 <- rast()
r3 <- rast()
compareGeom(r1, r2, r3)
nrow(r3) <- 10

## Not run:
compareGeom(r1, r3)

## End(Not run)
```

concats

Concatenate categorical rasters

Description

Combine two categorical rasters by concatenating their levels.

Usage

```
## S4 method for signature 'SpatRaster'
concats(x, y, filename="", ...)
```

Arguments

x	SpatRaster (with a single, categorical, layer)
y	SpatRaster (with a single, categorical, layer)
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[cats](#)

Examples

```
set.seed(0)
r <- rast(nrows=10, ncols=10)
values(r) <- sample(3, ncell(r), replace=TRUE)
levels(r) <- data.frame(id=1:3, cover=c("forest", "water", "urban"))

rr <- rast(r)
values(rr) <- sample(1:3, ncell(rr), replace=TRUE)
levels(rr) <- data.frame(id=c(1:3), color=c("red", "green", "blue"))

x <- concats(r, rr)
x
levels(x)[[1]]
```

contour

Contour plot

Description

Contour lines of a SpatRaster. Use `add=TRUE` to add the lines to the current plot. See [contour](#) for details.

if `filled=TRUE`, a new filled contour plot is made. See [filled.contour](#) for details.

`as.contour` returns the contour lines as a SpatVector.

Usage

```
## S4 method for signature 'SpatRaster'
contour(x, maxcells=100000, filled=FALSE, ...)
```

```
## S4 method for signature 'SpatRaster'
as.contour(x, maxcells=100000, ...)
```

Arguments

x	SpatRaster. Only the first layer is used
maxcells	maximum number of pixels used to create the contours
filled	logical. If TRUE, a <code>filled.contour</code> plot is made
...	any argument that can be passed to <code>contour</code> or <code>filled.contour</code> (graphics package)

See Also

[plot](#)

Examples

```
r <- rast(system.file("ex/elev.tif", package="terra"))
plot(r)
contour(r, add=TRUE)

v <- as.contour(r)
plot(r)
lines(v)

contour(r, filled=TRUE, nlevels=5)

## if you want a SpatVector with contour lines
template <- disagg(rast(r), 10)
rr <- resample(r, template)
rr <- floor(rr/100) * 100
v <- as.polygons(rr)
plot(v, 1, col=terrain.colors(7))
```

convHull

Convex hull and minimal rotated rectangle

Description

Get the convex hull or the minimal rotated rectangle of a SpatVector

Usage

```
## S4 method for signature 'SpatVector'
convHull(x, by="")

## S4 method for signature 'SpatVector'
minRect(x, by="")
```

Arguments

x SpatVector
 by character (variable name), to make convex hulls by group

Value

SpatVector

Examples

```
p <- vect(system.file("ex/lux.shp", package="terra"))
h <- convHull(p)

hh <- convHull(p, "NAME_1")
rr <- minRect(p, "NAME_1")

plot(rr, lwd=5, border="gray")
plot(hh, "NAME_1", col=rainbow(10, alpha=.5), lwd=3, add=TRUE, plg=list(x="topright"))
lines(aggregate(p, "NAME_1"), col="blue", lty=2, lwd=2)
```

costDistance

Cost distance

Description

Use a friction (cost) surface to compute the cost-distance from any cell to one or more target cells.

Distances are computed by summing local distances between cells, which are connected with their neighbors in 8 directions, and assuming that the path has to go through the centers of one of the neighboring raster cells.

Distances are multiplied with the friction, thus to get the cost-distance, the friction surface must express the cost per unit distance (speed) of travel.

If the coordinate reference system (CRS) of the SpatRaster is longitude/latitude (+proj=longlat) the distance is in meters divided by variable m. The default of m is 1000, expressing distance in km. Otherwise the distance is in the units of the CRS (typically meters).

Usage

```
## S4 method for signature 'SpatRaster'
costDistance(x, target=0, scale=1000, maxiter=50, filename="", ...)
```

Arguments

x	SpatRaster
target	numeric. value of the target cells (where to compute cost-distance to)
scale	numeric. Scale factor for longitude/latitude data (1 = m, 1000 = km)
maxiter	numeric. The maximum number of iterations. Increase this number if you get the warning that costDistance did not converge
filename	character. output filename (optional)
...	additional arguments as for writeRaster

Value

SpatRaster

See Also[gridDistance](#), [distance](#)**Examples**

```

r <- rast(ncols=5, nrows=5, crs="+proj=utm +zone=1 +datum=WGS84",
xmin=0, xmax=5, ymin=0, ymax=5, vals=1)
r[13] <- 0
d <- costDistance(r)
plot(d)
text(d, digits=1)

r <- rast(ncols=10, nrows=10, xmin=0, xmax=10, ymin=0, ymax=10,
vals=10, crs="+proj=utm +zone=1 +datum=WGS84")
r[5, 1] <- -10
r[2:3, 1] <- r[1, 2:4] <- r[2, 5] <- 0
r[3, 6] <- r[2, 7] <- r[1, 8:9] <- 0
r[6, 6:10] <- NA
r[6:9, 6] <- NA

d <- costDistance(r, -10)
plot(d)
text(d, digits=1, cex=.8)

```

cover

Replace values with values from another object

Description

Replace NA or other values in SpatRaster x with the values of SpatRaster y

For polygons: areas of x that overlap with y are replaced by y or, if identity=TRUE intersected with y.

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'  
cover(x, y, values=NA, filename="", ...)  
  
## S4 method for signature 'SpatVector,SpatVector'  
cover(x, y, identity=FALSE, expand=TRUE)
```

Arguments

x	SpatRaster or SpatVector
y	Same as x
values	numeric. The cell values in x to be replaced by the values in y
filename	character. Output filename
...	additional arguments for writing files as in writeRaster
identity	logical. If TRUE overlapping areas are intersected rather than replaced
expand	logical. Should parts of y that are outside of x be included?

Value

SpatRaster

Examples

```
r1 <- r2 <- rast(ncols=36, nrows=18)  
values(r1) <- 1:ncell(r1)  
values(r2) <- runif(ncell(r2))  
r2 <- classify(r2, cbind(-Inf, 0.5, NA))  
r3 <- cover(r2, r1)  
  
p <- vect(system.file("ex/lux.shp", package="terra"))  
e <- as.polygons(ext(6, 6.4, 49.75, 50))  
values(e) <- data.frame(y=10)  
  
cv <- cover(p, e)  
plot(cv, col=rainbow(12))  
ci <- cover(p, e, identity=TRUE)  
lines(e, lwd=3)  
  
plot(ci, col=rainbow(12))  
lines(e, lwd=3)
```

crds

Get the coordinates of SpatVector geometries or SpatRaster cells

Description

Get the coordinates of a SpatVector or SpatRaster cells. A matrix or data.frame of the x (longitude) and y (latitude) coordinates is returned.

Usage

```
## S4 method for signature 'SpatVector'
crds(x, df=FALSE)
```

```
## S4 method for signature 'SpatRaster'
crds(x, df=FALSE, na.rm=TRUE)
```

Arguments

x	SpatRaster or SpatVector
df	logical. If TRUE a data.frame is returned instead of a matrix
na.rm	logical. If TRUE cells that are NA are excluded

Value

matrix or data.frame

See Also

[geom](#) returns the complete structure of SpatVector geometries. For SpatRaster see [xyFromCell](#)

Examples

```
x1 <- rbind(c(-175,-20), c(-140,55), c(10, 0), c(-140,-60))
x2 <- rbind(c(-125,0), c(0,60), c(40,5), c(15,-45))
x3 <- rbind(c(-10,0), c(140,60), c(160,0), c(140,-55))
x4 <- rbind(c(80,0), c(105,13), c(120,2), c(105,-13))
z <- rbind(cbind(object=1, part=1, x1), cbind(object=2, part=1, x2),
          cbind(object=3, part=1, x3), cbind(object=3, part=2, x4))
colnames(z)[3:4] <- c('x', 'y')
z <- cbind(z, hole=0)
z[,z[, "object"]==3 & z[, "part"]==2, "hole"] <- 1

p <- vect(z, "polygons")
crds(p)

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
g <- crds(v)
```



```
head(g)
```

```
crop
```

Cut out a geographic subset

Description

Cut out a part of a `SpatRaster` with a `SpatExtent`, or another object from which an extent can be obtained.

You can only crop rectangular areas, but see [mask](#) for setting cell values within `SpatRaster` to NA. Also note that the `SpatRaster` returned may not have the exactly the same extent as the `SpatExtent` supplied because you can only select entire cells (rows and columns), and you cannot add new areas. See methods like [resample](#) and [disagg](#) to force `SpatRasters` to align and [extend](#) to add rows and/or columns.

You can crop a `SpatVector` with another `SpatVector`. If these are not polygons, the minimum convex hull is used). Unlike with [intersect](#) the geometries and attributes of `y` are not transferred to the output. You can also crop a `SpatVector` with a rectangle (`SpatRaster`, `SpatExtent`).

Usage

```
## S4 method for signature 'SpatRaster'
crop(x, y, snap="near", mask=FALSE, touches=TRUE, filename="", ...)

## S4 method for signature 'SpatRasterDataset'
crop(x, y, snap="near", filename="", ...)

## S4 method for signature 'SpatVector'
crop(x, y)
```

Arguments

<code>x</code>	<code>SpatRaster</code> or <code>SpatVector</code>
<code>y</code>	<code>SpatRaster</code> , <code>SpatVector</code> , <code>SpatExtent</code> or other object that has a <code>SpatExtent</code> (<code>extent</code> returns a <code>SpatExtent</code>)
<code>snap</code>	character. One of "near", "in", or "out". Used to align <code>y</code> to the geometry of <code>x</code>
<code>mask</code>	logical. Should <code>y</code> be used to mask? Only used if <code>y</code> is a <code>SpatVector</code>
<code>touches</code>	logical. If TRUE and <code>mask=TRUE</code> , all cells touched by lines or polygons will be masked, not just those on the line render path, or whose center point is within the polygon
<code>filename</code>	character. Output filename
<code>...</code>	additional arguments for writing files as in writeRaster

Value

`SpatRaster`

See Also

intersect

Examples

```
r <- rast(xmin=0, xmax=10, ymin=0, ymax=10, nrows=25, ncols=25)
values(r) <- 1:ncell(r)
e <- ext(-5, 5, -5, 5)
rc <- crop(r, e)

# crop and mask
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
cm <- crop(r, v[9:12,], mask=TRUE)
plot(cm)
lines(v)

# crop vector
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
e <- ext(6.15, 6.3, 49.7, 49.8)
x <- crop(v, e)
plot(x, "NAME_1")
```

crosstab

Cross-tabulate

Description

Cross-tabulate the layers of a SpatRaster to create a contingency table.

Usage

```
## S4 method for signature 'SpatRaster,missing'
crosstab(x, digits=0, long=FALSE, useNA=FALSE)
```

Arguments

x	SpatRaster
digits	integer. The number of digits for rounding the values before cross-tabulation
long	logical. If TRUE the results are returned in 'long' format data.frame instead of a table
useNA	logical, indicating if the table should includes counts of NA values

Value

A table or data.frame

See Also[freq](#), [zonal](#)**Examples**

```

r <- s <- rast(nc=5, nr=5)
set.seed(1)
values(r) <- runif(ncell(r)) * 2
values(s) <- runif(ncell(r)) * 3
x <- c(r, s)

crosstab(x)

rs <- r/s
r[1:5] <- NA
s[20:25] <- NA
x <- c(r, s, rs)
crosstab(x, useNA=TRUE, long=TRUE)

```

crs*Get or set a coordinate reference system*

Description

Get or set the coordinate reference system (CRS), also referred to as a "projection", of a `SpatRaster` or `SpatVector`.

Setting a new CRS does not change the data itself, it just changes the label. So you should only set the CRS of a dataset (if it does not come with one) to what it *is*, not to what you would *like it to be*. See [project](#) to *transform* an object from one CRS to another.

Usage

```

## S4 method for signature 'SpatRaster'
crs(x, proj=FALSE, describe=FALSE, parse=FALSE)

## S4 method for signature 'SpatVector'
crs(x, proj=FALSE, describe=FALSE, parse=FALSE)

## S4 replacement method for signature 'SpatRaster'
crs(x)<-value

## S4 replacement method for signature 'SpatVector'
crs(x)<-value

```

Arguments

x	SpatRaster or SpatVector
proj	logical. If TRUE the crs is returned in PROJ-string notation
describe	logical. If TRUE the name, EPSG code, and the name and extent of the area of use are returned if known
value	character string describing a coordinate reference system. This can be in a WKT format, as a <authority:number> code such as "EPSG:4326", or a PROJ-string format such as "+proj=utm +zone=12" (see Note)
parse	logical. If TRUE, wkt parts are parsed into a vector (each line becomes an element)

Value

character or modified SpatRaster/Vector

Note

Projections are handled by the PROJ/GDAL libraries. Recent changes in the PROJ library to improve transformations between datums have degraded the library's usability. The PROJ developers suggest to no longer use the proj-string notation to define a CRS, but use the WKT2 or <authority>:<code> notation instead. These alternative systems work for formally described CRSs that are in databases, but they do not cover the infinite number of CRSs that exist. It is not practical to define one's own custom CRS with WKT2. Moreover, unlike the proj-notation, these newer systems are hard to read and that leads to code that cannot be easily understood and, therefore, is more error-prone.

It is still possible to use the PROJ-string notation with one major caveat: the datum should be WGS84 (or the equivalent NAD83) – if you want to transform your data to a coordinate reference system with a different datum. Thus as long as you use WGS84, or an ellipsoid instead of a datum, you can safely use PROJ-strings to represent your CRS; including to define your own custom CRS.

Examples

```
r <- rast()
crs(r)
crs(r, describe=TRUE, proj=TRUE)

crs(r) <- "+proj=lcc +lat_1=48 +lat_2=33 +lon_0=-100 +ellps=WGS84"
crs(r)

# You can also use epsg codes
crs(r) <- "epsg:25831"
crs(r, describe=TRUE)$area
```

`deepcopy`*Deep copy*

Description

Make a deep copy of a `SpatRaster` or `SpatVector`. This is occasionally useful when wanting to use a replacement function in a shallow copy. That is a copy that was created like this: `x <- y`. If you use a replacement function to change an object, its shallow copies also change.

Usage

```
## S4 method for signature 'SpatRaster'  
deepcopy(x)
```

```
## S4 method for signature 'SpatVector'  
deepcopy(x)
```

Arguments

`x` `SpatRaster` or `SpatVector`

Value

Same as `x`

Examples

```
r <- rast(ncols=10, nrows=10, nl=3)  
tm <- as.Date("2001-05-03") + 1:3  
time(r) <- tm  
time(r)  
x <- r  
time(x) <- tm + 365  
time(x)  
time(r)  
  
y <- deepcopy(r)  
time(y) <- tm - 365  
time(y)  
time(r)  
  
# or make a new object like this  
z <- rast(r)  
time(z) <- tm  
time(z)  
time(r)
```

densify

*Add additional nodes to lines or polygons***Description**

Add additional nodes to lines or polygons. This can be useful to do prior to using project such that the path does not change too much.

Usage

```
## S4 method for signature 'SpatVector'
densify(x, interval, equalize=TRUE)
```

Arguments

x	SpatVector
interval	numeric larger than 1, specifying the desired minimum interval between nodes
equalize	logical. If TRUE, new nodes are spread at equal intervals between old nodes

Value

SpatVector

Examples

```
v <- vect(rbind(c(-120,-20), c(-80,5), c(-40,-60), c(-120,-20)),
  type="polygons", crs="+proj=longlat")
vd <- densify(v, 200000)

p <- project(v, "+proj=robin")
pd <- project(vd, "+proj=robin")

# good
plot(pd, col="gray", border="red", lwd=10)
points(pd, col="gray")

# bad
lines(p, col="blue", lwd=3)
points(p, col="blue", cex=2)
plot(p, col="blue", alpha=.1, add=TRUE)
legend("topright", c("good", "bad"), col=c("red", "blue"), lty=1, lwd=3)

## the other way around does not work
## unless the original data was truly planar (e.g. derived from a map)
x <- densify(p, 250000)
y <- project(x, "+proj=longlat")
# bad
plot(y)
# good
lines(vd, col="red")
```

density	<i>Density plot</i>
---------	---------------------

Description

Create density plots of the cell values of a SpatRaster

Usage

```
## S4 method for signature 'SpatRaster'
density(x, maxcells=100000, plot=TRUE, main, ...)
```

Arguments

x	SpatRaster
maxcells	the maximum number of (randomly sampled) cells to be used for creating the plot
plot	if TRUE produce a plot, else return a density object
main	character. Caption of plot(s)
...	additional arguments passed to <code>plot</code>

Value

density plot (and a density object, returned invisibly if plot=TRUE)

Examples

```
logo <- rast(system.file("ex/logo.tif", package="terra"))
density(logo)
```

deprecated	<i>deprecated methods</i>
------------	---------------------------

Description

These methods are deprecated and will be removed in future versions

Usage

```
## S4 method for signature 'SpatRaster'
area(x, ...)
```

Arguments

x	object
...	additional arguments

depth	<i>depth of SpatRaster layers</i>
-------	-----------------------------------

Description

Get or set the depth of the layers of a SpatRaster. Experimental.

Usage

```
## S4 method for signature 'SpatRaster'
depth(x)

## S4 replacement method for signature 'SpatRaster'
depth(x)<-value
```

Arguments

x	SpatRaster
value	numeric vector

Value

numeric

See Also

[time](#)

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))

depth(s) <- 1:3
depth(s)
```

describe	<i>describe</i>
----------	-----------------

Description

Describe the properties of spatial data in a file as generated with the "GDALinfo" tool.

Usage

```
## S4 method for signature 'character'
describe(x, sds=FALSE, meta=FALSE, parse=FALSE, options="", print=FALSE, open_opt="")
```


Arguments

x	character. The name of a file with spatial data. Or a fully specified subdataset within a file such as "NETCDF:\AVHRR.nc\":NDVI"
sds	logical. If TRUE the description or metadata of the subdatasets is returned (if available)
meta	logical. Get the file level metadata instead
parse	logical. If TRUE, metadata for subdatasets is parsed into components (if meta=TRUE)
options	character. A vector of valid options (if meta=FALSE) including "json", "mm", "stats", "hist", "nogcp", "nomd", "norat", "noct", "nofl", "checksum", "proj4", "listmdd", "mdd <value>" where <value> specifies a domain or 'all', "wkt_format <value>" where value is one of 'WKT1', 'WKT2', 'WKT2_2015', or 'WKT2_2018', "sd <subdataset>" where <subdataset> is the name or identifier of a sub-dataset. See https://gdal.org/programs/gdalinfo.html . Ignored if sds=TRUE
print	logical. If TRUE, print the results
open_opt	character. Driver specific open options

Value

character (invisibly, if print=FALSE)

Examples

```
f <- system.file("ex/elev.tif", package="terra")
describe(f)
describe(f, meta=TRUE)
#g <- describe(f, options=c("json", "nomd", "proj4"))
#head(g)
```

diff *Lagged differences*

Description

Compute the difference between consecutive layers in a SpatRaster.

Usage

```
## S4 method for signature 'SpatRaster'
diff(x, lag=1, filename="", ...)
```

Arguments

x	SpatRaster
lag	postive integer indicating which lag to use
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))
d <- diff(s)
```

dimensions

Dimensions of a SpatRaster or SpatVector and related objects

Description

Get the number of rows (nrow), columns (ncol), cells (ncell), layers (nlyr), sources (nsrc), the size size (nlyr(x)*ncell(x)), or spatial resolution of a SpatRaster.

length returns the number of sub-datasets in a SpatRasterDataset or SpatVectorCollection.

For a SpatVector length(x) is the same as nrow(x).

You can also set the number of rows or columns or layers. When setting dimensions, all cell values are dropped.

Usage

```
## S4 method for signature 'SpatRaster'
ncol(x)

## S4 method for signature 'SpatRaster'
nrow(x)

## S4 method for signature 'SpatRaster'
nlyr(x)

## S4 method for signature 'SpatRaster'
ncell(x)

## S4 method for signature 'SpatRaster'
nsrc(x)

## S4 replacement method for signature 'SpatRaster,numeric'
ncol(x)<-value

## S4 replacement method for signature 'SpatRaster,numeric'
nrow(x)<-value

## S4 replacement method for signature 'SpatRaster,numeric'
nlyr(x)<-value
```

```
## S4 method for signature 'SpatRaster'  
res(x)  
  
## S4 replacement method for signature 'SpatRaster,numeric'  
res(x)<-value  
  
## S4 method for signature 'SpatRaster'  
xres(x)  
  
## S4 method for signature 'SpatRaster'  
yres(x)  
  
## S4 method for signature 'SpatVector'  
ncol(x)  
  
## S4 method for signature 'SpatVector'  
nrow(x)  
  
## S4 method for signature 'SpatVector'  
length(x)
```

Arguments

x	SpatRaster or SpatVector or related objects
value	For ncol and nrow: positive integer. For res: one or two positive numbers

Value

integer

See Also

[ext](#)

Examples

```
r <- rast()  
ncol(r)  
nrow(r)  
nlyr(r)  
dim(r)  
nsrc(r)  
ncell(r)  
  
rr <- c(r,r)  
nlyr(rr)  
nsrc(rr)  
ncell(rr)
```

```
nrow(r) <- 18
ncol(r) <- 36
# equivalent to
dim(r) <- c(18, 36)

dim(r)
dim(r) <- c(10, 10, 5)
dim(r)

xres(r)
yres(r)
res(r)

res(r) <- 1/120
# different xres and yres
res(r) <- c(1/120, 1/60)
```

direction

Direction

Description

The direction (azimuth) to or from the nearest cell that is not NA. The direction is expressed in radians, unless you use argument `degrees=TRUE`.

Usage

```
## S4 method for signature 'SpatRaster'
direction(x, from=FALSE, degrees=FALSE, filename="", ...)
```

Arguments

x	SpatRaster
filename	Character. Output filename (optional)
degrees	Logical. If FALSE (the default) the unit of direction is radians.
from	Logical. Default is FALSE. If TRUE, the direction from (instead of to) the nearest cell that is not NA is returned
...	Additional arguments as for writeRaster

Value

SpatRaster

See Also

[distance](#)

Examples

```

r <- rast(ncol=36,nrow=18, crs="+proj=merc")
values(r) <- NA
r[306] <- 1
b <- direction(r, degrees=TRUE)
plot(b)

crs(r) <- "+proj=longlat"
b <- direction(r)
plot(b)

```

disagg

Disaggregate raster cells or vector geometries

Description

SpatRaster: Create a **SpatRaster** with a higher resolution (smaller cells). The values in the new **SpatRaster** are the same as in the larger original cells.

SpatVector: Separate multi-objects (points, lines, polygons) into single objects.

Usage

```

## S4 method for signature 'SpatRaster'
disagg(x, fact, method="near", filename="", ...)

## S4 method for signature 'SpatVector'
disagg(x)

```

Arguments

x	SpatRaster or SpatVector
fact	positive integer. Aggregation factor expressed as number of cells in each direction (horizontally and vertically). Or two integers (horizontal and vertical aggregation factor) or three integers (when also aggregating over layers)
method	character. Either "near" for nearest or "bilinear" for bilinear interpolation
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[aggregate](#), [resample](#)

Examples

```
r <- rast(ncols=10, nrows=10)
rd <- disagg(r, fact=c(10, 2))
ncol(rd)
nrow(rd)
values(r) <- 1:ncell(r)
rd <- disagg(r, fact=c(4, 2))
```

distance

Geographic distance

Description**If x is a SpatRaster:**

If y is missing this method computes the distance, for all cells that are NA in SpatRaster x to the nearest cell that is not NA. If argument grid=TRUE, the distance is computed using a path that goes through the centers of the 8 neighboring cells.

If y is a SpatVector, the distance to that SpatVector is computed for all cells. For lines and polygons this is done after rasterization; and only the overlapping areas of the vector and raster are considered (for now).

The distance is always expressed in meter if the coordinate reference system is longitude/latitude, and in map units otherwise. Map units are typically meter, but inspect crs(x) if in doubt.

Results are more precise, sometimes much more precise, when using longitude/latitude rather than a planar coordinate reference system, as these distort distance.

If x is a SpatVector:

If y is missing, a distance matrix between all object in x is computed. An distance matrix object of class "dist" is returned.

If y is a SpatVector the geographic distance between all objects is computed (and a matrix is returned). If both sets have the same number of points, and pairwise=TRUE, the distance between each pair of objects is computed, and a vector is returned.

The distance is always expressed in meter, except when the coordinate reference system is longitude/latitude AND one of the SpatVector(s) consists of lines or polygons. In that case the distance is in degrees, and thus not very useful (this will be fixed soon). Otherwise, results are more precise, sometimes much more precise, when using longitude/latitude rather than a planar coordinate reference system, as these distort distance.

Usage

```
## S4 method for signature 'SpatRaster,missing'
distance(x, y, grid=FALSE, filename="", ...)
```

```
## S4 method for signature 'SpatRaster,SpatVector'
distance(x, y, filename="", ...)
```

```
## S4 method for signature 'SpatVector,ANY'
distance(x, y, sequential=FALSE, pairs=FALSE, symmetrical=TRUE)

## S4 method for signature 'SpatVector,SpatVector'
distance(x, y, pairwise=FALSE)

## S4 method for signature 'matrix,matrix'
distance(x, y, lonlat, pairwise=FALSE)

## S4 method for signature 'matrix,missing'
distance(x, y, lonlat, sequential=FALSE)
```

Arguments

x	SpatRaster, SpatVector, or two-column matrix with coordinates (x,y) or (lon,lat)
y	missing or SpatVector, or two-column matrix
grid	logical. If TRUE, distance is computed using a path that goes through the centers of the 8 neighboring cells
filename	character. Output filename
...	additional arguments for writing files as in writeRaster
sequential	logical. If TRUE, the distance between sequential geometries is returned
pairwise	logical. If TRUE and if x and y have the same size (number of rows), the pairwise distances are returned instead of the distances between all elements
lonlat	logical. If TRUE the coordinates are interpreted as angular (longitude/latitude). If FALSE they are interpreted as planar
pairs	logical. If TRUE a "from", "to", "distance" matrix is returned
symmetrical	logical. If TRUE and pairs=TRUE, the distance between a pair is only included once. The distance between geometry 1 and 3 is included, but the (same) distance between 3 and 1 is not

Value

SpatRaster or numeric or matrix or distance matrix (object of class "dist")

Note

The distance unit is in meters.

A distance matrix can be coerced into a matrix with `as.matrix`

References

Karney, C.F.F., 2013. Algorithms for geodesics, *J. Geodesy* 87: 43-55. doi:10.1007/s00190-012-0578-z.

Examples

```

#lonlat
r <- rast(ncols=36, nrows=18, crs="+proj=longlat +datum=WGS84")
r[500] <- 1
d <- distance(r)
plot(d / 100000)

#planar
rr <- rast(ncols=36, nrows=18, crs="+proj=utm +zone=1 +datum=WGS84")
rr[500] <- 1
d <- distance(rr)

p1 <- vect(rbind(c(0,0), c(90,30), c(-90,-30)), crs="+proj=longlat +datum=WGS84")
dp <- distance(r, p1)

d <- distance(p1)
d
as.matrix(d)

p2 <- vect(rbind(c(30,-30), c(25,40), c(-9,-3)), crs="+proj=longlat +datum=WGS84")
dd <- distance(p1, p2)
dd
pd <- distance(p1, p2, pairwise=TRUE)
pd
pd == diag(dd)

# polygons, lines
crs <- "+proj=utm +zone=1"
p1 <- vect("POLYGON ((0 0, 8 0, 8 9, 0 9, 0 0))", crs=crs)
p2 <- vect("POLYGON ((5 6, 15 6, 15 15, 5 15, 5 6))", crs=crs)
p3 <- vect("POLYGON ((2 12, 3 12, 3 13, 2 13, 2 12))", crs=crs)
p <- rbind(p1, p2, p3)
L1 <- vect("LINESTRING(1 11, 4 6, 10 6)", crs=crs)
L2 <- vect("LINESTRING(8 14, 12 10)", crs=crs)
L3 <- vect("LINESTRING(1 8, 12 14)", crs=crs)
lns <- rbind(L1, L2, L3)
pts <- vect(cbind(c(7,10,10), c(3,5,6)), crs=crs)

distance(p1,p3)
distance(p)
distance(p,pts)
distance(p,lns)
distance(pts,lns)

```


Description

Create the dots for a dot-density map and add these to the current map. Dot-density maps are made to display count data. For example of population counts, where each dot represents n persons. The dots are returned as a `SpatVector`. If there is an active graphics device, the dots are added to it with [points](#).

Usage

```
## S4 method for signature 'SpatVector'
dots(x, field, size, ...)
```

Arguments

<code>x</code>	<code>SpatVector</code>
<code>field</code>	character of numeric indicating field name. Or numeric vector of the same length as <code>x</code>
<code>size</code>	positive number indicating the number of cases associated with each dot
<code>...</code>	graphical arguments passed to <code>points</code>

Value

`SpatVector` (invisibly)

See Also

[plot](#), [cartogram](#), [points](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
v$population <- 1000*(1:12)^2
plot(v, lwd=3, col="light gray", border="white")
d <- dots(v, "population", 1000, col="red", cex=.75)
lines(v)
d
```

draw

Draw a polygon, line, extent, or points

Description

Draw on a plot (map) to get a `SpatVector` or `SpatExtent` object for later use. After calling the function, start clicking on the map. When you are done, press ESC. You can also preset the maximum number of clicks.

Usage

```
## S4 method for signature 'character'
draw(x="extent", col="red", lwd=2, id=FALSE, n=1000, ...)
```

Arguments

x	character. The type of object to draw. One of "extent", "polygon", "line", or "points"
col	the color to be used
lwd	the width of the lines to be drawn
id	logical. If TRUE, a numeric ID is shown on the map
n	the maximum number of clicks (does not apply when x=="extent" in which case n is always 2)
...	additional graphics arguments for drawing

Value

SpatVector or SpatExtent

See Also

[click](#)

erase

Erase parts of a SpatVector object

Description

Erase parts of a SpatVector with another SpatVector or with a SpatExtent. You can also erase (parts of) polygons with the other polygons of the same SpatVector.

Usage

```
## S4 method for signature 'SpatVector,SpatVector'
erase(x, y)
```

```
## S4 method for signature 'SpatVector,missing'
erase(x)
```

```
## S4 method for signature 'SpatVector,SpatExtent'
erase(x, y)
```

Arguments

x	SpatVector
y	SpatVector or SpatExtent

Value

SpatVector or SpatExtent

See Also

[crop](#) and [intersect](#) for the inverse.

The equivalent for SpatRaster is [mask](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)

# polygons with polygons or extent

e <- ext(5.6, 6, 49.55, 49.7)
x <- erase(v, e)

p <- vect("POLYGON ((5.8 49.8, 6 49.9, 6.15 49.8, 6 49.6, 5.8 49.8))")
y <- erase(v, p)

# lines with polygons
lns <- as.lines(rast(v, ncol=10, nrow=10))[12:22]
eln <- erase(lns, v)
plot(v)
lines(lns, col='blue', lwd=4, lty=3)
lines(eln, col='red', lwd=2)

## self-erase
h <- convHull(v[-12], "NAME_1")
he <- erase(h)
plot(h, lwd=2, border="red", lty=2)
lines(he, col="gray", lwd=3)
```

expanse	<i>Get the expanse (area) of individual polygons or for all (summed) raster cells</i>
---------	---

Description

Compute the area covered by polygons or for all raster cells that are not NA.

This method computes areas for longitude/latitude rasters, as the size of the cells is constant in degrees, but not in square meters. But it can also be important if the coordinate reference system is planar, but not equal-area.

For vector data, the best way to compute area is to use the longitude/latitude CRS. This is contrary to (erroneous) popular belief that suggest that you should use a planar coordinate reference system. This is done automatically, if transform=TRUE.

Usage

```
## S4 method for signature 'SpatRaster'
expansE(x, unit="m", transform=TRUE, byValue=FALSE)

## S4 method for signature 'SpatVector'
expansE(x, unit="m", transform=TRUE)
```

Arguments

x	SpatRaster or SpatVector
unit	character. One of "m", "km", or "ha"
transform	logical. If TRUE, planar CRS are transformed to lon/lat for accuracy
byValue	logical. If TRUE, the area for each unique cell value is returned

Value

numeric. If x has no values, the total size of all cells. Otherwise, the total area size of all cells that are not NA, expressed in square meters, square kilometers, or hectares.

If byValue=TRUE a matrix is returned with three columns (layer, value, area)

See Also

[cellSize](#) for a the size of individual cells of a raster, that can be summed with [global](#) or with [zonal](#) to get the area for different categories.

Examples

```
### SpatRaster
r <- rast(nrows=18, ncols=36)
v <- 1:ncell(r)
v[200:400] <- NA
values(r) <- v

# summed area in km2
expansE(r, unit="km")

# all cells
expansE(rast(r), unit="km")

r <- rast(ncols=90, nrows=45, ymin=-80, ymax=80)
m <- project(r, "+proj=merc")

expansE(m, unit="km")
expansE(m, unit="km", transform=FALSE)

m2 <- c(m, m)
values(m2) <- cbind(c(1,2,NA,NA), c(11:14))
expansE(m2, unit="km", byValue=TRUE)
```

```
### SpatVector
v <- vect(system.file("ex/lux.shp", package="terra"))

a <- expanse(v)
a
sum(a)
```

 ext

Create, get or set a SpatExtent

Description

Get a SpatExtent of a SpatRaster, or coordinates from such an object. Or create a SpatExtent from a vector (length=4; order= xmin, xmax, ymin, ymax)

See [set.ext](#) to set the extent in place.

Usage

```
## S4 method for signature 'SpatRaster'
ext(x, cells=NULL)

## S4 method for signature 'SpatVector'
ext(x)

## S4 method for signature 'numeric'
ext(x, ...)

## S4 replacement method for signature 'SpatRaster,SpatExtent'
ext(x)<-value

## S4 replacement method for signature 'SpatRaster,numeric'
ext(x)<-value

## S4 method for signature 'SpatExtent'
x$name

## S4 replacement method for signature 'SpatExtent'
x$name<-value
```

Arguments

x	SpatRaster
cells	postive integer (cell) numbers to subset the extent to area covered by these cells

value	SpatExtent, or numeric vector of length four (xmin, xmax, ymin, ymax), or a single number with the \$ method
name	character, one of xmin, xmax, ymin, or ymax
...	if x is a single numeric value, additional numeric values for xmax, ymin, and ymax

Value

A [SpatExtent](#) object.

Examples

```
r <- rast()
e <- ext(r)
as.vector(e)
as.character(e)

ext(r) <- c(0, 2.5, 0, 1.5)
r
er <- ext(r)

round(er)
# go "in"
floor(er)
# go "out"
ceiling(er)

ext(r) <- e
```

extend

Extend

Description

Enlarge the spatial extent of a [SpatRaster](#). See [crop](#) if you (also) want to remove rows or columns.

You can also enlarge a [SpatExtent](#) with this method, or with algebraic notation (see examples)

Usage

```
## S4 method for signature 'SpatRaster'
extend(x, y, snap="near", filename="", overwrite=FALSE, ...)

## S4 method for signature 'SpatExtent'
extend(x, y)
```

Arguments

x	SpatRaster or SpatExtent
y	If x is a SpatRaster, y should be a SpatExtent, or an object from which it can be extracted (such as SpatRaster and SpatVector objects). Alternatively, you can provide two positive integers indicating the number of rows and columns that need to be added at each side (or a single positive integer when the number of rows and columns is equal) If x is a SpatExtent, y should be a numeric vector of 1, 2, or 4 elements
snap	character. One of "near", "in", or "out". Used to align y to the geometry of x
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
...	additional arguments for writing files as in writeRaster

Value

SpatRaster or SpatExtent

See Also

[crop](#), [merge](#), [ext](#)

Examples

```
r <- rast(xmin=-150, xmax=-120, ymin=30, ymax=60, ncols=36, nrows=18)
values(r) <- 1:ncell(r)
e <- ext(-180, -100, 40, 70)
re <- extend(r, e)

# extend with a number of rows and columns (at each side)
re2 <- extend(r, c(2,10))

# SpatExtent
e <- ext(r)
e
extend(e, 10)
extend(e, c(10, -10, 0, 20))
```

extract

Extract values from a SpatRaster

Description

Extract values from a SpatRaster for a set of locations. The locations can be a SpatVector (points, lines, polygons), a matrix with (x, y) or (longitude, latitude – in that order!) coordinates, or a vector with cell numbers.

When argument y is a SpatVector the first column has the ID (record number) of the SpatVector used (unless you set ID=FALSE).

Usage

```
## S4 method for signature 'SpatRaster,SpatVector'
extract(x, y, fun=NULL, method="simple", cells=FALSE, xy=FALSE,
        ID=TRUE, weights=FALSE, exact=FALSE, touches=is.lines(y),
        layer=NULL, bind=FALSE, ...)

## S4 method for signature 'SpatRaster,SpatExtent'
extract(x, y, cells=FALSE, xy=FALSE)

## S4 method for signature 'SpatRaster,matrix'
extract(x, y, ...)

## S4 method for signature 'SpatRaster,numeric'
extract(x, y, ...)

## S4 method for signature 'SpatVector,SpatVector'
extract(x, y, ...)
```

Arguments

x	SpatRaster or SpatVector of polygons
y	SpatVector (for points, lines, polygons), or for points, 2-column matrix or data.frame (x, y) or (lon, lat), or a vector with cell numbers
fun	function to summarize the data by geometry. If weights=TRUE or exact=TRUE only mean, sum, min and max are accepted).
method	character. method for extracting values with points ("simple" or "bilinear"). With "simple" values for the cell a point falls in are returned. With "bilinear" the returned values are interpolated from the values of the four nearest raster cells
cells	logical. If TRUE the cell numbers are also returned, unless fun is not NULL. Also see cells
xy	logical. If TRUE the coordinates of the cells are also returned, unless fun is not NULL. Also see xyFromCell
ID	logical. Should an ID column be added? If so, the first column returned has the IDs (record numbers) of input SpatVector y
weights	logical. If TRUE and y has polygons, the approximate fraction of each cell that is covered is returned as well, for example to compute a weighted mean
exact	logical. If TRUE and y has polygons, the exact fraction of each cell that is covered is returned as well, for example to compute a weighted mean
touches	logical. If TRUE, values for all cells touched by lines or polygons are extracted, not just those on the line render path, or whose center point is within the polygon. Not relevant for points; and always considered TRUE when weights=TRUE or exact=TRUE
layer	character or numeric to select the layer to extract from for each geometry. If layer is a character it can be a name in y or a vector of layer names. If it is numeric, it must be integer values between 1 and nlyr(x)

`bind` logical. If TRUE, a `SpatVector` is returned consisting of the input `SpatVector` `y` and the `cbind`-ed extracted values

`...` additional arguments to `fun` if `y` is a `SpatVector`. For example `na.rm=TRUE`. Or arguments passed to the `SpatRaster`, `SpatVector` method if `y` is a matrix (such as the `method` and `cells` arguments)

Value

`data.frame` or `SpatVector`

See Also

[values](#)

Examples

```
r <- rast(ncols=5, nrows=5, xmin=0, xmax=5, ymin=0, ymax=5)
values(r) <- 1:25
xy <- rbind(c(0.5,0.5), c(2.5,2.5))
p <- vect(xy, crs="+proj=longlat +datum=WGS84")
```

```
extract(r, xy)
extract(r, p)
```

```
r[1,]
r[5]
r[,5]
```

```
r[c(0:2, 99:101)]
```

```
f <- system.file("ex/meuse.tif", package="terra")
r <- rast(f)
```

```
xy <- cbind(179000, 330000)
xy <- rbind(xy-100, xy, xy+1000)
extract(r, xy)
```

```
p <- vect(xy)
g <- geom(p)
g
```

```
extract(r, p)
```

```
x <- r + 10
extract(x, p)
```

```
i <- cellFromXY(r, xy)
x[i]
r[i]
```

```
y <- c(x,x*2,x*3)
y[i]
```

```

## extract with a polygon
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
v <- v[1:2,]
z <- rast(v, resolution=.1, names="test")
values(z) <- 1:ncell(z)

rf <- system.file("ex/elev.tif", package="terra")
x <- rast(rf)
extract(x, v, mean, na.rm=TRUE)

e <- extract(z, v)
e
tapply(e[,2], e[,1], mean, na.rm=TRUE)

x <- c(z, z*2, z/3)
names(x) <- letters[1:3]

e <- extract(x, v)
de <- data.frame(e)
aggregate(de[,2:4], de[,1,drop=FALSE], mean)

```

extremes

Get or compute the minimum and maximum cell values

Description

The minimum and maximum value of a `SpatRaster` are returned or computed (from a file on disk if necessary) and stored in the object.

Usage

```

## S4 method for signature 'SpatRaster'
minmax(x)
## S4 method for signature 'SpatRaster'
hasMinMax(x)
## S4 method for signature 'SpatRaster'
setMinMax(x, force=FALSE)

```

Arguments

x	<code>SpatRaster</code>
force	logical. If TRUE min and max values are recomputed even if already available

Value

minmax: numeric matrix of minimum and maximum cell values by layer

hasMinMax: logical indicating whether the min and max values are available.

setMinMax: nothing. Used for the side-effect of computing the minimum and maximum values of a SpatRaster

Examples

```
r <- rast(system.file("ex/elev.tif", package="terra"))
minmax(r)
```

factors

Categorical rasters

Description

A SpatRaster layer can be a categorical variable (factor). Like [factors](#), categories are stored as indices (integer values) that have an associated label.

The categories can be inspected with `levels` and `cats`. They are represented by a `data.frame` that must have two or more columns, the first one identifying the cell values and the other column(s) providing the category labels.

If there are multiple columns with categories, you can set the "active" category.

`cats` returns the entire `data.frame`, whereas `levels` only return two columns: the index and the active category.

To set categories for the first layers you can provide `levels<-` with a `data.frame` or a list with a `data.frame`. To set categories for multiple layers you can provide `levels<-` with a list with one element for each layer. Use `categories` to set the categories for a specific layer or specific layers.

Usage

```
## S4 method for signature 'SpatRaster'
levels(x)
```

```
## S4 replacement method for signature 'SpatRaster'
levels(x)<-value
```

```
## S4 method for signature 'SpatRaster'
cats(x, layer, active=FALSE)
```

```
## S4 method for signature 'SpatRaster'
categories(x, layer=1, value, index)
```

```
## S4 method for signature 'SpatRaster'
is.factor(x)
```

```
## S4 method for signature 'SpatRaster'
as.factor(x)
```

Arguments

x	SpatRaster
layer	positive integer, the layer number or name
active	logical. If TRUE, only return the active category
value	a data.frame (ID, category) or vector with category names
index	positive integer, indicating the column in data.frame value to be used as the (active) category, (not counting the first column with the cell values)

Value

list of data.frames (levels, cats) or logical (is.factor)

See Also

[activeCat](#), [catalyze](#), [set.cats](#)

Examples

```
set.seed(0)
r <- rast(nrows=10, ncols=10)
values(r) <- sample(3, ncell(r), replace=TRUE)
is.factor(r)

cls <- data.frame(id=1:3, cover=c("forest", "water", "urban"))
levels(r) <- cls
is.factor(r)
r

plot(r, col=c("green", "blue", "light gray"))
text(r, digits=3, cex=.75, halo=TRUE)

# raster starts at 3
x <- r + 2
is.factor(x)

# Multiple categories
d <- data.frame(id=3:5, cover=cls[,2], letters=letters[1:3], value=10:12)
levels(x) <- d
x

# get current index
activeCat(x)
# set index
activeCat(x) <- 3
activeCat(x)
```

```

activeCat(x) <- "letters"
plot(x, col=c("green", "blue", "light gray"))
text(x, digits=3, cex=.75, halo=TRUE)

r <- as.numeric(x)
r

p <- as.polygons(x)
plot(p, "letters", col=c("green", "blue", "light gray"))

```

fillHoles

Remove holes from polygons

Description

Remove the holes in SpatVector polygons. If `inverse=TRUE` the holes are returned (as polygons).

Usage

```

## S4 method for signature 'SpatVector'
fillHoles(x, inverse=FALSE)

```

Arguments

<code>x</code>	SpatVector
<code>inverse</code>	logical. If TRUE the holes are returned as polygons

Value

SpatVector

Examples

```

x <- rbind(c(-10,0), c(140,60), c(160,0), c(140,-55))
hole <- rbind(c(80,0), c(105,13), c(120,2), c(105,-13))

z <- rbind(cbind(object=1, part=1, x, hole=0),
          cbind(object=1, part=1, hole, hole=1))
colnames(z)[3:4] <- c('x', 'y')
p <- vect(z, "polygons", atts=data.frame(id=1))
p

f <- fillHoles(p)
g <- fillHoles(p, inverse=TRUE)

plot(p, lwd=16, border="gray", col="light yellow")
polys(f, border="blue", lwd=3, density=4, col="orange")
polys(g, col="white", lwd=3)

```

fillTime	<i>Fill time gaps in a SpatRaster</i>
----------	---------------------------------------

Description

Add empty layers in between existing layers such that the time step between each layer is the same. See [approximate](#) to estimate values for these layer (and other missing values)

Usage

```
## S4 method for signature 'SpatRaster'  
fillTime(x, filename="", ...)
```

Arguments

x	SpatRaster
filename	character. Output filename
...	list with named options for writing files as in writeRaster

Value

SpatRaster

See Also

[approximate](#)

Examples

```
r <- rast(system.file("ex/logo.tif", package="terra"))  
s <- c(r, r)  
time(s) <- as.Date("2001-01-01") + c(0:2, 5:7)  
time(s)  
ss <- fillTime(s)  
time(ss)  
  
a <- approximate(ss)
```

flip *Flip or reverse a raster*

Description

Flip the values of a `SpatRaster` by inverting the order of the rows (`vertical=TRUE`) or the columns (`vertical=FALSE`).

`rev` is the same as a horizontal *and* a vertical flip.

Usage

```
## S4 method for signature 'SpatRaster'  
flip(x, direction="vertical", filename="", ...)
```

```
## S4 method for signature 'SpatVector'  
flip(x, direction="vertical")
```

```
## S4 method for signature 'SpatRaster'  
rev(x)
```

Arguments

<code>x</code>	<code>SpatRaster</code> or <code>SpatVector</code>
<code>direction</code>	character. Should (partially) match "vertical" to flip by rows, or "horizontal" to flip by columns
<code>filename</code>	character. Output filename
<code>...</code>	additional arguments for writing files as in writeRaster

Value

`SpatRaster`

See Also

[trans](#), [rotate](#)

Examples

```
r <- rast(nrow=18, ncol=36)  
m <- matrix(1:ncell(r), nrow=18)  
values(r) <- as.vector(t(m))  
rx <- flip(r, direction="h")  
  
values(r) <- as.vector(m)  
ry <- flip(r, direction="v")  
  
v <- rev(r)
```

focal	<i>Focal values</i>
-------	---------------------

Description

Calculate focal ("moving window") values for each cell.

Usage

```
## S4 method for signature 'SpatRaster'
focal(x, w=3, fun="sum", ..., na.policy="all", fillvalue=NA,
expand=FALSE, silent=TRUE, filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster
w	window. The window can be defined as one (for a square) or two numbers (row, col); or with an odd-sized weights matrix. See Details.
fun	function that takes multiple numbers, and returns a numeric vector (one or multiple numbers). For example mean, modal, min or max
...	additional arguments passed to fun such as na.rm
na.policy	character. Can be used to determine the cells of x for which focal values should be computed. Must be one of "all" (compute for all cells), "only" (only for cells that are NA) or "omit" (skip cells that are NA). Note that the value of this argument does not affect which cells around each focal cell are included in the computations (use na.rm=TRUE to ignore cells that are NA for that)
fillvalue	numeric. The value of the cells in the virtual rows and columns outside of the raster
expand	logical. If TRUE The value of the cells in the virtual rows and columns outside of the raster are set to be the same as the value on the border. Only available for "build-in" funs such as mean, sum, min and max
silent	logical. If TRUE error messages are printed that may occur when trying fun to determine the length of the returned value. This can be useful in debugging a fun that does not work
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	additional arguments for writing files as in writeRaster

Details

focal The window used must have odd dimensions. If you need even sides, you can use a matrix and add a column or row with weights of zero.

Window values are typically 0 or 1, or a value between 0 and 1 if you are using a rectangular area and/or the "sum" function. They can also be NA; these are ignored in the computation. That can be useful to compute, for example, the minimum or maximum value for a non-rectangular area.

The "mean" function is a special case, as zero weights are ignored automatically.

The "sum" function returns NA if all focal cells are NA and `na.rm=TRUE`. R would normally return a zero in these cases. See the difference between `focal(x, fun=sum, na.rm=TRUE)` and `focal(x, fun=\(i) sum(i, na.rm=TRUE))`

Example weight matrices

Laplacian filter: `filter=matrix(c(0,1,0,1,-4,1,0,1,0), nrow=3)`

Sobel filters (for edge detection): `fx=matrix(c(-1,-2,-1,0,0,0,1,2,1), nrow=3)` `fy=matrix(c(1,0,-1,2,0,-2,1,0,-1), nrow=3)`

Value

SpatRaster

See Also

[focalMat](#), [focalValues](#), [focal3D](#), [focalCor](#), [focalReg](#), [focalCpp](#)

Examples

```
r <- rast(ncols=10, nrows=10, ext(0, 10, 0, 10))
values(r) <- 1:ncell(r)

f <- focal(r, w=3, fun=function(x, ...) quantile(x, c(.25, .5, .75), ...), na.rm=TRUE)

f <- focal(r, w=3, fun="mean")

# the following two statements are equivalent:
a <- focal(r, w=matrix(1/9, nc=3, nr=3))
b <- focal(r, w=3, fun=mean, na.rm=FALSE)

# but this is different
d <- focal(r, w=3, fun=mean, na.rm=TRUE)

## illustrating the effect of different
## combinations of na.rm and na.policy
v <- vect(system.file("ex/lux.shp", package="terra"))
r <- rast(system.file("ex/elev.tif", package="terra"))
r[45:50, 45:50] <- NA

# also try "mean" or "min"
f <- "sum"
# na.rm=FALSE
plot(focal(r, 5, f) , fun=lines(v))

# na.rm=TRUE
plot(focal(r, 5, f, na.rm=TRUE), fun=lines(v))

# only change cells that are NA
plot(focal(r, 5, f, na.policy="only", na.rm=TRUE), fun=lines(v))

# do not change cells that are NA
```

```
plot(focal(r, 5, f, na.policy="omit", na.rm=TRUE), fun=lines(v))

# does not do anything
# focal(r, 5, f, na.policy="only", na.rm=FALSE)
```

focal3D

Three-dimensional focal values

Description

Calculate focal ("moving window") values for the three-dimensional neighborhood (window) of focal cells. See [focal](#) for two-dimensional focal computation.

Usage

```
## S4 method for signature 'SpatRaster'
focal3D(x, w=3, fun=mean, ..., na.policy="all", fillvalue=NA, pad=FALSE,
padvalue=fillvalue, expand=FALSE, silent=TRUE,
filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster
w	window. A rectangular prism (cuboid) defined by three numbers or by a three-dimensional array. The values are used as weights, and are usually zero, one, NA, or fractions. The window used must have odd dimensions. If you desire to use even sides, you can use an array, and pad the values with rows and/or columns that contain only NAs.
fun	function that takes multiple numbers, and returns one or multiple numbers for each focal area. For example mean, modal, min or max
...	additional arguments passed to fun such as na.rm
na.policy	character. Can be used to determine the cells of x for which focal values should be computed. Must be one of "all" (compute for all cells), "only" (only for cells that are NA) or "omit" (skip cells that are NA). Note that the value of this argument does not affect which cells around each focal cell are included in the computations (use na.rm=TRUE to ignore cells that are for that)
fillvalue	numeric. The value of the cells in the virtual rows and columns outside of the raster
pad	logical. Add virtual layers before the first and after the last layer
padvalue	numeric. The value of the cells in the virtual layers
expand	logical. Add virtual layers before the first or after the last layer that are the same as the first or last layers. If TRUE, arguments pad and padvalue are ignored
silent	logical. If TRUE error messages are printed that may occur when trying fun to determine the length of the returned value. This can be useful in debugging a function passed to fun that does not work

filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[focal](#)**Examples**

```
r <- rast(system.file("ex/logo.tif", package="terra"))
x <- focal3D(r, c(5,5,3), na.rm=TRUE)

a <- array(c(0,1,0,1,1,1,0,1,0, rep(1,9), 0,1,0,1,1,1,0,1,0), c(3,3,3))
a[a==0] <- NA
z <- focal3D(r, a, na.rm=TRUE)
```

focalCor

*Focal function across two layers***Description**

Calculate values such as a correlation coefficient for focal regions in two neighboring layers. A function is applied to the first and second layer, then to the second and third layer, etc.

Usage

```
## S4 method for signature 'SpatRaster'
focalCor(x, w=3, fun, ..., fillvalue=NA,
filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster with at least two layers
w	window. The window can be defined as one (for a square) or two numbers (row, col); or with an odd-sized weights matrix. See the Details section in focal
fun	A function with at least two arguments (one for each layer)
...	additional arguments for fun
fillvalue	numeric. The value of the cells in the virtual rows and columns outside of the raster
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[layerCor](#), [focalReg](#), [focal](#)**Examples**

```
r <- rast(system.file("ex/logo.tif", package="terra"))
set.seed(0)
r[[1]] <- flip(r[[1]], "horizontal")
r[[2]] <- flip(r[[2]], "vertical") + init(rast(r,1), runif)
r[[3]] <- init(rast(r,1), runif)

# suppress warning "the standard deviation is zero"
suppressWarnings(x <- focalCor(r, w=5, cor))

# this warning does not occur when using a larger window
x <- focalCor(r, w=9, function(x, y) cor(x, y))
plot(x)
```

focalCpp

*Compute focal values with an iterating C++ function***Description**

Calculate focal values with a C++ function that iterates over cells to speed up computations by avoiding an R loop (with `apply`).

See [focal](#) for an easier to use method.

Usage

```
## S4 method for signature 'SpatRaster'
focalCpp(x, w=3, fun, ..., fillvalue=NA,
silent=TRUE, filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster
w	window. The window can be defined as one (for a square) or two numbers (row, col); or with an odd-sized weights matrix. See the Details section in focal
fun	cppFunction that iterates over cells. For C++ functions that operate on a single focal window, or for R functions use focal instead. The function must have at least three arguments. The first argument can have any name, but it must be a <code>Rcpp::NumericVector</code> , <code>Rcpp::IntegerVector</code> or a <code>std::vector<double></code> . This is the container that receives the focal values. The other two arguments

	ni and wi must be of type size_t. ni represents the number of cells and nw represents the size of (number of elements in) the window
...	additional arguments to fun
fillvalue	numeric. The value of the cells in the virtual rows and columns outside of the raster
silent	logical. If TRUE error messages are printed that may occur when trying fun to determine the length of the returned value. This can be useful in debugging a fun that does not work
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[focal](#), [focalValues](#)**Examples**

```
## Not run:
library(Rcpp)
cppFunction(
  'NumericVector sum_and_multiply(NumericVector x, double m, size_t ni, size_t nw) {
    NumericVector out(ni);
    // loop over cells
    size_t start = 0;
    for (size_t i=0; i<ni; i++) {
      size_t end = start + nw;
      // compute something for a window
      double v = 0;
      // loop over the values of a window
      for (size_t j=start; j<end; j++) {
        v += x[j];
      }
      out[i] = v * m;
      start = end;
    }
    return out;
  }'
)

nr <- nc <- 10
r <- rast(ncols=nc, nrows=nr, ext= c(0, nc, 0, nr))
values(r) <- 1:ncell(r)

raw <- focalCpp(r, w=3, fun=sum_and_multiply, fillvalue=0, m=10)
```

```

# same as
f1 <- focal(r, w=3, fun=sum, fillvalue=0) *10
all(values(f1) == values(raw))

# and as
ffun <- function(x, m) { sum(x) * m }
f2 <- focal(r, w=3, fun=ffun, fillvalue=0, m=10)

# You can also use an R function with focalCpp but this
# is not recommended

R_sm_iter <- function(x, m, ni, nw) {
  out <- NULL
  for (i in 1:ni) {
    start <- (i-1) * nw + 1
    out[i] <- sum(x[start:(start+nw-1)]) * m
  }
  out
}

fr <- focalCpp(r, w=3, fun=R_sm_iter, fillvalue=0, m=10)

## End(Not run)

```

focalMat

Focal weights matrix

Description

Make a focal ("moving window") weight matrix for use in the `focal` function. The sum of the values adds up to one.

Usage

```
focalMat(x, d, type=c('circle', 'Gauss', 'rectangle'), fillNA=FALSE)
```

Arguments

<code>x</code>	SpatRaster
<code>d</code>	numeric. If <code>type=circle</code> , the radius of the circle (in units of the crs). If <code>type=rectangle</code> the dimension of the rectangle (one or two numbers). If <code>type=Gauss</code> the size of sigma, and optionally another number to determine the size of the matrix returned (default is $3*\text{sigma}$)
<code>type</code>	character indicating the type of filter to be returned
<code>fillNA</code>	logical. If TRUE, zeros are set to NA such that they are ignored in the computations. Only applies to <code>type="circle"</code>

Value

matrix that can be used with [focal](#)

Examples

```
r <- rast(ncols=180, nrows=180, xmin=0)
focalMat(r, 2, "circle")

focalMat(r, c(2,3), "rect")

# Gaussian filter for square cells
gf <- focalMat(r, 1, "Gauss")
```

focalReg

Focal regression

Description

Calculate the coefficients for a focal ("moving window") OLS regression model.

Usage

```
## S4 method for signature 'SpatRaster'
focalReg(x, w=3, na.rm=TRUE,
fillvalue=NA, filename="", ...)
```

Arguments

x	SpatRaster with at least two layers. The first is the "Y" (dependent) variable and the remainder are the "X" (independent) variables
w	window. The window can be defined as one (for a square) or two numbers (row, col); or with an odd-sized weights matrix. See the Details section in focal
na.rm	logical. Should missing values be removed?
fillvalue	numeric. The value of the cells in the virtual rows and columns outside of the raster
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[focal](#), [focalValues](#)

Examples

```
r <- rast(ncols=10, nrows=10, ext(0, 10, 0, 10))
values(r) <- 1:ncell(r)
x <- c(r, init(r, runif) * r)
f <- focalReg(x, 3)
```

focalValues

Get focal values

Description

Get a matrix in which each row had the focal values of a cell. These are the values of a cell and a rectangular window around it.

Usage

```
## S4 method for signature 'SpatRaster'
focalValues(x, w=3, row=1, nrows=nrow(x), fill=NA)
```

Arguments

x	SpatRaster or SpatVector
w	window. The window can be defined as one (for a square) or two odd numbers (row, col); or with an odd sized matrix
row	positive integer. Row number to start from, should be between 1 and nrow(x)
nrows	positive integer. How many rows?
fill	numeric used as values for imaginary cells outside the raster

Value

matrix

Examples

```
r <- rast(ncol=4, nrow=4, crs="+proj=utm +zone=1 +datum=WGS84")
values(r) <- 1:ncell(r)
focalValues(r)
```

freq	<i>Frequency table</i>
------	------------------------

Description

Frequency table of the values of a `SpatRaster`. NAs are not counted unless `value=NA`.

Usage

```
## S4 method for signature 'SpatRaster'  
freq(x, digits=0, value=NULL, bylayer=TRUE, usenames=FALSE)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>digits</code>	integer. Used for rounding the values before tabulation. Ignored if NA
<code>value</code>	numeric. An optional single value to only count the number of cells with that value. This value can be NA
<code>bylayer</code>	logical. If TRUE tabulation is done by layer
<code>usenames</code>	logical. If TRUE layers are identified by their names instead of their numbers. Only relevant if <code>bylayer</code> is TRUE

Value

A data.frame with 3 columns (layer, value, count) unless `bylayer=FALSE` in which case a data.frame with two columns is returned (value, count).

Examples

```
r <- rast(nrows=10, ncols=10)  
set.seed(2)  
values(r) <- sample(5, ncell(r), replace=TRUE)  
  
freq(r)  
  
x <- c(r, r/3)  
freq(x, bylayer=FALSE)  
freq(x)  
  
freq(x, digits=1)  
freq(x, digits=-1)  
  
freq(x, value=5)
```

gaps	<i>Find gaps between polygons</i>
------	-----------------------------------

Description

Get the gaps between polygons of a `SpatVector`

Usage

```
## S4 method for signature 'SpatVector'  
gaps(x)
```

Arguments

x `SpatVector`

Value

`SpatVector`

See Also

[sharedPaths](#), [topology](#), and [fillHoles](#) to get or remove polygon holes

Examples

```
f <- system.file("ex/lux.shp", package="terra")  
v <- vect(f)  
h <- convHull(v[-12], "NAME_1")  
g <- gaps(h)
```

gdal	<i>GDAL version, supported file formats, and cache size</i>
------	---

Description

Set the GDAL warning level or get a data frame with the available GDAL drivers (file formats), or, if `warn=NA` and `drivers=FALSE`, you get the version numbers of one or all of the GDAL, PROJ and GEOS libraries.

GDAL is the software library that terra builds on to read and write spatial data and for some raster data processing. PROJ is used for transformation of coordinates ("projection") and GEOS is used for geometric operations with vector data.

Usage

```
gdal(warn=NA, drivers=FALSE, lib="gdal")
gdalCache(size=NA)
setGDALconfig(option, value="")
getGDALconfig(option)
```

Arguments

warn	If NA and drivers=FALSE, the version of the library specified by lib is returned. Otherwise, the value should be an integer between 1 and 4 representing the level of GDAL warnings and errors that are passed to R. 1 = warnings and errors; 2 = errors only (recoverable errors as a warning); 3 = irrecoverable errors only; 4 = ignore all errors and warnings. The default setting is 3
drivers	logical. If TRUE a data.frame with the raster and vector data formats that are available.
lib	character. "gdal", "proj", or "geos", or any other value to get the versions numbers of all three
size	numeric. The new cache size in MB
option	character. GDAL configuration option name, or a "name=value" string (in which case the value argument is ignored)
value	character. value for GDAL configuratoin option. Use "" to reset it to its default value

Value

character

See Also

[describe](#) for file-level metadata "GDALinfo"

Examples

```
gdal()
gdal(2)
head(gdal(drivers=TRUE))
```

geom

Get the geometry (coordinates) of a SpatVector

Description

Get the geometry of a SpatVector. If wkt=FALSE, this is a five-column matrix or data.frame: the vector object ID, the IDs for the parts of each object (e.g. five polygons that together are one spatial object), the x (longitude) and y (latitude) coordinates, and a flag indicating whether the part is a "hole" (only relevant for polygons).

If wkt=TRUE, the "well-known text" representation is returned as a character vector.

Usage

```
## S4 method for signature 'SpatVector'
geom(x, wkt=FALSE, hex=FALSE, df=FALSE, list=FALSE, xnm="x", ynm="y")
```

Arguments

x	SpatVector
wkt	logical. If TRUE the WKT geometry is returned (unless hex is also TRUE)
hex	logical. If TRUE the hexadecimal geometry is returned
df	logical. If TRUE a data.frame is returned instead of a matrix (only if wkt=FALSE, hex=FALSE, and list=FALSE)
list	logical. If TRUE a nested list is returned with data.frames of coordinates
xnm	character. If list=TRUE the "x" column name for the coordinates data.frame
ynm	character. If list=TRUE the "y" column name for the coordinates data.frame

Value

matrix, vector, data.frame, or list

See Also

[crds](#), [xyFromCell](#)

Examples

```
x1 <- rbind(c(-175,-20), c(-140,55), c(10, 0), c(-140,-60))
x2 <- rbind(c(-125,0), c(0,60), c(40,5), c(15,-45))
x3 <- rbind(c(-10,0), c(140,60), c(160,0), c(140,-55))
x4 <- rbind(c(80,0), c(105,13), c(120,2), c(105,-13))
z <- rbind(cbind(object=1, part=1, x1), cbind(object=2, part=1, x2),
          cbind(object=3, part=1, x3), cbind(object=3, part=2, x4))
colnames(z)[3:4] <- c('x', 'y')
z <- cbind(z, hole=0)
z[z[, "object"]==3 & z[, "part"]==2, "hole"] <- 1

p <- vect(z, "polygons")
geom(p)

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
g <- geom(v)
head(g)

w <- geom(v, wkt=TRUE)
substr(w, 1, 60)
```

geomtype	<i>Geometry type of a SpatVector</i>
----------	--------------------------------------

Description

Get the geometry type (points, lines, or polygons) of a SpatVector or the data types of the fields (attributes, variables) of a SpatVector.

Usage

```
## S4 method for signature 'SpatVector'  
geomtype(x)  
  
## S4 method for signature 'SpatVector'  
datatype(x)  
  
## S4 method for signature 'SpatVector'  
is.points(x)  
  
## S4 method for signature 'SpatVector'  
is.lines(x)  
  
## S4 method for signature 'SpatVector'  
is.polygons(x)
```

Arguments

x SpatVector

Value

character

Examples

```
f <- system.file("ex/lux.shp", package="terra")  
v <- vect(f)  
  
geomtype(v)  
is.polygons(v)  
is.lines(v)  
is.points(v)  
  
names(v)  
datatype(v)
```

 global

global statistics

Description

Compute global statistics, that is summarized values of an entire `SpatRaster`.

If `x` is very large `global` will fail, except when `fun` is one of "mean", "min", "max", "sum", "prod", "range" (min and max), "rms" (root mean square), "sd" (sample standard deviation), "sdpop" (population standard deviation), "isNA" (number of cells that are NA), "notNA" (number of cells that are not NA).

You can compute a weighted mean or sum by providing a `SpatRaster` with weights.

Usage

```
## S4 method for signature 'SpatRaster'
global(x, fun="mean", weights=NULL, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>fun</code>	function to be applied to summarize the values by zone. Either as one of these character values: "max", "min", "mean", "sum", "range", "rms" (root mean square), "sd", "std" (population sd, using <code>n</code> rather than <code>n-1</code>), "isNA", "notNA"; or a proper R function (but these may fail for very large <code>SpatRasters</code>)
<code>...</code>	additional arguments passed on to <code>fun</code>
<code>weights</code>	<code>NULL</code> or <code>SpatRaster</code>

Value

A `data.frame` with a row for each layer

See Also

[zonal](#) for "zonal" statistics, and [app](#) or [Summary-methods](#) for "local" statistics, and [extract](#) for summarizing values for polygons. Also see [focal](#) for "focal" or "moving window" operations.

Examples

```
r <- rast(ncols=10, nrows=10)
values(r) <- 1:ncell(r)
global(r, "sum")
global(r, "mean", na.rm=TRUE)
```

gridDistance	<i>Distance on a grid</i>
--------------	---------------------------

Description

The function calculates the distance to cells of a `SpatRaster` when the path has to go through the centers of the eight neighboring raster cells.

The distance is in meters if the coordinate reference system (CRS) of the `SpatRaster` is longitude/latitude (+proj=longlat) and in the units of the CRS (typically meters) in other cases.

Distances are computed by summing local distances between cells, which are connected with their neighbors in 8 directions.

The shortest distance to the cells with the target value is computed for all cells that are not NA. Cells that are NA cannot be traversed and are ignored, unless the target itself is NA, in which case the distance to the nearest cell that is not NA is computed for all cells that are NA.

Usage

```
## S4 method for signature 'SpatRaster'  
gridDistance(x, target=0, scale=1000, maxiter=50, filename="", ...)
```

Arguments

x	<code>SpatRaster</code>
target	numeric. value of the target cells (where to compute distance to)
scale	numeric. Scale factor for longitude/latitude data (1 = m, 1000 = km)
maxiter	numeric. The maximum number of iterations. Increase this number if you get the warning that <code>costDistance</code> did not converge
filename	character. output filename (optional)
...	additional arguments as for writeRaster

Value

`SpatRaster`

See Also

See [distance](#) for "as the crow flies" distance, and [costDistance](#) for distance across a landscape with variable friction

Examples

```
# global lon/lat raster
r <- rast(ncol=10,nrow=10, vals=1)
r[48] <- 0
r[66:68] <- NA
d <- gridDistance(r)
plot(d)

# planar
crs(r) <- "+proj=utm +zone=15 +ellps=GRS80 +datum=NAD83 +units=m +no_defs"
d <- gridDistance(r)
plot(d)

# distance to cells that are not NA
rr <- classify(r, cbind(1, NA))
dd <- gridDistance(rr, NA)
```

head and tail

Show the head or tail of a Spat object*

Description

Show the head (first values) or tail (last values) of a SpatRaster or of the attributes of a SpatVector.

Usage

```
head(x, ...)
tail(x, ...)
```

Arguments

x	SpatRaster or SpatVector
...	additional arguments passed on to other methods

Value

matrix (SpatRaster) or data.frame (SpatVector)

See Also

[show](#), [geom](#)

Examples

```
r <- rast(nrows=25, ncols=25)
values(r) <- 1:ncell(r)
head(r)
tail(r)
```

hist	<i>Histogram</i>
------	------------------

Description

Create a histogram of the values of a `SpatRaster`. For large datasets a sample of `maxcell` is used.

Usage

```
## S4 method for signature 'SpatRaster'
hist(x, layer, maxcell=1000000, plot=TRUE, main, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>layer</code>	integer (or character) to indicate layer number (or name). Can be used to subset the layers to plot in a multilayer <code>SpatRaster</code>
<code>maxcell</code>	integer. To regularly sample very large objects
<code>plot</code>	logical. Plot the histogram or only return the histogram values
<code>main</code>	character. Main title(s) for the plot. Default is the value of <code>names</code>
<code>...</code>	additional arguments. See hist

Value

This function is principally used for plotting a histogram, but it also returns an object of class "histogram" (invisibly if `plot=TRUE`).

See Also

[pairs](#), [boxplot](#)

Examples

```
r1 <- r2 <- rast(nrows=50, ncols=50)
values(r1) <- runif(ncell(r1))
values(r2) <- runif(ncell(r1))
rs <- r1 + r2
rp <- r1 * r2

opar <- par(no.readonly =TRUE)
```

```

par(mfrow=c(2,2))
plot(rs, main='sum')
plot(rp, main='product')
hist(rs)
a <- hist(rp)
a
x <- c(rs, rp, sqrt(rs))
hist(x)
par(opar)

```

ifel

ifelse for SpatRasters

Description

Implementation of [ifelse](#) for SpatRasters. This method allows for a concise expression of what can otherwise be achieved with a combination of [classify](#), [mask](#), and [cover](#).

`ifel` is an R equivalent to the `Con` method in ArcGIS (`arcpy`).

Usage

```

## S4 method for signature 'SpatRaster'
ifel(test, yes, no, filename="", ...)

```

Arguments

<code>test</code>	SpatRaster
<code>yes</code>	SpatRaster or numeric
<code>no</code>	SpatRaster or numeric
<code>filename</code>	character. Output filename
<code>...</code>	additional arguments for writing files as in writeRaster

Value

SpatRaster

Examples

```

r <- rast(nrows=5, ncols=5, xmin=0, xmax=1, ymin=0, ymax=1)
values(r) <- c(-10:0, NA, NA, NA, 0:10)

x <- ifel(r > 1, 1, r)
# same as
a <- classify(r, cbind(1, Inf, 1))
# or
b <- app(r, fun=function(i) {i[i > 1] <- 1; i})
# or
d <- clamp(r, -Inf, 1)

```

```
# or (not recommended for large datasets)
e <- r
e[e>1] <- 1

## other examples
f <- ifel(is.na(r), 100, r)

z <- ifel(r > -2 & r < 2, 100, 0)

# nested expressions
y <- ifel(r > 1, 1, ifel(r < -1, -1, r))

k <- ifel(r > 0, r+10, ifel(r < 0, r-10, 3))
```

image

SpatRaster image method

Description

Plot (make a map of) the values of a `SpatRaster` via `image`. See `plot` if you need more fancy options such as a legend.

Usage

```
## S4 method for signature 'SpatRaster'
image(x, y=1, maxcell=500000, ...)
```

Arguments

x	<code>SpatRaster</code>
y	positive integer indicating the layer to be plotted, or a character indicating the name of the layer
maxcell	positive integer. Maximum number of cells to use for the plot
...	additional arguments as for graphics: <code>image</code>

See Also

[plot](#)

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
image(r)
image(r, col=rainbow(24))
```

impose	<i>Impose the geometry of a SpatRaster to those in a SpatRasterCollection.</i>
--------	--

Description

Warp the members of a SpatRasterCollection to match the geometry of a SpatRaster.

Usage

```
## S4 method for signature 'SpatRasterCollection'
impose(x, y, filename="", ...)
```

Arguments

x	SpatRasterCollection
y	SpatRaster
filename	character. Output filename
...	list with named options for writing files as in writeRaster

Value

SpatRaster

See Also

[resample](#)

initialize	<i>Initialize a SpatRaster with values</i>
------------	--

Description

Create a SpatRaster with values reflecting a cell property: 'x', 'y', 'col', 'row', 'cell' or 'chess'. Alternatively, a function can be used. In that case, cell values are initialized without reference to pre-existing values. E.g., initialize with a random number (fun=[runif](#)). While there are more direct ways of achieving this for small objects (see examples) for which a vector with all values can be created in memory, the `init` function will also work for SpatRaster objects with many cells.

Usage

```
## S4 method for signature 'SpatRaster'
init(x, fun, filename="", ...)
```

Arguments

x	SpatRaster
fun	function to be applied. This must be either single number, multiple numbers, a function, or one of a set of known character values. A function must take the number of cells as a single argument to return a vector of values with a length equal to the number of cells, such as <code>fun=runif</code> . Allowed character values are 'x', 'y', 'row', 'col', 'cell', and 'chess' to get the x or y coordinate, row, col or cell number or a chessboard pattern (alternating 0 and 1 values)
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

Examples

```
r <- rast(ncols=10, nrows=5, xmin=0, xmax=10, ymin=0, ymax=5)
x <- init(r, fun="cell")
y <- init(r, fun=runif)

# initialize with a single value
z <- init(r, fun=8)
```

inplace

*Change values in-place***Description**

These "in-place" replacement methods assign new value to an object without making a copy. That is efficient, but if there is a copy of the object that you made by standard assignment (e.g. with `y <- x`), that copy is also changed.

`set.names` is the in-place replacement version of `names<-`.

`set.ext` is the in-place replacement version of `ext<-`

`set.values` is the in-place replacement version of `[<-`.

`set.cats` is the in-place replacement version of `categories`

`set.crs` is the in-place replacement version of `crs<-`

Usage

```
## S4 method for signature 'SpatRaster'
set.names(x, value, index=1:nlyr(x), validate=FALSE)
## S4 method for signature 'SpatRasterDataset'
set.names(x, value, index=1:length(x), validate=FALSE)
## S4 method for signature 'SpatVector'
```

```

set.names(x, value, index=1:ncol(x), validate=FALSE)

## S4 method for signature 'SpatRaster'
set.ext(x, value)
## S4 method for signature 'SpatVector'
set.ext(x, value)

## S4 method for signature 'SpatRaster'
set.crs(x, value)
## S4 method for signature 'SpatVector'
set.crs(x, value)

## S4 method for signature 'SpatRaster'
set.values(x, cells, values)

## S4 method for signature 'SpatRaster'
set.cats(x, layer=1, value, index)

```

Arguments

x	SpatRaster
value	character (set.names). For set.cats: a data.frame with columns (value, category) or vector with category names
index	positive integer indicating layer(s) to assign a name to, or the index to select the active category
validate	logical. Make names valid and/or unique?
cells	cell numbers or missing
values	replacement values or missing to load all values into memory
layer	positive integer indicating to which layer to you want to assign these categories

Examples

```

s <- rast(ncols=5, nrows=5, nlyrs=3)
x <- s
names(s)
names(s) <- c("a", "b", "c")
names(s)
names(x)

x <- s
set.names(s, c("e", "f", "g"))
names(s)
names(x)

set.ext(x, c(0,180,0,90))

f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)

```

```
#values from file to memory
set.values(r)

# change values
set.values(r, 1:1000, 900)
```

inset	<i>Make an inset map</i>
-------	--------------------------

Description

Make an inset map or scale the extent of a SpatVector

Usage

```
## S4 method for signature 'SpatVector'
inset(x, e, loc="", scale=0.2, background="white",
      perimeter=TRUE, box=NULL, pper, pbox, ...)

## S4 method for signature 'SpatRaster'
inset(x, e, loc="", scale=0.2, background="white",
      perimeter=TRUE, box=NULL, pper, pbox, ...)

## S4 method for signature 'SpatVector'
inext(x, e, y=NULL, gap=0)
```

Arguments

x	SpatVector, SpatRaster
e	SpatExtent to set the size and location of the inset. Or missing
loc	character. One of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"
scale	numeric. The relative size of the inset, used when x is missing
background	color for the background of the inset. Use NA for no background color
perimeter	logical. If TRUE a perimeter (border) is drawn around the inset
box	SpatExtent or missing, to draw a box on the inset, e.g. to show where the map is located in a larger area
pper	list with graphical parameters (arguments) such as col and lwd for the perimeter line
pbox	list with graphical parameters (arguments) such as col and lwd for the box (line)
...	additional arguments passed to plot for the drawing of x
y	SpatVector. If not NULL, y is scaled based with the parameters for x. This is useful, for example, when x represent boundaries, and y points within these boundaries
gap	numeric to add space between the SpatVector and the SpatExtent

Value

scaled and shifted `SpatVector` or `SpatRaster` (returned invisibly)

See Also

[sbar](#), [rescale](#), [shift](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
x <- v[v$NAME_2 == "Diekirch", ]

plot(x, density=10, col="blue")
inset(v)

# more elaborate
plot(x, density=10, col="blue")
inset(v, col = "brown", border="lightgrey", perimeter=TRUE,
      pper=list(col="orange", lwd=3, lty=2),
      box=ext(x), pbox=list(col="blue", lwd=2))

cols <- rep("light grey", 12)
cols[2] <- "red"
e <- ext(c(6.2, 6.3, 49.9, 50))
b <- ext(x)+0.02
inset(v, e=e, col=cols, box=b)

# with a SpatRaster
ff <- system.file("ex/elev.tif", package="terra")
r <- rast(ff)
r <- crop(r, ext(x) + .01)
plot(r, type="int", mar=c(2,2,2,2), plg=list(x="topright"))
lines(v, lwd=1.5)
lines(x, lwd=2.5)
inset(v, col=cols, loc="topleft", scale=0.15)

# a more complex one
plot(r, plg=list(title="meter\n", shrink=.2, cex=.8))
lines(v, lwd=4, col="white")
lines(v, lwd=1.5)
lines(x, lwd=2.5)
text(x, "NAME_2", cex=1.5, halo=TRUE)
sbar(6, c(6.04, 49.785), type="bar", below="km", label=c(0,3,6), cex=.8)
s <- inset(v, col=cols, box=b, scale=.2, loc="topright", background="light yellow",
          pbox=list(lwd=2, lty=5, col="blue"))

# note the returned inset SpatVector
s
lines(s, col="orange")
```

`intersect`*Intersection*

Description

Intersect the geometries of two `SpatVectors`.

Intersecting points with points uses the extent of `y` to get the intersection. Intersecting of points and lines is not supported because of numerical inaccuracies with that. You can use [buffer](#), to create polygons from lines and use these with `intersect`.

See [crop](#) for intersection of a `SpatRaster`.

Usage

```
## S4 method for signature 'SpatVector,SpatVector'  
intersect(x, y)
```

```
## S4 method for signature 'SpatVector,SpatExtent'  
intersect(x, y)
```

```
## S4 method for signature 'SpatExtent,SpatVector'  
intersect(x, y)
```

```
## S4 method for signature 'SpatExtent,SpatExtent'  
intersect(x, y)
```

Arguments

<code>x</code>	<code>SpatVector</code> or <code>SpatExtent</code>
<code>y</code>	<code>SpatVector</code> or <code>SpatExtent</code>

Value

Same as `x`

See Also

[union](#), [crop](#), [relate](#)

Examples

```
e1 <- ext(-10, 10, -20, 20)  
e2 <- ext(0, 20, -40, 5)  
intersect(e1, e2)  
  
f <- system.file("ex/lux.shp", package="terra")  
v <- vect(f)  
e <- ext(5.6, 6, 49.55, 49.7)
```

```
x <- intersect(v, e)

p <- vect(c("POLYGON ((5.8 49.8, 6 49.9, 6.15 49.8, 6 49.6, 5.8 49.8))",
"POLYGON ((6.3 49.9, 6.2 49.7, 6.3 49.6, 6.5 49.8, 6.3 49.9))"), crs=crs(v))
values(p) <- data.frame(pid=1:2, area=expansion(p))

y <- intersect(v, p)
```

is.bool

Raster value types

Description

The values in a `SpatRaster` layer are by default numeric, but they can also be logical (Boolean), integer, or categorical

Note that `as.bool` and `as.int` return a new `SpatRaster`, whereas `is.bool` and `is.int` return a logical value for each layer. For a `SpatRaster`, `isTRUE` is equivalent to `as.bool`, `isFALSE` is equivalent to `!as.bool`, `as.integer` is the same as `as.int` and `as.logical` is the same as `as.bool`

Usage

```
## S4 method for signature 'SpatRaster'
is.bool(x)
## S4 method for signature 'SpatRaster'
as.bool(x, filename, ...)
## S4 method for signature 'SpatRaster'
is.int(x)
## S4 method for signature 'SpatRaster'
as.int(x, filename, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>filename</code>	character. Output filename
<code>...</code>	list with named options for writing files as in writeRaster

Value

logical or `SpatRaster`

Examples

```
r <- rast(nrows=10, ncols=10, vals=1:100)
is.bool(r)
z <- as.bool(r)
is.bool(z)
```

```
x <- r > 25
is.bool(x)

rr <- r/2
is.int(rr)
is.int(round(rr))
```

is.lonlat

Check for longitude/latitude crs

Description

Test whether a `SpatRaster` or `SpatVector` has a longitude/latitude coordinate reference system (CRS), or perhaps has one. That is when the CRS is unknown ("") but the x coordinates are within -181 and 181 and the y coordinates are within -90.1 and 90.1. For a `SpatRaster` you can also test if it is longitude/latitude and "global" (covers all longitudes).

Usage

```
## S4 method for signature 'SpatRaster'
is.lonlat(x, perhaps=FALSE, warn=TRUE, global=FALSE)

## S4 method for signature 'SpatVector'
is.lonlat(x, perhaps=FALSE, warn=TRUE)
```

Arguments

x	SpatRaster or SpatVector
perhaps	logical. If TRUE and the crs is unknown, the method returns TRUE if the coordinates are plausible for longitude/latitude
warn	logical. If TRUE, a warning is given if the CRS is unknown or when the CRS is longitude/latitude but the coordinates do not match that
global	logical. If TRUE, the method tests if the raster covers all longitudes (from -180 to 180 degrees) such that the extreme columns are in fact adjacent

Value

logical or NA

Examples

```
r <- rast()
is.lonlat(r)
is.lonlat(r, global=TRUE)

crs(r) <- ""
is.lonlat(r)
is.lonlat(r, perhaps=TRUE, warn=FALSE)
```

```
crs(r) <- "+proj=lcc +lat_1=48 +lat_2=33 +lon_0=-100 +ellps=WGS84"
is.lonlat(r)
```

lapp	<i>Apply a function to layers of a SpatRaster, or sub-datasets of a SpatRasterDataset</i>
------	---

Description

Apply a function to a SpatRaster, using layers as arguments.

The number of arguments in function `fun` must match the number of layers in the SpatRaster (or the number of sub-datasets in the SpatRasterDataset). For example, if you want to multiply two layers, you could use this function: `fun=function(x,y){return(x*y)}` percentage: `fun=function(x,y){return(100 * x / y)}`. If you combine three layers you could use `fun=function(x,y,z){return((x + y) * z)}`

Before you use the function, test it to make sure that it is vectorized. That is, it should work for vectors longer than one, not only for single numbers. Or if the input SpatRaster(s) have multiple layers, it should work for a matrix (multiple cells) of input data (or matrices in the case of a SpatRasterDataset). The function must return the same number of elements as its input vectors, or multiples of that. Also make sure that the function is NA-proof: it should return the same number of values when some or all input values are NA. And the function must return a vector or a matrix, not a data.frame. To test it, run it with `do.call(fun, data)` (see examples).

Use [app](#) for summarize functions such as `sum`, that take any number of arguments; and [tapp](#) to do so for groups of layers.

Usage

```
## S4 method for signature 'SpatRaster'
lapp(x, fun, ..., usernames=FALSE, cores=1, filename="", overwrite=FALSE, wopt=list())

## S4 method for signature 'SpatRasterDataset'
lapp(x, fun, ..., recycle=FALSE, filename="", overwrite=FALSE, wopt=list())
```

Arguments

<code>x</code>	SpatRaster or SpatRasterDataset
<code>fun</code>	a function that takes a vector and can be applied to each cell of <code>x</code>
<code>...</code>	additional arguments to be passed to <code>fun</code>
<code>usernames</code>	logical. Use the layer names to match the function arguments? If FALSE matching is by position
<code>cores</code>	positive integer. If <code>cores > 1</code> , a 'parallel' package cluster with that many cores is created and used. You can also supply a cluster object
<code>recycle</code>	logical. Recycle layers to match the subdataset with the largest number of layers
<code>filename</code>	character. Output filename
<code>overwrite</code>	logical. If TRUE, <code>filename</code> is overwritten
<code>wopt</code>	list with named options for writing files as in writeRaster

Value

SpatRaster

Note

Use [sapp](#) or [lapply](#) to apply a function that takes a `SpatRaster` as argument to each layer of a `SpatRaster` (that is rarely necessary).

See Also

[app](#), [tapp](#), [math](#)

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra")) + 1
ss <- s[[2:1]]

fvi <- function(x, y){ (x - y) / (x + y) }
# test the function
data <- list(c(1:5,NA), 6:1)
do.call(fvi, data)

x <- lapp(ss, fun=fvi )

# which is the same as supplying the layers to "fun"
# in some cases this will be much faster
y <- fvi(s[[2]], s[[1]])

f2 <- function(x, y, z){ (z - y + 1) / (x + y + 1) }
p1 <- lapp(s, fun=f2 )

p2 <- lapp(s[[1:2]], f2, z=200)

# the usenames argument

fvi2 <- function(red, green){ (red - green) / (red + green) }
names(s)
x1 <- lapp(s[[1:2]], fvi2, usenames=TRUE)
x2 <- lapp(s[[2:1]], fvi2, usenames=TRUE)
# x1 and x2 are the same, despite the change in the order of the layers
# x4 is also the same, but x3 is not
x3 <- lapp(s[[2:1]], fvi2, usenames=FALSE)
x4 <- lapp(s, fvi2, usenames=TRUE)

# while this would fail because
# there are too many layers in s
# x5 <- lapp(s, fvi2, usenames=FALSE)

pairs(c(x1, x2, x3, x4))

## SpatRasterDataset
x <- sds(s, s[[1]]+50)
```

```

fun <- function(x, y) { x/y }

# test "fun"
data <- list(matrix(1:9, ncol=3), matrix(9:1, ncol=3))
do.call(fun, data)

lapp(x, fun, recycle=TRUE)

# the same, more concisely
z <- s / (s[[1]]+50)

```

layerCor

Correlation and (weighted) covariance

Description

Compute correlation, (weighted) covariance, or similar summary statistics that compare the values of all pairs of the layers of a SpatRaster.

Usage

```

## S4 method for signature 'SpatRaster'
layerCor(x, fun, w, asSample=TRUE, na.rm=FALSE, maxcell=Inf, ...)

```

Arguments

x	SpatRaster
fun	character. The statistic to compute: either "cov" (covariance), "weighted.cov" (weighted covariance), or "pearson" (correlation coefficient) or your own function that takes two vectors as argument to compute a single number
w	SpatRaster with the weights to compute the weighted covariance. It should have a single layer and the same geometry as x
asSample	logical. If TRUE, the statistic for a sample (denominator is n-1) is computed, rather than for the population (denominator is n). Only for the standard functions
na.rm	logical. Should missing values be removed?
maxcell	positive integer. The number of cells to be regularly sampled. Only used when fun is a function
...	additional arguments for fun (if it is a proper function)

Value

If fun is one of the three standard statistics, you get a list with two items: the correlation or (weighted) covariance matrix, and the (weighted) means.

If fun is a function, you get a matrix.

References

For the weighted covariance:

- Canty, M.J. and A.A. Nielsen, 2008. Automatic radiometric normalization of multitemporal satellite imagery with the iteratively re-weighted MAD transformation. *Remote Sensing of Environment* 112:1025-1036.
- Nielsen, A.A., 2007. The regularized iteratively reweighted MAD method for change detection in multi- and hyperspectral data. *IEEE Transactions on Image Processing* 16(2):463-478.

See Also

[global](#), [cov.wt](#), [weighted.mean](#)

Examples

```
b <- rast(system.file("ex/logo.tif", package="terra"))
layerCor(b, "pearson")

layerCor(b, "cov")

# weigh by column number
w <- init(b, fun="col")
layerCor(b, "weighted.cov", w=w)
```

linearUnits

Linear units of the coordinate reference system

Description

Get the linear units of the coordinate reference system (crs) of a `SpatRaster` or `SpatVector` expressed in m. The value returned is used internally to transform area and perimeter measures to meters. The value returned for longitude/latitude crs is zero.

Usage

```
## S4 method for signature 'SpatRaster'
linearUnits(x)

## S4 method for signature 'SpatVector'
linearUnits(x)
```

Arguments

x `SpatRaster` or `SpatVector`

Value

numeric (meter)

See Also[crs](#)**Examples**

```
x <- rast()
crs(x) <- ""
linearUnits(x)

crs(x) <- "+proj=longlat +datum=WGS84"
linearUnits(x)

crs(x) <- "+proj=utm +zone=1 +units=cm"
linearUnits(x)

crs(x) <- "+proj=utm +zone=1 +units=km"
linearUnits(x)

crs(x) <- "+proj=utm +zone=1 +units=us-ft"
linearUnits(x)
```

lines*Add SpatVector data to a map*

Description

Add SpatVector data to a plot (map) with points, lines, or polys.

These are simpler alternatives for [plot\(x, add=TRUE\)](#)

Usage

```
## S4 method for signature 'SpatVector'
points(x, col, cex=1, pch=20, alpha=1, ...)

## S4 method for signature 'SpatVector'
lines(x, y=NULL, col, lwd=1, lty=1, arrows=FALSE, alpha=1, ...)

## S4 method for signature 'SpatVector'
polys(x, col, border="black", lwd=1, lty=1, alpha=1, ...)

## S4 method for signature 'SpatExtent'
points(x, col="black", alpha=1, ...)

## S4 method for signature 'SpatExtent'
lines(x, col="black", alpha=1, ...)

## S4 method for signature 'SpatExtent'
polys(x, col, alpha=1, ...)
```


Arguments

x	SpatVector or SpatExtent
y	missing or SpatVector. If both x and y have point geometry and the same number of rows, lines are drawn between pairs of points
col	character. Colors
border	character. color(s) of the polygon borders. Use NULL or NA to not draw a border
cex	numeric. point size magnifier. See par
pch	positive integer, line type. See points
alpha	number between 0 and 1 to set transparency
lwd	numeric, line-width. See par
lty	positive integer, line type. See par
arrows	logical. If TRUE and y is a SpatVector, arrows are drawn instead of lines. See ?arrows for additional arguments
...	additional graphical arguments such as lwd, cex and pch

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)

r <- rast(v)
values(r) <- 1:ncell(r)
plot(r)
lines(v)
points(v)
```

makeTiles

Make tiles

Description

Divide a SpatRaster into "tiles". The cell of another SpatRaster (normally with a much lower resolution) are used to define the tiles.

Usage

```
## S4 method for signature 'SpatRaster'
makeTiles(x, y, filename="tile_.tif", extend=FALSE, na.rm=FALSE, ...)
```

Arguments

x	SpatRaster
y	SpatRaster or SpatVector
filename	character. Output filename template. Filenames will be altered by adding the tile number for each tile
extend	logical. If TRUE, the extent of y is expanded to assure that it covers all of x
na.rm	logical. If TRUE, tiles with only missing values are ignored
...	additional arguments for writing files as in writeRaster

Value

character (filenames)

See Also

[vrt](#) to create a virtual raster from tiles

Examples

```
r <- rast(ncols=100, nrows=100)
values(r) <- 1:ncell(r)
x <- rast(ncols=2, nrows=2)
filename <- paste0(tempfile(), "_tif")
ff <- makeTiles(r, x, filename)
ff

vrt(ff)
```

makeVRT

Make a VRT header file

Description

Create a VRT header file for a "flat binary" raster file that needs a header file to be able to read it, but does not have it.

Usage

```
makeVRT(filename, nrow, ncol, nlyr=1, extent, xmin, ymin, xres, yres=xres, xycenter=TRUE,
  crs="+proj=longlat", lyrnms="", datatype, NAflag=NA, bandorder="BIL", byteorder="LSB",
  toptobottom=TRUE, offset=0, scale=1)
```

Arguments

filename	character. raster filename (without ".vrt" extension)
nrow	positive integer, the number of rows
ncol	positive integer, the number of columns
nlyr	positive integer, the number of layers
extent	SpatExtent or missing
xmin	numeric. minimum x coordinate (only used if extent is missing)
ymin	numeric. minimum y coordinate (only used if extent is missing)
xres	postive number. x resolution
yres	postive number. y resolution)
xycenter	logical. If TRUE, xmin and xmax represent the coordinates of the center of the extreme cell, in stead of the coordinates of the outside corner. Only used of extent is missing
crs	character. Coordinate reference system description
lyrnms	character. Layer names
datatype	character. One of "INT2S", "INT4S", "INT1U", "INT2U", "INT4U", "FLT4S", "FLT8S". If missing, this is guessed from the file size (INT1U for 1 byte per value, INT2S for 2 bytes and FLT4S for 4 bytes per value). This may be wrong because, for example, 2 bytes per value may in fact be INT2U (with the U for unsigned) values
NAflag	numeric. The value used as the "NA flag"
bandorder	character. One of "BIL", "BIP", or "BSQ". That is Band Interleaved by Line, or by Pixel, or Band SeQuential
byteorder	character. One of "LSB", "MSB". "MSB" is common for files generated on Linux systems, whereas "LSB" is common for files generated on windows
toptobottom	logical. If FALSE, the values are read bottom to top
offset	numeric. offset to be applied
scale	numeric. scale to be applied

Value

character (.VRT filename)

See Also

[vrt](#) to create a vrt for a collection of raster tiles

mask

*Mask values in a SpatRaster or SpatVector***Description**

If *x* is a *SpatRaster*: Create a new *SpatRaster* that has the same values as *SpatRaster* *x*, except for the cells that are NA (or another *maskvalue*) in another *SpatRaster* (the '*mask*'), or are not covered by a *SpatVector*. These cells become NA (or another *updatevalue*).

If *x* is a *SpatVector*: Select geometries of *x* that intersect, or not intersect, with the geometries of *y*.

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
mask(x, mask, inverse=FALSE, maskvalues=NA,
      updatevalue=NA, filename="", ...)
```

```
## S4 method for signature 'SpatRaster,SpatVector'
mask(x, mask, inverse=FALSE, updatevalue=NA,
      touches=TRUE, filename="", ...)
```

```
## S4 method for signature 'SpatVector,SpatVector'
mask(x, mask, inverse=FALSE)
```

Arguments

<i>x</i>	<i>SpatRaster</i> or <i>SpatVector</i>
<i>mask</i>	<i>SpatRaster</i> or <i>SpatVector</i>
<i>inverse</i>	logical. If TRUE, areas on <i>mask</i> that are <i>_not_</i> the <i>maskvalue</i> are masked
<i>maskvalues</i>	numeric. The value(s) in <i>mask</i> that indicates the cells of <i>x</i> that should become <i>updatevalue</i> (default = NA)
<i>updatevalue</i>	numeric. The value that cells of <i>x</i> should become if they are not covered by <i>mask</i> (and not NA)
<i>touches</i>	logical. If TRUE, all cells touched by lines or polygons will be masked, not just those on the line render path, or whose center point is within the polygon
<i>filename</i>	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[crop](#)

Examples

```

r <- rast(ncols=10, nrows=10)
m <- rast(ncols=10, nrows=10)
values(r) <- 1:100
set.seed(1965)
x <- round(3 * runif(ncell(r)))
x[x==0] <- NA
values(m) <- x
mr <- mask(r, m)

```

match	<i>Value matching for SpatRasters</i>
-------	---------------------------------------

Description

match returns a SpatRaster with the position of the matched values. The cell values are the index of the table argument.

%in% returns a 0/1 (FALSE/TRUE) SpatRaster indicating if the cells values were matched or not.

Usage

```
match(x, table, nomatch = NA_integer_, incomparables = NULL)
```

```
x %in% table
```

Arguments

x	SpatRaster
table	vector of the values to be matched against
nomatch	the value to be returned in the case when no match is found. Note that it is coerced to integer
incomparables	a vector of values that cannot be matched. Any value in x matching a value in this vector is assigned the nomatch value. For historical reasons, FALSE is equivalent to NULL

Value

SpatRaster

See Also

[app](#), [match](#)

Examples

```
r <- rast(nrows=10, ncols=10)
values(r) <- 1:100
m <- match(r, c(5:10, 50:55))
n <- r %in% c(5:10, 50:55)
```

 Math-methods

General mathematical methods

Description

Standard mathematical methods for computations with SpatRaster objects. Computations are local (applied on a cell by cell basis). If multiple SpatRaster objects are used, these must have the same extent and resolution. These have been implemented:

abs, sign, sqrt, ceiling, floor, trunc, cummax, cummin, cumprod, cumsum, log, log10, log2, log1p, acos, acosh, asin, asinh, atan, atanh, exp, expm1, cos, cosh, sin, sinh, tan, tanh, round, signif

Instead of directly calling these methods, you can also provide their name to the math method. This is useful if you want to provide an output filename.

The following methods have been implemented for SpatExtent: round, floor, ceiling
 round has also been implemented for SpatVector, to round the coordinates of the geometries.

Usage

```
## S4 method for signature 'SpatRaster'
sqrt(x)

## S4 method for signature 'SpatRaster'
log(x, base=exp(1))

## S4 method for signature 'SpatRaster'
round(x, digits=0)

## S4 method for signature 'SpatRaster'
math(x, fun, digits=0, filename="", overwrite=FALSE, ...)

## S4 method for signature 'SpatVector'
round(x, digits=4)
```

Arguments

x	SpatRaster
base	a positive or complex number: the base with respect to which logarithms are computed
digits	Number of digits for rounding

fun character. Math function name
 filename character. Output filename
 overwrite logical. If TRUE, filename is overwritten
 ... additional arguments for writing files as in [writeRaster](#)

Value

SpatRaster or SpatExtent

See Also

See [app](#) to use mathematical functions not implemented by the package, and [Arith-methods](#) for arithmetical operations

Examples

```

r1 <- rast(ncols=10, nrows=10)
v <- runif(ncell(r1))
v[10:20] <- NA
values(r1) <- v
r2 <- rast(r1)
values(r2) <- 1:ncell(r2) / ncell(r2)
r <- c(r1, r2)

s <- sqrt(r)
# same as
math(r, "sqrt")

round(s, 1)

```

mem

Memory available and needed

Description

mem_info prints the amount of RAM that is required and available to process a SpatRaster.
 free_RAM returns the amount of RAM that is available

Usage

```

mem_info(x, n=1)

free_RAM()

```

Arguments

x SpatRaster
 n positive integer. The number of copies of x that are needed

Value

free_RAM returns the amount of available RAM in kilobytes

Examples

```
mem_info(rast())
```

```
free_RAM()
```

merge	<i>Merge SpatRaster or SpatExtent objects, or a SpatVector with a data.frame</i>
-------	--

Description

Merge SpatRasters to form a new SpatRaster object with a larger spatial extent. If objects overlap, the values get priority in the same order as the arguments. The SpatRasters must have the same origin and spatial resolution. In areas where the SpatRaster objects overlap, the values of the SpatRaster that is last in the sequence of arguments will be retained. See [classify](#) to merge a SpatRaster and a data.frame. You can also merge SpatExtent objects.

There is also a method for merging SpatVector with a data.frame; that is, to join the data.frame to the attribute table of the SpatVector.

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
merge(x, y, ..., filename="", overwrite=FALSE, wopt=list())
```

```
## S4 method for signature 'SpatRasterCollection,missing'
merge(x, filename="", ...)
```

```
## S4 method for signature 'SpatExtent,SpatExtent'
merge(x, y, ...)
```

```
## S4 method for signature 'SpatVector,data.frame'
merge(x, y, ...)
```

Arguments

x	SpatRaster or SpatExtent
y	object of same class as x
...	if x is a SpatRaster: additional objects of the same class as x. If x is a SpatRasterCollection: options for writing files as in writeRaster . If x is a SpatVector, the same arguments as in merge
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in writeRaster

Value

SpatRaster or SpatExtent

Note

You can use `merge` with `do.call` to merge a list of SpatRasters (see example). But note that if the list is named, these names are used by `merge`. So if all elements are named, there should be one element with a SpatRaster called `x` and another one called `y`. For example with `names(x)[1:2] <- c("x" "m" "y")`. You can also removed the names of the the first two elements (assuming these are SpatRasters) with `names(x)[1:2] <- ""`.

See Also

Combining tiles with `vrt` may be more efficient. See `mosaic` for averaging overlapping regions.

Examples

```
x <- rast(xmin=-110, xmax=-80, ymin=40, ymax=70, ncols=30, nrows=30)
y <- rast(xmin=-85, xmax=-55, ymax=60, ymin=30, ncols=30, nrows=30)
z <- rast(xmin=-60, xmax=-30, ymax=50, ymin=20, ncols=30, nrows=30)
values(x) <- 1:ncell(x)
values(y) <- 1:ncell(y)
values(z) <- 1:ncell(z)

m1 <- merge(x, y, z)
m2 <- merge(z, y, x)
m3 <- merge(y, x, z)

# if you have many SpatRasters make a SpatRasterCollection from a list
rlist <- list(x, y, z)
rsrc <- sprc(rlist)

m <- merge(rsrc)

## SpatVector with data.frame
f <- system.file("ex/lux.shp", package="terra")
p <- vect(f)
dfr <- data.frame(District=p$NAME_1, Canton=p$NAME_2, Value=round(runif(length(p), 100, 1000)))
dfr <- dfr[1:5, ]
pm <- merge(p, dfr, all.x=TRUE, by.x=c('NAME_1', 'NAME_2'), by.y=c('District', 'Canton'))
pm
values(pm)
```

Description

Combine `SpatRasters` with partly overlapping time-stamps to create a single time series. If there is no overlap between the `SpatRasters` there is no point in using this function (use `c` instead).

Also note that time gaps are not filled. You can use `fillTime` to do that.

Usage

```
## S4 method for signature 'SpatRasterDataset'
mergeTime(x, fun=mean, filename="", ...)
```

Arguments

<code>x</code>	<code>SpatRasterDataset</code>
<code>fun</code>	A function that reduces a vector to a single number, such as <code>mean</code> or <code>min</code>
<code>filename</code>	character. Output filename
<code>...</code>	list with named options for writing files as in <code>writeRaster</code>

Value

`SpatRaster`

Examples

```
r <- rast(system.file("ex/logo.tif", package="terra"))
s1 <- c(r, r)
time(s1) <- as.Date("2001-01-01") + 0:5
s1 <- s1/10
time(s1) <- as.Date("2001-01-07") + 0:5
s2 <- s1*10
time(s2) <- as.Date("2001-01-05") + 0:5
x <- sds(s1, s1, s2)

m <- mergeTime(x, mean)
```

modal

modal value

Description

Compute the mode for each cell across the layers of a `SpatRaster`. The mode, or modal value, is the most frequent value in a set of values.

Usage

```
## S4 method for signature 'SpatRaster'
modal(x, ..., ties="first", na.rm=FALSE, filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster
...	additional argument of the same type as x or numeric
ties	character. Indicates how to treat ties. Either "random", "lowest", "highest", "first", or "NA"
na.rm	logical. If TRUE, NA values are ignored. If FALSE, NA is returned if x has any NA values
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in writeRaster

Value

SpatRaster

Examples

```
r <- rast(system.file("ex/logo.tif", package="terra"))
r <- c(r/2, r, r*2)
m <- modal(r)
```

mosaic

mosaic SpatRasters

Description

Combine adjacent and (partly) overlapping SpatRasters to form a single new SpatRaster. Values in overlapping cells are averaged (by default) or can be computed with another function.

The SpatRasters must have the same origin and spatial resolution.

This method is similar to the simpler, but faster [merge](#) method.

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
mosaic(x, y, ..., fun="mean", filename="", overwrite=FALSE, wopt=list())

## S4 method for signature 'SpatRasterCollection,missing'
mosaic(x, fun="mean", filename="", ...)
```

Arguments

x	SpatRaster
y	object of same class as x
...	additional SpatRasters
fun	character. One of "sum", "mean", "median", "min", "max"
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in writeRaster

Value

SpatRaster

See Also

[merge](#)

Examples

```
x <- rast(xmin=-110, xmax=-80, ymin=40, ymax=70, ncols=30, nrows=30)
y <- rast(xmin=-85, xmax=-55, ymax=60, ymin=30, ncols=30, nrows=30)
z <- rast(xmin=-60, xmax=-30, ymax=50, ymin=20, ncols=30, nrows=30)
values(x) <- 1:ncell(x)
values(y) <- 1:ncell(y)
values(z) <- 1:ncell(z)

m1 <- mosaic(x, y, z)
m2 <- mosaic(z, y, x)

# if you have many SpatRasters make a SpatRasterCollection from a list
rlist <- list(x, y, z)
rsrc <- sprc(rlist)

m <- mosaic(rsrc)
```

na.omit

na.omit for SpatVector

Description

Remove empty geometries and/or records that are NA from a SpatVector.

Usage

```
## S4 method for signature 'SpatVector'
na.omit(object, field=NA, geom=FALSE)
```

Arguments

object	SpatVector
field	character or NA. If NA, missing values in the attributes are ignored. Other values are either one or more field (variable) names, or "" to consider all fields
geom	logical. If TRUE empty geometries are removed

Value

SpatVector

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
v$test <- c(1,2,NA)
nrow(v)
x <- na.omit(v, "test")
nrow(x)
```

NAflag

Set the NA flag

Description

The main purpose of this method is to allow correct reading of a SpatRaster that is based on a file that has an incorrect NA flag. The file is not changed, but flagged value is set to NA when values are read from the file ("lazy evaluation"). In contrast, if the values are in memory the change is made immediately.

To change values, it is generally better to use [classify](#)

Usage

```
## S4 method for signature 'SpatRaster'
NAflag(x)

## S4 replacement method for signature 'SpatRaster'
NAflag(x)<-value
```

Arguments

x	SpatRaster
value	numeric. The value to be interpreted as NA; set this before reading the values from the file. This can be a single value, or multiple values, one for each data source (file / subdataset)

Value

none or numeric

See Also

[classify](#)

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))[[1]]
NAflag(s) <- 255
plot(s)
NAflag(s)
```

names

Names of Spat objects*

Description

Get or set the names of the layers of a `SpatRaster` or the attributes of a `SpatVector`. With `longnames` you can get or set the "long names" of a `SpatRaster` or `SpatRasterDataset`.

For a `SpatRaster`, you can also get/set a variable name or long name (one per data source).

See [set.names](#) for in-place setting of names.

Usage

```
## S4 method for signature 'SpatRaster'
names(x)

## S4 replacement method for signature 'SpatRaster'
names(x)<-value

## S4 method for signature 'SpatRaster'
varnames(x)

## S4 replacement method for signature 'SpatRaster'
varnames(x)<-value

## S4 method for signature 'SpatRaster'
longnames(x)

## S4 replacement method for signature 'SpatRaster'
longnames(x)<-value

## S4 method for signature 'SpatRasterDataset'
```

```
names(x)

## S4 replacement method for signature 'SpatRasterDataset'
names(x)<-value

## S4 method for signature 'SpatRasterDataset'
varnames(x)

## S4 replacement method for signature 'SpatRasterDataset'
varnames(x)<-value

## S4 method for signature 'SpatRasterDataset'
longnames(x)

## S4 replacement method for signature 'SpatRasterDataset'
longnames(x)<-value

## S4 method for signature 'SpatVector'
names(x)

## S4 replacement method for signature 'SpatVector'
names(x)<-value
```

Arguments

x	SpatRaster, SpatRasterDataset, or SpatVector
value	character (vector)

Value

character

Note

terra enforces neither unique nor valid names. See [make.unique](#) to create unique names and `{make.names}` to make syntactically valid names.

Examples

```
s <- rast(ncols=5, nrows=5, nlyrs=3)
nlyr(s)
names(s)
names(s) <- c("a", "b", "c")
names(s)

# space is not valid
names(s)[2] <- "hello world"
names(s)

# two invalid names
```

```

names(s) <- c("a", " a ", "3")
names(s)

# SpatVector names
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
names(v)
names(v) <- paste0(substr(names(v), 1, 2), "_", 1:ncol(v))
names(v)

```

nearest	<i>nearby geometries</i>
---------	--------------------------

Description

Identify geometries that are near to each other. Either get the index of all geometries within a certain distance, or the k nearest neighbors, or (with nearest) get the nearest points between two geometries.

Usage

```

## S4 method for signature 'SpatVector'
nearby(x, y=NULL, distance=0, k=1, centroids=TRUE, symmetrical=TRUE)

## S4 method for signature 'SpatVector'
nearest(x, y, pairs=FALSE, centroids=TRUE, lines=FALSE)

```

Arguments

x	SpatVector
y	SpatVector or NULL
distance	numeric. maximum distance
k	positive integer. number of neighbors. Ignored if distance > 0
centroids	logical. Should the centroids of polygons be used?
symmetrical	logical. If TRUE, a near pair is only included once. That is, if geometry 1 is near to geometry 3, the implied nearness between 3 and 1 is not reported. Ignored if k neighbors are returned
pairs	logical. If TRUE pairwise nearest points are returned (only relevant when using at least one SpatVector of lines or polygons)
lines	logical. If TRUE lines between the nearest points instead of (the nearest) points

Value

matrix

See Also[relate](#), [adjacent](#)**Examples**

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
nearby(v, distance=12000)
```

north	<i>North arrow</i>
-------	--------------------

Description

Add a (North) arrow to a map

Usage

```
north(xy=NULL, type=1, label="N", angle=0, d, head=0.1, xpd=TRUE, ...)
```

Arguments

xy	numeric. x and y coordinate to place the arrow. It can also be one of following character values: "bottomleft", "bottom", "bottomright", "topleft", "top", "topright", "left", "right", or NULL
type	integer between 1 and 12, or a character (unicode) representation of a right pointing arrow such as "\u27A9"
label	character, to be printed near the arrow
angle	numeric. The angle of the arrow in degrees
d	numeric. Distance covered by the arrow in plot coordinates. Only applies to type=1
head	numeric. The size of the arrow "head", for type=1
xpd	logical. If TRUE, the scale bar or arrow can be outside the plot area
...	graphical arguments to be passed to other methods

Value

none

See Also[sbar](#), [plot](#), [inset](#)

Examples

```
f <- system.file("ex/meuse.tif", package="terra")
r <- rast(f)
plot(r)
north()
north(c(178550, 332500), d=250)

## Not run:
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
plot(r, type="interval")
sbar(15, c(6.3, 50), type="bar", below="km", label=c(0,7.5,15), cex=.8)
north(type=3, cex=.8)
north(xy=c(6.7, 49.9), type=2, angle=45, label="NE")
north(xy=c(6.6, 49.7), type=5, cex=1.25)
north(xy=c(5.5, 49.6), type=9)
north(d=.05, xy=c(5.5, 50), angle=180, label="S", lwd=2, col="blue")

## all arrows
r <- rast(res=10)
values(r) <- 1
plot(r, col="white", axes=FALSE, legend=FALSE, mar=c(0,0,0,0), reset=TRUE)
for (i in 1:12) {
  x = -200+i*30
  north(xy=cbind(x,30), type=i)
  text(x, -20, i, xpd=TRUE)
}

## End(Not run)
```

not.na

is not NA

Description

Shortcut method to avoid the two-step `!is.na(x)`

Usage

```
## S4 method for signature 'SpatRaster'
not.na(x, filename="", ...)
```

Arguments

x	SpatRaster
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

seealso[Compare-methods](#)**Examples**

```
r <- rast(ncols=10, nrows=10, vals=1)
r[10:20] <- NA
x <- not.na(r)
```

options

*Options***Description**

Class and methods for showing and setting general options for terra.

Usage

```
terraOptions(...)
```

Arguments

... option names and values (see Details). Or missing, to show the current options

Details

The following options are available.

memfrac - value between 0 and 0.9 (larger values give a warning). The fraction of RAM that may be used by the program.

memmin - if memory required is below this threshold (in GB), the memory is assumed to be available. Otherwise, terra checks if it is available.

memmax - the maximum amount of RAM (in GB) that terra is allowed to use when processing a raster dataset. Should be less than what is detected (see [mem_info](#)), and higher values are ignored. Set it to a negative number or NA to not set this option. `terraOptions` only shows the value of `memmax` if it is set.

tempdir - directory where temporary files are written. The default what is returned by `tempdir()`.

datatype - default data type. See [writeRaster](#)

todisk - logical. If TRUE write all raster data to disk (temp file if no file name is specified). For debugging.

progress - non-negative integer. A progress bar is shown if the number of chunks in which the data is processed is larger than this number. No progress bar is shown if the value is zero

verbose - logical. If TRUE debugging info is printed for some functions

Examples

```
terraOptions()
terraOptions(memfrac=0.5, tempdir = "c:/temp")
terraOptions(progress=10)
terraOptions()
```

origin	<i>Origin</i>
--------	---------------

Description

Get or set the coordinates of the point of origin of a `SpatRaster`. This is the point closest to (0, 0) that you could get if you moved towards that point in steps of the x and y resolution.

Usage

```
## S4 method for signature 'SpatRaster'
origin(x)

## S4 replacement method for signature 'SpatRaster'
origin(x)<-value
```

Arguments

x	<code>SpatRaster</code>
value	numeric vector of length 1 or 2

Value

A vector of two numbers (x and y coordinates)

Examples

```
r <- rast(xmin=-0.5, xmax = 9.5, ncols=10)
origin(r)
origin(r) <- c(0,0)
r
```

pairs	<i>Pairs plot (matrix of scatterplots)</i>
-------	--

Description

Pair plots of layers in a `SpatRaster`. This is a wrapper around graphics function `pairs`.

Usage

```
## S4 method for signature 'SpatRaster'  
pairs(x, hist=TRUE, cor=TRUE, use="pairwise.complete.obs", maxcells=100000, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>hist</code>	logical. If TRUE a histogram of the values is shown on the diagonal
<code>cor</code>	logical. If TRUE the correlation coefficient is shown in the upper panels
<code>use</code>	argument passed to the <code>cor</code> function
<code>maxcells</code>	integer. Number of pixels to sample from each layer of a large <code>SpatRaster</code>
<code>...</code>	additional arguments (graphical parameters)

See Also

`boxplot`, `hist`

Examples

```
r <- rast(system.file("ex/elev.tif", package="terra"))  
s <- c(r, 1/r, sqrt(r))  
names(s) <- c("elevation", "inverse", "sqrt")  
pairs(s)  
  
# to make individual histograms:  
hist(r)  
# or scatter plots:  
plot(s[[1]], s[[2]])
```

patches *Detect patches (clumps) of cells*

Description

Detect patches (clumps). Patches are groups of cells that are surrounded by cells that are NA. Set `zeroAsNA` to TRUE to also identify patches separated by cells with values of zero.

Usage

```
## S4 method for signature 'SpatRaster'
patches(x, directions=4, zeroAsNA=FALSE, allowGaps=TRUE, filename="", ...)
```

Arguments

<code>x</code>	SpatRaster
<code>directions</code>	integer indicating which cells are considered adjacent. Should be 8 (Queen's case) or 4 (Rook's case)
<code>zeroAsNA</code>	logical. If TRUE treat cells that are zero as if they were NA
<code>allowGaps</code>	logical. If TRUE there may be gaps in the patch IDs (e.g. you may have patch IDs 1, 2, 3 and 5, but not 4). If it is FALSE, these numbers will be recoded from 1 to the number of patches (4 in this example)
<code>filename</code>	character. Output filename
<code>...</code>	options for writing files as in writeRaster

Value

SpatRaster. Cell values are patch numbers

See Also

[focal](#), [boundaries](#)

Examples

```
r <- rast(nrows=18, ncols=36, xmin=0)
r[1:2, 5:8] <- 1
r[5:8, 2:6] <- 1
r[7:12, 22:36] <- 1
r[15:16, 18:29] <- 1
p <- patches(r)

# zero as background instead of NA
r <- rast(nrows=10, ncols=10, xmin=0, vals=0)
r[3, 3] <- 10
r[4, 4] <- 10
r[5, 5:8] <- 12
```

```

r[6, 6:9] <- 12

# treat zeros as NA

p4 <- patches(r, zeroAsNA=TRUE)
p8 <- patches(r, 8, zeroAsNA=TRUE)

### patches for different values
# remove zeros manually
rr <- classify(r, cbind(0, NA))

# make layers for each value
s <- segregate(rr, keep=TRUE, other=NA)
p <- patches(s)

### patch ID values are not guaranteed to be consecutive
r <- rast(nrows=5, ncols=10, xmin=0)
set.seed(0)
values(r) <- round(runif(ncell(r))*0.7)
rp <- patches(r, directions=8, zeroAsNA=TRUE)
plot(rp, type="classes"); text(rp)

## unless you set allowGaps=FALSE
rp <- patches(r, directions=8, zeroAsNA=TRUE, allowGaps=FALSE)
plot(rp, type="classes"); text(rp)

### use zonal to remove small patches
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
x <- classify(r, cbind(-Inf, 400, NA))
y <- patches(x)
# remove patches smaller than 100 ha
rz <- zonal(cellSize(y, unit="ha"), y, sum, as.raster=TRUE)
s <- ifel(rz < 100, NA, y)

```

perim

Perimeter or length

Description

This method returns the length of lines or the perimeter of polygons.

When the crs is not longitude/latitude, you may get more accurate results by first un-projecting the `SpatVector` (you can use [project](#) to transform the crs to longitude/latitude)

Usage

```

## S4 method for signature 'SpatVector'
perim(x)

```

Arguments

x SpatVector

Value

numeric (m)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
perim(v)
```

persp

Perspective plot

Description

Perspective plot of a SpatRaster. This is an implementation of a generic function in the graphics package.

Usage

```
## S4 method for signature 'SpatRaster'
persp(x, maxcells=100000, ...)
```

Arguments

x SpatRaster. Only the first layer is used

maxcells integer > 0. Maximum number of cells to use for the plot. If maxpixels < ncell(x), spatSample(method="regular") is used before plotting

... Any argument that can be passed to [persp](#) (graphics package)

See Also

[persp](#), [contour](#), [plot](#)

Examples

```
r <- rast(system.file("ex/elev.tif", package="terra"))
persp(r)
```


plet

*Plot with leaflet***Description**

Plot the values of a `SpatRaster` or `SpatVector` to make an interactive leaflet map that is displayed in a browser.

These methods require that `packageVersion("leaflet") > "2.1.1"` or the development version of leaflet that you can install with `remotes::install_github("rstudio/leaflet")`.

Usage

```
## S4 method for signature 'SpatRaster'
plet(x, y=1, col, alpha=0.8, main=names(x), tiles=NULL,
     wrap=TRUE, maxcell=500000, legend="bottomright",
     shared=FALSE, panel=FALSE, collapse=TRUE, map=NULL)

## S4 method for signature 'SpatVector'
plet(x, y="", col, alpha=1, fill=0, main=y, cex=1, lwd=2, popup=TRUE,
     label=FALSE, split=FALSE, tiles=c("Streets", "Esri.WorldImagery", "OpenTopoMap"),
     wrap=TRUE, legend="bottomright", collapse=FALSE, map=NULL)

## S4 method for signature 'SpatVectorCollection'
plet(x, col, alpha=1, fill=0, cex=1, lwd=2, popup=TRUE,
     label=FALSE, tiles=c("Streets", "Esri.WorldImagery", "OpenTopoMap"), wrap=TRUE,
     legend="bottomright", collapse=FALSE, map=NULL)

## S4 method for signature 'leaflet'
lines(x, y, col, lwd=2, alpha=1)

## S4 method for signature 'leaflet'
points(x, y, col, cex=1, alpha=1, popup=FALSE)
```

Arguments

<code>x</code>	<code>SpatRaster</code> , <code>SpatVector</code> , or leaflet object
<code>y</code>	missing, or positive integer, or character (variable or layer name) indicating the layer(s) to be plotted. If <code>x</code> is a <code>SpatRaster</code> , you can select multiple layers
<code>col</code>	character. Vector of colors or color generating function
<code>alpha</code>	Number between 0 and 1 to set the transparency for lines (0 is transparent, 1 is opaque)
<code>fill</code>	Number between 0 and 1 to set the transparency for polygon areas (0 is transparent, 1 is opaque)

tiles	character or NULL. Names of background tile providers
wrap	logical. if TRUE, tiles wrap around
maxcell	positive integer. Maximum number of cells to use for the plot
legend	character to indicate the legend position ("bottomleft", "bottomright", "topleft" or "topright") or NULL to suppress the legend
main	character. Title for the legend. The length should be 1 if x is a SpatVector and length nlyr(x) if x is a SpatRaster
shared	logical. Should the legend be the same for all rasters (if multiple layers of SpatRaster x are mapped)
map	leaflet object
collapse	logical. Should the layers "control" panel be collapsed?
split	logical. IF TRUE a check-box is created to toggle each value in codey (If x is a SpatVector)
cex	numeric. point size magnifier. See par
lwd	numeric, line-width. See par
popup	logical. Should pop-ups be created?
label	logical. Should mouse-over labels be added?
panel	logical. Should SpatRaster layers be shown as a panel"

See Also[plot](#)**Examples**

```
## Not run:
if (require(leaflet)) {
  if (packageVersion("leaflet") > "2.1.1") {

    v <- vect(system.file("ex/lux.shp", package="terra"))
    p <- spatSample(as.polygons(v, ext=T), 10)
    values(p) = data.frame(id=11:20, name=letters[1:10])

    m <- plet(v, "NAME_1", alpha=.5, tiles="")
    m <- points(m, p, col="gray", cex=2, popup=T)
    lines(m, v)

    plet(v, "NAME_1", split=TRUE, alpha=.2) |>
      points(p, col="gray", cex=2, popup=T) |> lines(v)

    s <- svc(v, p)
    names(s) <- c("the polys", "set of points")
    plet(s, col=c("red", "blue"), lwd=1)

    r <- rast(system.file("ex/elev.tif", package="terra"))
```

```

plet(r, main="Hi\nthere") |> lines(v, lwd=1)

plet(r, tiles="Streets") |> lines(v, lwd=2, col="blue")

x <- c(r, 50*classify(r, 5))
names(x) <- c("first", "second")

# each their own legend
plet(x, 1:2, tiles="Streets", collapse=FALSE) |> lines(v, lwd=2, col="blue")

# shared legend
plet(x, 1:2, tiles="Streets", shared=TRUE, collapse=FALSE) |> lines(v, lwd=2, col="blue")

}}
## End(Not run)

```

plot

Make a map

Description

Plot the values of a `SpatRaster` or `SpatVector` to make a map. See [lines](#) to add a `SpatVector` to an existing map.

Usage

```

## S4 method for signature 'SpatRaster,numeric'
plot(x, y=1, col, type, mar=NULL, legend=TRUE, axes=TRUE, plg=list(), pax=list(),
      maxcell=500000, smooth=FALSE, range=NULL, levels=NULL, all_levels=FALSE,
      fun=NULL, colNA=NULL, alpha=NULL, sort=FALSE, decreasing=FALSE, grid=FALSE,
      ext=NULL, reset=FALSE, ...)

## S4 method for signature 'SpatRaster,missing'
plot(x, y, maxcell=500000, main, mar=NULL, nc, nr, maxnl=16, legend, ...)

## S4 method for signature 'SpatRaster,character'
plot(x, y, ...)

## S4 method for signature 'SpatVector,character'
plot(x, y, col, type, mar=NULL, legend=TRUE, add=FALSE, axes=!add,
      main=y, buffer=TRUE, background=NULL, grid=FALSE, ext=NULL,
      sort=TRUE, decreasing=FALSE, plg=list(), pax=list(), nr, nc, ...)

## S4 method for signature 'SpatVector,numeric'
plot(x, y, ...)

## S4 method for signature 'SpatVector,missing'
plot(x, y, values=NULL, ...)

```

```
## S4 method for signature 'SpatExtent,missing'
plot(x, y, ...)
```

Arguments

<code>x</code>	SpatRaster or SpatVector
<code>y</code>	missing or positive integer or name indicating the layer(s) to be plotted
<code>col</code>	character. Colors. The default is <code>rev(grDevices::terrain.colors(50))</code> . If <code>x</code> is a SpatRaster, it can also be a data.frame with two columns (value, color) to get a "classes" type legend or with three columns (from, to, color) to get an "interval" type legend
<code>type</code>	character. Type of map/legend. One of "continuous", "classes", or "interval". If not specified, the type is chosen based on the data
<code>mar</code>	numeric vector of length 4 to set the margins of the plot (to make space for the legend). The default is (3.1, 3.1, 2.1, 7.1) for a single plot with a legend and (3.1, 3.1, 2.1, 2.1) otherwise. Use <code>mar=NA</code> to not set the margins
<code>legend</code>	logical or character. If not FALSE a legend is drawn. The character value can be used to indicate where the legend is to be draw. For example "topright" or "bottomleft". Use <code>plg</code> for more refined placement (SpatVector data only)
<code>axes</code>	logical. Draw axes?
<code>buffer</code>	logical. If TRUE the plotting area is slightly larger than the extent of <code>x</code>
<code>background</code>	background color. Default is no color (white)
<code>plg</code>	list with parameters for drawing the legend. See the arguments for legend
<code>pax</code>	list with parameters for drawing axes. See the arguments for axis
<code>maxcell</code>	positive integer. Maximum number of cells to use for the plot
<code>smooth</code>	logical. If TRUE the cell values are smoothed (for continuous legend)
<code>range</code>	numeric. minimum and maximum values to be used for the continuous legend
<code>levels</code>	character. labels for the legend when <code>type="classes"</code>
<code>all_levels</code>	logical. If TRUE, the legend shows all levels of a categorical raster, even if they are not present in the data
<code>fun</code>	function to be called after plotting each SpatRaster layer to add something to each map (such as text, legend, lines). For example, with SpatVector <code>v</code> , you could do <code>fun=function() lines(v)</code> . The function may have one argument, representing the the layer that is plotted (1 to the number of layers)
<code>colNA</code>	character. color for the NA values
<code>alpha</code>	Either a single numeric between 0 and 1 to set the transparency for all colors (0 is transparent, 1 is opaque) or a SpatRaster with values between 0 and 1 to set the transparency by cell. To set the transparency for a given color, set it to the colors directly
<code>sort</code>	logical. If TRUE legends with categorical values are sorted. If <code>x</code> is a SpatVector you can also supply a vector of the unique values, in the order in which you want them to appear in the legend

decreasing	logical. If TRUE, legends are sorted in decreasing order
grid	logical. If TRUE grid lines are drawn. Their properties such as type and color can be set with the <code>pax</code> argument
nc	positive integer. Optional. The number of columns to divide the plotting device in (when plotting multiple layers)
nr	positive integer. Optional. The number of rows to divide the plotting device in (when plotting multiple layers)
main	character. Main plot titles (one for each layer to be plotted)
maxnl	positive integer. Maximum number of layers to plot (for a multi-layer object)
add	logical. If TRUE add the object to the current plot
ext	SpatExtent. Can be use instead of <code>xlim</code> and <code>ylim</code> to set the extent of the plot
reset	logical. If TRUE add the margins (see argument <code>mar</code>) are reset to what they were before calling <code>plot</code> ; doing so may affect the display of additional objects that are added to the map (e.g. with <code>lines</code>)
values	Either a vector with values to be used for plotting or a two-column data.frame, where the first column matches a variable in <code>x</code> and the second column has the values to be plotted
...	arguments passed to <code>plot("SpatRaster", "numeric")</code> and additional graphical arguments

See Also

[points](#), [lines](#), [polys](#), [image](#), [scatterplot](#), [sbar](#)

Examples

```
## raster
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
plot(r)

plot(r, type="interval")

e <- c(6.3, 6.35, 49.9, 50.1)
plot(r, plg=list(ext=e, title="Title\n", title.cex=1.25), pax=list(sides=1:2))

d <- classify(r, c(100,200,300,400,500,600))
plot(d, type="classes")

plot(d, type="interval", breaks=1:5)
plot(d, type="interval", breaks=c(1,4,5), plg=list(legend=c("1-4", "4-5")))
plot(d, type="classes", plg=list(legend=c("Mr", "Xx", "As", "Zx", "Bb"), x="bottomright"))

x <- trunc(r/200)
levels(x) <- data.frame(id=0:2, element=c("earth", "wind", "fire"))
plot(x, plg=list(x="topright"), mar=c(2,2,2,2))
```

```

# two plots with the same legend
dev.new(width=6, height=4, noRStudioGD = TRUE)
par(mfrow=c(1,2))
plot(r, range=c(50,600))
plot(r/2, range=c(50,600))

# as you only need one legend:
par(mfrow=c(1,2))
plot(r, range=c(50,600), mar=c(4, 3, 4, 3), plg=list(shrink=0.9, cex=.8),
pax=list(sides=1:2, cex.axis=.6))
#text(182500, 335000, "Two maps, one plot", xpd=NA)
plot(r/2, range=c(50,600), mar=c(4, 2, 4, 4), legend=FALSE,
pax=list(sides=c(1,4), cex.axis=.6))

# multi-layer with RGB
s <- rast(system.file("ex/logo.tif", package="terra"))
s
plot(s)
# remove RGB
plot(s*1)
# or use layers
plot(s, 1)
plot(s, 1:3)

# fix legend by linking values and colors

x = rast(nrows = 2, ncols = 2, vals=1)
y = rast(nrows = 2, ncols = 2, vals=c(1,2,2,1))
cols = data.frame(id=1:2, col=c("red", "blue"))
plot(c(x,y), col=cols)

r = rast(nrows=10, ncols=10, vals=1:100)
dr = data.frame(from=c(5,33,66,150), to=c(33, 66, 95,200), col=rainbow(4))
plot(r, col=dr)

### SpatVector

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)

plot(v)

plot(v, 2, pax=list(sides=1:2), plg=list(x=6.2, y=50.2, cex=1.2))

plot(v, 4, pax=list(sides=1:2), plg=list(x=6.2, y=50.2, ncol=2), main="")

plot(v, 1, plg=list(x=5.9, y=49.37, horiz=TRUE, cex=1.1), main="", mar=c(5,2,0.5,0.5))

plot(v, density=1:12, angle=seq(18, 360, 20), col=rainbow(12))

plot(v, "NAME_2", col=rainbow(12), border=c("gray", "blue"), lwd=3, type="classes")

```

```
plot(v, "AREA", type="interval", breaks=3, mar=c(3.1, 3.1, 2.1, 3.1),
     plg=list(x="topright"), main="")

plot(v, "AREA", type="interval", breaks=c(0,200,250,350), mar=c(2,2,2,2),
     plg=list(legend=c("<200", "200-250", ">250"), cex=1,
             bty="o", x=6.4, y=50.125, box.lwd=2, bg="light yellow", title="My Legend"))
```

plotRGB

Red-Green-Blue plot of a multi-layered SpatRaster

Description

Make a Red-Green-Blue plot based on three layers in a `SpatRaster`. The layers (sometimes referred to as "bands" because they may represent different bandwidths in the electromagnetic spectrum) are combined such that they represent the red, green and blue channel. This function can be used to make "true" (or "false") color images from Landsat and other multi-spectral satellite images.

Note that the margins of the plot are set to zero (no axes or titles are visible) but can be set with the `mar` argument.

An alternative way to plot RGB images is to first use `colorize` to create a single layer `SpatRaster` with a color-table and then use `plot`.

Usage

```
## S4 method for signature 'SpatRaster'
plotRGB(x, r=1, g=2, b=3, a=NULL, scale, maxcell=500000, mar=0,
        stretch=NULL, ext=NULL, smooth=FALSE, colNA="white", alpha, bgamma,
        addfun=NULL, zlim=NULL, zlimcol=NULL, axes=FALSE, xlab="", ylab="",
        asp=NULL, add=FALSE, interpolate, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>r</code>	integer. Index of the Red channel, between 1 and <code>nlyr(x)</code>
<code>g</code>	integer. Index of the Green channel, between 1 and <code>nlyr(x)</code>
<code>b</code>	integer. Index of the Blue channel, between 1 and <code>nlyr(x)</code>
<code>a</code>	integer. Index of the alpha (transparency) channel, between 1 and <code>nlyr(x)</code> . If not NULL, argument <code>alpha</code> is ignored
<code>scale</code>	integer. Maximum (possible) value in the three channels. Defaults to 255 or to the maximum value of <code>x</code> if that is known and larger than 255
<code>maxcell</code>	integer > 0. Maximum number of pixels to use
<code>mar</code>	numeric vector recycled to length 4 to set the margins of the plot. Use <code>mar=NULL</code> or <code>mar=NA</code> to not set the margins
<code>stretch</code>	character. Option to stretch the values to increase contrast: "lin" (linear) or "hist" (histogram)

ext	An SpatExtent object to zoom in to a region of interest (see draw)
smooth	logical. If TRUE, smooth the image when drawing to get the appearance of a higher spatial resolution
colNA	color for the background (NA values)
alpha	transparency. Integer between 0 (transparent) and 255 (opaque)
bgalpha	Background transparency. Integer between 0 (transparent) and 255 (opaque)
addfun	Function to add additional items such as points or polygons to the plot (map). See plot
zlim	numeric vector of length 2. Range of values to plot (optional)
zlimcol	If NULL the values outside the range of zlim get the color of the extremes of the range. If zlimcol has any other value, the values outside the zlim range get the color of NA values (see colNA)
axes	logical. If TRUE axes are drawn (and arguments such as main="title" will be honored)
xlab	character. Label of x-axis
ylab	character. Label of y-axis
asp	numeric. Aspect (ratio of x and y. If NULL, and appropriate value is computed to match data for the longitude/latitude coordinate reference system, and 1 for planar coordinate reference systems
add	logical. If TRUE add values to current plot
interpolate	logical. Do not use, to be removed
...	graphical parameters as in plot or rasterImage

See Also

[plot](#), [colorize](#), [RGB](#)

Examples

```
b <- rast(system.file("ex/logo.tif", package="terra"))
plotRGB(b)
plotRGB(b, mar=c(2,2,2,2))
plotRGB(b, 3, 2, 1)

b[1000:2000] <- NA
plotRGB(b, 3, 2, 1, stretch='hist')
```

 predict

Spatial model predictions

Description

Make a `SpatRaster` object with predictions from a fitted model object (for example, obtained with `glm` or `randomForest`). The first argument is a `SpatRaster` object with the predictor variables. The `names` in the Raster object should exactly match those expected by the model. Any regression like model for which a `predict` method has been implemented (or can be implemented) can be used.

The method should work if the model's `predict` function returns a vector, matrix or `data.frame` (or a list that can be coerced to a `data.frame`). In other cases it may be necessary to provide a custom "predict" function that wraps the model's `predict` function to return the values in the required form. See the examples.

This approach of using model predictions is commonly used in remote sensing (for the classification of satellite images) and in ecology, for species distribution modeling.

Usage

```
## S4 method for signature 'SpatRaster'
predict(object, model, fun=predict, ..., factors=NULL, const=NULL, na.rm=FALSE,
        index=NULL, cores=1, cpkgs=NULL, filename="", overwrite=FALSE, wopt=list())
```

Arguments

<code>object</code>	<code>SpatRaster</code>
<code>model</code>	fitted model of any class that has a "predict" method (or for which you can supply a similar method as <code>fun</code> argument. E.g. <code>glm</code> , <code>gam</code> , or <code>randomForest</code>)
<code>fun</code>	function. The <code>predict</code> function that takes <code>model</code> as first argument. The default value is <code>predict</code> , but can be replaced with e.g. <code>predict.se</code> (depending on the type of model), or your own custom function
<code>...</code>	additional arguments for <code>fun</code>
<code>const</code>	<code>data.frame</code> . Can be used to add a constant value as a predictor variable so that you do not need to make a <code>SpatRaster</code> layer for it
<code>factors</code>	list with levels for factor variables. The list elements should be named with names that correspond to names in <code>object</code> such that they can be matched. This argument may be omitted for standard models such as "glm" as the <code>predict</code> function will extract the levels from the <code>model</code> object, but it is necessary in some other cases (e.g. <code>cforest</code> models from the <code>party</code> package)
<code>na.rm</code>	logical. If <code>TRUE</code> , cells with NA values in the predictors are removed from the computation. This option prevents errors with models that cannot handle NA values. In most other cases this will not affect the output. An exception is when predicting with a model that returns predicted values even if some (or all!) variables are NA
<code>index</code>	integer. To select subset of output variables

cores	positive integer. If cores > 1, a 'parallel' package cluster with that many cores is created and used
cpkgs	character. The package(s) that need to be loaded on the nodes to be able to run the model.predict function (see examples)
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in writeRaster

Value

SpatRaster

Examples

```
logo <- rast(system.file("ex/logo.tif", package="terra"))
names(logo) <- c("red", "green", "blue")
p <- matrix(c(48, 48, 48, 53, 50, 46, 54, 70, 84, 85, 74, 84, 95, 85,
  66, 42, 26, 4, 19, 17, 7, 14, 26, 29, 39, 45, 51, 56, 46, 38, 31,
  22, 34, 60, 70, 73, 63, 46, 43, 28), ncol=2)

a <- matrix(c(22, 33, 64, 85, 92, 94, 59, 27, 30, 64, 60, 33, 31, 9,
  99, 67, 15, 5, 4, 30, 8, 37, 42, 27, 19, 69, 60, 73, 3, 5, 21,
  37, 52, 70, 74, 9, 13, 4, 17, 47), ncol=2)

xy <- rbind(cbind(1, p), cbind(0, a))

# extract predictor values for points
e <- extract(logo, xy[,2:3])

# combine with response (excluding the ID column)
v <- data.frame(cbind(pa=xy[,1], e))

#build a model, here with glm
model <- glm(formula=pa~., data=v)

#predict to a raster
r1 <- predict(logo, model)

plot(r1)
points(p, bg='blue', pch=21)
points(a, bg='red', pch=21)

# logistic regression
model <- glm(formula=pa~., data=v, family="binomial")
r1log <- predict(logo, model, type="response")

# to get the probability and standard error
r1se <- predict(logo, model, se.fit=TRUE)

# or provide a custom predict function
```

```

predfun <- function(model, data) {
  v <- predict(model, data, se.fit=TRUE)
  cbind(p=as.vector(v$fit), se=as.vector(v$se.fit))
}

r2 <- predict(logo, model, fun=predfun)

# principal components of a SpatRaster
# here using sampling to simulate an object too large
# to feed all its values to prcomp

sr <- values(spatSample(logo, 100, as.raster=TRUE))
pca <- prcomp(sr)

x <- predict(logo, pca)
plot(x)

## parallelization
## Not run:
## simple case with GLM
model <- glm(formula=pa~., data=v)
p <- predict(logo, model, cores=2)

## The above does not work with a model from a contributed
## package, as the package needs to be loaded in each core.
## Below are three approaches to deal with that

library(randomForest)
rfm <- randomForest(formula=pa~., data=v)

## approach 0 (not parallel)
rp0 <- predict(logo, rfm)

## approach 1, use the "cpkgs" argument
rp1 <- predict(logo, rfm, cores=2, cpkgs="randomForest")

## approach 2, write a custom predict function that loads the package
rfun <- function(mod, dat, ...) {
  library(randomForest)
  predict(mod, dat, ...)
}
rp2 <- predict(logo, rfm, fun=rfun, cores=2)

## approach 3, write a parallelized custom predict function
rfun <- function(mod, dat, ...) {
  ncls <- length(cls)
  nr <- nrow(dat)
  s <- split(dat, rep(1:ncls, each=ceiling(nr/ncls), length.out=nr))
  unlist( parallel::clusterApply(cls, s, function(x, ...) predict(mod, x, ...)) )
}

library(parallel)
cls <- parallel::makeCluster(2)

```

```

parallel::clusterExport(cls, c("rfm", "rfun", "randomForest"))
rp3 <- predict(logo, rfm, fun=rfun)
parallel::stopCluster(cls)

plot(c(rp0, rp1, rp2, rp3))

### with two output variables (probabilities for each class)
v$pa <- as.factor(v$pa)
rfm2 <- randomForest(formula=pa~., data=v)
rfp <- predict(logo, rfm2, cores=2, type="prob", cpkgs="randomForest")

## End(Not run)

```

project

Change the coordinate reference system

Description

Change the coordinate reference system ("project") of a `SpatVector`, `SpatRaster` or a matrix with coordinates.

Usage

```

## S4 method for signature 'SpatVector'
project(x, y)

## S4 method for signature 'SpatRaster'
project(x, y, method, mask=FALSE, align=FALSE,
        gdal=TRUE, res=NULL, origin=NULL, filename="", ...)

## S4 method for signature 'matrix'
project(x, from, to)

```

Arguments

x	<code>SpatRaster</code> or <code>SpatVector</code>
y	if (x is a <code>SpatRaster</code> , the preferred approach is for y to be a <code>SpatRaster</code> as well, serving as a template for the geometry (extent and resolution) of the output <code>SpatRaster</code> . Alternatively, you can provide a coordinate reference system (CRS) description. You can use the following formats to define coordinate reference systems: WKT, PROJ.4 (e.g., <code>+proj=longlat +datum=WGS84</code>), or an EPSG code (e.g., <code>"epsg:4326"</code>). But note that the PROJ.4 notation has been deprecated, and you can only use it with the WGS84/NAD83 and NAD27 datums. Other datums are silently ignored. If x is a <code>SpatVector</code> , you can provide a crs definition as discussed above, or any other object from which such a crs can be extracted with crs

method	character. Method used for estimating the new cell values of a SpatRaster. One of: near: nearest neighbor. This method is fast, and it can be the preferred method if the cell values represent classes. It is not a good choice for continuous values. This is used by default if the first layer of x is categorical. bilinear: bilinear interpolation. This is the default if the first layer of x is numeric (not categorical). cubic: cubic interpolation. cubicspline: cubic spline interpolation.
mask	logical. If TRUE, mask out areas outside the input extent (see example with Robinson projection)
align	logical. If TRUE, and y is a SpatRaster, the template is used for the spatial resolution and origin, but the extent is set such that all of the extent of x is included
gdal	logical. If TRUE the GDAL-warp algorithm is used. Otherwise a slower internal algorithm is used that may be more accurate if there is much variation in the cell sizes of the output raster. Only the near and bilinear algorithms are available for the internal algorithm
res	numeric. Can be used to set the resolution of the output raster if y is a CRS
origin	numeric. Can be used to set the origon of the output raster if y is a CRS
filename	character. Output filename
...	additional arguments for writing files as in writeRaster
from	character. Coordinate reference system for x
to	character. Output coordinate reference system

Value

SpatVector or SpatRaster

Note

User beware. Sadly, the PROJ.4 notation has been partly deprecated in the GDAL/PROJ library that is used by this function. You can still use this notation, but **only** with the the WGS84 datum. Other datums are silently ignored.

When printing a Spat* object the PROJ.4 notation is shown because it is the most concise and clear format available. However, internally a WKT representation is used (see [crs](#)).

Transforming (projecting) raster data is fundamentally different from transforming vector data. Vector data can be transformed and back-transformed without loss in precision and without changes in the values. This is not the case with raster data. In each transformation the values for the new cells are estimated in some fashion. Therefore, if you need to match raster and vector data for analysis, you should generally transform the vector data.

When using this method with a SpatRaster, the preferable approach is to provide a template SpatRaster as argument y. The template is then another raster dataset that you want your data to align with. If you do not have a template to begin with, you can do `project(x, crs)` and then

manipulate the output to get the template you want. For example, where possible use whole numbers for the extent and resolution so that you do not have to worry about small differences in the future. You can use commands like `dim(z) = c(180, 360)` or `res(z) <- 100000`.

The output resolution should be similar to the input resolution, but there is not "correct" resolution in raster transformation; but it is not obvious what this resolution is if you are using lon/lat data that spans a large North-South extent.

See `sf_proj_pipelines` to use specific "projection pipe lines"

See Also

[crs](#), [resample](#)

Examples

```
## SpatRaster
a <- rast(ncols=40, nrows=40, xmin=-110, xmax=-90, ymin=40, ymax=60,
          crs="+proj=longlat +datum=WGS84")
values(a) <- 1:ncell(a)
newcrs="+proj=lcc +lat_1=48 +lat_2=33 +lon_0=-100 +datum=WGS84"
b <- rast(ncols=94, nrows=124, xmin=-944881, xmax=935118, ymin=4664377, ymax=7144377, crs=newcrs)
w <- project(a, b)

## SpatVector
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
crs(v, proj=TRUE)
cat(crs(v), "\n")

crs <- "+proj=moll +lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84"
p <- project(v, crs)
p

project(v, "EPSG:2169")
```

quantile

Quantiles of spatial data

Description

Compute quantiles for each cell across the layers of a `SpatRaster`.

You can use `global(x, fun=quantile)` to instead compute quantiles across cells for each layer.

You can also use this method to compute quantiles of the numeric variables of a `SpatVector`.

Usage

```
## S4 method for signature 'SpatRaster'
quantile(x, probs=seq(0, 1, 0.25), na.rm=FALSE, filename="", ...)

## S4 method for signature 'SpatVector'
quantile(x, probs=seq(0, 1, 0.25), ...)
```

Arguments

x	SpatRaster or SpatVector
probs	numeric vector of probabilities with values in [0,1]
na.rm	logical. If TRUE, NA's are removed from x before the quantiles are computed
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster with layers representing quantiles

See Also

[app](#)

Examples

```
r <- rast(system.file("ex/logo.tif", package="terra"))
rr <- c(r/2, r, r*2)
qr <- quantile(rr)
qr

## Not run:
# same but slower
qa <- app(rr, quantile)

## End(Not run)

#quantile by layer instead of by cell
qg <- global(r, quantile)
```

query

Query a SpatVectorProxy object

Description

Query a SpatVectorProxy to extract a subset

Usage

```
## S4 method for signature 'SpatVectorProxy'
query(x, start=1, n=nrow(x), vars=NULL, where=NULL,
      extent=NULL, filter=NULL)
```

Arguments

x	SpatVectorProxy
start	positive integer. The record to start reading at
n	positive integer. The number of records requested
vars	character. Variable names. Must be a subset of names(x)
where	character. expression like "NAME_1='California' AND ID > 3" , to subset records. Note that start and n are applied after executing the where statement
extent	Spat* object. The extent of the object is used as a spatial filter to select the geometries to read. Ignored if filter is not NULL
filter	SpatVector. Used as a spatial filter to select geometries to read (the convex hull is used for lines or points)

Value

SpatVector

See Also

[vect](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f, proxy=TRUE)
v

x <- query(v, vars=c("ID_2", "NAME_2"), start=5, n=2)
x

query(v, vars=c("ID_2", "NAME_1", "NAME_2"), where="NAME_1='Grevenmacher' AND ID_2 > 6")

## with an extent
e <- ext(5.9, 6.3, 49.9, 50)
x <- query(v, extent=e)

## with polygons
p <- as.polygons(e)
x <- query(v, filter=p)
x
```


rapp

*Range-Apply***Description**

Apply a function to a range of the layers of a `SpatRaster` that varies by cell. The range is specified for each cell one or two `SpatRasters` (arguments `first` and `last`). For either `first` or `last` you can use a numeric constant instead.

See [selectRange](#) to create a new `SpatRaster` by extracting one or more values starting at a cell-varying layer.

See [app](#) or [Summary-methods](#) if you want to apply a function to all cells (not a range), perhaps after making a [subset](#) of a `SpatRaster`.

Usage

```
## S4 method for signature 'SpatRaster'
rapp(x, first, last, fun, ..., allys=FALSE, fill=NA,
      clamp=FALSE, circular=FALSE, filename="", overwrite=FALSE, wopt=list())
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>first</code>	<code>SpatRaster</code> or positive integer between 1 and <code>nlyr(x)</code> , indicating the first layer in the range of layers to be considered
<code>last</code>	<code>SpatRaster</code> or positive integer between 1 and <code>nlyr(x)</code> , indicating the last layer in the range to be considered
<code>fun</code>	function to be applied
<code>...</code>	additional arguments passed to <code>fun</code>
<code>allys</code>	logical. If TRUE, values for all layers are passed to <code>fun</code> but the values outside of the range are set to <code>fill</code>
<code>fill</code>	numeric. The fill value for the the values outside of the range, for when <code>allys=TRUE</code>
<code>clamp</code>	logical. If FALSE and the specified range is outside <code>1:nlyr(x)</code> all cells are considered NA. Otherwise, the invalid part of the range is ignored
<code>circular</code>	logical. If TRUE the values are considered circular, such as the days of the year. In that case, if <code>first > last</code> the layers used are <code>c(first:nlyr(x), 1:last)</code> . Otherwise, the range would be considered invalid and NA would be returned
<code>filename</code>	character. Output filename
<code>overwrite</code>	logical. If TRUE, <code>filename</code> is overwritten
<code>wopt</code>	list with named options for writing files as in writeRaster

Value

`SpatRaster`

See Also

[selectRange](#), [app](#), [Summary-methods](#), [lapp](#), [tapp](#)

Examples

```
r <- rast(ncols=9, nrows=9)
values(r) <- 1:ncell(r)
s <- c(r, r, r, r, r, r)
s <- s * 1:6
s[1:2] <- NA
start <- end <- rast(r)
start[] <- 1:3
end[] <- 4:6
a <- rapp(s, start, end, fun="mean")
b <- rapp(s, start, 2, fun="mean")

# cumsum from start to nlyr(x). return all layers
r <- rapp(s, start, nlyr(s), cumsum, alllyrs=TRUE, fill=0)
# return only the final value
rr <- rapp(s, start, nlyr(s), function(i) max(cumsum(i)))
```

 rast

Create a SpatRaster

Description

Methods to create a SpatRaster. These objects can be created from scratch, from a filename, or from another object.

A SpatRaster represents a spatially referenced surface divided into three dimensional cells (rows, columns, and layers).

When a SpatRaster is created from a file, it does not load the cell (pixel) values into memory (RAM). It only reads the parameters that describe the geometry of the SpatRaster, such as the number of rows and columns and the coordinate reference system. The actual values will be read when needed.

Usage

```
## S4 method for signature 'character'
rast(x, subds=0, lyrs=NULL, drivers=NULL, opts=NULL)

## S4 method for signature 'missing'
rast(x, nrows=180, ncols=360, nlyrs=1, xmin=-180, xmax=180,
      ymin=-90, ymax=90, crs, extent, resolution, vals, names, time, units)

## S4 method for signature 'SpatRaster'
rast(x, nlyrs=nlyr(x), names, vals, keeptime=TRUE, keepunits=FALSE, props=FALSE)

## S4 method for signature 'matrix'
```

```

rast(x, type="", crs="", digits=6, extent=NULL)

## S4 method for signature 'data.frame'
rast(x, type="xyz", crs="", digits=6, extent=NULL)

## S4 method for signature 'array'
rast(x, crs="", extent=NULL)

## S4 method for signature 'list'
rast(x, warn=TRUE)

## S4 method for signature 'SpatRasterDataset'
rast(x)

## S4 method for signature 'SpatVector'
rast(x, ...)

## S4 method for signature 'SpatExtent'
rast(x, ...)

```

Arguments

x	filename (character), missing, SpatRaster, SpatRasterDataset, SpatExtent, SpatVector, matrix, array, list of SpatRaster objects. For other types it will be attempted to create a SpatRaster via ('as(x, "SpatRaster")')
subds	positive integer or character to select a sub-dataset. If zero or "", all sub-datasets are returned (if possible)
lyrs	positive integer or character to select a subset of layers (a.k.a. "bands")
drivers	character. GDAL drivers to consider
opts	character. GDAL dataset open options
nrows	positive integer. Number of rows
ncols	positive integer. Number of columns
nlyrs	positive integer. Number of layers
xmin	minimum x coordinate (left border)
xmax	maximum x coordinate (right border)
ymin	minimum y coordinate (bottom border)
ymax	maximum y coordinate (top border)
crs	character. Description of the Coordinate Reference System (map projection) in PROJ.4, WKT or authority:code notation. If this argument is missing, and the x coordinates are within -360 .. 360 and the y coordinates are within -90 .. 90, longitude/latitude is assigned
keptime	logical. If FALSE the time stamps are discarded
keepunits	logical. If FALSE the layer units are discarded
props	logical. If TRUE the properties (categories and color-table) are kept

extent	object of class <code>SpatExtent</code> . If present, the arguments <code>xmin</code> , <code>xmax</code> , <code>ymin</code> and <code>ymax</code> are ignored
resolution	numeric vector of length 1 or 2 to set the resolution (see res). If this argument is used, arguments <code>ncol</code> and <code>nrow</code> are ignored
vals	numeric. An optional vector with cell values (if fewer values are provided, these are recycled to reach the number of cells)
names	character. An optional vector with layer names (must match the number of layers)
time	time or date stamps for each layer
units	character. units for each layer
type	character. If the value is not "xyz", the raster has the same number of rows and columns as the matrix. If the value is "xyz", the matrix must have at least two columns, the first with <code>x</code> (or longitude) and the second with <code>y</code> (or latitude) coordinates that represent the centers of raster cells. The additional columns are the values associated with the raster cells.
digits	integer to set the precision for detecting whether points are on a regular grid (a low number of digits is a low precision). Only used when <code>type="xyz"</code>
warn	logical. If TRUE, a warnings about empty rasters may be emitted
...	additional arguments passed on to the <code>rast,missing-method</code>

Details

Files are read with the GDAL library. GDAL guesses the file format from the name, and/or tries reading it with different "drivers" (see [gdal](#)) until it succeeds. In very few cases this may cause a file to be opened with the wrong driver, and some information may be lost. For example, when a netCDF file is opened with the HDF5 driver. You can avoid that by using argument `rast("filename.ncdf", drivers="NETCDF")`

These classes hold a C++ pointer to the data "reference class" and that creates some limitations. They cannot be recovered from a saved R session either or directly passed to nodes on a computer cluster. Generally, you should use [writeRaster](#) to save `SpatRaster` objects to disk (and pass a filename or cell values of cluster nodes). Also see [wrap](#).

Value

`SpatRaster`

See Also

[sds](#) to create a `SpatRasterDataset` (4 dimensions) and [vect](#) for vector (points, lines, polygons) data

Examples

```
# Create a SpatRaster from scratch
x <- rast(nrows=108, ncols=21, xmin=0, xmax=10)

# Create a SpatRaster from a file
f <- system.file("ex/elev.tif", package="terra")
```

```

r <- rast(f)

s <- rast(system.file("ex/logo.tif", package="terra"))

# Create a skeleton with no associated cell values
rast(s)

# from a matrix
m <- matrix(1:25, nrow=5, ncol=5)
rm <- rast(m)

# from a "xyz" data.frame
d <- as.data.frame(rm, xy=TRUE)
head(d)
rast(d, type="xyz")

```

rasterize

Rasterize vector data

Description

Transfer values associated with the geometries of vector data to a raster

Usage

```

## S4 method for signature 'SpatVector,SpatRaster'
rasterize(x, y, field="", fun, ..., background=NA, touches=FALSE,
update=FALSE, sum=FALSE, cover=FALSE, filename="", overwrite=FALSE, wopt=list())

## S4 method for signature 'matrix,SpatRaster'
rasterize(x, y, values=1, fun, ..., background=NA,
update=FALSE, filename="", overwrite=FALSE, wopt=list())

```

Arguments

x	SpatVector or a two-column matrix (point coordinates)
y	SpatRaster
field	character or numeric. If field is a character, it should a variable name in x. If field is numeric it typically is a single number or a vector of length nrow(x). The values are recycled to nrow(x)
values	numeric. For when x is a matrix. Normally of length 1 or nrow(x). The values will be recycled to nrow(x)
fun	function, summarizing function that returns a single number; for when there are multiple points in one cell. For example mean, length (to get a count), min or max. Only used if x consists of points
...	additional arguments passed to fun if x has point geometries

background	numeric. Value to put in the cells that are not covered by any of the features of x. Default is NA
touches	logical. If TRUE, all cells touched by lines or polygons are affected, not just those on the line render path, or whose center point is within the polygon. If touches=TRUE, add cannot be TRUE
update	logical. If TRUE, the values of the input SpatRaster are updated
sum	logical. If TRUE, the values of overlapping geometries are summed instead of replaced; and background is set to zero. Only used if x does not consists of points
cover	logical. If TRUE and the geometry of x is polygons, the fraction of a cell that is covered by the polygons is returned. This is estimated by determining presence/absence of the polygon in at least 100 sub-cells (more of there are very few cells)
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[mask](#)**Examples**

```

r <- rast(xmin=0, ncols=18, nrows=18)

# generate points
set.seed(1)
p <- spatSample(r, 1000, xy=TRUE, replace=TRUE)

# rasterize points as a matrix
x <- rasterize(p, r, fun=sum)
y <- rasterize(p, r, value=1:nrow(p), fun=max)

# rasterize points as a SpatVector
pv <- vect(p)
xv <- rasterize(pv, r, fun=sum)

# Polygons
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
r <- rast(v, ncols=75, nrows=100)
z <- rasterize(v, r, "NAME_2")
plot(z)
lines(v)

```

rasterizeGeom

*Rasterize geometric properties of vector data***Description**

Rasterization of geometric properties of vector data. You can get the count of the number of geometries in each cell; the area covered by polygons; the length of the lines; or the number of lines that cross each cell. See [rasterize](#) for standard rasterization (of attribute values associated with geometries).

The area of polygons is intended for summing the area of polygons that are relatively small relative to the raster cells, and for when there may be multiple polygons per cell. See `rasterize(sum=TRUE)` for counting large polygons and `rasterize(cover=TRUE)` to get the fraction that is covered by larger polygons.

Usage

```
## S4 method for signature 'SpatVector,SpatRaster'
rasterizeGeom(x, y, fun="count", unit="m", filename="", ...)
```

Arguments

x	SpatVector
y	SpatRaster
fun	character. "count", "area", "length", or "crosses"
unit	character. "m" or "km"
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[rasterize](#)**Examples**

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
r <- rast(v, res=.1)

# length of lines
lms <- as.lines(v)
x <- rasterizeGeom(lms, r, fun="length", "km")
```

```

# count of points
set.seed(44)
pts <- spatSample(v, 100)
y <- rasterizeGeom(pts, r)

# area of polygons
pols <- buffer(pts, 1000)
z <- rasterizeGeom(pols, r, fun="area")

```

read and write

Read from, or write to, file

Description

Methods to read from or write chunks of values to or from a file. These are low level methods for programmers. Use `writeRaster` if you want to save an entire `SpatRaster` to file in one step. It is much easier to use.

To write chunks, begin by opening a file with `writeStart`, then write values to it in chunks using the list that is returned by `writeStart`. When writing is done close the file with `writeStop`.

`blocks` also returns chunk size information. For example for only reading raster data.

Usage

```

## S4 method for signature 'SpatRaster'
readStart(x)

## S4 method for signature 'SpatRaster'
readStop(x)

## S4 method for signature 'SpatRaster'
readValues(x, row=1, nrow=nrow(x), col=1, ncol=ncol(x), mat=FALSE, dataframe=FALSE, ...)

## S4 method for signature 'SpatRaster,character'
writeStart(x, filename="", overwrite=FALSE, n=4, ...)

## S4 method for signature 'SpatRaster'
writeStop(x)

## S4 method for signature 'SpatRaster,vector'
writeValues(x, v, start, nrow)

## S4 method for signature 'SpatRaster'
blocks(x, n=4)

fileBlockSize(x)

```


Arguments

x	SpatRaster
filename	character. Output filename
v	vector with cell values to be written
start	integer. Row number (counting starts at 1) from where to start writing v
row	positive integer. Row number to start from, should be between 1 and nrow(x)
nrows	positive integer. How many rows?
col	positive integer. Column number to start from, should be between 1 and ncol(x)
ncols	positive integer. How many columns? Default is the number of columns left after the start column
mat	logical. If TRUE, values are returned as a numeric matrix instead of as a vector, except when dataframe=TRUE. If any of the layers of x is a factor, the level index is returned, not the label. Use dataframe=TRUE to get the labels
dataframe	logical. If TRUE, values are returned as a data.frame instead of as a vector (also if matrix is TRUE)
overwrite	logical. If TRUE, filename is overwritten
n	positive integer indicating how many copies the data may be in memory at any point in time. This is used to determine how many blocks (large) datasets need to be read
...	For writeStart: additional arguments for writing files as in writeRaster For readValues: additional arguments for data.frame (and thus only relevant when dataframe=TRUE)

Value

readValues returns a vector, matrix, or data.frame

writeStart returns a list that can be used for processing the file in chunks.

The other methods invisibly return a logical value indicating whether they were successful or not. Their purpose is the side-effect of opening or closing files.

rectify

Rectify a SpatRaster

Description

Rectify a rotated SpatRaster into a non-rotated object

Usage

```
## S4 method for signature 'SpatRaster'
rectify(x, method="bilinear", aoi=NULL, snap=TRUE,
        filename="", ...)
```

Arguments

x	SpatRaster to be rectified
method	character. Method used to for resampling. See resample
aoi	SpatExtent or SpatRaster to crop x to a smaller area of interest; Using a SpatRaster allowing to set the exact output extent and output resolution
snap	logical. If TRUE, the origin and resolution of the output are the same as would the case when aoi = NULL. Only relevant if aoi is a SpatExtent
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

relate	<i>relate</i>
--------	---------------

Description

Get a matrix indicating the presence or absence of spatial relationships between geometries.

Usage

```
## S4 method for signature 'SpatVector,SpatVector'
relate(x, y, relation)

## S4 method for signature 'SpatVector,SpatVector'
is.related(x, y, relation)

## S4 method for signature 'SpatVector,missing'
relate(x, y, relation, pairs=FALSE, symmetrical=FALSE)
```

Arguments

x	SpatVector or SpatExtent
y	missing or as for x
relation	character. One of "intersects", "touches", "crosses", "overlaps", "within", "contains", "covers", "coveredby", "disjoint". Or a "DE-9IM" string such as "FF*FF*****". See wikipedia or geotools doc
pairs	logical. If TRUE a "from", "to" matrix is returned for the cases where the requested relation is TRUE
symmetrical	logical. If TRUE and pairs=TRUE, the relation between a pair is only included once. For example, the relation between geometry 1 and 3 is included, but the relation between 3 and 1 is not. Note that whole some relationships are symmetrical (e.g. "touches"), but that others are not (e.g. "within")

Value

matrix (relate) or vector (is.related)

See Also

[adjacent](#), [nearby](#), [intersect](#), [crop](#)

Examples

```
# polygons
p1 <- vect("POLYGON ((0 0, 8 0, 8 9, 0 9, 0 0))")
p2 <- vect("POLYGON ((5 6, 15 6, 15 15, 5 15, 5 6))")
p3 <- vect("POLYGON ((8 2, 9 2, 9 3, 8 3, 8 2))")
p4 <- vect("POLYGON ((2 6, 3 6, 3 8, 2 8, 2 6))")
p5 <- vect("POLYGON ((2 12, 3 12, 3 13, 2 13, 2 12))")
p6 <- vect("POLYGON ((10 4, 12 4, 12 7, 11 7, 11 6, 10 6, 10 4))")

p <- rbind(p1, p2, p3, p4, p5, p6)
plot(p, col=rainbow(6, alpha=.5))
lines(p, lwd=2)
text(p)

## relate SpatVectors
relate(p1, p2, "intersects")
relate(p1, p3, "touches")
relate(p1, p5, "disjoint")
relate(rbind(p1, p2), p4, "disjoint")

## relate geometries within SpatVectors
# which are completely separated?
relate(p, relation="disjoint")

# which touch (not overlap or within)?
relate(p, relation="touches")
# which overlap (not merely touch, and not within)?
relate(p, relation="overlaps")
# which are within (not merely overlap)?
relate(p, relation="within")

# do they touch or overlap or are within?
relate(p, relation="intersects")

all(relate(p, relation="intersects") ==
     (relate(p, relation="overlaps") |
      relate(p, relation="touches") |
      relate(p, relation="within")))

#for polygons, "coveredby" is "within"
relate(p, relation="coveredby")

# polygons, lines, and points
```

```

pp <- rbind(p1, p2)
L1 <- vect("LINESTRING(1 11, 4 6, 10 6)")
L2 <- vect("LINESTRING(8 14, 12 10)")
L3 <- vect("LINESTRING(1 8, 12 14)")
lns <- rbind(L1, L2, L3)
pts <- vect(cbind(c(7,10,10), c(3,5,6)))

plot(pp, col=rainbow(2, alpha=.5))
text(pp, paste0("POL", 1:2), halo=TRUE)
lines(pp, lwd=2)
lines(lns, col=rainbow(3), lwd=4)
text(lns, paste0("L", 1:3), halo=TRUE)
points(pts, cex=1.5)
text(pts, paste0("PT", 1:3), halo=TRUE, pos=4)

relate(lns, relation="crosses")
relate(lns, pp, relation="crosses")
relate(lns, pp, relation="touches")
relate(lns, pp, relation="intersects")

relate(lns, pp, relation="within")
# polygons can contain lines or points, not the other way around
relate(lns, pp, relation="contains")
relate(pp, lns, relation="contains")
# points and lines can be covered by polygons
relate(lns, pp, relation="coveredby")

relate(pts, pp, "within")
relate(pts, pp, "touches")
relate(pts, lns, "touches")

```

 rep

Replicate layers

Description

Replicate layers in a SpatRaster

Usage

```
## S4 method for signature 'SpatRaster'
rep(x, ...)
```

Arguments

x	SpatRaster
...	arguments as in rep

Value

SpatRaster

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))
x <- rep(s, 2)
nlyr(x)
names(x)
x
```

replace

*Replace values of a SpatRaster***Description**

Replace values of a SpatRaster. These are convenience functions for smaller objects only. For larger rasters see [link{classify}](#)

Value

SpatRaster

See Also

[link{classify}](#), [values](#), [replace](#)

Examples

```
r <- rast(ncols=5, nrows=5, xmin=0, xmax=5, ymin=0, ymax=5)
r[] <- 1:25
r[,1] <- 5
r[,2] <- 10
r[r>10] <- NA
```

resample

*Transfer values of a SpatRaster to another one with a different geometry***Description**

resample transfers values between SpatRaster objects that do not align (have a different origin and/or resolution). See [project](#) to change the coordinate reference system (crs).

If the origin and crs are the same, you should consider using these other functions instead: [aggregate](#), [disagg](#), [extend](#) or [crop](#).

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
resample(x, y, method, filename="", ...)
```

Arguments

x	SpatRaster to be resampled
y	SpatRaster with the geometry that x should be resampled to
method	character. Method used for estimating the new cell values. One of: near: nearest neighbor. This method is fast, and it can be the preferred method if the cell values represent classes. It is not a good choice for continuous values. This is used by default if the first layer of x is categorical. bilinear: bilinear interpolation. This is the default if the first layer of x is numeric (not categorical). cubic: cubic interpolation. cubicspline: cubic spline interpolation. lanczos: Lanczos windowed sinc resampling. sum: the weighted sum of all non-NA contributing grid cells. min, q1, med, q3, max, average, mode, rms: the minimum, first quartile, median, third quartile, maximum, mean, mode, or root-mean-square value of all non-NA contributing grid cells.
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[aggregate](#), [disagg](#), [crop](#), [project](#),

Examples

```
r <- rast(nrows=3, ncols=3, xmin=0, xmax=10, ymin=0, ymax=10)
values(r) <- 1:ncell(r)
s <- rast(nrows=25, ncols=30, xmin=1, xmax=11, ymin=-1, ymax=11)
x <- resample(r, s, method="bilinear")

opar <- par(no.readonly =TRUE)
par(mfrow=c(1,2))
plot(r)
plot(x)
par(opar)
```

rescale	<i>rescale</i>
---------	----------------

Description

Rescale a `SpatVector` or `SpatRaster`. This may be useful to make small [inset](#) maps or for georeferencing.

Usage

```
## S4 method for signature 'SpatRaster'  
rescale(x, fx=0.5, fy=fx, x0, y0)
```

```
## S4 method for signature 'SpatVector'  
rescale(x, fx=0.5, fy=fx, x0, y0)
```

Arguments

<code>x</code>	<code>SpatVector</code> or <code>SpatRaster</code>
<code>fx</code>	numeric > 0. The horizontal scaling factor
<code>fy</code>	numeric > 0. The vertical scaling factor
<code>x0</code>	numeric. x-coordinate of the center of rescaling. If missing, the center of the extent of <code>x</code> is used
<code>y0</code>	numeric. y-coordinate of the center of rescaling. If missing, the center of the extent of <code>x</code> is used

Value

Same as `x`

See Also

[t](#), [shift](#), [flip](#), [rotate](#), [inset](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")  
v <- vect(f)  
w <- rescale(v, 0.2)  
plot(v)  
lines(w, col="red")
```

 RGB

Layers representing colors

Description

With RGB you can get or set the layers to be used as Red, Green and Blue when plotting a SpatRaster. Currently, a benefit of this is that `plot` will send the object to `plotRGB`

With `colorize` you can convert a three-layer RGB SpatRaster into other color spaces. You can also convert it into a single-layer SpatRaster with a color-table.

Usage

```
## S4 method for signature 'SpatRaster'
RGB(x)

## S4 replacement method for signature 'SpatRaster'
RGB(x)<-value

## S4 method for signature 'SpatRaster'
colorize(x, to="hsv", alpha=FALSE, stretch=NULL,
grays=FALSE, NAzero=FALSE, filename="", overwrite=FALSE, ...)

## S4 method for signature 'SpatRaster'
has.RGB(x)
```

Arguments

<code>x</code>	SpatRaster
<code>value</code>	vector of three (or four) positive integers indicating the layers that are red, green and blue (and optionally a fourth transparency layer). Or NULL to remove the RGB settings
<code>to</code>	character. The color space to transform the values to. If <code>x</code> has RGB set, you can transform these to "hsv", "hsi" and "hsl", or use "col" to create a single layer with a color table. You can also use "rgb" to backtransform to RGB
<code>alpha</code>	logical. Should an alpha (transparency) channel be included? Only used if <code>x</code> has a color-table and <code>to="rgb"</code>
<code>stretch</code>	character. Option to stretch the values to increase contrast: "lin" (linear) or "hist" (histogram). Only used for transforming RGB to col
<code>grays</code>	logical. If TRUE, a gray-scale color-table is created. Only used for transforming RGB to col
<code>NAzero</code>	logical. If TRUE, NAs are treated as zeros such that a color can be returned if at least one of the three channels has a value. Only used for transforming RGB to ("col")
<code>filename</code>	character. Output filename
<code>overwrite</code>	logical. If TRUE, filename is overwritten
<code>...</code>	additional arguments for writing files as in <code>writeRaster</code>

Examples

```

r <- rast(system.file("ex/logo.tif", package="terra"))
plot(r)
has.RGB(r)
RGB(r) <- NULL
has.RGB(r)
plot(r)
RGB(r) <- c(3,1,2)
plot(r)

RGB(r) <- 1:3
x <- colorize(r, "col")
y <- colorize(r, "hsv")
z <- colorize(y, "rgb")

```

rotate

Rotate a SpatRaster along longitude

Description

Rotate a `SpatRaster` that has longitude coordinates from 0 to 360, to standard coordinates between -180 and 180 degrees (or vice-versa). Longitude between 0 and 360 is frequently used in global climate models.

Usage

```

## S4 method for signature 'SpatRaster'
rotate(x, left=TRUE, filename="", ...)

```

Arguments

x	SpatRaster or SpatVector
left	logical. If TRUE, rotate to the left, else to the right
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[shift](#) and [spin](#)

Examples

```
x <- rast(nrows=9, ncols=18, nl=3, xmin=0, xmax=360)
v <- rep(as.vector(t(matrix(1:ncell(x), nrow=9, ncol=18))), 3)
values(x) <- v
z <- rotate(x)
```

sapp	<i>Apply a terra function that takes only a single layer and returns a SpatRaster to all layers of a SpatRaster</i>
------	---

Description

Apply to all layers of a SpatRaster a function that only takes a single layer SpatRaster and returns a SpatRaster (these are rare). In most cases you can also use `lapply` or `sapply` for this.

Or apply the same method to each sub-dataset (SpatRaster) in a SpatRasterDataset

Usage

```
## S4 method for signature 'SpatRaster'
sapp(x, fun, ..., filename="", overwrite=FALSE, wopt=list())

## S4 method for signature 'SpatRasterDataset'
sapp(x, fun, ..., filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster or SpatRasterDataset
fun	if x is a SpatRaster: a function that takes a SpatRaster argument and can be applied to each layer of x (e.g. <code>terrain</code>). if x is a SpatRasterDataset: a function that is applied to all layers of the SpatRasters in x (e.g. <code>mean</code>)
...	additional arguments to be passed to fun
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in <code>writeRaster</code>

Value

SpatRaster

See Also

[lapp](#), [app](#), [tapp](#), [lapply](#)

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra")) + 1
x <- sapp(s, terrain)

sd <- sds(s*2, s/2)
y <- sapp(sd, mean)
```

sbar *scale bar*

Description

Add a scale bar to a map

Usage

```
sbar(d, xy=NULL, type="line", divs=2, below="",
     lonlat=NULL, label, adj=c(0.5, -1), lwd=2, xpd=TRUE, ...)
```

Arguments

d	numeric. Distance covered by the scale bar. For the scale bar, it should be in the units of the coordinates of the plot (map), and in km for angular (longitude/latitude) data; see argument lonlat. It can also be missing
xy	numeric. x and y coordinate to place the scale bar. It can also be one of following character values: "bottomleft", "bottom", "bottomright", "topleft", "top", "topright", "left", "right", or NULL
type	for sbar: "line" or "bar"
divs	number of divisions for a bar: 2 or 4
below	character. Text to go below the scale bar (e.g., "kilometers")
lonlat	logical or NULL. If logical, TRUE indicates if the plot is using longitude/latitude coordinates. If NULL this is guessed from the plot's coordinates
label	vector of three numbers to label the scale bar (beginning, midpoint, end)
adj	adjustment for text placement
lwd	line width for the "line" type of the scale bar
xpd	logical. If TRUE, the scale bar can be outside the plotting area
...	graphical arguments to be passed to other methods

Value

none

See Also

[north](#), [plot](#), [inset](#)

Examples

```
f <- system.file("ex/meuse.tif", package="terra")
r <- rast(f)
plot(r)
sbar()
sbar(1000, xy=c(178500, 333500), type="bar", divs=4, cex=.8)
sbar(1000, xy="bottomright", divs=4, cex=.8)
north(d=250, c(178550, 332500))
```

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
plot(r, type="interval")
sbar(20, c(6.2, 50.1), type="bar", cex=.8, divs=4)
sbar(15, c(6.3, 50), type="bar", below="km", label=c(0,7.5,15), cex=.8)
sbar(15, c(6.65, 49.8), cex=.8, label=c(0,"km",15))
north(type=2)
sbar(15, c(6.65, 49.7), cex=.8, label="15 kilometer", lwd=5)
sbar(15, c(6.65, 49.6), divs=4, cex=.8, below="km")
```

scale

Scale values

Description

Center and/or scale raster data. For details see [scale](#)

Usage

```
## S4 method for signature 'SpatRaster'
scale(x, center=TRUE, scale=TRUE)
```

Arguments

x	SpatRaster
center	logical or numeric. If TRUE, centering is done by subtracting the layer means (omitting NAs), and if FALSE, no centering is done. If center is a numeric vector (recycled to $n_{\text{lyr}}(x)$), then each layer of x has the corresponding value from center subtracted from it.
scale	logical or numeric. If TRUE, scaling is done by dividing the (centered) layers of x by their standard deviations if center is TRUE, and the root mean square otherwise. If scale is FALSE, no scaling is done. If scale is a numeric vector (recycled to $n_{\text{lyr}}(x)$), each layer of x is divided by the corresponding value. Scaling is done after centering.

Value

SpatRaster

See Also[scale](#)**Examples**

```

r <- rast(system.file("ex/logo.tif", package="terra"))
s <- scale(r)

## the equivalent, computed in steps
m <- global(r, "mean")
rr <- r - m[,1]
rms <- global(rr, "rms")
ss <- rr / rms[,1]

```

scatterplot

*Scatterplot of two SpatRaster layers***Description**

Scatterplot of the values of two SpatRaster layers

Usage

```

## S4 method for signature 'SpatRaster,SpatRaster'
plot(x, y, maxcell=100000, warn=TRUE, nc, nr,
     maxnl=16, gridded=FALSE, ncol=25, nrow=25, ...)

```

Arguments

x	SpatRaster
y	SpatRaster
maxcell	positive integer. Maximum number of cells to use for the plot
nc	positive integer. Optional. The number of columns to divide the plotting device in (when plotting multiple layers)
nr	positive integer. Optional. The number of rows to divide the plotting device in (when plotting multiple layers)
maxnl	positive integer. Maximum number of layers to plot (for multi-layer objects)
gridded	logical. If TRUE the scatterplot is gridded (counts by cells)
warn	boolean. Show a warning if a sample of the pixels is used (for scatterplot only)
ncol	positive integer. Number of columns for gridding
nrow	positive integer. Number of rows for gridding
...	additional graphical arguments

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))
plot(s[[1]], s[[2]])
plot(s, sqrt(s[[3:1]]))
```

sds

Create a SpatRasterDataset

Description

Methods to create a `SpatRasterDataset`. This is an object to hold "sub-datasets", each a `SpatRaster` that in most cases will have multiple layers.

See [describe](#) for getting information about the sub-datasets present in a file.

Usage

```
## S4 method for signature 'missing'
sds(x)

## S4 method for signature 'character'
sds(x, ids=0)

## S4 method for signature 'SpatRaster'
sds(x, ...)

## S4 method for signature 'list'
sds(x)

## S4 method for signature 'array'
sds(x, crs="", extent=NULL)
```

Arguments

x	character (filename), or <code>SpatRaster</code> , or list of <code>SpatRaster</code> objects, or missing. If multiple filenames are provided, it is attempted to make <code>SpatRasters</code> from these, and combine them into a <code>SpatRasterDataset</code>
ids	optional. vector of integer subdataset ids. Ignored if the first value is not a positive integer
crs	character. Description of the Coordinate Reference System (map projection) in PROJ.4, WKT or authority:code notation. If this argument is missing, and the x coordinates are within -360 .. 360 and the y coordinates are within -90 .. 90, longitude/latitude is assigned
extent	SpatExtent
...	additional <code>SpatRaster</code> objects

Value

SpatRasterDataset

See Also[describe](#)**Examples**

```
s <- rast(system.file("ex/logo.tif", package="terra"))
x <- sds(s, s/2)
names(x) <- c("first", "second")
x
length(x)

# extract the second SpatRaster
x[2]

a <- array(1:9, c(3,3,3,3))
sds(a)
```

segregate

*segregate***Description**

Create a SpatRaster with a layer for each class (value, or subset of the values) in the input SpatRaster. For example, if the input has vegetation types, this function will create a layer (presence/absence; dummy variable) for each of these classes.

This is called "one-hot encoding" or "dummy encoding" (for a dummy encoding scheme you can remove (any) one of the output layers as it is redundant).

Usage

```
## S4 method for signature 'SpatRaster'
segregate(x, classes=NULL, keep=FALSE, other=0, round=FALSE, digits=0, filename="", ...)
```

Arguments

x	SpatRaster
classes	numeric. The values (classes) for which layers should be made. If NULL all classes are used
keep	logical. If TRUE, cells that are of the class represented by a layer get that value, rather than a value of 1
other	numeric. Value to assign to cells that are not of the class represented by a layer
round	logical. Should the values be rounded first?

digits integer. Number of digits to round the values to
filename character. Output filename
... additional arguments for writing files as in [writeRaster](#)

Value

SpatRaster

Examples

```

r <- rast(nrows=5, ncols=5)
values(r) <- rep(c(1:4, NA), each=5)
b <- segregate(r)
bb <- segregate(r, keep=TRUE, other=NA)

```

sel *Spatial selection*

Description

Geometrically subset SpatRaster or SpatVector (to be done) by drawing on a plot (map).

Usage

```

## S4 method for signature 'SpatRaster'
sel(x, ...)

## S4 method for signature 'SpatVector'
sel(x, use="rec", draw=TRUE, col="cyan", ...)

```

Arguments

x SpatRaster or SpatVector
use character indicating what to draw. One of "rec" (rectangle) or "pol" (polygon)
draw logical. If TRUE the selection is drawn on the map
col color to be used for drawing if draw=TRUE
... additional graphics arguments for drawing

Value

SpatRaster or SpatVector

See Also

[crop](#) and [intersect](#) to make an intersection and [click](#) and [text](#) to see cell values or geometry attributes

Examples

```
## Not run:
# select a subset of a SpatRaster
r <- rast(nrows=10, ncols=10)
values(r) <- 1:ncell(r)
plot(r)
s <- sel(r) # now click on the map twice

# plot the selection on a new canvas:
x11()
plot(s)

# vector
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
plot(v)
x <- sel(v) # now click on the map twice
x

## End(Not run)
```

selectHighest	<i>select cells with high or low values</i>
---------------	---

Description

Identify *n* cells that have the highest of lowest values in the first layer of a SpatRaster.

Usage

```
## S4 method for signature 'SpatRaster'
selectHighest(x, n, low=FALSE)
```

Arguments

<i>x</i>	SpatRaster. Only the first layer is processed
<i>n</i>	The number of cells to select
<i>low</i>	logical. If TRUE, the lowest values are selected instead of the highest values

Value

SpatRaster

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
x <- selectHighest(r, 1000)
y <- selectHighest(r, 1000, TRUE)

m <- merge(y-1, x)
levels(m) <- data.frame(id=0:1, elevation=c("low", "high"))
plot(m)
```

selectRange	<i>Select the values of a range of layers, as specified by cell values in another SpatRaster</i>
-------------	--

Description

Use a single layer SpatRaster to select cell values from different layers in a multi-layer SpatRaster. The values of the SpatRaster to select layers (y) should be whole numbers between 1 and nlyr(x) (values outside this range are ignored).

See [rapp](#) for applying a function to a range of variable size.

See [extract](#) for extraction of values by cell, point, or otherwise.

Usage

```
## S4 method for signature 'SpatRaster'
selectRange(x, y, z=1, repint=0, filename="", ...)
```

Arguments

x	SpatRaster
y	SpatRaster. Cell values must be positive integers. They indicate the first layer to select for each cell
z	positive integer. The number of layers to select
repint	integer > 1 and < nlyr(x) allowing for repeated selection at a fixed interval. For example, if x has 36 layers, and the value of a cell in y=2 and repint = 12, the values for layers 2, 14 and 26 are returned
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[rapp](#), [tapp](#), [extract](#)

Examples

```

r <- rast(ncols=10, nrows=10)
values(r) <- 1
s <- c(r, r+2, r+5)
s <- c(s, s)
set.seed(1)
values(r) <- sample(3, ncell(r), replace=TRUE)
x <- selectRange(s, r)

x <- selectRange(s, r, 3)

```

serialize

serialize and saveRDS for SpatRaster and SpatVector

Description

serialize and saveRDS for SpatRaster and SpatVector. Note that these objects will first be "packed" with [wrap](#), and after unserialize/readRDS they need to be unpacked with [rast](#) or [vect](#).

Use of these functions is not recommended. Especially for SpatRaster it is generally much more efficient to use [writeRaster](#) and [write](#), e.g., a GTiff file.

SpatRaster objects must have all values in memory (that is, the cell values are not in files) to be serialized. These functions use [set.values](#) to load values into memory if needed and if deemed possible given the amount of RAM available.

Usage

```

## S4 method for signature 'SpatRaster'
saveRDS(object, file="", ascii = FALSE, version = NULL, compress=TRUE, refhook = NULL)

## S4 method for signature 'SpatVector'
saveRDS(object, file="", ascii = FALSE, version = NULL, compress=TRUE, refhook = NULL)

## S4 method for signature 'SpatRaster'
serialize(object, connection, ascii = FALSE, xdr = TRUE, version = NULL, refhook = NULL)

## S4 method for signature 'SpatVector'
serialize(object, connection, ascii = FALSE, xdr = TRUE, version = NULL, refhook = NULL)

```

Arguments

object	SpatVector or SpatRaster
file	file name to save object to
connection	see serialize
ascii	see serialize or saveRDS
version	see serialize or saveRDS

compress see [serialize](#) or [saveRDS](#)
 refhook see [serialize](#) or [saveRDS](#)
 xdr see [serialize](#) or [saveRDS](#)

Value

Packed* object

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
p <- serialize(v, NULL)
head(p)
x <- unserialize(p)
x
vect(x)
```

setValues *Set the values of raster cells or of geometry attributes*

Description

Set cell values of a `SpatRaster` or the attributes of a `SpatVector`. For large `SpatRaster` objects use [init](#) instead to set values.

Usage

```
## S4 replacement method for signature 'SpatRaster,ANY'
values(x)<-value

## S4 method for signature 'SpatRaster,ANY'
setValues(x, values, keeptime=TRUE, keepunits=TRUE, props=FALSE)

## S4 replacement method for signature 'SpatVector,ANY'
values(x)<-value
```

Arguments

`x` `SpatRaster` or `SpatVector`
`value` For `SpatRaster`: matrix or numeric, the length must match the total number of cells (`ncell(x) * nlyr(x)`), or be a single value. For `SpatVector`: data.frame, matrix, vector, or `NULL`
`values` Same as for `value`
`keeptime` logical. If `TRUE` the time stamps are kept
`keepunits` logical. If `FALSE` the units are discarded
`props` logical. If `TRUE` the properties (categories and color-table) are kept

Value

The same object type as `x`

See Also

[values](#), [init](#)

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
x <- setValues(r, 1:ncell(r))
x
values(x) <- runif(ncell(x))
x
head(x)

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
values(v) <- data.frame(ID=1:12, name=letters[1:12])
head(v)
```

shade

Hill shading

Description

Compute hill shade from slope and aspect layers (both in radians). Slope and aspect can be computed with function [terrain](#).

A hill shade layer is often used as a backdrop on top of which another, semi-transparent, layer is drawn.

Usage

```
shade(slope, aspect, angle=45, direction=0, normalize=FALSE,
      filename="", overwrite=FALSE, ...)
```

Arguments

<code>slope</code>	SpatRaster with slope values (in radians)
<code>aspect</code>	SpatRaster with aspect values (in radians)
<code>angle</code>	The the elevation angle of the light source (sun), in degrees
<code>direction</code>	The direction (azimuth) angle of the light source (sun), in degrees
<code>normalize</code>	Logical. If TRUE, values below zero are set to zero and the results are multiplied with 255
<code>filename</code>	character. Output filename
<code>overwrite</code>	logical. If TRUE, filename is overwritten
<code>...</code>	additional arguments for writing files as in writeRaster

References

Horn, B.K.P., 1981. Hill shading and the reflectance map. Proceedings of the IEEE 69(1):14-47

See Also

[terrain](#)

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
alt <- disagg(r, 10, method="bilinear")
slope <- terrain(alt, "slope", unit="radians")
aspect <- terrain(alt, "aspect", unit="radians")
hill <- shade(slope, aspect, 40, 270)
plot(hill, col=grey(0:100/100), legend=FALSE, mar=c(2,2,1,4))
plot(alt, col=rainbow(25, alpha=0.35), add=TRUE)
```

sharedPaths

Shared paths

Description

Get shared paths of line or polygon geometries. This can for geometries in a single SpatVector, or between two SpatVectors

Usage

```
## S4 method for signature 'SpatVector'
sharedPaths(x, y=NULL)
```

Arguments

x	SpatVector of lines or polygons
y	missing or SpatVector of lines or polygons

Value

SpatVector

See Also

[gaps](#), [topology](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
plot(v, col="light gray")
text(v, halo=TRUE)

x <- sharedPaths(v)
lines(x, col="red", lwd=2)
text(x, col="blue", halo=TRUE, cex=0.8)
head(x)

z <- sharedPaths(v[3,], v[12,])
```

 shift

Shift

Description

Shift a `SpatRaster`, `SpatVector` or `SpatExtent` to another location.

Usage

```
## S4 method for signature 'SpatRaster'
shift(x, dx=0, dy=0, filename="", ...)

## S4 method for signature 'SpatVector'
shift(x, dx=0, dy=0)

## S4 method for signature 'SpatExtent'
shift(x, dx=0, dy=0)
```

Arguments

<code>x</code>	<code>SpatRaster</code> , <code>SpatVector</code> or <code>SpatExtent</code>
<code>dx</code>	numeric. The shift in horizontal direction
<code>dy</code>	numeric. The shift in vertical direction
<code>filename</code>	character. Output filename
<code>...</code>	additional arguments for writing files as in writeRaster

Value

Same as `x`

See Also

[flip](#), [rotate](#)

Examples

```
r <- rast(xmin=0, xmax=1, ymin=0, ymax=1)
r <- shift(r, dx=1, dy=-1)

e <- ext(r)
shift(e, 5, 5)
```

simplifyGeom

simplifyGeom geometries

Description

Reduce the number of nodes used to represent geometries.

Usage

```
## S4 method for signature 'SpatVector'
simplifyGeom(x, tolerance=0.1, preserveTopology=TRUE, makeValid=TRUE)
```

Arguments

x	SpatVector of lines or polygons
tolerance	numeric. The minimum distance between nodes in units of the crs (i.e. degrees for long/lat)
preserveTopology	logical. If TRUE the topology of output geometries is preserved
makeValid	logical. If TRUE, link{makeValid} is run after simplification to assure that the output polygons are valid

Value

SpatVector

See Also

[sharedPaths](#), [gaps](#), [link{is.valid\(\)}](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
w <- simplifyGeom(v, .02, makeValid=FALSE)
e <- erase(w)
g <- gaps(e)
plot(e, lwd=5, border="light gray")
polys(g, col="red", border="red")
```

sort	<i>Sort a SpatRaster</i>
------	--------------------------

Description

Sort the cell values of a SpatRaster across layers

Usage

```
## S4 method for signature 'SpatRaster'  
sort(x, decreasing=FALSE, filename="", ...)
```

Arguments

x	SpatRaster
decreasing	logical
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

Examples

```
f <- system.file("ex/elev.tif", package="terra")  
r <- rast(f)  
r <- c(r, r/2, r*2)  
sort(r)
```

sources	<i>Data sources of a SpatRaster</i>
---------	-------------------------------------

Description

Get the data sources of a SpatRaster or SpatVector or related object. Sources are either files (or similar resources) or "", meaning that they are in memory. You can use `hasValues` to check if in-memory layers actually have cell values.

Usage

```
## S4 method for signature 'SpatRaster'
sources(x, nlyr=FALSE, bands=FALSE)

## S4 method for signature 'SpatVector'
sources(x)

## S4 method for signature 'SpatRaster'
hasValues(x)

## S4 method for signature 'SpatRaster'
inMemory(x, bylayer=FALSE)
```

Arguments

x	SpatRaster, SpatRasterCollection, SpatVector or SpatVectorProxy
nlyr	logical. If TRUE for each source, the number of layers is returned
bands	logical. If TRUE for each source, the "bands" used, that is, the layer number in the source file, are returned
bylayer	logical. If TRUE a value is returned for each layer instead of for each source

Value

A vector of filenames, or "" when there is no filename, if nlyr and bands are both FALSE. Otherwise a data.frame

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
s <- rast(r)
values(s) <- 1:ncell(s)
rs <- c(r,r,s,r)
sources(rs)
hasValues(r)
x <- rast()
hasValues(x)
```

SpatExtent-class

Class "SpatExtent"

Description

Objects of class SpatExtent are used to define the spatial extent (extremes) of objects of the SpatRaster class.

Objects from the Class

You can use the `ext` function to create `SpatExtent` objects, or to extract them from `SpatRaster` objects.

Slots

`ptr`: pointer to the C++ class

Methods

`show` display values of a `SpatExtent` object

Examples

```
e <- ext(-180, 180, -90, 90)
e
```

Spatial interpolation *Interpolate*

Description

Make a `SpatRaster` with interpolated values using a fitted model object of classes such as "gstat" (gstat package) or "Krige" (fields package), or any other model that has location (e.g., "x" and "y", or "longitude" and "latitude") as predictors (independent variables). If x and y are the only predictors, it is most efficient if you provide an empty (no associated data in memory or on file) `SpatRaster` for which you want predictions. If there are more spatial predictor variables provide these as a `SpatRaster` in the first argument of the function. If you do not have x and y locations as implicit predictors in your model you should use `predict` instead.

Usage

```
## S4 method for signature 'SpatRaster'
interpolate(object, model, fun=predict, ..., xyNames=c("x", "y"),
            factors=NULL, const=NULL, index = NULL, na.rm=FALSE,
            filename="", overwrite=FALSE, wopt=list())
```

Arguments

<code>object</code>	<code>SpatRaster</code>
<code>model</code>	model object
<code>fun</code>	function. Default value is "predict", but can be replaced with e.g. "predict.se" (depending on the class of <code>model</code>), or a custom function (see examples)
<code>...</code>	additional arguments passed to <code>fun</code>
<code>xyNames</code>	character. variable names that the model uses for the spatial coordinates. E.g., <code>c("longitude", "latitude")</code>

factors	list with levels for factor variables. The list elements should be named with names that correspond to names in object such that they can be matched. This argument may be omitted for some models from which the levels can be extracted from the model object
const	data.frame. Can be used to add a constant for which there is no SpatRaster for model predictions. This is particularly useful if the constant is a character-like factor value
index	positive integer or NULL. Allows for selecting of the variable returned if the model returns multiple variables
na.rm	logical. If TRUE, cells with NA values in the predictors are removed from the computation. This option prevents errors with models that cannot handle NA values. In most other cases this will not affect the output. An exception is when predicting with a model that returns predicted values even if some (or all!) variables are NA
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
wopt	list with named options for writing files as in writeRaster

Value

SpatRaster

See Also[predict](#)**Examples**

```

r <- rast(system.file("ex/elev.tif", package="terra"))
ra <- aggregate(r, 10)
xy <- data.frame(xyFromCell(ra, 1:ncell(ra)))
v <- values(ra)
i <- !is.na(v)
xy <- xy[i,]
v <- v[i]

## Not run:
library(fields)
tps <- Tps(xy, v)
p <- rast(r)

# use model to predict values at all locations
p <- interpolate(p, tps)
p <- mask(p, r)
plot(p)

### change "fun" from predict to fields::predictSE to get the TPS standard error
## need to use "rast(p)" to remove the values

```

```

se <- interpolate(rast(p), tps, fun=predictSE)
se <- mask(se, r)
plot(se)

### another predictor variable, "e"
e <- (init(r, "x") * init(r, "y")) / 100000000
names(e) <- "e"

z <- as.matrix(extract(e, xy)[-1])

## add as another independent variable
xyz <- cbind(xy, z)
tps2 <- Tps(xyz, v)
p2 <- interpolate(e, tps2, xyOnly=FALSE)

## as a linear coveriate
tps3 <- Tps(xy, v, Z=z)

## Z is a separate argument in Krig.predict, so we need a new function
## Internally (in interpolate) a matrix is formed of x, y, and elev (Z)

pfun <- function(model, x, ...) {
  predict(model, x[,1:2], Z=x[,3], ...)
}
p3 <- interpolate(e, tps3, fun=pfun)

#### gstat examples
library(gstat)
library(sp)
data(meuse)

### inverse distance weighted (IDW)
r <- rast(system.file("ex/meuse.tif", package="terra"))
mg <- gstat(id = "zinc", formula = zinc~1, locations = ~x+y, data=meuse,
           nmax=7, set=list(idp = .5))
z <- interpolate(r, mg, debug.level=0, index=1)
z <- mask(z, r)

## with a model built with an `sf` object you need to provide custom function

library(sf)
sfmeuse <- st_as_sf(meuse, coords = c("x", "y"), crs=crs(r))
mgssf <- gstat(id = "zinc", formula = zinc~1, data=sfmeuse, nmax=7, set=list(idp = .5))

interpolate_gstat <- function(model, x, crs, ...) {
  v <- st_as_sf(x, coords=c("x", "y"), crs=crs)
  p <- predict(model, v, ...)
  as.data.frame(p)[,1:2]
}

zsf <- interpolate(r, mgssf, debug.level=0, fun=interpolate_gstat, crs=crs(r), index=1)
zsf <- mask(zsf, r)

```

```

### kriging

### ordinary kriging
v <- variogram(log(zinc)~1, ~x+y, data=meuse)
mv <- fit.variogram(v, vgm(1, "Sph", 300, 1))
gOK <- gstat(NULL, "log.zinc", log(zinc)~1, meuse, locations=~x+y, model=mv)
OK <- interpolate(r, gOK, debug.level=0)

## universal kriging
vu <- variogram(log(zinc)~elev, ~x+y, data=meuse)
mu <- fit.variogram(vu, vgm(1, "Sph", 300, 1))
gUK <- gstat(NULL, "log.zinc", log(zinc)~elev, meuse, locations=~x+y, model=mu)
names(r) <- "elev"
UK <- interpolate(r, gUK, debug.level=0)

## co-kriging
gCoK <- gstat(NULL, 'log.zinc', log(zinc)~1, meuse, locations=~x+y)
gCoK <- gstat(gCoK, 'elev', elev~1, meuse, locations=~x+y)
gCoK <- gstat(gCoK, 'cadmium', cadmium~1, meuse, locations=~x+y)
gCoK <- gstat(gCoK, 'copper', copper~1, meuse, locations=~x+y)
coV <- variogram(gCoK)
plot(coV, type='b', main='Co-variogram')
coV.fit <- fit.lmc(coV, gCoK, vgm(model='Sph', range=1000))
coV.fit
plot(coV, coV.fit, main='Fitted Co-variogram')
coK <- interpolate(r, coV.fit, debug.level=0)
plot(coK)

## End(Not run)

```

SpatRaster-class

SpatRaster class

Description

A `SpatRaster` represents a rectangular part of the world that is sub-divided into rectangular cells of equal area (in terms of the units of the coordinate reference system). For each cell can have multiple values ("layers").

An object of the `SpatRaster` class can point to one or more files on disk that hold the cell values, and/or it can hold these values in memory. These objects can be created with the `rast` method.

The underlying C++ class "Rcpp_SpatRaster" is not intended for end-users. It is for internal use within this package only.

Examples

```
rast()
```

spatSample	<i>Take a regular sample</i>
------------	------------------------------

Description

Take a spatial sample from a SpatRaster, SpatVector or SpatExtent. Sampling a SpatVector or SpatExtent always returns a SpatVector of points.

With a SpatRaster, you can get cell values, cell numbers (`cells=TRUE`), coordinates (`xy=TRUE`) or (when `type="regular"` and `as.raster=TRUE`) get a new SpatRaster with the same extent, but fewer cells.

In order to assure regularity when requesting a regular sample, the number of cells or points returned may not be exactly the same as the size requested.

Usage

```
## S4 method for signature 'SpatRaster'
spatSample(x, size, method="random", replace=FALSE, na.rm=FALSE,
as.raster=FALSE, as.df=TRUE, as.points=FALSE, values=TRUE,
cells=FALSE, xy=FALSE, ext=NULL, warn=TRUE, weights=NULL)
```

```
## S4 method for signature 'SpatVector'
spatSample(x, size, method="random", strata=NULL, chess="")
```

```
## S4 method for signature 'SpatExtent'
spatSample(x, size, method="random", lonlat, as.points=FALSE)
```

Arguments

<code>x</code>	SpatRaster, SpatVector or SpatExtent
<code>size</code>	numeric. The sample size. If <code>x</code> is a SpatVector, you can also provide a vector of the same length as <code>x</code> in which case sampling is done separately for each geometry. If <code>x</code> is a SpatRaster, and you are using <code>method="regular"</code> you can specify the size as two numbers (number of rows and columns)
<code>method</code>	character. Should be "regular" or "random", If <code>x</code> is a SpatRaster, it can also be "stratified" (each value in <code>x</code> is a stratum) or "weights" (each value in <code>x</code> is a probability weight)
<code>replace</code>	logical. If TRUE, sampling is with replacement (if <code>method="random"</code>)
<code>na.rm</code>	logical. If TRUE, codeNAs are removed. Only used with random sampling of cell values. That is with <code>method="random"</code> , <code>as.raster=FALSE</code> , <code>cells=FALSE</code>
<code>as.raster</code>	logical. If TRUE, a SpatRaster is returned
<code>as.df</code>	logical. If TRUE, a data.frame is returned instead of a matrix
<code>as.points</code>	logical. If TRUE, a SpatVector of points is returned
<code>values</code>	logical. If TRUE cell values are returned

cells	logical. If TRUE, cell numbers are returned. If method="stratified" this is always set to TRUE if xy=FALSE
xy	logical. If TRUE, cell coordinates are returned
ext	SpatExtent or NULL to restrict sampling to a subset of the area of x
warn	logical. Give a warning if the sample size returned is smaller than requested
weights	SpatRaster. Used to provide weights when method="stratified"
strata	if not NULL, stratified random sampling is done, taking size samples from each stratum. If x has polygon geometry, strata must be a field name (or index) in x. If x has point geometry, strata can be a SpatVector of polygons or a SpatRaster
chess	character. One of "", "white", or "black". For stratified sampling if strata is a SpatRaster. If not "", samples are only taken from alternate cells, organized like the "white" or "black" fields on a chessboard
lonlat	logical. If TRUE, sampling of a SpatExtent is weighted by cos(latitude). For SpatRaster and SpatVector this done based on the crs , but it is ignored if as.raster=TRUE

Value

numeric matrix, data.frame, SpatRaster or SpatVector

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
s <- spatSample(r, 10, as.raster=TRUE)
spatSample(r, 10)
spatSample(r, 10, "random")

## if you require cell numbers and/or coordinates
size <- 6
# random cells
cells <- spatSample(r, 6, "random", cells=TRUE, values=FALSE)
cells <- as.vector(cells)

v <- r[cells]
xy <- xyFromCell(r, cells)
cbind(xy, v)

# regular
cells <- spatSample(r, 6, "regular", cells=TRUE, values=FALSE)
cells <- as.vector(cells)
v <- r[cells]
xy <- xyFromCell(r, cells)
cbind(xy, v)

# stratified
rr <- rast(ncol=10, nrow=10, names="stratum")
set.seed(1)
values(rr) <- round(runif(ncell(rr), 1, 3))
```



```

spatSample(rr, 2, "stratified", xy=TRUE)

s <- spatSample(rr, 5, "stratified", as.points=TRUE)
plot(rr, plg=list(title="raster"))
plot(s, 1, add=TRUE, plg=list(x=185, y=1, title="points"))

## SpatExtent
e <- ext(r)
spatSample(e, 10, "random", lonlat=TRUE)

## SpatVector
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
#sample geometries
i <- sample(nrow(v), 5)
vv <- v[i,]

```

SpatVector-class	<i>Class "SpatVector"</i>
------------------	---------------------------

Description

Objects of class SpatVector.

Objects from the Class

You can use the [vect](#) method to create SpatVector objects.

Slots

ptr: pointer to the C++ class

Methods

show display values of a SpatVector

spin	<i>spin a SpatVector</i>
------	--------------------------

Description

Spin (rotate) the geometry of a SpatVector.

Usage

```
## S4 method for signature 'SpatVector'
spin(x, angle, x0, y0)
```

Arguments

x	SpatVector
angle	numeric. Angle of rotation in degrees
x0	numeric. x-coordinate of the center of rotation. If missing, the center of the extent of x is used
y0	numeric. y-coordinate of the center of rotation. If missing, the center of the extent of x is used

Value

SpatVector

See Also

[rescale](#), [t](#), [shift](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
w <- spin(v, 180)
plot(v)
lines(w, col="red")

# lower-right corner as center
e <- as.vector(ext(v))
x <- spin(v, 45, e[1], e[3])
```

split

Split

Description

Split a SpatVector or SpatRaster

Usage

```
## S4 method for signature 'SpatRaster'
split(x, f)

## S4 method for signature 'SpatVector'
split(x, f)
```

Arguments

x SpatRaster or SpatVector
 f If x is a SpatVector: a field (variable) name or a vector of the same length as x.
 If x is a SpatRaster: a vector of the length `nlyr(x)`

Value

Same as x

Examples

```
v <- vect(system.file("ex/lux.shp", package="terra"))
x <- split(v, "NAME_1")

s <- rast(system.file("ex/logo.tif", package="terra"))
y <- split(s, c(1,2,1))
sds(y)
```

 sprc

Create a SpatRasterCollection

Description

Methods to create a SpatRasterCollection. This is an object to hold a collection (list) of SpatRaster objects. There are no restrictions on the similarity of the SpatRaster geometry.

They can be used to combine several SpatRasters to be used with [merge](#) or [mosaic](#)

Usage

```
## S4 method for signature 'SpatRaster'
sprc(x, ...)

## S4 method for signature 'list'
sprc(x)

## S4 method for signature 'missing'
sprc(x)
```

Arguments

x SpatRaster, list with SpatRaster objects, or missing
 ... additional SpatRaster objects

Value

SpatRasterCollection

See Also[sds](#)**Examples**

```
x <- rast(xmin=-110, xmax=-50, ymin=40, ymax=70, ncols=60, nrows=30)
y <- rast(xmin=-80, xmax=-20, ymax=60, ymin=30)
res(y) <- res(x)
values(x) <- 1:ncell(x)
values(y) <- 1:ncell(y)

z <- sprc(x, y)
z
```

`stretch`*Stretch*

Description

Linear or histogram equalization stretch of values in a `SpatRaster`.

For linear stretch, provide the desired output range (`minv` and `maxv`) and the lower and upper bounds in the original data, either as quantiles (`minq` and `maxq`, or as cell values (`smin` and `smax`). If `smin` and `smax` are both not `NA`, `minq` and `maxq` are ignored.

For histogram equalization, these arguments are ignored, but you can provide the desired scale of the output.

Usage

```
## S4 method for signature 'SpatRaster'
stretch(x, minv=0, maxv=255, minq=0, maxq=1, smin=NA, smax=NA,
histeq=FALSE, scale=1, filename="", ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>minv</code>	numeric ≥ 0 and smaller than <code>maxv</code> . lower bound of stretched value
<code>maxv</code>	numeric ≤ 255 and larger than <code>minv</code> . upper bound of stretched value
<code>minq</code>	numeric ≥ 0 and smaller than <code>maxq</code> . lower quantile bound of original value. Ignored if <code>smin</code> is supplied
<code>maxq</code>	numeric ≤ 1 and larger than <code>minq</code> . upper quantile bound of original value. Ignored if <code>smax</code> is supplied
<code>smin</code>	numeric $< smax$. user supplied lower value for the layers, to be used instead of a quantile computed by the function itself
<code>smax</code>	numeric $> smin$. user supplied upper value for the layers, to be used instead of a quantile computed by the function itself

histeq	logical. If TRUE histogram equalization is used instead of linear stretch
scale	numeric. The scale (maximum value) of the output if histeq=TRUE
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

Examples

```
r <- rast(nc=10, nr=10)
values(r) <- rep(1:25, 4)
rs <- stretch(r)
s <- c(r, r*2)
sr <- stretch(s)
```

subset	<i>Subset a SpatRaster or a SpatVector</i>
--------	--

Description

Select a subset of layers from a SpatRaster or select a subset of records (row) and/or variables (columns) from a SpatVector.

Usage

```
## S4 method for signature 'SpatRaster'
subset(x, subset, negate=FALSE, NSE=FALSE, filename="", overwrite=FALSE, ...)

## S4 method for signature 'SpatVector'
subset(x, subset, select, drop=FALSE, NSE=FALSE)
```

Arguments

x	SpatRaster or SpatVector
subset	if x is a SpatRaster: integer or character to select layers if x is a SpatVector: logical expression indicating the rows to keep (missing values are taken as FALSE)
select	expression, indicating columns to select
negate	logical. If TRUE all layers that are not in the subset are selected
NSE	logical. If TRUE, non-standard evaluation (the use of unquoted variable names) is allowed. Set this to FALSE when calling subset from a function
drop	logical. If TRUE, the geometries will be dropped, and a data.frame is returned
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
...	additional arguments for writing files as in writeRaster

Value

if x is a SpatRaster: SpatRaster

if x is a SpatVector: SpatVector or, if drop=TRUE, a data.frame.

Examples

```
### SpatRaster
s <- rast(system.file("ex/logo.tif", package="terra"))
subset(s, 2:3)
subset(s, c(3,2,3,1))
#equivalent to
s[[ c(3,2,3,1) ]]

s[[c("red", "green")]]
s$red

# expression based (partial) matching of names with single brackets
s["re"]
s["^re"]

# not with double brackets
# s[["re"]]

### SpatVector

v <- vect(system.file("ex/lux.shp", package="terra"))
v[2:3,]
v[1:2, 2:3]

subset(v, v$NAME_1 == "Diekirch", c("NAME_1", "NAME_2"))

subset(v, NAME_1 == "Diekirch", c(NAME_1, NAME_2), NSE=TRUE)
```

 subst

replace cell values

Description

Substitute(replace) cell values of a SpatRaster with a new value. See [classify](#) for more complex/flexible replacement.

Usage

```
## S4 method for signature 'SpatRaster'
subst(x, from, to, filename="", ...)
```

Arguments

x	SpatRaster
from	numeric value(s). Normally a vector of the same length as 'to'. If x has multiple layers, it can also be a matrix of numeric value(s) where <code>nrow(x) == length(to)</code> . In that case the output has a single layer, with values based on the combination of the values of the input layers
to	numeric value(s). Normally a vector of the same length as 'from'. If x has a single layer, it can also be a matrix of numeric value(s) where <code>nrow(x) == length(from)</code> . In that case the output has multiple layers, one for each column in to
filename	character. Output filename
...	Additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also[classify](#)**Examples**

```

r <- rast(ncols=5, nrows=5, xmin=0, xmax=1, ymin=0, ymax=1, crs="")
r <- init(r, 1:6)
x <- subst(r, 3, 7)
x <- subst(r, 2:3, NA)
x <- subst(x, NA, 10)

# multiple output layers
z <- subst(r, 2:3, cbind(20,30))

# multiple input layers
rr <- c(r, r+1, r+2)
m <- rbind(c(1:3), c(3:5))
zz <- subst(rr, m, c(100, 200))

```

summarize

*Summarize***Description**

Compute summary statistics for cells, either across layers or between layers (parallel summary).

The following summary methods are available for SpatRaster: `any`, `all`, `max`, `min`, `mean`, `median`, `prod`, `range`, `stdev`, `sum`, `which.min`, `which.max`. See [modal](#) to compute the mode and [app](#) to compute summary statistics that are not included here.

Because generic functions are used, the method applied is chosen based on the first argument: "x". This means that if *r* is a *SpatRaster*, `mean(r, 5)` will work, but `mean(5, r)` will not work.

The `mean` method has an argument "trim" that is ignored.

If `pop=TRUE` `stdev` computes the population standard deviation, computed as:

```
f <- function(x) sqrt(sum((x-mean(x))^2) / length(x))
```

This is different than the sample standard deviation returned by `sd` (which uses *n*-1 as denominator).

Usage

```
## S4 method for signature 'SpatRaster'
min(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
max(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
range(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
prod(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
sum(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
any(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
all(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
range(x, ..., na.rm=FALSE)

## S4 method for signature 'SpatRaster'
which.min(x)

## S4 method for signature 'SpatRaster'
which.max(x)

## S4 method for signature 'SpatRaster'
stdev(x, ..., pop=TRUE, na.rm=FALSE)

## S4 method for signature 'SpatRaster'
mean(x, ..., trim=NA, na.rm=FALSE)

## S4 method for signature 'SpatRaster'
median(x, na.rm=FALSE, ...)
```


Arguments

x	SpatRaster
...	additional SpatRaster objects or numeric values; and arguments filename, overwrite and wopt as for writeRaster
na.rm	logical. If TRUE, NA values are ignored. If FALSE, NA is returned if x has any NA values
trim	ignored
pop	logical. If TRUE, the population standard deviation is computed. Otherwise the sample standard deviation is computed

Value

SpatRaster

See Also

[app](#), [Math-methods](#), [modal](#), [which.lyr](#)

Examples

```
set.seed(0)
r <- rast(nrows=10, ncols=10, nlyrs=3)
values(r) <- runif(ncell(r) * nlyr(r))

x <- mean(r)
# note how this returns one layer
x <- sum(c(r, r[[2]]), 5)

# and this returns three layers
y <- sum(r, r[[2]], 5)

max(r)
max(r, 0.5)

y <- stdev(r)
# not the same as
yy <- app(r, sd)

z <- stdev(r, r*2)

x <- mean(r, filename=paste0(tempfile(), ".tif"))
```

summary

summary

Description

Compute summary statistics (min, max, mean, and quartiles) for `SpatRaster` using base [summary](#) method. A sample is used for very large files.

For single or other statistics see [Summary-methods](#), [global](#), and [quantile](#)

Usage

```
## S4 method for signature 'SpatRaster'  
summary(object, size=100000, warn=TRUE, ...)  
  
## S4 method for signature 'SpatVector'  
summary(object, ...)
```

Arguments

<code>object</code>	<code>SpatRaster</code> or <code>SpatVector</code>
<code>size</code>	positive integer. Size of a regular sample used for large datasets (see spatSample)
<code>warn</code>	logical. If TRUE a warning is given if a sample is used
<code>...</code>	additional arguments passed on to the base summary method

Value

matrix with (an estimate of) the median, minimum and maximum values, the first and third quartiles, and the number of cells with NA values

See Also

[Summary-methods](#), [global](#), [quantile](#)

Examples

```
set.seed(0)  
r <- rast(nrows=10, ncols=10, nlyrs=3)  
values(r) <- runif(nlyr(r)*ncell(r))  
summary(r)
```

`svc`*Create a `SpatVectorCollection`*

Description

Methods to create a `SpatVectorCollection`. This is an object to hold "sub-datasets", each a `SpatVector`, perhaps of different geometry type.

Usage

```
## S4 method for signature 'missing'
svc(x)

## S4 method for signature 'SpatVector'
svc(x, ...)

## S4 method for signature 'list'
svc(x)
```

Arguments

<code>x</code>	<code>SpatVector</code> , or list of a <code>SpatVector</code> , or missing
<code>...</code>	Additional <code>SpatVectors</code>

Value

`SpatVectorCollection`

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
x <- svc()
x <- svc(v, v[1:3,], as.lines(v[3:5,]), as.points(v))
length(x)
x

# extract
x[3]

# replace
x[2] <- as.lines(v[1,])
```

symdif	<i>Symmetrical difference</i>
--------	-------------------------------

Description

Symmetrical difference of polygons

Usage

```
## S4 method for signature 'SpatVector,SpatVector'
symdif(x, y)
```

Arguments

x	SpatVector
y	SpatVector

Value

SpatVector

See Also

[erase](#)

Examples

```
p <- vect(system.file("ex/lux.shp", package="terra"))
b <- as.polygons(ext(6, 6.4, 49.75, 50))
#sd <- symdif(p, b)
#plot(sd, col=rainbow(12))
```

tapp	<i>Apply a function to subsets of layers of a SpatRaster</i>
------	--

Description

Apply a function to subsets of layers of a SpatRaster (similar to [tapply](#) and [aggregate](#)). The layers are combined based on the index.

The function used should return a single value, and the number of layers in the output SpatRaster equals the number of unique values in index.

For example, if you have a SpatRaster with 6 layers, you can use `index=c(1,1,1,2,2,2)` and `fun=sum`. This will return a SpatRaster with two layers. The first layer is the sum of the first three layers in the input SpatRaster, and the second layer is the sum of the last three layers in the input

SpatRaster. Indices are recycled such that `index=c(1,2)` would also return a SpatRaster with two layers (one based on the odd layers (1,3,5), the other based on the even layers (2,4,6)).

The index can also be one of the following values to group by time period (if `x` has the appropriate [time](#) values): "years", "months", "yearmonths", "days", "week" (ISO 8601 week number), or "doy" (day of the year).

See [app](#) or [Summary-methods](#) if you want to use a more efficient function that returns multiple layers based on **all** layers in the SpatRaster.

Usage

```
## S4 method for signature 'SpatRaster'
tapp(x, index, fun, ..., cores=1, filename="", overwrite=FALSE, wopt=list())
```

Arguments

<code>x</code>	SpatRaster
<code>index</code>	factor or numeric (integer). Vector of length <code>nlyr(x)</code> (shorter vectors are recycled) grouping the input layers. It can also be one of the following values: "years", "months", "yearmonths", "days", "week" (ISO 8601 week number), or "doy" (day of the year)
<code>fun</code>	function to be applied. The following functions have been re-implemented in C++ for speed: "sum", "mean", "median", "modal", "which", "which.min", "which.max", "min", "max", "prod", "any", "all", "sd", "std", "first". To use the base-R function for say, "min", you could use something like <code>fun = \(\i) min(i)</code>
<code>...</code>	additional arguments passed to <code>fun</code>
<code>cores</code>	positive integer. If <code>cores > 1</code> , a 'parallel' package cluster with that many cores is created and used. You can also supply a cluster object. Ignored for functions that are implemented by terra in C++ (see under <code>fun</code>)
<code>filename</code>	character. Output filename
<code>overwrite</code>	logical. If TRUE, <code>filename</code> is overwritten
<code>wopt</code>	list with named options for writing files as in writeRaster

Value

SpatRaster

See Also

[app](#), [Summary-methods](#)

Examples

```
r <- rast(ncols=10, nrows=10)
values(r) <- 1:ncell(r)
s <- c(r, r, r, r, r, r)
s <- s * 1:6
```

```
b1 <- tapp(s, index=c(1,1,1,2,2,2), fun=sum)
b1
b2 <- tapp(s, c(1,2,3,1,2,3), fun=sum)
b2
```

 terrain

terrain characteristics

Description

Compute terrain characteristics from elevation data. The elevation values should be in the same units as the map units (typically meter) for projected (planar) raster data. They should be in meter when the coordinate reference system is longitude/latitude.

For accuracy, always compute these values on the original data (do not first change the projection). Distances (needed for slope and aspect) for longitude/latitude data are computed on the WGS84 ellipsoid with Karney's algorithm.

Usage

```
## S4 method for signature 'SpatRaster'
terrain(x, v="slope", neighbors=8, unit="degrees", filename="", ...)
```

Arguments

x	SpatRaster, single layer with elevation values. Values should have the same unit as the map units, or in meters when the crs is longitude/latitude
v	character. One or more of these options: slope, aspect, TPI, TRI, roughness, flowdir (see Details)
unit	character. "degrees" or "radians" for the output of "slope" and "aspect"
neighbors	integer. Indicating how many neighboring cells to use to compute slope or aspect with. Either 8 (queen case) or 4 (rook case)
filename	character. Output filename
...	list. Options for writing files as in writeRaster

Details

When neighbors=4, slope and aspect are computed according to Fleming and Hoffer (1979) and Ritter (1987). When neighbors=8, slope and aspect are computed according to Horn (1981). The Horn algorithm may be best for rough surfaces, and the Fleming and Hoffer algorithm may be better for smoother surfaces (Jones, 1997; Burrough and McDonnell, 1998).

If slope = 0, aspect is set to 0.5*pi radians (or 90 degrees if unit="degrees"). When computing slope or aspect, the coordinate reference system of x must be known for the algorithm to differentiate between planar and longitude/latitude data.

terrain is not vectorized over "neighbors" or "unit" – only the first value is used.

`flowdir` returns the "flow direction" (of water), that is the direction of the greatest drop in elevation (or the smallest rise if all neighbors are higher). They are encoded as powers of 2 (0 to 7). The cell to the right of the focal cell is 1, the one below that is 2, and so on:

32	64	128
16	x	1
8	4	2

If two cells have the same drop in elevation, a random cell is picked. That is not ideal as it may prevent the creation of connected flow networks. ArcGIS implements the approach of Greenlee (1987) and I might adopt that in the future.

The terrain indices are according to Wilson et al. (2007), as in `gdaldem`. TRI (Terrain Ruggedness Index) is the mean of the absolute differences between the value of a cell and the value of its 8 surrounding cells. TPI (Topographic Position Index) is the difference between the value of a cell and the mean value of its 8 surrounding cells. Roughness is the difference between the maximum and the minimum value of a cell and its 8 surrounding cells.

Such measures can also be computed with the `focal` function:

```
f <- matrix(1, nrow=3, ncol=3)
TRI <- focal(x, w=f, fun=function(x, ...) sum(abs(x[-5]-x[5]))/8)
TPI <- focal(x, w=f, fun=function(x, ...) x[5] - mean(x[-5]))
rough <- focal(x, w=f, fun=function(x, ...) max(x) - min(x), na.rm=TRUE)
```

References

- Burrough, P., and R.A. McDonnell, 1998. Principles of Geographical Information Systems. Oxford University Press.
- Fleming, M.D. and Hoffer, R.M., 1979. Machine processing of Landsat MSS data and DMA topographic data for forest cover type mapping. LARS Technical Report 062879. Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana.
- Horn, B.K.P., 1981. Hill shading and the reflectance map. Proceedings of the IEEE 69:14-47
- Jones, K.H., 1998. A comparison of algorithms used to compute hill terrain as a property of the DEM. Computers & Geosciences 24: 315-323
- Karney, C.F.F., 2013. Algorithms for geodesics, J. Geodesy 87: 43-55. doi:10.1007/s00190-012-0578-z.
- Ritter, P., 1987. A vector-based terrain and aspect generation algorithm. Photogrammetric Engineering and Remote Sensing 53: 1109-1111

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
x <- terrain(r, "slope")
```

text *Add labels to a map*

Description

Plots labels, that is a textual (rather than color) representation of values, on top an existing plot (map).

Usage

```
## S4 method for signature 'SpatRaster'
text(x, labels, digits=0, halo=FALSE, ...)

## S4 method for signature 'SpatVector'
text(x, labels, halo=FALSE, ...)
```

Arguments

x	SpatRaster or SpatVector
labels	character. Optional. Vector of labels with length(x) or a variable name from names(x)
digits	integer. how many digits should be used?
halo	logical. If TRUE a "halo" is printed around the text. If TRUE, additional arguments hc="white" and hw=0.1 can be modified to set the colour and width of the halo
...	additional arguments to pass to graphics function text

See Also

[text](#), [plot](#)

Examples

```
r <- rast(nrows=4, ncols=4)
values(r) <- 1:ncell(r)
plot(r)
text(r)

plot(r)
text(r, halo=TRUE, hc="blue", col="white", hw=0.2)

plot(r, col=rainbow(16))
text(r, col=c("black", "white"), vfont=c("sans serif", "bold"), cex=2)
```

tighten	<i>tighten SpatRaster or SpatRasterDataset objects</i>
---------	--

Description

Combines data sources within a SpatRaster object (that are in memory, or from the same file) to allow for faster processing.

Or combine sub-datsets into a SpatRaster.

Usage

```
## S4 method for signature 'SpatRaster'
tighten(x)

## S4 method for signature 'SpatRasterDataset'
tighten(x)
```

Arguments

x SpatRaster or SpatRasterDataset

Value

SpatRaster

Examples

```
r <- rast(nrow=5, ncol=9, vals=1:45)
x <- c(r, r*2, r*3)
x
tighten(x)
```

time	<i>time of SpatRaster layers</i>
------	----------------------------------

Description

Get or set the time of the layers of a SpatRaster. Time can be stored as [POSIXlt](#) (date and time, with a resolution of seconds, and a time zone), [Date](#), "months", "years", or "yearmonths".

Usage

```
## S4 method for signature 'SpatRaster'
time(x, format="")

## S4 replacement method for signature 'SpatRaster'
time(x, tstep="")<-value

## S4 method for signature 'SpatRaster'
timeInfo(x)
```

Arguments

x	SpatRaster
format	One of "", "seconds" (POSIXlt), "days" (Date), "yearmonths" (decimal years), "years", "months". If "", the returned format is (based on) the format that was used to set the time
value	Date, POSIXt, yearmon (defined in package zoo), or numeric
tstep	One of "years", "months", "yearmonths". Used when value is numeric. Ignored when value is of type Date, POSIXt, or yearmon

Value

time: POSIXlt, Date, or numeric timeInfo: data.frame

See Also

[depth](#)

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))

# Date"
d <- as.Date("2001-05-04") + 0:2
time(s) <- d
time(s)

# POSIX (date/time with a resolution of seconds)
time(s) <- as.POSIXlt(d)
time(s)

# with time zone
time(s) <- as.POSIXlt(Sys.time(), "America/New_York") + 0:2
time(s)
timeInfo(s)

# years
time(s, tstep="years") <- 2000 + 0:2
s
```

```
time(s, tstep="months") <- 1:3
s
```

tmpFiles	<i>Temporary files</i>
----------	------------------------

Description

List and optionally remove temporary files created by the terra package. These files are created when an output SpatRaster may be too large to store in memory (RAM). This can happen when no filename is provided to a function and when using functions where you cannot provide a filename.

Temporary files are automatically removed at the end of each R session that ends normally. You can use tmpFiles to see the files in the current sessions, including those that are orphaned (not connect to a SpatRaster object any more) and from other (perhaps old) sessions, and remove all the temporary files.

Usage

```
tmpFiles(current=TRUE, orphan=FALSE, old=FALSE, remove=FALSE)
```

Arguments

current	logical. If TRUE, temporary files from the current R session are included
orphan	logical. If TRUE, temporary files from the current R session that are no longer associated with a SpatRaster object (if current is TRUE these are also included)
old	logical. If TRUE, temporary files from other "R" sessions. Unless you are running multiple instances of R at the same time, these are from old (possibly crashed) R sessions and should be removed
remove	logical. If TRUE, temporary files are removed

Value

character

See Also

[terraOptions](#)

Examples

```
tmpFiles()
```

topology

*Vector topology methods***Description**

makeNodes create nodes on lines

mergeLines connect lines to form polygons

removeDupNodes removes duplicate nodes in geometries and optionally rounds the coordinates

emptyGeoms returns the indices of empty (null) geometries

snap makes boundaries of geometries identical if they are very close to each other.

Usage

```
## S4 method for signature 'SpatVector'
mergeLines(x)
## S4 method for signature 'SpatVector'
snap(x, y=NULL, tolerance)
## S4 method for signature 'SpatVector'
removeDupNodes(x, digits = -1)
## S4 method for signature 'SpatVector'
makeNodes(x)
```

Arguments

x	SpatVector of lines or polygons
y	SpatVector of lines or polygons to snap to. If NULL snapping is to the other geometries in x
tolerance	numeric. Snapping tolerance (distance between geometries)
digits	numeric. Number of digits used in rounding. Ignored if < 0

Value

SpatVector

See Also

[sharedPaths](#), [gaps](#), [simplifyGeom](#)

Examples

```
p1 <- as.polygons(ext(0,1,0,1))
p2 <- as.polygons(ext(1.1,2,0,1))

p <- rbind(p1, p2)

y <- snap(p, tol=.15)
```

```
plot(p, lwd=3, col="light gray")
lines(y, col="red", lwd=2)
```

transpose

Transpose

Description

Transpose a SpatRaster

Usage

```
## S4 method for signature 'SpatRaster'
t(x)

## S4 method for signature 'SpatVector'
t(x)

## S4 method for signature 'SpatRaster'
trans(x, filename="", ...)
```

Arguments

x	SpatRaster
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

See Also

[flip](#), [rotate](#)

Examples

```
r <- rast(nrows=18, ncols=36)
values(r) <- 1:ncell(r)
tr1 <- t(r)
tr2 <- trans(r)
ttr <- trans(tr2)
```

trim	<i>Trim a SpatRaster</i>
------	--------------------------

Description

Trim (shrink) a SpatRaster by removing outer rows and columns that are NA or another value.

Usage

```
## S4 method for signature 'SpatRaster'
trim(x, padding=0, value=NA, filename="", ...)
```

Arguments

x	SpatRaster
padding	integer. Number of outer rows/columns to keep
value	numeric. The value of outer rows or columns that are to be removed
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

Examples

```
r <- rast(ncols=10, nrows=10, xmin=0,xmax=10,ymin=0,ymax=10)
v <- rep(NA, ncell(r))
v[c(12,34,69)] <- 1:3
values(r) <- v
s <- trim(r)
```

union	<i>Union SpatVector or SpatExtent objects</i>
-------	---

Description

Overlapping polygons (between, not within, objects) are intersected. Union for lines and points simply combines the two data sets; without any geometric intersections. This is equivalent to [c](#). Attributes are joined. See [c](#) if you want to combine polygons without intersection.

If x and y have a different geometry type, a SpatVectorCollection is returned.

If a single SpatVector is supplied, overlapping polygons are intersected. Original attributes are lost. New attributes allow for determining how many, and which, polygons overlapped.

SpatExtent: Objects are combined into their union; this is equivalent to [+](#).

Usage

```
## S4 method for signature 'SpatVector,SpatVector'
union(x, y)

## S4 method for signature 'SpatVector,missing'
union(x, y)

## S4 method for signature 'SpatExtent,SpatExtent'
union(x, y)
```

Arguments

x	SpatVector or SpatExtent
y	Same as x or missing

Value

SpatVector or SpatExtent

See Also

[intersect](#)
[merge](#) and [mosaic](#) to union SpatRaster objects.
[crop](#) and [extend](#) for the union of SpatRaster and SpatExtent.
[merge](#) for merging a data.frame with attributes of a SpatVector.
[aggregate](#) to dissolve SpatVector objects.

Examples

```
e1 <- ext(-10, 10, -20, 20)
e2 <- ext(0, 20, -40, 5)
union(e1, e2)

#SpatVector
v <- vect(system.file("ex/lux.shp", package="terra"))
v <- v[,3:4]
p <- vect(c("POLYGON ((5.8 49.8, 6 49.9, 6.15 49.8, 6 49.65, 5.8 49.8))",
"POLYGON ((6.3 49.9, 6.2 49.7, 6.3 49.6, 6.5 49.8, 6.3 49.9))"), crs=crs(v))
values(p) <- data.frame(pid=1:2, value=expansion(p))
u <- union(v, p)
plot(u, "pid")

b <- buffer(v, 1000)

u <- union(b)
u$sum <- rowSums(as.data.frame(u))
plot(u, "sum")
```

unique	<i>Unique values</i>
--------	----------------------

Description

This method returns the unique values in a `SpatRaster`, or removes duplicates records (geometry and attributes) in a `SpatVector`.

Usage

```
## S4 method for signature 'SpatRaster'
unique(x, incomparables=FALSE, na.rm=TRUE, as.raster=FALSE)
```

```
## S4 method for signature 'SpatVector'
unique(x, incomparables=FALSE, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code> or <code>SpatVector</code>
<code>incomparables</code>	logical. If <code>FALSE</code> and <code>x</code> is a <code>SpatRaster</code> : the unique values are determined for all layers together, and the result is a matrix. If <code>TRUE</code> , each layer is evaluated separately, and a list is returned. If <code>x</code> is a <code>SpatVector</code> this argument is as for a <code>data.frame</code>
<code>na.rm</code>	logical. If <code>TRUE</code> , <code>NaN</code> is included if there are any missing values
<code>as.raster</code>	logical. If <code>TRUE</code> , a single-layer categorical <code>SpatRaster</code> with the unique values is returned
<code>...</code>	additional arguments passed on to unique

Value

If `x` is a `SpatRaster`: `data.frame` or list (if `incomparables=FALSE`)

If `x` is a `SpatVector`: `SpatVector`

Examples

```
r <- rast(ncols=5, nrows=5)
values(r) <- rep(1:5, each=5)
unique(r)
s <- c(r, round(r/3))
unique(s)
unique(s, TRUE)

unique(s, as.raster=TRUE)

v <- vect(cbind(x=c(1:5,1:5), y=c(5:1,5:1)),
crs="+proj=utm +zone=1 +datum=WGS84")
```

```
nrow(v)
u <- unique(v)
nrow(u)

values(v) <- c(1:5, 1:3, 5:4)
unique(v)
```

units	<i>units of SpatRaster or SpatRasterDataSet</i>
-------	---

Description

Get or set the units of the layers of a SpatRaster or the datasets in a SpatRasterDataSet.

Usage

```
## S4 method for signature 'SpatRaster'
units(x)

## S4 replacement method for signature 'SpatRaster'
units(x)<-value

## S4 method for signature 'SpatRasterDataset'
units(x)

## S4 replacement method for signature 'SpatRasterDataset'
units(x)<-value
```

Arguments

x	SpatRaster
value	character

Value

character

See Also

[time](#), [names](#)

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))

units(s) <- c("m/s", "kg", "ha")
units(s)
s
```

```
units(s) <- "kg"  
units(s)
```

valid *Check or fix polygon validity*

Description

Check the validity of polygons or attempt to fix it

Usage

```
## S4 method for signature 'SpatVector'  
is.valid(x, messages=FALSE, as.points=FALSE)  
  
## S4 method for signature 'SpatVector'  
makeValid(x)
```

Arguments

x	SpatVector
messages	logical. If TRUE the error messages are returned
as.points	logical. If TRUE, it is attempted to return locations where polygons are invalid as a SpatVector or points

Value

logical

Examples

```
w <- vect("POLYGON ((0 -5, 10 0, 10 -10, 0 -5))")  
is.valid(w)  
  
w <- vect("POLYGON ((0 -5, 10 0, 10 -10, 4 -2, 0 -5))")  
is.valid(w)  
is.valid(w, TRUE)  
  
plot(w)  
points(cbind(4.54, -2.72), cex=2, col="red")
```

values

Cell values and geometry attributes

Description

Get the cell values of a `SpatRaster` or the attributes of a `SpatVector`.

By default all values returned are numeric. This is because a vector or matrix can only store one data type, and a `SpatRaster` may consist of multiple data types. However, with `values(x, dataframe=TRUE)` and `as.data.frame(x)` the values returned match the type of each layer, and can be numeric, logical, integer, or factor.

Usage

```
## S4 method for signature 'SpatRaster'
values(x, mat=TRUE, dataframe=FALSE, row=1,
       nrows=nrow(x), col=1, ncols=ncol(x), na.rm=FALSE, ...)

## S4 method for signature 'SpatVector'
values(x, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code> or <code>SpatVector</code>
<code>mat</code>	logical. If <code>TRUE</code> , values are returned as a matrix instead of as a vector, except when <code>dataframe</code> is <code>TRUE</code>
<code>dataframe</code>	logical. If <code>TRUE</code> , values are returned as a <code>data.frame</code> instead of as a vector (also if <code>matrix</code> is <code>TRUE</code>)
<code>row</code>	positive integer. Row number to start from, should be between 1 and <code>nrow(x)</code>
<code>nrows</code>	positive integer. How many rows?
<code>col</code>	positive integer. Column number to start from, should be between 1 and <code>ncol(x)</code>
<code>ncols</code>	positive integer. How many columns? Default is the number of columns left after the start column
<code>na.rm</code>	logical. Remove NAs?
<code>...</code>	additional arguments passed to <code>data.frame</code>

Details

If `x` is a `SpatRaster`, and `mat=FALSE`, the values are returned as a vector. In cell-order by layer. If `mat=TRUE`, a matrix is returned in which the values of each layer are represented by a column (with `ncell(x)` rows). The values per layer are in cell-order, that is, from top-left, to top-right and then down by row. Use `as.matrix(x, wide=TRUE)` for an alternative matrix representation where the number of rows and columns matches that of `x`.

Value

matrix or data.frame

See Also

[values<-](#), [focalValues](#)

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
r
x <- values(r)
x[3650:3655, ]
r[3650:3655]

ff <- system.file("ex/lux.shp", package="terra")
v <- vect(ff)
y <- values(v)
head(y)
```

vect

Create SpatVector objects

Description

Methods to create a SpatVector from a filename or other R object.

A filename can be for a shapefile or any spatial file format.

You can use a data.frame to make a SpatVector of points; or a "geom" matrix to make a SpatVector of any supported geometry (see examples and [geom](#)).

You can supply a list of SpatVectors to append them into a single SpatVector.

SpatVectors can also be created from "Well Known Text", and from spatial vector data objects defined in the sf or sp packages.

Usage

```
## S4 method for signature 'character'
vect(x, layer="", query="", extent=NULL, filter=NULL, crs="", proxy=FALSE)
```

```
## S4 method for signature 'matrix'
vect(x, type="points", atts=NULL, crs="")
```

```
## S4 method for signature 'data.frame'
vect(x, geom=c("lon", "lat"), crs="", keepgeom=FALSE)
```

```
## S4 method for signature 'list'
vect(x)
```

```
## S4 method for signature 'sf'
vect(x)
```

Arguments

x	character. A filename; or a "Well Known Text" string; or a data.frame (only to make a SpatVector of points); or a "geom" matrix to make a SpatVector of any supported geometry (see examples and geom); or a spatial vector data object defined in the sf or sp packages
layer	character. layer name to select a layer from a file (database) with multiple layers
query	character. An query to subset the dataset in the OGR-SQL dialect
extent	Spat* object. The extent of the object is used as a spatial filter to select the geometries to read. Ignored if filter is not NULL
filter	SpatVector. Used as a spatial filter to select geometries to read (the convex hull is used for lines or points). It is guaranteed that all features that overlap with the extent of filter will be returned. It can happen that additional geometries are returned
type	character. Geometry type. Must be "points", "lines", or "polygons"
atts	data.frame with the attributes. The number of rows must match the number of geometrical elements
crs	character. The coordinate reference system in one of the following formats: WKT/WKT2, <authority>:<code>, or PROJ-string notation (see crs)
proxy	logical. If TRUE a SpatVectorProxy is returned
geom	character. The field name(s) with the geometry data. Either two names for x and y coordinates of points, or a single name for a single column with WKT geometries)
keepgeom	logical. If TRUE the geom variable(s) is (are) also included in the attributes

Value

SpatVector

See Also

[geom](#)

Examples

```
### SpatVector from file
f <- system.file("ex/lux.shp", package="terra")
f
v <- vect(f)
v
```

```

## subsetting (large) files
## with attribute query
v <- vect(f, query="SELECT NAME_1, NAME_2, ID_2 FROM lux WHERE ID_2 < 4")
v

## with an extent
e <- ext(5.9, 6.3, 49.9, 50)
v <- vect(f, extent=e)

## with polygons
p <- as.polygons(e)
v <- vect(f, filter=p)

### SpatVector from a geom matrix
x1 <- rbind(c(-180,-20), c(-140,55), c(10, 0), c(-140,-60))
x2 <- rbind(c(-10,0), c(140,60), c(160,0), c(140,-55))
x3 <- rbind(c(-125,0), c(0,60), c(40,5), c(15,-45))
hole <- rbind(c(80,0), c(105,13), c(120,2), c(105,-13))
z <- rbind(cbind(object=1, part=1, x1, hole=0), cbind(object=2, part=1, x3, hole=0),
cbind(object=3, part=1, x2, hole=0), cbind(object=3, part=1, hole, hole=1))
colnames(z)[3:4] <- c('x', 'y')

p <- vect(z, "polygons")
p

z[z[, "hole"]==1, "object"] <- 4
lns <- vect(z[,1:4], "lines")
plot(p)
lines(lns, col="red", lwd=2)

### from wkt
v <- vect("POLYGON ((0 -5, 10 0, 10 -10, 0 -5))")

wkt <- c("MULTIPOLYGON ( ((40 40, 20 45, 45 30, 40 40)),
((20 35, 10 30, 10 10, 30 5, 45 20, 20 35),(30 20, 20 15, 20 25, 30 20)))",
"POLYGON ((0 -5, 10 0, 10 -10, 0 -5))")
w <- vect(wkt)

# combine two SpatVectors
vw <- rbind(w, v)

# add a data.frame
d <- data.frame(id=1:2, name=c("a", "b"))
values(w) <- d

# add data.frame on creation, here from a geom matrix
g <- geom(w)
d <- data.frame(id=1:2, name=c("a", "b"))
m <- vect(g, "polygons", atts=d, crs="+proj=longlat +datum=WGS84")

### SpatVector from a data.frame
d$wkt <- wkt

```

```
x <- vect(d, geom="wkt")

d$wkt <- NULL
d$lon <- c(0,10)
d$lat <- c(0,10)
x <- vect(d, geom=c("lon", "lat"))

# SpatVector to sf
#sf::st_as_sf(x)
```

vector-attributes *Get or replace attribute values of a SpatVector*

Description

Replace values of a SpatVector.

Usage

```
## S4 method for signature 'SpatVector'
x$name

## S4 replacement method for signature 'SpatVector'
x$name<-value
```

Arguments

x	SpatVector
name	character (field name) or numeric (column number)
value	vector of new values

Value

vector

See Also

[values](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
v$NAME_1
v$NAME_1[3] <- "my name"
v$ID_1 <- LETTERS[1:12]
v$new <- sample(12)
values(v)
```



```
v[2,2] <- "hello"
v[1,] <- v[10,]
v[,3] <- v[,1]
v[2, "NAME_2"] <- "terra"
head(v, 3)
```

vector_layers	<i>List or remove layers from a vector file</i>
---------------	---

Description

List or remove layers from a vector file that supports layers such as GPGK

Usage

```
vector_layers(filename, delete="", return_error=FALSE)
```

Arguments

filename	character. filename
delete	character. layers to be deleted (ignored if the value is "")
return_error	logical. If TRUE, an error occurs if some layers cannot be deleted. Otherwise a warning is given

voronoi	<i>Voronoi diagram and Delaunay triangles</i>
---------	---

Description

Get a Voronoi diagram or Delaunay triangles for points, or the nodes of lines or polygons

Usage

```
## S4 method for signature 'SpatVector'
voronoi(x, bnd=NULL, tolerance=0, as.lines=FALSE, deldir=FALSE)
```

```
## S4 method for signature 'SpatVector'
delaunay(x, tolerance=0, as.lines=FALSE)
```

Arguments

x	SpatVector
bnd	SpatVector to set the outer boundary of the voronoi diagram
tolerance	numeric >= 0, snapping tolerance (0 is no snapping)
as.lines	logical. If TRUE, lines are returned without the outer boundary
deldir	logical. If TRUE, the <code>deldir</code> is used instead of the GEOS C++ library method. It has been reported that <code>deldir</code> does not choke on very large data sets

Value

SpatVector

Examples

```
wkt <- c("MULTIPOLYGON ( ((40 40, 20 45, 45 30, 40 40)),
  ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35),(30 20, 20 15, 20 25, 30 20)))",
  "POLYGON ((0 -5, 10 0, 10 -10, 0 -5))")
x <- vect(wkt)
v <- voronoi(x)
v
d <- delaunay(x)
d

plot(v, lwd=2, col=rainbow(15))
lines(x, col="gray", lwd=2)
points(x)
```

vrt

*Virtual Raster Dataset***Description**

Create a Virtual Raster Dataset (VRT) from a collection of file-based raster datasets (tiles).

Usage

```
## S4 method for signature 'character'
vrt(x, filename="", options=NULL, overwrite=FALSE)
```

Arguments

x	character. Filenames of raster "tiles". See tiles
filename	character. Output VRT filename
options	character. All arguments as separate vector elements. Options as for gdalbuild-vrt
overwrite	logical. Should filename be overwritten if it exists?

Value

SpatRaster

See Also

[makeTiles](#) to create tiles; [makeVRT](#) to create a .vrt file for a file without a header

Examples

```

r <- rast(ncols=100, nrows=100)
values(r) <- 1:ncell(r)
x <- rast(ncols=2, nrows=2)
filename <- paste0(tempfile(), "_tif")
ff <- makeTiles(r, x, filename)
ff

## output in lower resolution
#vrtfile <- paste0(tempfile(), ".vrt")
#v <- vrt(ff, vrtfile, options = c("-tr", 5, 5))
#head(readLines(vrtfile))
#v

```

weighted.mean

Weighted mean of layers

Description

Compute the weighted mean for each cell of the layers of a `SpatRaster`. The weights can be spatially variable or not.

Usage

```

## S4 method for signature 'SpatRaster,numeric'
weighted.mean(x, w, na.rm=FALSE, filename="", ...)

## S4 method for signature 'SpatRaster,SpatRaster'
weighted.mean(x, w, na.rm=FALSE, filename="", ...)

```

Arguments

x	<code>SpatRaster</code>
w	A vector of weights (one number for each layer), or for spatially variable weights, a <code>SpatRaster</code> with weights (should have the same extent, resolution and number of layers as x)
na.rm	Logical. Should missing values be removed?
filename	character. Output filename
...	options for writing files as in writeRaster

Value

`SpatRaster`

See Also

[Summary-methods](#), [weighted.mean](#)

Examples

```

b <- rast(system.file("ex/logo.tif", package="terra"))

# give least weight to first layer, most to last layer
wm1 <- weighted.mean(b, w=1:3)

# spatially varying weights
# weigh by column number
w1 <- init(b, "col")

# weigh by row number
w2 <- init(b, "row")
w <- c(w1, w2, w2)

wm2 <- weighted.mean(b, w=w)

```

where

Where are the cells with the min or max values?

Description

This method returns the cell numbers for the cells with the min or max values of a `SpatRaster`.

Usage

```

## S4 method for signature 'SpatRaster'
where.min(x, list)
## S4 method for signature 'SpatRaster'
where.max(x, list)

```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>list</code>	logical. If TRUE a list is returned instead of a matrix

Value

matrix or list

See Also

[which](#) and [Summary-methods](#) for `which.min` and `which.max`

Examples

```

f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
where.min(r)

```

 which.lyr

Which cells are TRUE?

Description

This method returns a single layer SpatRaster with cell values indicating the the first layer in the input that is TRUE. All numbers that are not zero (or FALSE), are considered to be TRUE.

Usage

```
## S4 method for signature 'SpatRaster'
which.lyr(x)
```

Arguments

x SpatRaster

Value

SpatRaster

See Also

[isTRUE](#), [which](#), See [Summary-methods](#) for [which.min](#) and [which.max](#)

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))
x <- which.lyr(s > 100)
```

 width

SpatVector geometric properties

Description

width returns the minimum diameter of the geometry, defined as the smallest band that contains the geometry, where a band is a strip of the plane defined by two parallel lines. This can be thought of as the smallest hole that the geometry can be moved through, with a single rotation.

clearance returns the minimum clearance of a geometry. The minimum clearance is the smallest amount by which a vertex could be moved to produce an invalid polygon, a non-simple linestring, or a multipoint with repeated points. If a geometry has a minimum clearance of 'mc', it can be said that:

No two distinct vertices in the geometry are separated by less than 'mc' No vertex is closer than 'mc' to a line segment of which it is not an endpoint. If the minimum clearance cannot be defined for a geometry (such as with a single point, or a multipoint whose points are identical, NA is returned.

Usage

```
## S4 method for signature 'SpatVector'
width(x, as.lines=FALSE)
## S4 method for signature 'SpatVector'
clearance(x, as.lines=FALSE)
```

Arguments

x SpatVector of lines or polygons
as.lines logical. If TRUE lines are returned that define the width or clearance

Value

numeric or SpatVector

See Also

[minRect](#)

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
width(v)
clearance(v)
```

window

Set a window

Description

Experimental: Assign a window (area of interest) to a SpatRaster with a SpatExtent, or set it to NULL to remove the window. This is similar to [crop](#) without actually creating a new dataset.

Currently, the window will be forced to intersect with the extent of the SpatRaster. It is envisioned that in future versions, the window may also go outside these boundaries.

Usage

```
## S4 replacement method for signature 'SpatRaster'
window(x)<-value

## S4 method for signature 'SpatRaster'
window(x)
```

Arguments

x SpatRaster
value SpatExtent

Value

none for window<- and logical for window

See Also

[crop](#), [extend](#)

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
global(r, "mean", na.rm=TRUE)
e <- ext(c(5.9, 6, 49.95, 50))

window(r) <- e
global(r, "mean", na.rm=TRUE)
r

x <- rast(f)
xe <- crop(x, e)
global(xe, "mean", na.rm=TRUE)

b <- c(xe, r)
window(b)
b

window(r) <- NULL
r
```

wrap

wrap (pack) a SpatRaster or SpatVector object

Description

Wrap a SpatRaster or SpatVector object to create a Packed* object. Packed objects can be saved as an R object to disk (.rds or .RData), or passed over a connection that serializes (e.g. to nodes on a computer cluster); but with large datasets passing a filename could be more sensible in that context.

Usage

```
## S4 method for signature 'SpatRaster'
wrap(x)

## S4 method for signature 'SpatVector'
wrap(x)
```

Arguments

x SpatVector or SpatRaster

Value

Packed* object

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
p <- wrap(v)
p
vv <- vect(p)
vv
```

writeCDF

Write raster data to a NetCDF file

Description

Write a SpatRaster or SpatRasterDataset to a NetCDF file.

When using a SpatRasterDataset, the varname, longname, and unit should be set in the object (see examples).

Always use the ".nc" or ".cdf" file extension to assure that the file can be properly read again by GDAL

Usage

```
## S4 method for signature 'SpatRaster'
writeCDF(x, filename, varname, longname="", unit="", ...)

## S4 method for signature 'SpatRasterDataset'
writeCDF(x, filename, overwrite=FALSE, zname="time",
         prec="float", compression=NA, missval, ...)
```

Arguments

x SpatRaster or SpatRasterDataset

filename character. Output filename

varname character. Name of the dataset

longname character. Long name of the dataset

unit character. Unit of the data

overwrite logical. If TRUE, filename is overwritten

zname character. The name of the "time" dimension

prec	character. One of "double", "float", "integer", "short", "byte" or "char"
compression	Can be set to an integer between 1 (least compression) and 9 (most compression)
missval	numeric, the number used to indicate missing values
...	additional arguments passed on to <code>ncvar_def</code>

Value

SpatRaster or SpatDataSet

See Also

see [writeRaster](#) for writing other file formats

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
fname <- paste0(tempfile(), ".nc")
#rr <- writeCDF(r, fname, overwrite=TRUE, varname="alt",
#             longname="elevation in m above sea level", unit="m")

a <- rast(ncols=5, nrows=5, nl=50)
values(a) <- 1:prod(dim(a))
time(a) <- as.Date("2020-12-31") + 1:nlyr(a)
#aa <- writeCDF(a, fname, overwrite=TRUE, varname="power",
#             longname="my nice data", unit="U/Pa")

b <- sqrt(a)
s <- sds(a, b)
names(s) <- c("temp", "prec")
longnames(s) <- c("temperature (C)", "precipitation (mm)")
units(s) <- c("C", "mm")
#ss <- writeCDF(s, fname, overwrite=TRUE)

# for CRAN
#file.remove(fname)
```

writeRaster

Write raster data to a file

Description

Write a SpatRaster object to a file.

Usage

```
## S4 method for signature 'SpatRaster,character'
writeRaster(x, filename, overwrite=FALSE, ...)
```

Arguments

x	SpatRaster
filename	character. Output filename. Can be a single filename, or as many filenames as <code>nlyr(x)</code> to write a file for each layer
overwrite	logical. If TRUE, filename is overwritten
...	additional arguments for for writing files. See Details

Details

In `writeRaster`, and in other methods that generate `SpatRaster` objects, options for writing raster files to disk can be provided as additional arguments or, in a few cases, as the `wopt` argument (a named list) if the additional arguments are already used for a different purpose. The following options are available:

name	description
datatype	values for datatype are "INT1U", "INT2U", "INT2S", "INT4U", "INT4S", "FLT4S", "FLT8S". The first three le
filetype	file format expresses as GDAL driver names . If this argument is not supplied, the driver is derived from the filena
gdal	GDAL driver specific datasource creation options. See the GDAL documentation. For example, with the GeoTiff
tempdir	the path where temporary files are to be written to.
progress	positive integer. If the number of chunks is larger, a progress bar is shown.
memfrac	numeric between 0 and 0.9 (higher values give a warning). The fraction of available RAM that terra is allowed to
memmax	memmax - the maximum amount of RAM (in GB) that terra can use when processing a raster dataset. Should be
names	output layer names.
NAflag	numeric. value to represent missing (NA or NaN) values. See note
verbose	logical. If TRUE debugging information is printed.
steps	positive integers. In how many steps (chunks) do you want to process the data (for debugging)
todisk	logical. If TRUE processing operates as if the dataset is very large and needs to be written to a temporary file (for c

Value

`SpatRaster`. This function is used for the side-effect of writing values to a file.

Note

GeoTiff files are, by default, written with LZW compression. If you do not want compression, use `gdal="COMPRESS=NONE"`.

When writing integer values the lowest available value (given the datatype) is used to represent NA for signed types, and the highest value is used for unsigned values. This can be a problem with byte data (between 0 and 255) as the value 255 is reserved for NA. To keep the value 255, you need to set another value as `NAflag`, or do not set a `NAflag` (with `NAflag=NA`)

See Also

see [writeCDF](#) for writing NetCDF files.

Examples

```

library(terra)
r <- rast(nrows=5, ncols=5, vals=1:25)

# create a temporary filename for the example
f <- file.path(tempdir(), "test.tif")

writeRaster(r, f, overwrite=TRUE)

writeRaster(r, f, overwrite=TRUE, gdal=c("COMPRESS=NONE", "TFW=YES"), datatype='INT1U')

## Or with a wopt argument:

writeRaster(r, f, overwrite=TRUE, wopt= list(gdal=c("COMPRESS=NONE"), datatype='INT1U'))

## remove the file
unlink(f)

```

writeVector*Write SpatVector data to a file*

Description

Write a SpatVector to a file. You can choose one of many file formats.

Usage

```

## S4 method for signature 'SpatVector,character'
writeVector(x, filename, filetype=NULL, layer=NULL, insert=FALSE,
           overwrite=FALSE, options="ENCODING=UTF-8")

```

Arguments

x	SpatVector
filename	character. Output filename
filetype	character. A file format associated with a GDAL "driver" such as "ESRI Shapefile". See <code>gdal(drivers=TRUE)</code> or the GDAL docs . If NULL it is attempted to guess the filetype from the filename extension
layer	character. Output layer name. If NULL the filename is used
insert	logical. If TRUE, a new layer is inserted into the file, if the format allows it (e.g. GPKG allows that). See vector_layers to remove a layer
overwrite	logical. If TRUE, filename is overwritten
options	character. Format specific GDAL options such as "ENCODING=UTF-8". Use NULL or "" to not use any options

Examples

```
v <- vect(cbind(1:5,1:5))
crs(v) <- "+proj=longlat +datum=WGS84"
v$id <- 1:length(v)
v$name <- letters[1:length(v)]
tmpf1 <- tempfile()
writeVector(v, tmpf1)
x <- vect(tmpf1)

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
tmpf2 <- tempfile()
writeVector(v, tmpf2)
y <- vect(tmpf2)
```

xmin

Get or set single values of an extent

Description

Get or set single values of an extent. Values can be set for a `SpatExtent` or `SpatRaster`, but not for a `SpatVector`)

Usage

```
## S4 method for signature 'SpatExtent'
xmin(x)

## S4 method for signature 'SpatExtent'
xmax(x)

## S4 method for signature 'SpatExtent'
ymin(x)

## S4 method for signature 'SpatExtent'
ymax(x)

## S4 method for signature 'SpatRaster'
xmin(x)

## S4 method for signature 'SpatRaster'
xmax(x)

## S4 method for signature 'SpatRaster'
ymin(x)

## S4 method for signature 'SpatRaster'
```

```
ymax(x)

## S4 method for signature 'SpatVector'
xmin(x)

## S4 method for signature 'SpatVector'
xmax(x)

## S4 method for signature 'SpatVector'
ymin(x)

## S4 method for signature 'SpatVector'
ymax(x)

## S4 replacement method for signature 'SpatRaster,numeric'
xmin(x)<-value

## S4 replacement method for signature 'SpatRaster,numeric'
xmax(x)<-value

## S4 replacement method for signature 'SpatRaster,numeric'
ymin(x)<-value

## S4 replacement method for signature 'SpatRaster,numeric'
ymax(x)<-value
```

Arguments

x	SpatRaster, SpatExtent, or SpatVector
value	numeric

Value

SpatExtent or numeric coordinate

Examples

```
r <- rast()
ext(r)
ext(c(0, 20, 0, 20))

xmin(r)
xmin(r) <- 0
xmin(r)
```

`xyRowColCell`*Coordinates from a row, column or cell number and vice versa*

Description

Get coordinates of the center of raster cells for a row, column, or cell number of a `SpatRaster` object. Or get row, column, or cell numbers from coordinates or from each other.

Cell numbers start at 1 in the upper left corner, and increase from left to right, and then from top to bottom. The last cell number equals the number of cells of the `SpatRaster` object. row numbers start at 1 at the top, column numbers start at 1 at the left.

Usage

```
## S4 method for signature 'SpatRaster,numeric'  
xFromCol(object, col)  
  
## S4 method for signature 'SpatRaster,numeric'  
yFromRow(object, row)  
  
## S4 method for signature 'SpatRaster,numeric'  
xyFromCell(object, cell)  
  
## S4 method for signature 'SpatRaster,numeric'  
xFromCell(object, cell)  
  
## S4 method for signature 'SpatRaster,numeric'  
yFromCell(object, cell)  
  
## S4 method for signature 'SpatRaster,numeric'  
colFromX(object, x)  
  
## S4 method for signature 'SpatRaster,numeric'  
rowFromY(object, y)  
  
## S4 method for signature 'SpatRaster,numeric,numeric'  
cellFromRowCol(object, row, col)  
  
## S4 method for signature 'SpatRaster,numeric,numeric'  
cellFromRowColCombine(object, row, col)  
  
## S4 method for signature 'SpatRaster,numeric'  
rowFromCell(object, cell)  
  
## S4 method for signature 'SpatRaster,numeric'  
colFromCell(object, cell)
```

```
## S4 method for signature 'SpatRaster,numeric'
rowColFromCell(object, cell)
```

```
## S4 method for signature 'SpatRaster,matrix'
cellFromXY(object, xy)
```

Arguments

object	SpatRaster
cell	integer. cell number(s)
col	integer. column number(s) or missing (equivalent to all columns)
row	integer. row number(s) or missing (equivalent to all rows)
x	x coordinate(s)
y	y coordinate(s)
xy	matrix of x and y coordinates

Details

Cell numbers start at 1 in the upper left corner, and increase from left to right, and then from top to bottom. The last cell number equals the number of cells of the SpatRaster (see [ncell](#)).

Value

xFromCol, yFromCol, xFromCell, yFromCell: vector of x or y coordinates

xyFromCell: matrix(x,y) with coordinate pairs

colFromX, rowFromY, cellFromXY, cellFromRowCol, rowFromCell, colFromCell: vector of row, column, or cell numbers

rowColFromCell: matrix of row and column numbers

See Also

[crds](#)

Examples

```
r <- rast()

xFromCol(r, c(1, 120, 180))
yFromRow(r, 90)
xyFromCell(r, 10000)
xyFromCell(r, c(0, 1, 32581, ncell(r), ncell(r)+1))

cellFromRowCol(r, 5, 5)
cellFromRowCol(r, 1:2, 1:2)
cellFromRowCol(r, 1, 1:3)

# all combinations
cellFromRowColCombine(r, 1:2, 1:2)
```

```

colFromX(r, 10)
rowFromY(r, 10)
xy <- cbind(lon=c(10,5), lat=c(15, 88))
cellFromXY(r, xy)

# if no row/col specified all are returned
range(xFromCol(r))
length(yFromRow(r))

```

zonal

Zonal statistics

Description

Compute zonal statistics, that is summarize values of a `SpatRaster` for each "zone" defined by another `SpatRaster`.

If `fun` is a true R function, `zonal` may fail for very large `SpatRaster` objects, except for the functions ("mean", "min", "max", "sum", "isNA", and "notNA").

You can also summarize values of a `SpatVector` for each polygon (zone) defined by another `SpatVector`.

See [extract](#) to summarize values of a `SpatRaster` with a `SpatVector`.

Usage

```

## S4 method for signature 'SpatRaster,SpatRaster'
zonal(x, z, fun=mean, ..., as.raster=FALSE, filename="", wopt=list())

## S4 method for signature 'SpatVector,SpatVector'
zonal(x, z, fun=mean, ..., weighted=FALSE, as.polygons=FALSE)

```

Arguments

<code>x</code>	<code>SpatRaster</code> or <code>SpatVector</code>
<code>z</code>	Object of the same class as <code>x</code> . That is, either a <code>SpatRaster</code> with cell-values representing zones or a <code>SpatVector</code> with each polygon geometry representing a zone
<code>fun</code>	function to be applied to summarize the values by zone. Either as character: "mean", "min", "max", "sum", "isNA", and "notNA" and, for relatively small <code>SpatRasters</code> , a proper function
<code>...</code>	additional arguments passed to <code>fun</code>
<code>as.raster</code>	logical. If TRUE, a <code>SpatRaster</code> is returned with the zonal statistic for each zone
<code>filename</code>	character. Output filename (ignored if <code>as.raster=FALSE</code>)
<code>wopt</code>	list with additional arguments for writing files as in writeRaster

weighted logical. If TRUE, a weighted.mean is computed and fun is ignored. Weights are based on the length of the lines or the area of the polygons in x that intersect with z. This argument is ignored if x is a SpatVector or points

as.polygons logical. Should the results be merged with the attributes of z?

Value

A data.frame with a value for each zone) or a SpatRaster or SpatVector of polygons

See Also

See [global](#) for "global" statistics (i.e., all of x is considered a single zone), [app](#) for local statistics, and [extract](#) for summarizing values for polygons

Examples

```
r <- rast(ncols=10, nrows=10)
values(r) <- 1:ncell(r)
z <- rast(r)
values(z) <- rep(c(1:2, NA, 3:4), each=20)
names(z) <- "zone"
zonal(r, z, "sum", na.rm=TRUE)

# multiple layers
r <- rast(system.file("ex/logo.tif", package = "terra"))
# zonal layer
z <- rast(r, 1)
names(z) <- "zone"
values(z) <- rep(c(1:2, NA, c(3:4)), each=ncell(r)/5, length.out=ncell(r))

zonal(r, z, "mean", na.rm = TRUE)

# raster of zonal values
zr <- zonal(r, z, "mean", na.rm = TRUE, as.raster=TRUE)

### SpatVector

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)[,c(2,4)]

p <- spatSample(v, 100)
values(p) <- data.frame(b2=1:100, ssep1=100:1)

zonal(p, v, mean)
```

`zoom`*Zoom in on a map*

Description

Zoom in on a map (plot) by providing a new extent, by default this is done by clicking twice on the map.

Usage

```
## S4 method for signature 'SpatRaster'  
zoom(x, e=draw(), maxcell=100000, layer=1, new=FALSE, ...)  
  
## S4 method for signature 'SpatVector'  
zoom(x, e=draw(), new=FALSE, ...)
```

Arguments

<code>x</code>	<code>SpatRaster</code>
<code>e</code>	<code>SpatExtent</code>
<code>maxcell</code>	positive integer. Maximum number of cells used for the map
<code>layer</code>	positive integer to select the layer to be used
<code>new</code>	logical. If TRUE, the zoomed in map will appear on a new device (window)
<code>...</code>	additional arguments passed to plot

Value

`SpatExtent` (invisibly)

See Also

[draw](#), [plot](#)

Index

- !, SpatRaster-method (Compare-methods),
57
- * **classes**
 - options, 147
 - SpatExtent-class, 202
 - SpatRaster-class, 206
 - SpatVector-class, 209
- * **math**
 - Arith-methods, 30
 - atan2, 36
 - Compare-methods, 57
 - Math-methods, 134
 - modal, 138
- * **methods**
 - activeCat, 18
 - aggregate, 21
 - animate, 25
 - app, 26
 - Arith-methods, 30
 - as.data.frame, 31
 - as.list, 32
 - as.raster, 33
 - barplot, 38
 - boundaries, 39
 - cartogram, 43
 - catalyze, 44
 - cells, 45
 - cellSize, 46
 - colors, 54
 - combineGeoms, 55
 - Compare-methods, 57
 - concats, 58
 - contour, 59
 - convHull, 60
 - cover, 62
 - crosstab, 66
 - deepcopy, 69
 - densify, 70
 - diff, 73
 - disagg, 77
 - dots, 80
 - erase, 82
 - expanse, 83
 - extract, 87
 - extremes, 90
 - factors, 91
 - fillHoles, 93
 - fillTime, 94
 - focalValues, 104
 - gaps, 106
 - geomtype, 109
 - head and tail, 112
 - hist, 113
 - image, 115
 - impose, 116
 - inset, 119
 - intersect, 121
 - is.bool, 122
 - lapp, 124
 - lines, 128
 - makeTiles, 129
 - makeVRT, 130
 - mask, 132
 - match, 133
 - Math-methods, 134
 - merge, 136
 - mergeTime, 137
 - mosaic, 139
 - not.na, 146
 - patches, 150
 - perim, 151
 - persp, 152
 - plet, 153
 - plot, 155
 - plotRGB, 159
 - predict, 161
 - quantile, 166
 - query, 167

- rapp, 169
- rast, 170
- read and write, 176
- relate, 178
- replace, 181
- RGB, 184
- sapp, 186
- scatterplot, 189
- sds, 190
- selectRange, 194
- serialize, 195
- setValues, 196
- sharedPaths, 198
- simplifyGeom, 200
- sources, 201
- Spatial interpolation, 203
- split, 210
- sprc, 211
- summarize, 215
- summary, 218
- svc, 219
- syndif, 220
- tapp, 220
- text, 225
- topology, 229
- union, 231
- values, 236
- vect, 237
- vector_layers, 241
- vrt, 242
- width, 245
- window, 246
- wrap, 247
- writeCDF, 248
- writeRaster, 249
- writeVector, 251
- * **package**
 - terra-package, 6
- * **spatial**
 - activeCat, 18
 - add, 19
 - adjacent, 20
 - aggregate, 21
 - align, 23
 - all.equal, 24
 - animate, 25
 - app, 26
 - approximate, 28
 - Arith-methods, 30
 - as.character, 31
 - as.data.frame, 31
 - as.list, 32
 - as.raster, 33
 - as.spatvector, 34
 - atan2, 36
 - autocorrelation, 37
 - barplot, 38
 - boundaries, 39
 - boxplot, 40
 - buffer, 41
 - c, 42
 - cartogram, 43
 - catalyze, 44
 - cells, 45
 - cellSize, 46
 - centroids, 48
 - clamp, 49
 - classify, 50
 - click, 52
 - coerce, 53
 - colors, 54
 - combineGeoms, 55
 - Compare-methods, 57
 - compareGeom, 57
 - concat, 58
 - contour, 59
 - convHull, 60
 - costDistance, 61
 - cover, 62
 - crds, 64
 - crop, 65
 - crosstab, 66
 - crs, 67
 - deepcopy, 69
 - densify, 70
 - density, 71
 - depth, 72
 - describe, 72
 - diff, 73
 - dimensions, 74
 - direction, 76
 - disagg, 77
 - distance, 78
 - dots, 80
 - draw, 81
 - erase, 82

expansion, 83
ext, 85
extend, 86
extract, 87
extremes, 90
factors, 91
fillHoles, 93
fillTime, 94
flip, 95
focal, 96
focal3D, 98
focalCor, 99
focalCpp, 100
focalMat, 102
focalReg, 103
focalValues, 104
freq, 105
gaps, 106
gdal, 106
geom, 107
geomtype, 109
global, 110
gridDistance, 111
head and tail, 112
hist, 113
ifel, 114
image, 115
impose, 116
initialize, 116
inplace, 117
inset, 119
intersect, 121
is.bool, 122
is.lonlat, 123
lapp, 124
linearUnits, 127
lines, 128
makeTiles, 129
makeVRT, 130
mask, 132
match, 133
Math-methods, 134
mem, 135
merge, 136
mergeTime, 137
mosaic, 139
na.omit, 140
NAflag, 141
names, 142
nearest, 144
north, 145
not.na, 146
options, 147
origin, 148
pairs, 149
patches, 150
perim, 151
persp, 152
plet, 153
plot, 155
plotRGB, 159
predict, 161
project, 164
quantile, 166
query, 167
rapp, 169
rast, 170
rasterize, 173
rasterizeGeom, 175
read and write, 176
rectify, 177
relate, 178
rep, 180
replace, 181
resample, 181
rescale, 183
RGB, 184
rotate, 185
sapp, 186
sbar, 187
scale, 188
scatterplot, 189
sds, 190
segregate, 191
sel, 192
selectHighest, 193
selectRange, 194
serialize, 195
setValues, 196
shade, 197
sharedPaths, 198
shift, 199
simplifyGeom, 200
sort, 201
sources, 201
SpatExtent-class, 202

- Spatial interpolation, 203
- SpatRaster-class, 206
- spatSample, 207
- SpatVector-class, 209
- spin, 209
- split, 210
- sprc, 211
- stretch, 212
- subset, 213
- subst, 214
- summarize, 215
- summary, 218
- svc, 219
- symdif, 220
- tapp, 220
- terra-package, 6
- terrain, 222
- text, 225
- tighten, 226
- time, 226
- tmpFiles, 228
- topology, 229
- transpose, 230
- trim, 231
- union, 231
- unique, 233
- units, 234
- valid, 235
- values, 236
- vect, 237
- vector-attributes, 240
- vector_layers, 241
- voronoi, 241
- vrt, 242
- where, 244
- which.lyr, 245
- width, 245
- window, 246
- wrap, 247
- writeCDF, 248
- writeRaster, 249
- writeVector, 251
- xmin, 252
- xyRowColCell, 254
- zonal, 256
- zoom, 258
- * **univar**
 - freq, 105
 - modal, 138
 - [, 15
 - [, SpatExtent, missing, missing-method (ext), 85
 - [, SpatExtent, numeric, missing-method (ext), 85
 - [, SpatRaster, SpatExtent, missing-method (extract), 87
 - [, SpatRaster, SpatRaster, missing-method (extract), 87
 - [, SpatRaster, SpatVector, missing-method (extract), 87
 - [, SpatRaster, character, missing-method (subset), 213
 - [, SpatRaster, data.frame, missing-method (extract), 87
 - [, SpatRaster, logical, missing-method (replace), 181
 - [, SpatRaster, matrix, missing-method (extract), 87
 - [, SpatRaster, missing, missing-method (extract), 87
 - [, SpatRaster, missing, numeric-method (extract), 87
 - [, SpatRaster, numeric, missing-method (extract), 87
 - [, SpatRaster, numeric, numeric-method (extract), 87
 - [, SpatRasterCollection, numeric, missing-method (subset), 213
 - [, SpatRasterDataset, character, missing-method (subset), 213
 - [, SpatRasterDataset, logical, missing-method (subset), 213
 - [, SpatRasterDataset, numeric, missing-method (subset), 213
 - [, SpatRasterDataset, numeric, numeric-method (subset), 213
 - [, SpatVector, SpatExtent, missing-method (extract), 87
 - [, SpatVector, SpatVector, missing-method (extract), 87
 - [, SpatVector, data.frame, missing-method (subset), 213
 - [, SpatVector, logical, character-method (subset), 213
 - [, SpatVector, logical, logical-method (subset), 213

- [, SpatVector, logical, missing-method (subset), [213](#)
- [, SpatVector, logical, numeric-method (subset), [213](#)
- [, SpatVector, matrix, missing-method (subset), [213](#)
- [, SpatVector, missing, character-method (subset), [213](#)
- [, SpatVector, missing, logical-method (subset), [213](#)
- [, SpatVector, missing, missing-method (subset), [213](#)
- [, SpatVector, missing, numeric-method (subset), [213](#)
- [, SpatVector, numeric, character-method (subset), [213](#)
- [, SpatVector, numeric, logical-method (subset), [213](#)
- [, SpatVector, numeric, missing-method (subset), [213](#)
- [, SpatVector, numeric, numeric-method (subset), [213](#)
- [, SpatVectorCollection, numeric, missing-method (svc), [219](#)
- [<-, SpatExtent, numeric, missing-method (ext), [85](#)
- [<-, SpatRaster, SpatRaster, ANY-method (replace), [181](#)
- [<-, SpatRaster, SpatVector, missing-method (replace), [181](#)
- [<-, SpatRaster, logical, missing-method (replace), [181](#)
- [<-, SpatRaster, missing, missing-method (replace), [181](#)
- [<-, SpatRaster, missing, numeric-method (replace), [181](#)
- [<-, SpatRaster, numeric, missing-method (replace), [181](#)
- [<-, SpatRaster, numeric, numeric-method (replace), [181](#)
- [<-, SpatRasterDataset, numeric, missing-method (sds), [190](#)
- [<-, SpatVector, ANY, ANY-method (vector-attributes), [240](#)
- [<-, SpatVector, ANY, missing-method (vector-attributes), [240](#)
- [<-, SpatVector, missing, ANY-method (vector-attributes), [240](#)
- [<-, SpatVectorCollection, numeric, missing-method (svc), [219](#)
- [<-, SpatRaster, character, missing-method (replace), [181](#)
- [<-, SpatRaster, numeric, missing-method (replace), [181](#)
- [<-, SpatVector, character, missing-method (vector-attributes), [240](#)
- [<-, SpatVector, numeric, missing-method (vector-attributes), [240](#)
- [<-, SpatVectorCollection, numeric, missing-method (svc), [219](#)
- \$(vector-attributes), [240](#)
- \$(SpatExtent-method (ext), [85](#)
- \$(SpatRaster-method (subset), [213](#)
- \$(SpatRasterDataset-method (subset), [213](#)
- \$(SpatVector-method (vector-attributes), [240](#)
- \$<- (vector-attributes), [240](#)
- \$<-, SpatExtent-method (ext), [85](#)
- \$<-, SpatRaster-method (replace), [181](#)
- \$<-, SpatVector-method (vector-attributes), [240](#)
- %in% (match), [133](#)
- %in%, SpatRaster-method (match), [133](#)
- activeCat, [11](#), [18](#), [45](#), [92](#)
- activeCat, SpatRaster-method (activeCat), [18](#)
- activeCat<- (activeCat), [18](#)
- activeCat<-, SpatRaster-method (activeCat), [18](#)
- add, [19](#)
- add<-, [7](#), [17](#)
- add<- (add), [19](#)

- add<- , SpatRaster , SpatRaster-method
(add), 19
- adjacent, 9, 13, 20, 145, 179
- adjacent , SpatRaster-method (adjacent),
20
- adjacent , SpatVector-method (adjacent),
20
- aggregate, 7, 13, 21, 77, 181, 182, 220, 232
- aggregate , SpatRaster-method
(aggregate), 21
- aggregate , SpatVector-method
(aggregate), 21
- align, 15, 23
- align , SpatExtent, numeric-method
(align), 23
- align , SpatExtent, SpatRaster-method
(align), 23
- all (summarize), 215
- all , SpatRaster-method (summarize), 215
- all.equal, 24, 57
- all.equal , SpatRaster , SpatRaster-method
(all.equal), 24
- animate, 25
- animate , SpatRaster-method (animate), 25
- any (summarize), 215
- any , SpatRaster-method (summarize), 215
- app, 7, 17, 26, 30, 57, 110, 124, 125, 133, 135,
167, 169, 170, 186, 215, 217, 221,
257
- app , SpatRaster-method (app), 26
- app , SpatRasterDataset-method (app), 26
- apply, 26
- approx, 28
- approximate, 8, 17, 28, 94
- approximate , SpatRaster-method
(approximate), 28
- area (deprecated), 71
- area , SpatRaster-method (deprecated), 71
- Arith, missing, SpatRaster-method
(Arith-methods), 30
- Arith, numeric, SpatExtent-method
(Arith-methods), 30
- Arith, numeric, SpatRaster-method
(Arith-methods), 30
- Arith, SpatExtent, numeric-method
(Arith-methods), 30
- Arith, SpatExtent, SpatExtent-method
(Arith-methods), 30
- Arith, SpatRaster, missing-method
(Arith-methods), 30
- Arith, SpatRaster, numeric-method
(Arith-methods), 30
- Arith, SpatRaster, SpatRaster-method
(Arith-methods), 30
- Arith, SpatVector, SpatVector-method
(Arith-methods), 30
- Arith-methods, 8, 30, 57
- arrows, 129
- as.array, 9
- as.array (coerce), 53
- as.array , SpatRaster-method (coerce), 53
- as.bool, 8
- as.bool (is.bool), 122
- as.bool , SpatRaster-method (is.bool), 122
- as.character, 31
- as.character , SpatExtent-method
(as.character), 31
- as.character , SpatRaster-method
(as.character), 31
- as.contour, 16
- as.contour (contour), 59
- as.contour , SpatRaster-method (contour),
59
- as.data.frame, 14, 31, 53, 236
- as.data.frame , SpatRaster-method
(as.data.frame), 31
- as.data.frame , SpatVector-method
(as.data.frame), 31
- as.factor, 11
- as.factor (factors), 91
- as.factor , SpatRaster-method (factors),
91
- as.int, 8
- as.int (is.bool), 122
- as.int , SpatRaster-method (is.bool), 122
- as.integer , SpatRaster-method (is.bool),
122
- as.lines, 16
- as.lines (as.spatvector), 34
- as.lines , SpatExtent-method
(as.spatvector), 34
- as.lines , SpatRaster-method
(as.spatvector), 34
- as.lines , SpatVector-method
(as.spatvector), 34
- as.list, 14, 32, 32

- as.list, SpatRaster-method (as.list), 32
- as.list, SpatRasterCollection-method (as.list), 32
- as.list, SpatRasterDataset-method (as.list), 32
- as.list, SpatVector-method (as.list), 32
- as.logical, SpatRaster-method (is.bool), 122
- as.matrix, 9, 32, 236
- as.matrix (coerce), 53
- as.matrix, SpatRaster-method (coerce), 53
- as.numeric, 11
- as.numeric (catalyze), 44
- as.numeric, SpatRaster-method (catalyze), 44
- as.points, 16, 17
- as.points (as.spatvector), 34
- as.points, SpatExtent-method (as.spatvector), 34
- as.points, SpatRaster-method (as.spatvector), 34
- as.points, SpatVector-method (as.spatvector), 34
- as.polygons, 16–18, 53
- as.polygons (as.spatvector), 34
- as.polygons, SpatExtent-method (as.spatvector), 34
- as.polygons, SpatRaster-method (as.spatvector), 34
- as.polygons, SpatVector-method (as.spatvector), 34
- as.raster, 33, 33
- as.raster, SpatRaster-method (as.raster), 33
- as.spatvector, 34
- as.vector (coerce), 53
- as.vector, SpatRaster-method (coerce), 53
- atan2, 36
- atan2, SpatRaster, SpatRaster-method (atan2), 36
- atan_2 (atan2), 36
- atan_2, SpatRaster, SpatRaster-method (atan2), 36
- autocor, 9
- autocor (autocorrelation), 37
- autocor, numeric-method (autocorrelation), 37
- autocor, SpatRaster-method (autocorrelation), 37
- autocorrelation, 37
- axis, 156
- barplot, 16, 38, 39
- barplot, SpatRaster-method (barplot), 38
- blocks (read and write), 176
- blocks, SpatRaster-method (read and write), 176
- boundaries, 9, 39, 150
- boundaries, SpatRaster-method (boundaries), 39
- boxplot, 17, 39, 40, 40, 113, 149
- boxplot, SpatRaster-method (boxplot), 40
- buffer, 13, 41, 121
- buffer, SpatRaster-method (buffer), 41
- buffer, SpatVector-method (buffer), 41
- bxp, 41
- c, 7, 17, 19, 20, 42, 138, 231
- c, SpatRaster-method (c), 42
- c, SpatRasterDataset-method (c), 42
- c, SpatVector-method (c), 42
- c, SpatVectorCollection-method (c), 42
- cartogram, 16, 43, 81
- cartogram, SpatVector-method (cartogram), 43
- catalyze, 11, 19, 44, 92
- catalyze, SpatRaster-method (catalyze), 44
- categories, 117
- categories (factors), 91
- categories, SpatRaster-method (factors), 91
- cats, 11, 19, 45, 59
- cats (factors), 91
- cats, SpatRaster-method (factors), 91
- cellFromRowCol, 11
- cellFromRowCol (xyRowColCell), 254
- cellFromRowCol, SpatRaster, numeric, numeric-method (xyRowColCell), 254
- cellFromRowColCombine, 11
- cellFromRowColCombine (xyRowColCell), 254
- cellFromRowColCombine, SpatRaster, numeric, numeric-method (xyRowColCell), 254
- cellFromXY, 10
- cellFromXY (xyRowColCell), 254

- cellFromXY, SpatRaster, data.frame-method (xyRowColCell), 254
- cellFromXY, SpatRaster, matrix-method (xyRowColCell), 254
- cells, 11, 17, 45, 88
- cells, SpatRaster, missing-method (cells), 45
- cells, SpatRaster, numeric-method (cells), 45
- cells, SpatRaster, SpatExtent-method (cells), 45
- cells, SpatRaster, SpatVector-method (cells), 45
- cellSize, 8, 17, 46, 84
- cellSize, SpatRaster-method (cellSize), 46
- centroids, 12, 48
- centroids, SpatVector-method (centroids), 48
- clamp, 49
- clamp, numeric-method (clamp), 49
- clamp, SpatRaster-method (clamp), 49
- classify, 8, 17, 49, 50, 114, 136, 141, 142, 214, 215
- classify, SpatRaster-method (classify), 50
- clearance, 14
- clearance (width), 245
- clearance, SpatVector-method (width), 245
- click, 14, 16, 52, 82, 192
- click, missing-method (click), 52
- click, SpatRaster-method (click), 52
- click, SpatVector-method (click), 52
- coerce, 33, 53
- colFromCell (xyRowColCell), 254
- colFromCell, SpatRaster, numeric-method (xyRowColCell), 254
- colFromX, 10
- colFromX (xyRowColCell), 254
- colFromX, SpatRaster, numeric-method (xyRowColCell), 254
- colorize, 16, 159, 160
- colorize (RGB), 184
- colorize, SpatRaster-method (RGB), 184
- colors, 54
- coltab (colors), 54
- coltab, SpatRaster-method (colors), 54
- coltab<- (colors), 54
- coltab<-, SpatRaster-method (colors), 54
- combineGeoms, 15, 55
- combineGeoms, SpatVector, SpatVector-method (combineGeoms), 55
- Compare, numeric, SpatRaster-method (Compare-methods), 57
- Compare, SpatExtent, SpatExtent-method (Compare-methods), 57
- Compare, SpatRaster, character-method (Compare-methods), 57
- Compare, SpatRaster, numeric-method (Compare-methods), 57
- Compare, SpatRaster, SpatRaster-method (Compare-methods), 57
- Compare-methods, 8, 57
- compareGeom, 10, 17, 25, 57
- compareGeom, SpatRaster, SpatRaster-method (compareGeom), 57
- concats, 11, 58
- concats, SpatRaster-method (concats), 58
- contour, 16, 59, 59, 60
- contour, SpatRaster-method (contour), 59
- convHull, 13, 60
- convHull, SpatVector-method (convHull), 60
- cor, 149
- costDistance, 9, 61, 111
- costDistance, SpatRaster-method (costDistance), 61
- cov.wt, 127
- cover, 8, 13, 62, 114
- cover, SpatRaster, SpatRaster-method (cover), 62
- cover, SpatVector, SpatVector-method (cover), 62
- cppFunction, 100
- crds, 13, 17, 64, 108, 255
- crds, SpatRaster-method (crds), 64
- crds, SpatVector-method (crds), 64
- crop, 7, 14, 15, 65, 83, 86, 87, 121, 132, 179, 181, 182, 192, 232, 246, 247
- crop, SpatRaster-method (crop), 65
- crop, SpatRasterDataset-method (crop), 65
- crop, SpatVector-method (crop), 65
- crosstab, 8, 66
- crosstab, SpatRaster, missing-method (crosstab), 66
- crs, 10, 13, 35, 67, 128, 164–166, 208, 238

- crs, sf-method (crs), 67
- crs, SpatRaster-method (crs), 67
- crs, SpatRasterDataset-method (crs), 67
- crs, SpatVector-method (crs), 67
- crs, SpatVectorProxy-method (crs), 67
- crs<- (crs), 67
- crs<-, SpatRaster, ANY-method (crs), 67
- crs<-, SpatRaster-method (crs), 67
- crs<-, SpatVector, ANY-method (crs), 67
- crs<-, SpatVector-method (crs), 67
- cut, 39
- data.frame, 32, 177, 236
- datatype (geomtype), 109
- datatype, SpatVector-method (geomtype), 109
- Date, 226
- deepcopy, 69
- deepcopy, SpatRaster-method (deepcopy), 69
- deepcopy, SpatVector-method (deepcopy), 69
- delaunay, 12
- delaunay (voronoi), 241
- delaunay, SpatVector-method (voronoi), 241
- deldir, 241
- densify, 70
- densify, SpatVector-method (densify), 70
- density, 16, 71
- density, SpatRaster-method (density), 71
- deprecated, 71
- depth, 72, 227
- depth, SpatRaster-method (depth), 72
- depth<- (depth), 72
- depth<-, SpatRaster-method (depth), 72
- describe, 72, 107, 190, 191
- describe, character-method (describe), 72
- diff, 73
- diff, SpatRaster-method (diff), 73
- dim (dimensions), 74
- dim, SpatRaster-method (dimensions), 74
- dim, SpatRasterDataset-method (dimensions), 74
- dim, SpatVector-method (dimensions), 74
- dim, SpatVectorProxy-method (dimensions), 74
- dim<-, SpatRaster-method (dimensions), 74
- dimensions, 74
- direction, 9, 76
- direction, SpatRaster-method (direction), 76
- disagg, 7, 13, 17, 22, 65, 77, 181, 182
- disagg, SpatRaster-method (disagg), 77
- disagg, SpatVector-method (disagg), 77
- distance, 9, 17, 42, 62, 76, 78, 111
- distance, matrix, matrix-method (distance), 78
- distance, matrix, missing-method (distance), 78
- distance, SpatRaster, missing-method (distance), 78
- distance, SpatRaster, SpatVector-method (distance), 78
- distance, SpatVector, ANY-method (distance), 78
- distance, SpatVector, SpatVector-method (distance), 78
- dots, 16, 80
- dots, SpatVector-method (dots), 80
- draw, 15–17, 24, 53, 81, 160, 258
- draw, character-method (draw), 81
- draw, missing-method (draw), 81
- emptyGeoms (topology), 229
- emptyGeoms, SpatVector-method (topology), 229
- erase, 13, 15, 56, 82, 220
- erase, SpatVector, missing-method (erase), 82
- erase, SpatVector, SpatExtent-method (erase), 82
- erase, SpatVector, SpatVector-method (erase), 82
- expanse, 8, 13, 17, 47, 83
- expanse, SpatRaster-method (expanse), 83
- expanse, SpatVector-method (expanse), 83
- ext, 10, 13, 15, 17, 24, 65, 75, 85, 87, 203
- ext, Extent-method (ext), 85
- ext, missing-method (ext), 85
- ext, numeric-method (ext), 85
- ext, Raster-method (ext), 85
- ext, sf-method (ext), 85
- ext, SpatExtent-method (ext), 85
- ext, Spatial-method (ext), 85
- ext, SpatRaster-method (ext), 85
- ext, SpatRasterCollection-method (ext), 85

- ext, SpatRasterDataset-method (ext), 85
- ext, SpatVector-method (ext), 85
- ext, SpatVectorProxy-method (ext), 85
- ext<- (ext), 85
- ext<-, SpatRaster, numeric-method (ext), 85
- ext<-, SpatRaster, SpatExtent-method (ext), 85
- extend, 7, 65, 86, 181, 232, 247
- extend, SpatExtent-method (extend), 86
- extend, SpatRaster-method (extend), 86
- extract, 9, 14, 18, 87, 110, 194, 256, 257
- extract, SpatRaster, data.frame-method (extract), 87
- extract, SpatRaster, matrix-method (extract), 87
- extract, SpatRaster, numeric-method (extract), 87
- extract, SpatRaster, sf-method (extract), 87
- extract, SpatRaster, SpatExtent-method (extract), 87
- extract, SpatRaster, SpatVector-method (extract), 87
- extract, SpatVector, SpatVector-method (extract), 87
- extremes, 90
- factor, 91
- factors, 91
- fileBlocksize (read and write), 176
- filled.contour, 59, 60
- fillHoles, 13, 14, 93, 106
- fillHoles, SpatVector-method (fillHoles), 93
- fillTime, 11, 29, 94, 138
- fillTime, SpatRaster-method (fillTime), 94
- flip, 7, 14, 95, 183, 199, 230
- flip, SpatRaster-method (flip), 95
- flip, SpatVector-method (flip), 95
- focal, 9, 28, 29, 40, 96, 98–103, 110, 150, 224
- focal, SpatRaster-method (focal), 96
- focal3D, 97, 98
- focal3D, SpatRaster-method (focal3D), 98
- focalCor, 9, 17, 97, 99
- focalCor, SpatRaster-method (focalCor), 99
- focalCpp, 9, 97, 100
- focalCpp, SpatRaster-method (focalCpp), 100
- focalMat, 97, 102
- focalReg, 9, 97, 100, 103
- focalReg, SpatRaster-method (focalReg), 103
- focalValues, 97, 101, 103, 104, 237
- focalValues, SpatRaster-method (focalValues), 104
- free_RAM (mem), 135
- freq, 8, 67, 105
- freq, SpatRaster-method (freq), 105
- gaps, 14, 106, 198, 200, 229
- gaps, SpatVector, SpatExtent-method (gaps), 106
- gaps, SpatVector-method (gaps), 106
- gdal, 106, 172
- gdalCache (gdal), 106
- geom, 13, 32, 33, 64, 107, 112, 237, 238
- geom, SpatVector-method (geom), 107
- geomtype, 109
- geomtype, Spatial-method (geomtype), 109
- geomtype, SpatVector-method (geomtype), 109
- geomtype, SpatVectorProxy-method (geomtype), 109
- getGDALconfig (gdal), 106
- global, 8, 17, 84, 110, 127, 166, 218, 257
- global, SpatRaster-method (global), 110
- gridDistance, 9, 62, 111
- gridDistance, SpatRaster-method (gridDistance), 111
- has.colors (colors), 54
- has.colors, SpatRaster-method (colors), 54
- has.RGB (RGB), 184
- has.RGB, SpatRaster-method (RGB), 184
- hasMinMax (extremes), 90
- hasMinMax, SpatRaster-method (extremes), 90
- hasValues (sources), 201
- hasValues, SpatRaster-method (sources), 201
- head (head and tail), 112
- head and tail, 112
- head, SpatRaster-method (head and tail), 112

- head, SpatVector-method (head and tail), 112
- hist, 16, 39, 41, 113, 113, 149
- hist, SpatRaster-method (hist), 113
- ifel, 30, 57, 114
- ifel, SpatRaster-method (ifel), 114
- ifelse, 114
- image, 16, 115, 115, 157
- image, SpatRaster-method (image), 115
- impose, 116
- impose, SpatRasterCollection-method (impose), 116
- inext (inset), 119
- inext, SpatVector-method (inset), 119
- init, 8, 196, 197
- init (initialize), 116
- init, SpatRaster-method (initialize), 116
- initialize, 116
- inMemory, 10, 12
- inMemory (sources), 201
- inMemory, SpatRaster-method (sources), 201
- inplace, 117
- inset, 16, 119, 145, 183, 187
- inset, SpatRaster-method (inset), 119
- inset, SpatVector-method (inset), 119
- interpolate, 9
- interpolate (Spatial interpolation), 203
- interpolate, SpatRaster-method (Spatial interpolation), 203
- intersect, 13, 15, 56, 65, 83, 121, 179, 192, 232
- intersect, SpatExtent, SpatExtent-method (intersect), 121
- intersect, SpatExtent, SpatVector-method (intersect), 121
- intersect, SpatVector, SpatExtent-method (intersect), 121
- intersect, SpatVector, SpatVector-method (intersect), 121
- is.bool, 122
- is.bool, SpatRaster-method (is.bool), 122
- is.factor, 11
- is.factor (factors), 91
- is.factor, SpatRaster-method (factors), 91
- is.finite, SpatRaster-method (Compare-methods), 57
- is.infinite, SpatRaster-method (Compare-methods), 57
- is.int (is.bool), 122
- is.int, SpatRaster-method (is.bool), 122
- is.lines (geomtype), 109
- is.lines, SpatVector-method (geomtype), 109
- is.lonlat, 10, 13, 17, 123
- is.lonlat, SpatRaster-method (is.lonlat), 123
- is.lonlat, SpatVector-method (is.lonlat), 123
- is.na, SpatRaster-method (Compare-methods), 57
- is.nan, SpatRaster-method (Compare-methods), 57
- is.points (geomtype), 109
- is.points, SpatVector-method (geomtype), 109
- is.polygons (geomtype), 109
- is.polygons, SpatVector-method (geomtype), 109
- is.related (relate), 178
- is.related, SpatExtent, SpatVector-method (relate), 178
- is.related, SpatVector, SpatExtent-method (relate), 178
- is.related, SpatVector, SpatVector-method (relate), 178
- is.valid, 14
- is.valid (valid), 235
- is.valid, SpatVector-method (valid), 235
- isFALSE, SpatRaster-method (is.bool), 122
- isTRUE, 245
- isTRUE, SpatRaster-method (is.bool), 122
- lapp, 7, 17, 27, 124, 170, 186
- lapp, SpatRaster-method (lapp), 124
- lapp, SpatRasterDataset-method (lapp), 124
- lapply, 186
- layerCor, 8, 17, 100, 126
- layerCor, SpatRaster-method (layerCor), 126
- legend, 156
- length, 15
- length (dimensions), 74
- length, SpatRasterCollection-method (dimensions), 74

- length, SpatRasterDataset-method (dimensions), [74](#)
- length, SpatVector-method (dimensions), [74](#)
- length, SpatVectorCollection-method (dimensions), [74](#)
- levels, [11](#)
- levels (factors), [91](#)
- levels, SpatRaster-method (factors), [91](#)
- levels<- (factors), [91](#)
- levels<-, SpatRaster-method (factors), [91](#)
- linearUnits, [13](#), [127](#)
- linearUnits, SpatRaster-method (linearUnits), [127](#)
- linearUnits, SpatVector-method (linearUnits), [127](#)
- lines, [16](#), [128](#), [155](#), [157](#)
- lines, leaflet-method (plet), [153](#)
- lines, SpatExtent-method (lines), [128](#)
- lines, SpatRaster-method (lines), [128](#)
- lines, SpatVector-method (lines), [128](#)
- log (Math-methods), [134](#)
- log, SpatRaster-method (Math-methods), [134](#)
- Logic, logical, SpatRaster-method (Compare-methods), [57](#)
- Logic, numeric, SpatRaster-method (Compare-methods), [57](#)
- Logic, SpatRaster, logical-method (Compare-methods), [57](#)
- Logic, SpatRaster, numeric-method (Compare-methods), [57](#)
- Logic, SpatRaster, SpatRaster-method (Compare-methods), [57](#)
- Logic-methods, [8](#)
- Logic-methods (Compare-methods), [57](#)
- longnames (names), [142](#)
- longnames, SpatRaster-method (names), [142](#)
- longnames, SpatRasterDataset-method (names), [142](#)
- longnames<- (names), [142](#)
- longnames<- , SpatRaster-method (names), [142](#)
- longnames<- , SpatRasterDataset-method (names), [142](#)
- make.unique, [143](#)
- makeNodes, [14](#)
- makeNodes (topology), [229](#)
- makeNodes, SpatVector-method (topology), [229](#)
- makeTiles, [129](#), [242](#)
- makeTiles, SpatRaster-method (makeTiles), [129](#)
- makeValid, [14](#)
- makeValid (valid), [235](#)
- makeValid, SpatVector-method (valid), [235](#)
- makeVRT, [130](#), [242](#)
- mask, [8](#), [65](#), [83](#), [114](#), [132](#), [174](#)
- mask, SpatRaster, sf-method (mask), [132](#)
- mask, SpatRaster, SpatRaster-method (mask), [132](#)
- mask, SpatRaster, SpatVector-method (mask), [132](#)
- mask, SpatVector, sf-method (mask), [132](#)
- mask, SpatVector, SpatVector-method (mask), [132](#)
- match, [133](#), [133](#)
- match, SpatRaster-method (match), [133](#)
- math, [125](#)
- math (Math-methods), [134](#)
- Math, SpatExtent-method (Math-methods), [134](#)
- Math, SpatRaster-method (Math-methods), [134](#)
- math, SpatRaster-method (Math-methods), [134](#)
- Math-methods, [8](#), [15](#), [134](#)
- Math2, SpatExtent-method (Math-methods), [134](#)
- Math2, SpatRaster-method (Math-methods), [134](#)
- Math2, SpatVector-method (Math-methods), [134](#)
- Math2-methods (Math-methods), [134](#)
- max (summarize), [215](#)
- max, SpatRaster-method (summarize), [215](#)
- mean (summarize), [215](#)
- mean, SpatExtent-method (summarize), [215](#)
- mean, SpatRaster-method (summarize), [215](#)
- mean, SpatVector-method (summarize), [215](#)
- median (summarize), [215](#)
- median, SpatRaster-method (summarize), [215](#)
- median, SpatVector-method (summarize), [215](#)
- mem, [135](#)

- mem_info, [12](#), [147](#), [250](#)
- mem_info (mem), [135](#)
- merge, [7](#), [14](#), [87](#), [136](#), [136](#), [139](#), [140](#), [211](#), [232](#)
- merge, SpatExtent, SpatExtent-method (merge), [136](#)
- merge, SpatRaster, SpatRaster-method (merge), [136](#)
- merge, SpatRasterCollection, missing-method (merge), [136](#)
- merge, SpatVector, data.frame-method (merge), [136](#)
- merge, SpatVector, SpatVector-method (merge), [136](#)
- mergeLines, [14](#)
- mergeLines (topology), [229](#)
- mergeLines, SpatVector-method (topology), [229](#)
- mergeTime, [11](#), [137](#)
- mergeTime, SpatRasterDataset-method (mergeTime), [137](#)
- min (summarize), [215](#)
- min, SpatRaster-method (summarize), [215](#)
- minmax, [9](#)
- minmax (extremes), [90](#)
- minmax, SpatRaster-method (extremes), [90](#)
- minRect, [246](#)
- minRect (convHull), [60](#)
- minRect, SpatVector-method (convHull), [60](#)
- modal, [138](#), [215](#), [217](#)
- modal, SpatRaster-method (modal), [138](#)
- mosaic, [7](#), [137](#), [139](#), [211](#), [232](#)
- mosaic, SpatRaster, SpatRaster-method (mosaic), [139](#)
- mosaic, SpatRasterCollection, missing-method (mosaic), [139](#)

- na.omit, [12](#), [140](#)
- na.omit, SpatVector-method (na.omit), [140](#)
- NAflag, [10](#), [17](#), [141](#)
- NAflag, SpatRaster-method (NAflag), [141](#)
- NAflag<- (NAflag), [141](#)
- NAflag<-, SpatRaster-method (NAflag), [141](#)
- name (names), [142](#)
- name<- (names), [142](#)
- names, [10](#), [12](#), [13](#), [113](#), [142](#), [161](#), [234](#)
- names, SpatRaster-method (names), [142](#)
- names, SpatRasterDataset-method (names), [142](#)
- names, SpatVector-method (names), [142](#)
- names, SpatVectorCollection-method (names), [142](#)
- names, SpatVectorProxy-method (names), [142](#)
- names<- (names), [142](#)
- names<-, SpatRaster-method (names), [142](#)
- names<-, SpatRasterDataset-method (names), [142](#)
- names<-, SpatVector-method (names), [142](#)
- names<-, SpatVectorCollection-method (names), [142](#)
- ncell, [10](#), [255](#)
- ncell (dimensions), [74](#)
- ncell, ANY-method (dimensions), [74](#)
- ncell, SpatRaster-method (dimensions), [74](#)
- ncell, SpatRasterDataset-method (dimensions), [74](#)
- ncol, [10](#), [13](#)
- ncol (dimensions), [74](#)
- ncol, SpatRaster-method (dimensions), [74](#)
- ncol, SpatRasterDataset-method (dimensions), [74](#)
- ncol, SpatVector-method (dimensions), [74](#)
- ncol<- (dimensions), [74](#)
- ncol<-, SpatRaster, numeric-method (dimensions), [74](#)
- ncvar_def, [249](#)
- nearby, [13](#), [21](#), [179](#)
- nearby (nearest), [144](#)
- nearby, SpatVector-method (nearest), [144](#)
- nearest, [13](#), [144](#)
- nearest, SpatVector-method (nearest), [144](#)
- nlyr, [10](#), [17](#)
- nlyr (dimensions), [74](#)
- nlyr, SpatRaster-method (dimensions), [74](#)
- nlyr, SpatRasterDataset-method (dimensions), [74](#)
- nlyr<- (dimensions), [74](#)
- nlyr<-, SpatRaster, numeric-method (dimensions), [74](#)
- north, [16](#), [145](#), [187](#)
- not.na, [146](#)
- not.na, SpatRaster-method (not.na), [146](#)
- nrow, [10](#), [13](#)
- nrow (dimensions), [74](#)
- nrow, SpatRaster-method (dimensions), [74](#)
- nrow, SpatRasterDataset-method (dimensions), [74](#)

- nrow, SpatVector-method (dimensions), 74
- nrow<- (dimensions), 74
- nrow<-, SpatRaster, numeric-method (dimensions), 74
- nsrc (dimensions), 74
- nsrc, SpatRaster-method (dimensions), 74
- options, 147
- origin, 10, 148
- origin, SpatRaster-method (origin), 148
- origin<- (origin), 148
- origin<-, SpatRaster-method (origin), 148
- PackedSpatRaster-class (SpatRaster-class), 206
- PackedSpatVector-class (SpatVector-class), 209
- pairs, 17, 41, 113, 149, 149
- pairs, SpatRaster-method (pairs), 149
- par, 129, 154
- patches, 9, 17, 40, 150
- patches, SpatRaster-method (patches), 150
- perim, 13, 151
- perim, SpatVector-method (perim), 151
- perimeter (perim), 151
- perimeter, SpatVector-method (perim), 151
- persp, 16, 152, 152
- persp, SpatRaster-method (persp), 152
- plet, 153
- plet, missing-method (plet), 153
- plet, SpatRaster-method (plet), 153
- plet, SpatVector-method (plet), 153
- plet, SpatVectorCollection-method (plet), 153
- plot, 16, 25, 26, 44, 60, 71, 81, 115, 145, 154, 155, 157, 159, 160, 184, 187, 225, 258
- plot, SpatExtent, missing-method (plot), 155
- plot, SpatRaster, character-method (plot), 155
- plot, SpatRaster, missing-method (plot), 155
- plot, SpatRaster, numeric-method (plot), 155
- plot, SpatRaster, SpatRaster-method (scatterplot), 189
- plot, SpatVector, character-method (plot), 155
- plot, SpatVector, data.frame-method (plot), 155
- plot, SpatVector, missing-method (plot), 155
- plot, SpatVector, numeric-method (plot), 155
- plot, SpatVectorProxy, missing-method (plot), 155
- plotRGB, 16, 159, 184
- plotRGB, SpatRaster-method (plotRGB), 159
- points, 16, 81, 129, 157
- points (lines), 128
- points, leaflet-method (plet), 153
- points, SpatExtent-method (lines), 128
- points, SpatVector-method (lines), 128
- polys, 16, 157
- polys (lines), 128
- polys, SpatExtent-method (lines), 128
- polys, SpatVector-method (lines), 128
- POSIXlt, 226
- predict, 9, 51, 161, 203, 204
- predict, SpatRaster-method (predict), 161
- prod (summarize), 215
- prod, SpatRaster-method (summarize), 215
- project, 7, 10, 12, 17, 67, 151, 164, 181, 182
- project, matrix-method (project), 164
- project, SpatRaster-method (project), 164
- project, SpatVector-method (project), 164
- quantile, 8, 18, 166, 218
- quantile, SpatRaster-method (quantile), 166
- quantile, SpatVector-method (quantile), 166
- query, 167
- query, SpatVectorProxy-method (query), 167
- rainbow, 39
- range (summarize), 215
- range, SpatRaster-method (summarize), 215
- rapp, 7, 169, 194
- rapp, SpatRaster-method (rapp), 169
- rast, 7, 16, 17, 170, 206
- rast, ANY-method (rast), 170
- rast, array-method (rast), 170
- rast, character-method (rast), 170
- rast, data.frame-method (rast), 170
- rast, list-method (rast), 170

- rast,matrix-method (rast), 170
- rast,missing-method (rast), 170
- rast,PackedSpatRaster-method (rast), 170
- rast,SpatExtent-method (rast), 170
- rast,SpatRaster-method (rast), 170
- rast,SpatRasterDataset-method (rast), 170
- rast,SpatVector-method (rast), 170
- rast,stars-method (rast), 170
- rast,stars_proxy-method (rast), 170
- rasterImage, 33, 160
- rasterize, 16, 173, 175
- rasterize,matrix,SpatRaster-method (rasterize), 173
- rasterize,sf,SpatRaster-method (rasterize), 173
- rasterize,SpatVector,SpatRaster-method (rasterize), 173
- rasterizeGeom, 16, 175
- rasterizeGeom,SpatVector,SpatRaster-method (rasterizeGeom), 175
- RasterSource (SpatRaster-class), 206
- RasterSource-class (SpatRaster-class), 206
- Rcpp_RasterSource-class (SpatRaster-class), 206
- Rcpp_SpatCategories-class (SpatRaster-class), 206
- Rcpp_SpatExtent-class (SpatExtent-class), 202
- Rcpp_SpatRaster-class (SpatRaster-class), 206
- Rcpp_SpatVector-class (SpatVector-class), 209
- read and write, 176
- readStart, 11
- readStart (read and write), 176
- readStart,SpatRaster-method (read and write), 176
- readStart,SpatRasterDataset-method (read and write), 176
- readStop, 11
- readStop (read and write), 176
- readStop,SpatRaster-method (read and write), 176
- readStop,SpatRasterDataset-method (read and write), 176
- readValues (read and write), 176
- readValues,SpatRaster-method (read and write), 176
- rectify, 177
- rectify,SpatRaster-method (rectify), 177
- relate, 13, 21, 121, 145, 178
- relate,SpatExtent,SpatExtent-method (relate), 178
- relate,SpatExtent,SpatVector-method (relate), 178
- relate,SpatVector,missing-method (relate), 178
- relate,SpatVector,SpatExtent-method (relate), 178
- relate,SpatVector,SpatVector-method (relate), 178
- removeDupNodes, 14
- removeDupNodes (topology), 229
- removeDupNodes,SpatVector-method (topology), 229
- rep, 180, 180
- rep,SpatRaster-method (rep), 180
- replace, 181, 181
- res, 10, 172
- res (dimensions), 74
- res,SpatRaster-method (dimensions), 74
- res,SpatRasterDataset-method (dimensions), 74
- res<- (dimensions), 74
- res<-,SpatRaster,numeric-method (dimensions), 74
- res<-,SpatRaster-method (dimensions), 74
- resample, 7, 23, 65, 77, 116, 166, 178, 181
- resample,SpatRaster,SpatRaster-method (resample), 181
- rescale, 14, 44, 120, 183, 210
- rescale,SpatRaster-method (rescale), 183
- rescale,SpatVector-method (rescale), 183
- rev (flip), 95
- rev,SpatRaster-method (flip), 95
- RGB, 160, 184
- RGB,SpatRaster-method (RGB), 184
- RGB<- (RGB), 184
- RGB<-,SpatRaster-method (RGB), 184
- rotate, 7, 95, 183, 185, 199, 230
- rotate,SpatRaster-method (rotate), 185
- round, 39
- round (Math-methods), 134
- round,SpatRaster-method (Math-methods),

- 134
- round, SpatVector-method (Math-methods), 134
- rowColFromCell, 10
- rowColFromCell (xyRowColCell), 254
- rowColFromCell, SpatRaster, numeric-method (xyRowColCell), 254
- rowFromCell (xyRowColCell), 254
- rowFromCell, SpatRaster, numeric-method (xyRowColCell), 254
- rowFromY, 10
- rowFromY (xyRowColCell), 254
- rowFromY, SpatRaster, numeric-method (xyRowColCell), 254
- runif, 116

- sapp, 7, 125, 186
- sapp, SpatRaster-method (sapp), 186
- sapp, SpatRasterDataset-method (sapp), 186
- saveRDS, 195, 196
- saveRDS (serialize), 195
- saveRDS, SpatRaster-method (serialize), 195
- saveRDS, SpatVector-method (serialize), 195
- sbar, 16, 120, 145, 157, 187
- scale, 8, 188, 188, 189
- scale, SpatRaster-method (scale), 188
- scatterplot, 189
- sds, 12, 172, 190, 212
- sds, array-method (sds), 190
- sds, character-method (sds), 190
- sds, list-method (sds), 190
- sds, missing-method (sds), 190
- sds, SpatRaster-method (sds), 190
- sds, stars-method (sds), 190
- sds, stars_proxy-method (sds), 190
- segregate, 8, 17, 191
- segregate, SpatRaster-method (segregate), 191
- sel, 14, 16, 192
- sel, SpatRaster-method (sel), 192
- sel, SpatVector-method (sel), 192
- selectHighest, 193
- selectHighest, SpatRaster-method (selectHighest), 193
- selectRange, 7, 18, 169, 170, 194
- selectRange, SpatRaster-method (selectRange), 194
- serialize, 195, 195, 196
- serialize, SpatRaster-method (serialize), 195
- serialize, SpatVector-method (serialize), 195
- set.cats, 11, 92
- set.cats (inplace), 117
- set.cats, SpatRaster-method (inplace), 117
- set.crs (inplace), 117
- set.crs, SpatRaster-method (inplace), 117
- set.crs, SpatVector-method (inplace), 117
- set.ext, 85
- set.ext (inplace), 117
- set.ext, SpatRaster-method (inplace), 117
- set.ext, SpatVector-method (inplace), 117
- set.names, 142
- set.names (inplace), 117
- set.names, SpatRaster-method (inplace), 117
- set.names, SpatRasterDataset-method (inplace), 117
- set.names, SpatVector-method (inplace), 117
- set.names, SpatVectorCollection-method (inplace), 117
- set.values, 195
- set.values (inplace), 117
- set.values, SpatRaster-method (inplace), 117
- setCats (inplace), 117
- setCats, SpatRaster-method (inplace), 117
- setGDALconfig (gdal), 106
- setMinMax, 9
- setMinMax (extremes), 90
- setMinMax, SpatRaster-method (extremes), 90
- setValues, 9, 196
- setValues, SpatRaster, ANY-method (setValues), 196
- setValues, SpatRaster-method (setValues), 196
- setValues, SpatVector, ANY-method (setValues), 196
- setValues, SpatVector-method (setValues), 196

- shade, [9](#), [197](#)
- sharedPaths, [14](#), [56](#), [106](#), [198](#), [200](#), [229](#)
- sharedPaths, SpatVector-method (sharedPaths), [198](#)
- shift, [7](#), [14](#), [120](#), [183](#), [185](#), [199](#), [210](#)
- shift, SpatExtent-method (shift), [199](#)
- shift, SpatRaster-method (shift), [199](#)
- shift, SpatVector-method (shift), [199](#)
- show, [112](#)
- show, SpatExtent-method (SpatExtent-class), [202](#)
- show, SpatRaster-method (SpatRaster-class), [206](#)
- show, SpatVector-method (SpatVector-class), [209](#)
- simplifyGeom, [14](#), [200](#), [229](#)
- simplifyGeom, SpatVector-method (simplifyGeom), [200](#)
- size (dimensions), [74](#)
- size, SpatRaster-method (dimensions), [74](#)
- snap, [14](#)
- snap (topology), [229](#)
- snap, SpatVector-method (topology), [229](#)
- sort, [201](#)
- sort, SpatRaster-method (sort), [201](#)
- sources, [10](#), [12](#), [201](#)
- sources, SpatRaster-method (sources), [201](#)
- sources, SpatRasterCollection-method (sources), [201](#)
- sources, SpatVector-method (sources), [201](#)
- sources, SpatVectorProxy-method (sources), [201](#)
- SpatCategories (SpatRaster-class), [206](#)
- SpatCategories-class (SpatRaster-class), [206](#)
- SpatExtent, [86](#), [160](#), [190](#)
- SpatExtent (SpatExtent-class), [202](#)
- SpatExtent-class, [202](#)
- Spatial interpolation, [203](#)
- SpatRaster (SpatRaster-class), [206](#)
- SpatRaster-class, [206](#)
- spatSample, [9](#), [17](#), [207](#), [218](#)
- spatSample, SpatExtent-method (spatSample), [207](#)
- spatSample, SpatRaster-method (spatSample), [207](#)
- spatSample, SpatVector-method (spatSample), [207](#)
- SpatVector (SpatVector-class), [209](#)
- SpatVector-class, [209](#)
- spin, [14](#), [185](#), [209](#)
- spin, SpatVector-method (spin), [209](#)
- split, [210](#)
- split, SpatRaster-method (split), [210](#)
- split, SpatVector-method (split), [210](#)
- sprc, [211](#)
- sprc, list-method (sprc), [211](#)
- sprc, missing-method (sprc), [211](#)
- sprc, SpatRaster-method (sprc), [211](#)
- sqrt (Math-methods), [134](#)
- sqrt, SpatRaster-method (Math-methods), [134](#)
- stdev (summarize), [215](#)
- stdev, SpatRaster-method (summarize), [215](#)
- stretch, [8](#), [212](#)
- stretch, SpatRaster-method (stretch), [212](#)
- subset, [7](#), [17](#), [169](#), [213](#)
- subset, SpatRaster-method (subset), [213](#)
- subset, SpatVector-method (subset), [213](#)
- subst, [8](#), [51](#), [214](#)
- subst, SpatRaster-method (subst), [214](#)
- sum (summarize), [215](#)
- sum, SpatRaster-method (summarize), [215](#)
- summarize, [215](#)
- summary, [8](#), [218](#), [218](#)
- Summary, SpatExtent-method (summary), [218](#)
- Summary, SpatRaster-method (summary), [218](#)
- summary, SpatRaster-method (summary), [218](#)
- Summary, SpatVector-method (summary), [218](#)
- summary, SpatVector-method (summary), [218](#)
- Summary-methods, [8](#), [18](#)
- Summary-methods (summarize), [215](#)
- svc, [15](#), [219](#)
- svc, list-method (svc), [219](#)
- svc, missing-method (svc), [219](#)
- svc, sf-method (svc), [219](#)
- svc, SpatVector-method (svc), [219](#)
- syndif, [13](#), [220](#)
- syndif, SpatVector, SpatVector-method (syndif), [220](#)
- t, [7](#), [14](#), [183](#), [210](#)
- t (transpose), [230](#)
- t, SpatRaster-method (transpose), [230](#)
- t, SpatVector-method (transpose), [230](#)
- tail (head and tail), [112](#)

- tail, SpatRaster-method (head and tail), 112
- tail, SpatVector-method (head and tail), 112
- tapp, 7, 18, 27, 124, 125, 170, 186, 194, 220
- tapp, SpatRaster-method (tapp), 220
- tapply, 220
- terra (terra-package), 6
- terra-package, 6
- terrain, 9, 186, 197, 198, 222
- terrain, SpatRaster-method (terrain), 222
- terraOptions, 12, 228
- terraOptions (options), 147
- text, 16, 192, 225, 225
- text, SpatRaster-method (text), 225
- text, SpatVector-method (text), 225
- tighten, 226
- tighten, SpatRaster-method (tighten), 226
- tighten, SpatRasterDataset-method (tighten), 226
- tiles, 242
- tiles (makeTiles), 129
- tiles, SpatRaster-method (makeTiles), 129
- time, 11, 72, 221, 226, 234
- time, SpatRaster-method (time), 226
- time<- (time), 226
- time<-, SpatRaster-method (time), 226
- timeInfo (time), 226
- timeInfo, SpatRaster-method (time), 226
- tmpFiles, 12, 228
- topology, 106, 198, 229
- trans, 95
- trans (transpose), 230
- trans, SpatRaster-method (transpose), 230
- transpose, 230
- Trig, 36
- trim, 7, 231
- trim, SpatRaster-method (trim), 231
- union, 13, 15, 56, 121, 231
- union, SpatExtent, SpatExtent-method (union), 231
- union, SpatVector, missing-method (union), 231
- union, SpatVector, SpatExtent-method (union), 231
- union, SpatVector, SpatVector-method (union), 231
- unique, 8, 12, 233, 233
- unique, SpatRaster, ANY-method (unique), 233
- unique, SpatRaster-method (unique), 233
- unique, SpatVector, ANY-method (unique), 233
- unique, SpatVector-method (unique), 233
- units, 234
- units, SpatRaster-method (units), 234
- units, SpatRasterDataset-method (units), 234
- units<- (units), 234
- units<-, SpatRaster-method (units), 234
- units<-, SpatRasterDataset-method (units), 234
- valid, 235
- values, 9, 14, 18, 89, 181, 197, 236, 240
- values, SpatRaster-method (values), 236
- values, SpatVector-method (values), 236
- values<-, 9, 14
- values<- (setValues), 196
- values<-, SpatRaster, ANY-method (setValues), 196
- values<-, SpatVector, ANY-method (setValues), 196
- values<-, SpatVector, data.frame-method (setValues), 196
- values<-, SpatVector, matrix-method (setValues), 196
- values<-, SpatVector, NULL-method (setValues), 196
- varnames (names), 142
- varnames, SpatRaster-method (names), 142
- varnames, SpatRasterDataset-method (names), 142
- varnames<- (names), 142
- varnames<-, SpatRaster-method (names), 142
- varnames<-, SpatRasterDataset-method (names), 142
- vect, 12, 16, 17, 168, 172, 209, 237
- vect, character-method (vect), 237
- vect, data.frame-method (vect), 237
- vect, list-method (vect), 237
- vect, matrix-method (vect), 237
- vect, missing-method (vect), 237
- vect, PackedSpatVector-method (vect), 237
- vect, sf-method (vect), 237
- vect, sfc-method (vect), 237

- vect, Spatial-method (vect), 237
- vect, XY-method (vect), 237
- vector-attributes, 240
- vector_layers, 12, 241, 251
- voronoi, 12, 241
- voronoi, SpatVector-method (voronoi), 241
- vrt, 130, 131, 137, 242
- vrt, character-method (vrt), 242

- weighted.mean, 127, 243, 243
- weighted.mean, SpatRaster, numeric-method (weighted.mean), 243
- weighted.mean, SpatRaster, SpatRaster-method (weighted.mean), 243
- where, 244
- which, 244, 245
- which.lyr, 8, 217, 245
- which.lyr, SpatRaster-method (which.lyr), 245
- which.max (summarize), 215
- which.max, SpatRaster-method (summarize), 215
- which.min (summarize), 215
- which.min, SpatRaster-method (summarize), 215
- width, 14, 245
- width, SpatVector-method (width), 245
- window, 246
- window, SpatRaster-method (window), 246
- window<- (window), 246
- window<-, SpatRaster-method (window), 246
- wrap, 6, 172, 195, 247
- wrap, Spatial-method (wrap), 247
- wrap, SpatRaster-method (wrap), 247
- wrap, SpatVector-method (wrap), 247
- writeCDF, 11, 248, 250
- writeCDF, SpatRaster-method (writeCDF), 248
- writeCDF, SpatRasterDataset-method (writeCDF), 248
- writeRaster, 6, 11, 22, 27, 29, 36, 40, 42, 44, 47, 49, 51, 59, 62, 63, 65, 73, 76, 77, 79, 87, 94–96, 99, 101, 103, 111, 114, 116, 117, 122, 124, 130, 132, 135, 136, 138–140, 146, 147, 150, 162, 165, 167, 169, 172, 174, 175, 177, 178, 182, 184–186, 192, 194, 195, 197, 199, 201, 204, 213, 215, 217, 221, 222, 230, 231, 243, 249, 249, 256
- writeRaster, SpatRaster, character-method (writeRaster), 249
- writeStart, 11
- writeStart (read and write), 176
- writeStart, SpatRaster, character-method (read and write), 176
- writeStop, 12
- writeStop (read and write), 176
- writeStop, SpatRaster-method (read and write), 176
- writeValues, 11
- writeValues (read and write), 176
- writeValues, SpatRaster, vector-method (read and write), 176
- writeVector, 12, 251
- writeVector, SpatVector, character-method (writeVector), 251

- xFromCell, 10
- xFromCell (xyRowColCell), 254
- xFromCell, SpatRaster, numeric-method (xyRowColCell), 254
- xFromCol, 10
- xFromCol (xyRowColCell), 254
- xFromCol, SpatRaster, missing-method (xyRowColCell), 254
- xFromCol, SpatRaster, numeric-method (xyRowColCell), 254
- xmax, 10
- xmax (xmin), 252
- xmax, SpatExtent-method (xmin), 252
- xmax, SpatRaster-method (xmin), 252
- xmax, SpatVector-method (xmin), 252
- xmax<- (xmin), 252
- xmax<-, SpatExtent, numeric-method (xmin), 252
- xmax<-, SpatRaster, numeric-method (xmin), 252
- xmin, 10, 252
- xmin, SpatExtent-method (xmin), 252
- xmin, SpatRaster-method (xmin), 252
- xmin, SpatVector-method (xmin), 252
- xmin<- (xmin), 252
- xmin<-, SpatExtent, numeric-method (xmin), 252
- xmin<-, SpatRaster, numeric-method (xmin), 252

- xres, [10](#)
- xres (dimensions), [74](#)
- xres, SpatRaster-method (dimensions), [74](#)
- xyFromCell, [10](#), [64](#), [88](#), [108](#)
- xyFromCell (xyRowColCell), [254](#)
- xyFromCell, SpatRaster, numeric-method (xyRowColCell), [254](#)
- xyRowColCell, [254](#)

- yFromCell, [10](#)
- yFromCell (xyRowColCell), [254](#)
- yFromCell, SpatRaster, numeric-method (xyRowColCell), [254](#)
- yFromRow, [10](#)
- yFromRow (xyRowColCell), [254](#)
- yFromRow, SpatRaster, missing-method (xyRowColCell), [254](#)
- yFromRow, SpatRaster, numeric-method (xyRowColCell), [254](#)

- ymin, [10](#)
- ymin (xmin), [252](#)
- ymin, SpatExtent-method (xmin), [252](#)
- ymin, SpatRaster-method (xmin), [252](#)
- ymin, SpatVector-method (xmin), [252](#)
- ymin<- (xmin), [252](#)
- ymin<-, SpatExtent, numeric-method (xmin), [252](#)
- ymin<-, SpatRaster, numeric-method (xmin), [252](#)

- ymin, [10](#)
- ymin (xmin), [252](#)
- ymin, SpatExtent-method (xmin), [252](#)
- ymin, SpatRaster-method (xmin), [252](#)
- ymin, SpatVector-method (xmin), [252](#)
- ymin<- (xmin), [252](#)
- ymin<-, SpatExtent, numeric-method (xmin), [252](#)
- ymin<-, SpatRaster, numeric-method (xmin), [252](#)

- yes, [10](#)
- yes (dimensions), [74](#)
- yes, SpatRaster-method (dimensions), [74](#)

- zonal, [8](#), [67](#), [84](#), [110](#), [256](#)
- zonal, SpatRaster, SpatRaster-method (zonal), [256](#)
- zonal, SpatVector, SpatVector-method (zonal), [256](#)

- zoom, [16](#), [258](#)
- zoom, SpatRaster-method (zoom), [258](#)
- zoom, SpatVector-method (zoom), [258](#)