Package 'terra'

November 15, 2021

Type Package

Title Spatial Data Analysis

Version 1.4-19

Date 2021-11-14

Depends R (>= 3.5.0)

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

LinkingTo Rcpp

Imports methods, Rcpp

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

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>)

Repository CRAN

Date/Publication 2021-11-15 12:10:02 UTC

R topics documented:

`terra-package` *The terra package*

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 (R 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 vectors of SpatRaster or SpatVector.

These classes hold a C++ pointer to the data and they cannot be directly saved to a ".Rds" file or used in cluster computing. They cannot be recovered from a saved R session either. See wrap or writeRaster to work around that limitation.

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. Some of these may not have been implemented yet (they are not hyperlinked).

SpatRaster

I. Creating, combining and sub-setting SpatRaster objects

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

II. Changing the spatial extent and/or resolution of a SpatRaster

Also see the methods in section VIII

merge	Combine SpatRasters with different extents (but same origin and resolution)
mosaic	Combine SpatRasters with different extents using a function for overlapping cells
crop	Select a geographic subset of a SpatRaster
extend	Enlarge a SpatRaster
trim	Trim a SpatRaster by removing exterior rows and/or columns that only have NAs
aggregate	Combine cells of a SpatRaster to create larger cells
disagg	Subdivide cells
resample	Resample (warp) values to a SpatRaster with a different origin and/or resolution
project	Project (warp) values to a SpatRaster with a different coordinate reference system
shift	Adjust the location of SpatRaster
flip	Flip values horizontally or vertically

rotate	Rotate values around the date-line (for lon/lat data)
t	Transpose a SpatRaster

III. Local (cell based) computation

Apply-like methods:

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

Other methods:

cellSize	Compute the area of cells
classify	(Re-)classify 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
subst	Substitute (replace) cell values
which.lyr	which is the first layer that is TRUE?

IV. Zonal and global computation

expanse	Compute the summed area of cells
crosstab	Cross-tabulate two SpatRasters
freq	Frequency table of SpatRaster cell values
global	Summarize SpatRaster cell values with a function
quantile	Quantiles
stretch	Stretch values

scale	Scale values
summary	Summary of the values of a SpatRaster (quartiles and mean)
unique	Get the unique values in a SpatRaster
zonal	Summarize a SpatRaster by zones in another SpatRaster

V. Focal and other spatial contextual computation

focal	Focal (neighborhood; moving window) functions
adjacent	Identify cells that are adjacent to a set of cells of a SpatRaster
boundaries	Detection of boundaries (edges)
distance	Shortest distance to a cell that is not NA or to or from a vector object
direction	Direction (azimuth) to or from cells that are not NA
localFun	Local association (using neighborhoods) functions
patches	Find patches (clumps)
terrain	Compute slope, aspect and other terrain characteristics from elevation data
shade	Compute hill shade from slope and aspect layers
autocor	Compute global or local spatial autocorrelation

VI. Model predictions

predict	Predict a non-spatial model to a SpatRaster
interpolate	Predict a spatial model to a SpatRaster

VII. Accessing cell values

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

values	cell values (fails with very large rasters)
values<-	Set new values to the cells of a SpatRaster
setValues	Set new values to the cells of a SpatRaster
as.matrix	Get cell values as a matrix
as.array	Get cell values as an array
extract	Extract cell values from a SpatRaster (e.g., by cell, coordinates, polygon)

spatSample	Regular or random sample
minmax	Get the minimum and maximum value of the cells of a SpatRaster (if known)
setMinMax	Compute the minimum and maximum value of a SpatRaster if these are not known
extract	spatial queries of a SpatRaster with a SpatVector

VIII. Getting and setting SpatRaster 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).

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

xFromCol	x-coordinates from column numbers
yFromRow	y-coordinates from row numbers
xFromCell	x-coordinates from row numbers
yFromCell	y-coordinates from cell numbers
xyFromCell	x and y coordinates from cell numbers

colFromX	Column numbers from x-coordinates (or longitude)
rowFromY	Row numbers from y-coordinates (or latitude)
rowColFromCell	Row and column numbers from cell numbers
cellFromXY	Cell numbers from x and y coordinates
cellFromRowCol	Cell numbers from row and column numbers
cellFromRowColCombine	Cell numbers from all combinations of row and column numbers
cells	Cell numbers from an SpatVector or SpatExtent

X. Writing SpatRaster files

Basic:

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

Advanced:

writeStart	Open a file for writing
writeValues	Write some values
writeStop	Close the file after writing

XI. Miscellaneous SpatRaster methods

terraOptions	Show, set, or get session options, mostly to control memory use and to set write options
sources	Show the data sources of a SpatRaster
tmpFiles	Show or remove temporary files
mem_info	memory needs and availability
readStart	Open file connections for efficient multi-chunk reading
readStop	Close file connections
inMemory	Are the cell values in memory?

SpatRasterDataSet

XII. SpatRasterDataset

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

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

SpatVector

XIII. Create SpatVector objects

vect	Create a SpatVector from a file (for example a "shapefile") or from another object
rbind	append SpatVectors of the same geometry type
unique	remove duplicates
na.omit	remove empty geometries and/or fields that are NA
project	Project a SpatVector to a different coordinate reference system
writeVector	Write SpatVector data to disk
centroids	Get the centroids of a SpatVector
voronoi	Voronoi diagram
delauny	Delauny triangles
convHull	Compute the convex hull of a SpatVector
fillHoles	Remove or extract holes from polygons

XIV. Properties of SpatVector objects

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

XV. Geometric queries

adjacent	find adjacent polygons

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

XVI. Geometric operations

erase or "-"	erase (parts of) geometries
intersect or "*"	intersect geometries
union or "+"	Merge geometries
cover	update polygons
symdif	symmetrical difference of two polygons
aggregate	dissolve smaller polygons into larger ones
buffer	buffer geometries
disagg	split multi-geometries into separate geometries
crop	clip geometries using a rectangle (SpatExtent) or SpatVector

XVII. SpatVector attributes

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

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

XVIII. Change geometries (for display, experimentation)

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

XIX. Geometry properties and topology

width	the minimum diameter of the geometries
clearance	the minimum clearance of the geometries
sharedPaths	shared paths (arcs) between line or polygon geometries
simplify	simplify geometries
gaps	find gaps between polygon geometries
fillHoles	get or remove the polygon holes
makeNodes	create nodes on lines
mergeLines	connect lines to form polygons
removeDupNodes	remove duplicate nodes in geometries and optionally rounds the coordinates
snap	make boundaries of geometries identical if they are very close to each other
erase (single argument)	remove parts of geometries that overlap
union (single argument)	create new polygons such that there are no overlapping polygons

---

Spat* Collections

XX. 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.

svc	create a SpatRasterCollection from a set of SpatRaster objects
length	how many SpatRasters does the SpatRasterCollection have?
[	extract a SpatRastert

SpatExtent

XXI. SpatExtent

ext	Create a SpatExtent object. For example to crop a Spatial dataset
intersect	Intersect two SpatExtent objects, same as -
union	Combine two SpatExtent objects, same as +
Math-methods	round/floor/ceiling of a SpatExtent
align	Align a SpatExtent with a SpatRaster

draw	Create a SpatExtent by drawing it on top of a map (plot)

General methods

XXII. Data type conversion

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)`

rast	SpatRaster from matrix and other objects
vect	SpatVector from `sf` or `Spatial*` vector data
sf::st_as_sf	sf object from SpatVector
rasterize	Rasterizing points, lines or polygons
as.points	Create points from a SpatRaster or SpatVector
as.lines	Create points from a SpatRaster or SpatVector
as.polygons	Create polygons from a SpatRaster
as.contour	Contour lines from a SpatRaster

XXIII. Plotting

Maps:

plot	Plot a SpatRaster or SpatVector. The main method to create a map
points	Add points to a map
lines	Add lines to a map
polys	Add polygons to a map
text	Add text (such as the values of a SpatRaster or SpatVector) to a map
image	Alternative to plot to make a map with a SpatRaster
plotRGB	Combine three layers (red, green, blue channels) into a single "real color" plot
RGB2col	Combine three layers (red, green, blue channels) into a single layer with a color-table
dots	Make a dot-density map
cartogram	Make a cartogram
persp	Perspective plot of a SpatRaster
contour	Contour plot or filled-contour plot of a SpatRaster
inset	Add a small inset (overview) map
sbar	Add a scalebar

Interacting with a map:

zoom	Zoom in to a part of a map by drawing a bounding box on it

click	Query values of SpatRaster or SpatVector by clicking on a map
sel	Select a spatial subset of a SpatRaster or SpatVector by drawing on a map
draw	Create a SpatExtent or SpatVector by drawing on a map

Other plots:

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

XXIV. 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 (combining Raster* objects or files)	c
addLayer	add<-
area	cellSize
disaggregate	disagg
extent	ext
calc	app
overlay	lapp
stackApply	tapp
nlayers	nlyr
NAvalue	NAflag
stackSelect	selectRange
reclassify, subs, cut	classify
cellStats	global
projectRaster	project
dropLayer	subset
isLonLat, isGlobalLonLat, couldBeLonLat	is.lonlat
shapefile	vect
gridDistance, distanceFromPoints	distance
drawExtent, drawPoly, drawLine	draw

compareRaster	compareGeom
sampleRandom, sampleRegular	spatSample
rasterToPoints	as.points
rasterToPolygons	as.polygons
cellFromLine, cellFromPolygon, cellsFromExtent	cells
layerize	segregate
clump	patches

XXV. 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). It other cases it simply seemed better. Here are some examples:

as.polygons	By default, `terra` returns dissolved polygons
extract	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
values	`terra` always returns a matrix. `raster` returns a vector for a `RasterLayer`
Summary-methods	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)

Author

Except where indicated otherwise, the methods and functions in this package were written by Robert Hijmans. The configuration scripts were written by Roger Bivand for `rgdal` and `sf`. 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

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. The feedback and suggestions by Kendon Bell, Jean-Luc Dupouey, Krzysztof Dyba, Jakub Nowosad, Gerald Nelson, and Michael Sumner have been especially helpful.

activeCat *Active category*

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 withouth 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
pairs	logical. If `TRUE`, a two-column matrix of pairs of adjacent cells is returned. If `x` is a `SpatRaster` and `pairs` is `FALSE`, an `n*m` matrix is returned where the number of rows `n` is `length(cells)` and the number of columns `m` is the number of neighbors requested with `directions`
include	logical. Should the focal cells be included in the result? They are always included if `pairs=TRUE`
type	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
symmetrical	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", ...)
```

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?

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 then 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, 2)


## 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)
```

`animate` *Animate a SpatRaster*

Description

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

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 *never* be another SpatRaster
`cores`	positive integer. If `cores > 1`, a 'parallel' package cluster with that many cores is created and used. 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`

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)
```

`as.character` *Create a text representation of (the skeleton of) an object*

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` *SpatRaster or SpatVector to data.frame*

Description

Coerce a SpatRaster or SpatVector to a data.frame or coerce a SpatVector to a list

Usage

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

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

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

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

Arguments

x	SpatVector
geom	character or NULL. If not NULL, either "WKT" or "HEX", to get the geometry included in Well-Known-Text or hexadecimal notation
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

Value

data.frame

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.data.frame(v)
as.list(v)
```

---

as.raster *Coerce to a "raster" object*

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, extent=FALSE)

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

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

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

## 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

x	SpatRaster or SpatVector
trunc	logical; truncate values to integers. Cels with the same value are merged. Therefore, if trunc=FALSE the object returned can be very large
dissolve	logical; combine cells with the same values? If TRUE only the first layer in x is processed
values	logical; include cell values as attributes? If FALSE the cells are not dissolved and the object returned can be very large
multi	logical. If TRUE a multipoint geometry is returned
skiplast	logical. If TRUE the last point of a polygon (which is the same as the first point) is not included
extent	logical. if TRUE, a polygon for the extent of the SpatRaster is returned. It has vertices for each grid cell, not just the four corners of the raster. This can be useful for more precise projection. In other cases it is better to do as.polygons(ext(x)) to get a much smaller object returned that covers the same extent
na.rm	logical. If TRUE cells that are NA are ignored
crs	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 *Two argument arc-tangent*

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)
```

Arguments

y	SpatRaster
x	SpatRaster

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 *Spatial autocorrelation*

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

`height`	SpatRaster
`maxcell`	integer. To regularly subsample very large datasets
`digits`	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
`breaks`	breaks used to group the data as in `cut`
`col`	a color generating function such as `rainbow` (the default), or a vector of colors
`...`	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

`x`	SpatRaster
`inner`	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 graphics::boxplot

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 objects or raster patches*

Description

Calculate a buffer around all cells that are not `NA` in a SpatRaster, or around the geometries of a SpatVector (currently only implemented for points)

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. 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, ...)

## 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
...	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` transfers all categories to new layers.

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)
```

---

cellSize *Area covered by each raster cell*

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 crs.

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, 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
filename	character. Output filename
...	additional arguments for writing files as in writeRaster

Value

numeric (area)

See Also

expanse

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 *Get centroids*

Description

Get the centroids for the polygons of a SpatVector

Usage

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

Arguments

x	SpatVector

Value

SpatVector of points

Examples

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

---

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

x	SpatRaster
lower	numeric. lowest value
upper	numeric. highest value
values	logical. If FALSE values outside the clamping range become NA, if TRUE, they get the extreme values
filename	character. Output filename
...	additional argumments 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` *Classify (or reclassify) cell values*

Description

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

The classification is done based on the argument `rcl`. 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, rcl, include.lowest=FALSE, right=TRUE,
     othersNA=FALSE, filename="", ...)
```

Arguments

`x`	SpatRaster
`rcl`	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 re-classifying integer values. In that case, the `right` argument is automatically set to `NA`. 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 inbetween 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.
`include.lowest`	logical, indicating if a value equal to the lowest value in `rcl` (or highest value in the second column, for `right=FALSE`) should be included.
`right`	logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa. The default is `TRUE`. A special case is to use right=NA. In this case both the left and right intervals are open

othersNA	logical. If TRUE, values that are not matched become NA. If FALSE, they retain their original value.
filename	character. Output filename
...	Additional arguments for writing files as in writeRaster

Value

SpatRaster

Note

For 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
rc2 <- classify(r, c(0, 0.25, 0.5, 1), include.lowest=TRUE)

## 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, othersNA=TRUE)
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 other object types*

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
mode	this argument is ignored
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

Value

vector, matrix, 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
```

Arguments

x	SpatRaster
layer	positive integer, the layer number or name
value	a vector of colors; or a three (red,green,blue) or four (alpha) column data.frame with no more than 256 rows; 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)
coltab(r) <- coltb
plot(r)
```

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

---

compareGeom *Compare geometries of SpatRasters*

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)
```

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

Examples

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

## Not run:
compareGeom(r1, r3)

## End(Not run)
```

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 filled.contour plot is made
...	any argument that can be passed to contour or filled.contour (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)
```

copy *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'
copy(x)

## S4 method for signature 'SpatVector'
copy(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 <- copy(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)
```

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)
```

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

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 in stead 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 codexyFromCell

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. With a SpatRaster you can only extract rectangular areas, but see mask for setting cell values within SpatRaster to NA.

You can crop a SpatVector with a rectangle, or with another vector (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.

Usage

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

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

Arguments

x	SpatRaster or SpatVector
y	SpatRaster, SpatVector, SpatExtent or other object that has a SpatExtent (ext returns a SpatExtent)
snap	character. One of "near", "in", or "out"
filename	character. Output filename
...	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)

# 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, indicting 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 object.

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 libaries. 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
```

---

density *Density plot*

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 plot

Value

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

Examples

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

---

depth *depth of SpatRaster layers*

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 *describe*

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 <- desc(f, options=c("json", "nomd", "proj4"))
#cat(g, "\n")
```

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 objecs
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)
```

---

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).

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.

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,ANY'
distance(x, y, lonlat, sequential=FALSE)
```

Arguments

`x`	SpatRaster, SpatVector, or two-column matrix (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

Examples

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

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

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)
```

`dots` *Make a dot-density map*

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`. It 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

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

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)
#d <- dots(v, "population", 10000)
#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

intersect, crop. The equivalent for SpatRaster is mask

Examples

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

# 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)
```

expand *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, 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
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)

# expand 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))
```

---

expanse *Get the expanse (area) of individual polygons or for all (summed) raster cells*

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 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)

## 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

Value

numeric. The sum of the size of the cells that are not NA

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")

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)

### 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)

Usage

```
## S4 method for signature 'SpatRaster'
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
value	SpatExtent, or numeric vector of lenght four (xmin, xmax, ymin, ymax), or a single number with the $ method
name	charcter, one of xmin, xmax, ymin, or ymax
...	if x is a single numeric value, additional numeric values for xmax, ymin, and ymax

Value

SpatExtent

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
```

---

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, and list=FALSE, the first column has the ID (record number) of the SpatVector used.

Usage

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

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

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

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

Arguments

x	SpatRaster
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 codemean, sum, min and max are accepted)
...	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)
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
list	logical. If FALSE the output is simplified to a matrix (if fun=NULL)
factors	logical. If TRUE the categories are returned as factors instead of their numerical representation. The value returned becomes a data.frame if it otherwise would have been a matrix, even if there are no factors
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
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 exctract 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)`

Value

matrix, list, or data.frame

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)

#ee <- extract(z, v, list=TRUE)
#rapply(ee, mean)

#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)

#ee <- extract(x, v, list=TRUE)
#matrix(rapply(ee, mean), ncol=nlyr(x), byrow=TRUE)
```

`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'
setMinMax(x, force=FALSE)
```

Arguments

`x`	SpatRaster
`force`	logical. If `TRUE` min and max values are recomputed even if already avaialbe

Value

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

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

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 (integers) that have an associated label. For a SpatRaster, the index starts at 0, and cannot exceed 255.

The categories can be inspected with `levels` and `cats`. With `levels<-` you can set the categories of the first layer by providing a vector of labels (the first value will be for cells with value 0, the second for 1, etc). You can also provide 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. To set categories for multiple layers you can provide `levels<-` with a list with one element for each layer.

With `setCats` you can set it for any layer and you can also set the 'active" category if there are multiple categories.

Usage

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

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

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

## S4 method for signature 'SpatRaster'
cats(x, layer)

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

Arguments

x	SpatRaster
layer	positive integer, the layer number or name
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 category, skipping the first column with the ID.

Value

list (levels, cats) or data.frame (cats for a single layer); logical (is.factor, setCats)

See Also

activeCat, catalyze

Examples

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

cls <- c("forest", "water", "urban")
# make the raster start at zero
x <- r - 1
levels(x) <- cls
names(x) <- "land cover"
is.factor(x)
x

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

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

# approach 1
levels(x) <- c("", "", "", "forest", "water", "urban")

# approach 2, also showing the use of two categories
d <- data.frame(id=3:5, cover=cls, letters=letters[1:3], value=10:12)
levels(x) <- d
x

## switch categories
cats(x, 1)
# get current index
activeCat(x)
# set index
```

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

r <- as.numeric(x)
r

#activeCat(x) <- 2
#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

`x`	SpatVector
`inverse`	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)
```

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

`x`	SpatRaster or SpatVector
`direction`	character. Should (partially) match "vertical" to flip by rows, or "horizontal" to flip by columns
`filename`	character. Output filename
`...`	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` *Focal values*

Description

Calculate focal ("moving window") values for the neighborhood of focal cells using a matrix of weights, perhaps in combination with a function.

Usage

```
## S4 method for signature 'SpatRaster'
focal(x, w=3, fun="sum", na.rm=TRUE, na.only=FALSE,
fillvalue=NA, expand=FALSE, filename="", ...)
```

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 single number. For example mean, modal, min or max. It should also accept a `na.rm` argument, either as actual argument or through use of `...`
`na.rm`	logical. Should missing values be removed?
`na.only`	logical. Should only missing values in `x` be changed?
`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
`filename`	character. Output filename
`...`	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 the mean, min, or max value for a non-rectangular area.

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,-`

Value

SpatRaster

See Also

focalMat, focalValues

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, .25, ...), 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)
```

---

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'))
```

Arguments

x	SpatRaster
d	numeric. If type=circle, the radius of the circle (in units of the crs). If type=rectangle the dimension of the rectangle (one or two numbers). If type=Gauss the size of sigma, and optionally another number to determine the size of the matrix returned (default is 3*sigma)
type	character indicating the type of filter to be returned

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")
```

---

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 *Frequency table*

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

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

Value

matrix or data.frame with 3 columns (layer, value, count) or, if bylayer=FALSE two columns (value, count). If any of the layers of x is categorical, there is an additional column (label). A data.frame is returned if usenames=TRUE or if any of the layers of x is categorical.

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 *Find gaps between polygons*

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 *GDAL version, supported file formats, and cache size*

Description

Set the GDAL warning level or get a data.frame with the available GDAL drivers (file formats), or get the GDAL version (if warn=NA and drivers=FALSE, the default.

GDAL is the software library that terra builds on to read and write spatial data and for some raster data processing.

Usage

```
gdal(warn=NA, drivers=FALSE)
gdalCache(size=NA)
```

Arguments

warn	ignored if NA. 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.
size	numeric. The new cache size in MB

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)
```

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 in stead of a matrix (only for wkt=FALSE and hex=FALSE

Value

matrix

See Also

See xyFromCell to get the coordinates of the cells of a SpatRaster

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 *Geometry type of a SpatVector*

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", or "sum".

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

x	SpatRaster
fun	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", "sdpop" (population sd, using n rather than n-1); or, for relatively small SpatRasters, a proper function
...	additional arguments passed on to fun
weights	NULL or SpatRaster

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)
```

---

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 *Histogram*

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

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

`test`	SpatRaster
`yes`	SpatRaster or numeric
`no`	SpatRaster or numeric
`filename`	character. Output filename
`...`	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=50000, ...)
```

Arguments

`x`	SpatRaster
`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::image`

See Also

`plot`

Examples

```
f <- system.file("ex/elev.tif", package="terra")
r <- rast(f)
image(r)
image(r, col=rainbow(24))
```

`initialize` *Initialize a SpatRaster with values*

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 a 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 `fun=runif`. 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)
```

inset *Make an inset map*

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", border="black", box=NULL, pbx, ...)

## S4 method for signature 'SpatRaster'
inset(x, e, loc="", scale=0.2, background="white", border="black", box=NULL, pbx, ...)

## 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
border	color for the border around the inset. Use NA for no border
box	SpatExtent or missing, to draw a box on the inset
pbx	list with graphical arguments for the box
...	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 (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)

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",
pbx=list(lwd=2, lty=5, col="blue"))

# note the returned inset SpatVector
s
```

`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

x	SpatVector or SpatExtent
y	SpatVector or SpatExtent

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=expanse(p))

y <- intersect(v, p)
```

`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 wen 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).

The form `isLonLat` is depracated. Use `is.lonlat`

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)

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

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

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

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)
```

`is.valid` *Check polygon validity*

Description

Check the validity of polygons

Usage

```
## S4 method for signature 'SpatVector'
is.valid(x, messages=FALSE, as.points=FALSE)
```

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")

# invalid polygon
#w <- vect("POLYGON ((0 -5, 10 0, 10 -10, 4 -2))")
#is.valid(w)
```

lapp *Apply a function to layers of a SpatRaster, or sub-datasets of a SpatRasterDataset*

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. 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 returns 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`.

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, ..., usenames=FALSE, filename="", overwrite=FALSE, wopt=list())

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

Arguments

`x`	SpatRaster or SpatRasterDataset
`fun`	a function that takes a vector and can be applied to each cell of `x`
`...`	additional arguments to be passed to `fun`
`usenames`	logical. Use the layer names to match the function arguments? If `FALSE` matching is by position
`recycle`	logical. Recycle layers to match the subdataset with the largest number of layers
`filename`	character. Output filename
`overwrite`	logical. If `TRUE`, `filename` is overwritten
`wopt`	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) }
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)
lapp(x, function(x, y) x/y, recycle=TRUE)
```

---

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 perimenter 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, ...)

## S4 method for signature 'SpatVector'
lines(x, y=NULL, col, lwd=1, lty=1, arrows=FALSE, ...)

## S4 method for signature 'SpatVector'
polys(x, col, border="black", lwd=1, lty=1, ...)

## S4 method for signature 'SpatExtent'
points(x, col, ...)

## S4 method for signature 'SpatExtent'
lines(x, col, ...)
```

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
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 intead 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", ...)
```

Arguments

`x`	SpatRaster
`y`	SpatRaster or SpatVector
`filename`	character. Output filename template. Filenames will be altered by adding the tilenumber for each tile
`...`	additional arguments for writing files as in `writeRaster`

Value

character (filenames)

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)
```

mask *Mask values in a SpatRaster*

Description

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 not covered by a SpatVector. These cells become NA (or another updatevalue).

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="", ...)
```

Arguments

x	SpatRaster
mask	SpatRaster
inverse	logical. If TRUE, areas on mask that are _not_ the maskvalue are masked
maskvalues	numeric. The value(s) in mask that indicates the cells of x that should become updatevalue (default = NA)
updatevalue	numeric. The value that cells of x should become if they are not covered by mask (and not NA)
touches	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
filename	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 *Value matching for SpatRasters*

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 *Arithmetic, logical and general mathematical methods*

Description

Standard operators and 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:

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

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`.

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

Summary: `max,min,range,prod,sum,any,all`

Math: `abs,sign,sqrt,ceiling,floor,trunc,cummax,cummin,cumprod,cumsum,log,log10,log2,log1p,acos,acosh`

The following methods have been implemented for **SpatExtent**: `round,floor,ceil,==`, for **(SpatExtent, SpatExtent)**: `==,+,-`, and for **(SpatExtent, numeric)** : `+,-,*,/,%%`

Usage

```
## S4 method for signature 'SpatRaster'
log(x, base=exp(1))
```

Arguments

`x`	SpatRaster
`base`	a positive or complex number: the base with respect to which logarithms are computed

Value

SpatRaster or SpatExtent

seealso

`ifel` to convieniently combine operations and `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
s <- sqrt(b2)
round(s, 1)

max(s)
max(s, na.rm=TRUE)

x <- is.na(s)

y <- which.max(s)

### 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
```

---

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	SpatRastaer
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 *Merge SpatRaster or SpatExtent objects, or a SpatVector with a data.frame*

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. See classify to merge a SpatRaster and a data.frame. You can also merge SpatExtent objects.

There is a 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

Details

The SpatRaster objects 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.

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

mosaic

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 <- src(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)
```

`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

mosaic SpatRasters to form a new SpatRaster object with a larger spatial extent. Unlike with merge, values on overlapping areas are averaged.

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

Details

The SpatRaster objects 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.

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 <- src(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)

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 *nearby geometries*

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)
```

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 0.1 and 0.9 (larger values give a warning). The fraction of RAM that may be used by the program.

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 *Origin*

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	SpatRaster
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 *Pairs plot (matrix of scatterplots)*

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

x	SpatRaster
hist	logical. If TRUE a histogram of the values is shown on the diagonal
cor	logical. If TRUE the correlation coefficient is shown in the upper panels
use	argument passed to the cor function
maxcells	integer. Number of pixels to sample from each layer of a large SpatRaster
...	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 indvidual 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, filename="", ...)
```

Arguments

x	SpatRaster
directions	integer indicating which cells are considered adjacent. Should be 8 (Queen's case) or 4 (Rook's case)
zeroAsNA	logical. If TRUE treat cells that are zero as if they were NA
filename	character. Output filename
...	options for writing files as in writeRaster

Value

SpatRaster. Cell values are either a patch number

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)

r <- rast(nrows=10, ncols=10, xmin=0)
r[] <- 0
r[3, 3] <- 10
r[4, 4] <- 10
r[5, 5:8] <- 12
r[6, 6:9] <- 12

# remove zeros if need be with zeroAsNA

p4 <- patches(r, zeroAsNA=TRUE)
p8 <- patches(r, 8, zeroAsNA=TRUE)

# patches for different values
# remove zeros
rr <- classify(r, cbind(0, NA))
# first make layers for each value
s <- segregate(rr, keep=TRUE, other=NA)
p <- patches(s)
```

---

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)
```

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, fun=NULL,
     colNA=NULL, alpha=NULL, reset=FALSE, ...)

## S4 method for signature 'SpatRaster,missing'
plot(x, y, maxcell=500000, main, mar=NULL, nc, nr, maxnl=16, ...)

## S4 method for signature 'SpatVector,character'
plot(x, y, col, type, mar=NULL, legend=TRUE,
     add=FALSE, axes=!add, main=y,
     plg=list(), pax=list(), nr, nc, ...)

## S4 method for signature 'SpatVector,numeric'
plot(x, y, ...)

## S4 method for signature 'SpatVector,missing'
plot(x, y, ...)

## S4 method for signature 'SpatExtent,missing'
plot(x, y, ...)
```

Arguments

x	SpatRaster or SpatVector
y	missing or positive integer or name indicating the layer(s) to be plotted
col	character. Colors
type	character. Type of map/legend. One of "continuous", "classes", or "interval"
mar	numeric vector of lenght 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 `mar=NA` to not set the margins
legend	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 `plg` for more refined placement (SpatVector data only)
axes	logical. Draw axes?

plg	list with parameters for drawing the legend. See the arguments for legend
pax	list with parameters for drawing axes. See the arguments for axis
maxcell	positive integer. Maximum number of cells to use for the plot
smooth	logical. If TRUE the cell values are smoothed (for continuous legend)
range	numeric. minimum and maximum values to be used for the continuous legend
levels	character. labels to be used for the classes legend
fun	function to be called after plotting each SpatRaster layer to add something to each map (such as text, legend, lines). For example, with SpatVector v, you could do fun=function() lines(v). The function may have one argument, representing the the layer that is plotted (1 to the number of layers)
colNA	character. color for the NA values
alpha	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
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
reset	logical. If TRUE add the margins (see argument mar) are reset to what they were before calling plot; doing so may affect the display of additional objects that are added to the map (e.g. with lines
...	arguments passed to plot("SpatRaster","numeric") 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) <- 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)


## vector

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.

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, bgalpha,
addfun=NULL, zlim=NULL, zlimcol=NULL, axes=FALSE, xlab="", ylab="",
asp=NULL, add=FALSE, interpolate, ...)
```

Arguments

x	SpatRaster
r	integer. Index of the Red channel, between 1 and nlyr(x)
g	integer. Index of the Green channel, between 1 and nlyr(x)
b	integer. Index of the Blue channel, between 1 and nlyr(x)
a	integer. Index of the alpha (transparancy) channel, between 1 and nlyr(x). If not NULL, argument alpha is ignored
scale	integer. Maximum (possible) value in the three channels. Defaults to 255 or to the maximum value of x if that is known and larger than 255
maxcell	integer > 0. Maximum number of pixels to use
mar	numeric vector recycled to length 4 to set the margins of the plot. Use mar=NULL or mar=NA to not set the margins
stretch	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, 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.

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

object	SpatRaster
model	fitted model of any class that has a "predict" method (or for which you can supply a similar method as fun argument. E.g. glm, gam, or randomForest
fun	function. The predict function that takes model as first argument. The default value is predict, but can be replaced with e.g. predict.se (depending on the type of model), or your own custom function
...	additional arguments for fun
const	data.frame. Can be used to add a constant value as a predictor variable so that you do not need to make a SpatRaster layer for it
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 standard models such as "glm" as the predict function will extract the levels from the model object, but it is necessary in some other cases (e.g. cforest models from the party package)
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
index	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[,-1]))

#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")

# use a modified function to get the probability and standard error
# from the glm model. The values returned by "predict" are in a list,
# and this list needs to be transformed to a matrix

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))

## End(Not run)
```

`project` *Change the coordinate reference system*

Description

Change the coordinate reference system ("project") of a SpatVector or SpatRaster.

Usage

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

## S4 method for signature 'SpatRaster'
project(x, y, method, mask=FALSE, align=FALSE, filename="", ...)
```

Arguments

x	SpatVector
y	if (x is a SpatRaster, the prefered approach is for y to be a SpatRaster as well, serving as a template for the geometry (extent and resolution) of the output SpatRaster. 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., `+proj=longlat +datum=WGS84`), or an EPSG code (e.g., `"epsg:4326"`). But note that the PROJ.4 notation has been deprecated, and you can only use if with the WGS84/NAD83 and NAD27 datums. Other datums are silently ignored. If x is a SpatVector, 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
filename	character. Output filename
...	additional arguments for writing files as in `writeRaster`

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 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 <- "+proj=moll +lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84"
p <- project(v, crs)
p
```

---

quantile *Quantiles of spatial data*

Description

Compute quantiles for each cell across the layers of a SpatRaster.

You can also use this method to compute the 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"))
r <- c(r/2, r, r*2)
q <- quantile(r)
q

# same but slower
# qa <- app(r, quantile)
```

---

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, ..., allyrs=FALSE, fill=NA,
        clamp=FALSE, filename="", overwrite=FALSE, wopt=list())
```

Arguments

x	SpatRaster
first	SpatRaster or positive integer between 1 and nlyr(x), indicating the first layer in the range of layers to be considered
last	SpatRaster or positive integer between 1 and nlyr(x), indicating the last layer in the range to be considered
fun	function to be applied
...	additional arguments passed to fun
allyrs	logical. Should the values of all layers be passed to fun. The values outside of the range are set to fill
fill	numeric. The fill value for the the values outside of the range, for when allyrs=TRUE
clamp	logical. If FALSE and the specified range is outside 1:nlyr(x) all cells are considered NA. Otherwise, the invalid part of the range is ignored
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

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, allyrs=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 or from a file.

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 basic parameters that describe the SpatRaster such as the number of rows and columns and the coordinate reference system. The actual values will be read, perhaps in chunks – to avoid memory overflows – as needed.

Usage

```
## S4 method for signature 'character'
rast(x, subds=0, 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)

## S4 method for signature 'SpatRaster'
rast(x, nlyrs=nlyr(x), names, vals, time=FALSE, props=FALSE)

## S4 method for signature 'matrix'
rast(x, type="", crs="", digits=6)

## S4 method for signature 'data.frame'
rast(x, type="", crs="", digits=6)

## S4 method for signature 'list'
rast(x)

## 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 subdataset. If zero or "", all subdatasets are returned (if possible)
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. PROJ.4 type description of a Coordinate Reference System (map projection). If this argument is missing, and the x coordinates are within -360 .. 360 and the y coordinates are within -90 .. 90, "+proj=longlat +datum=WGS84" is used
time	logical. If TRUE the time stamps are kept
props	logical. If TRUE the properties (categories and color-table) are kept
extent	object of class SpatExtent. If present, the arguments xmin, xmax, ymin and ymax are ignored
resolution	numeric vector of length 1 or 2 to set the resolution (see res). If this argument is used, arguments ncol and nrow 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)
type	character. If the value is not "xyz", the raster has the same number of rows and colums as the matrix. If the value is "xyz", the matrix must have at least two columns, the first with x (or longitude) and the second with y (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 type="xyz"
...	additional arguments, in some cases passed on to the rast,missing-method

Details

The files are opened and read via GDAL. GDAL guesses the file format from the name, and/or else tries reading it with different "drivers" untill it succeeds. In very few cases this may cause a file to be opened with wrong driver, and some information may be lost (for example because of opening a netCDF file with the HDF5 driver. You can avoid that by prepending the driver name to the filename like this: `rast('NETCDF:"filename.ncdf"')`

Value

SpatRaster

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)
```

---

rasterize *Rasterize vector data*

Description

Transfer 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())
```

Arguments

x	SpatVector
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)
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

Note

To update existing raster data use the output of rasterize as second argument in cover

See Also

mask

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
r <- rast(v, ncols=75, nrows=100)
#x <- rasterize(v, r, "NAME_2")

#plot(x)
#lines(v)
```

---

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. When writing is done close the file with writeStop.

Usage

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

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

## S4 method for signature 'SpatRaster'
readValues(x, row=1, nrows=nrow(x), col=1, ncols=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, nrows)

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 is 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	poistive 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
...	additional arguments for writing files as in writeRaster

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 succesful 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 *relate*

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,missing'
relate(x, y, relation, pairs=FALSE, symmetrical=FALSE)
```

Arguments

x	SpatVector or any object for which vect returns a SpatVector or else for which ext returns a 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", other are not (e.g. "within")

Value

matrix

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.

Value

SpatRaster

See Also

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`, `expand` 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.
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 *rescale*

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

x	SpatVector or SpatRaster
fx	numeric > 0. The horizontal scaling factor
fy	numeric > 0. The vertical scaling factor
x0	numeric. x-coordinate of the center of rescaling. If missing, the center of the extent of x is used
y0	numeric. y-coordinate of the center of rescaling. If missing, the center of the extent of x 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 *Declare RGB channels*

Description

With RGB you can get or set the layers to be used as Red, Green and Blue when plotting a SpatRaster. Currently, the sole benefit of this is that plot will send the object to plotRGB

With RGB2col you can convert a three-layer RGB SpatRaster 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'
RGB2col(x, value, stretch=NULL, grays=FALSE, filename="", overwrite=FALSE, ...)
```

Arguments

x	SpatRaster
value	vector of three (or four) positive integers indicating the layers that are red, green and blue (and optionally a fourth transparancy layer). Or NULL to remove the RGB settings. With RGB2col, this argument can be missing if RGB(x) is not NULL
stretch	character. Option to stretch the values to increase contrast: "lin" (linear) or "hist" (histogram)
grays	logical. If TRUE, a gray-scale color-table is created
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
...	additional arguments for writing files as in writeRaster

Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))
RGB(s) <- c(1,2,3)
plot(s)
RGB(s) <- NULL
```

---

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 *Apply a terra function that takes only a single layer to all layers*

Description

Apply to all layers of a SpatRaster a function that only takes a single layer SpatRaster (these are rare). in most cases you can also use lapply or sapply for this.

Usage

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

Arguments

x	SpatRaster
fun	a function that takes a SpatRaster argument and can be applied to each layer of x
...	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 writeRaster

Value

SpatRaster

See Also

lapp,app,tapp,lapply

Examples

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

---

sbar *scalebar*

Description

Add a scalebar to a plot

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 scalebar. In the units of the coordinates of the plot, and in km for angular (longitude/latitude) data; see lonlat
xy	x and y coordinate to place the plot. Can be NULL. Use xy=click() to make this interactive
type	"line" or "bar"
divs	number of divisions for a bar: 2 or 4
below	character. Text to go below scalebar (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
adj	adjustment for text placement
label	vector of three numbers to label the scale bar (beginning, midpoint, end)
lwd	line width for the "line" type scalebar
xpd	logical. If TRUE, the scalebar can be (partly) outside the plot area
...	graphical arguments to be passed to other methods

Value

none

See Also

plot, inset

Examples

```
f <- system.file("ex/meuse.tif", package="terra")
r <- rast(f)
plot(r)
sbar(1000)
sbar(1000, xy=c(178500, 333500), type="bar", divs=4, cex=.8)

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, 49.9), cex=.8, label=c(0,"km",15))

sbar(15, c(6.65, 49.8), cex=.8, label="15 kilometer", lwd=5)

sbar(15, c(6.65, 49.7), divs=4, cex=.8, below="km")
```

---

scale *Scale values*

Description

Center and/or scale raster data. For details see link{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 nlyr(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 nlyr(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)
```

Arguments

x	character (filename), or SpatRaster, or list of SpatRaster objects, or missing. If multiple filenames are provided, it is attempted to make SpatRasters from these, and combine them into a SpatRasterDataset
ids	optional. vector of integer subdataset ids. Ignored if the first value is not a positive integer
...	additional arguments. Can be other SpatRaster objects if x is a SpatRaster

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]
```

---

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. Classes and cell values are always truncated to integers.

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, 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
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	logial. 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` *select cells with high or low values*

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

`x`	SpatRaster. Only the first layer is processed
`n`	The number of cells to select
`low`	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) <- c("low", "high")
plot(m)
```

`selectRange` *Select the values of a range of layers, as specified by cell values in another SpatRaster*

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 af 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)
```

---

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, time=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`
`time`	logical. If `TRUE` the time stamps are kept
`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="", ...)
```

Arguments

slope	SpatRasterwith slope values (in radians)
aspect	SpatRaster with aspect values (in radians)
angle	The the elevation angle of the light source (sun), in degrees
direction	The direction (azimuth) angle of the light source (sun), in degrees
normalize	Logical. If TRUE, values below zero are set to zero and the results are multiplied with 255
filename	character. Output filename
...	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

Usage

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

Arguments

x	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)
```

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

x	SpatRaster, SpatVector or SpatExtent
dx	numeric. The shift in horizontal direction
dy	numeric. The shift in vertical direction
filename	character. Output filename
...	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)
```

simplify *Simplify geometries*

Description

Reduce the number of nodes used to represent geometries.

Usage

```
## S4 method for signature 'SpatVector'
simplify(x, tolerance=0.1)
```

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

Value

SpatVector

See Also

sharedPaths, gaps

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
#w <- simplify(v, .02)
#e <- erase(w)
#g <- gaps(e)
#plot(e, lwd=5, border="light gray")
#polys(g, col="red", border="red")
```

sources *Data sources of a SpatRaster*

Description

Get the data sources of a SpatRaster and the number of layers by source. 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, bands=FALSE)

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

## S4 method for signature 'SpatRaster'
inMemory(x, bylayer=FALSE)
```

Arguments

x	SpatRaster
bands	logical. If TRUE for each source, the "bands" used, that is, the layers in the source file, are returned
bylayer	logical. If TRUE a value is retured for each layer instead of for each source

Value

sources returns a data.frame with the source names (if any) and the number of layers by source

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

`object`	SpatRaster
`model`	model object
`fun`	function. Default value is "predict", but can be replaced with e.g. "predict.se" (depending on the class of `model`), or a custom function (see examples)
`...`	additional arguments passed to `fun`
`xyNames`	character. variable names that the model uses for the spatial coordinates. E.g., `c("longitude","latitude")`
`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))
mgsf <- 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, mgsf, 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 *Take a regular sample*

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)


## 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

`x`	SpatRaster
`size`	numeric. The sample size. If `x` is a SpatVector, you can also provide a vector of the same length as `x` in wich case sampling is done seperately for each geometry. If `x` is a SpatRaster, and you are using `method="regular"` you can specify the size as two numbers (number of rows and columns)
`method`	character. Should be "regular" or "random"
`replace`	logical. If `TRUE`, sampling is with replacement (if `method="random"`
`na.rm`	logical. If `TRUE`, codeNAs are removed. Only used with random sampling of cell values. That is with `method="random",as.raster=FALSE,cells=FALSE`
`as.raster`	logical. If `TRUE`, a SpatRaster is returned
`as.df`	logical. If `TRUE`, a data.frame is returned instead of a matrix
`as.points`	logical. If `TRUE`, a SpatVector of points is returned
`values`	logical. If `TRUE` cell values are returned
`cells`	logical. If `TRUE`, cell numbers are returned

xy	logical. If TRUE, cell coordinates are returned
ext	SpatExtent or NULL to restrict sampling to a a subset of the area of x
warn	logical. Give a warning if the sample size returned is smaller than requested
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 or SpatRaster

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)

## 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 *Class "SpatVector"*

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 *spin a SpatVector*

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

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 SpatVector: 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)
```

---

src *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'
src(x, ...)

## S4 method for signature 'list'
src(x)

## S4 method for signature 'missing'
src(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 <- src(x, y)
z
```

`stretch` *Stretch*

Description

Linear stretch of values in a SpatRaster object. 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.

Usage

```
## S4 method for signature 'SpatRaster'
stretch(x, minv=0, maxv=255, minq=0, maxq=1, smin=NA, smax=NA, filename="", ...)
```

Arguments

`x`	SpatRaster
`minv`	numeric >= 0 and smaller than maxv. lower bound of stretched value
`maxv`	numeric <= 255 and larger than maxv. upper bound of stretched value
`minq`	numeric >= 0 and smaller than maxq. lower quantile bound of original value. Ignored if smin is supplied
`maxq`	numeric <= 1 and larger than minq. upper quantile bound of original value. Ignored if smax is supplied
`smin`	numeric < smax. user supplied lower value for the layers, to be used instead of a quantile computed by the function itself
`smax`	numeric > smin. user supplied upper value for the layers, to be used instead of a quantile computed by the function itself
`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 *Subset of a SpatRaster*

Description

Select a subset of layers from a SpatRaster.

Usage

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

Arguments

x	SpatRaster
subset	integer or character. Should indicate the layers (represented as integer or by their names)
filename	character. Output filename
overwrite	logical. If TRUE, filename is overwritten
...	additional arguments for writing files as in writeRaster

Value

SpatRaster

Examples

```
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

# expresion based (partial) matching of names with single brackets
s["re"]
s["^re"]
```

```
# not with double brackets
# s[["re"]]
```

---

subset-vector *Subset of a SpatVector*

Description

Select a subset of variables or records from a SpatVector.

Usage

```
## S4 method for signature 'SpatVector'
subset(x, subset, drop=FALSE)
```

Arguments

x	SpatVector
subset	logical expression indicating elements or rows to keep: missing values are taken as false
drop	logical. If TRUE, the geometries will be dropped, and a data.frame is returned

Value

SpatVector or, if drop=TRUE, a data.frame.

Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
v[2:3,]
v[,2:3]
subset(v, v$NAME_1 == "Diekirch")
```

---

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)
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 nrow(x) == length(from). 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)
```

---

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,w
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'
mean(x, ..., trim=NA, na.rm=FALSE)

## S4 method for signature 'SpatRaster'
median(x, na.rm=FALSE)

## S4 method for signature 'SpatRaster'
stdev(x, ..., pop=TRUE, na.rm=FALSE)

## S4 method for signature 'SpatRaster'
which.min(x)

## S4 method for signature 'SpatRaster'
which.max(x)
```

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)
```

---

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

object	SpatRaster or SpatVector
size	positive integer. Size of a regular sample used for large datasets (see spatSample)
warn	logical. If TRUE a warning is given if a sample is used
...	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

x	SpatVector, or list of a SpatVector, or missing
...	Additional SpatVectors

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` *Symetrical difference*

Description

Symetrical 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 *Apply a function to subsets of layers of a SpatRaster*

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)).

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 object.

Usage

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

Arguments

x	SpatRaster
index	factor or numeric (integer). Vector of length nlyr(x) (shorter vectors are recycled) grouping the input layers
fun	function to be applied
...	additional arguments passed to fun
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

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 characteristic*

Description

Compute terrain characteristic from elevation data. The elevation values should be the same as the map units (typically meter) for projected (planar) raster data. They should be in meter when the coordinate reference system (CRS) is longitude/latitude.

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 `neigbors=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

Ritter, P., 1987. A vector-based terrain and aspect generation algorithm. Photogrammetric Engineering and Remote Sensing 53: 1109-1111

---

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 *tighten SpatRaster or SpatRasterDataset objects*

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 *time of SpatRaster layers*

Description

Get or set the time of the layers of a SpatRaster.

Usage

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

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

Arguments

x	SpatRaster
value	"Date", "POSIXt", or numeric

Value

Date

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 (time stored as seconds)
time(s) <- as.POSIXlt(d)
time(s)

# "raw" time
time(s) <- as.numeric(d)
time(s)
```

---

tmpFiles *Temporary files*

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

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) or simplifying tolerance (distance between nodes)
digits	numeric. Number of digits used in rounding. Ignored if < 0

Value

SpatVector

See Also

sharedPaths, gaps, simplify

---

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 *Trim a SpatRaster*

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 *Union SpatVector or SpatExtent objects*

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 codex 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.

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=expanse(p))
#u <- union(v, p)
#plot(u, "pid")

#b <- buffer(v, .015)
#u <- union(b)
#u$sum <- rowSums(as.data.frame(u))
#plot(u, "sum")
```

unique *Unique values*

Description

This function returns the unique values in a SpatRaster or removes duplicates in a SpatVector.

Usage

```
## S4 method for signature 'SpatRaster'
unique(x, incomparables=FALSE)

## S4 method for signature 'SpatVector'
unique(x, incomparables=FALSE, ...)
```

Arguments

x	SpatRaster or SpatVector
incomparables	logical. If FALSE and x is a SpatRaster: the unique values are determined for all layers together, and the result is a matrix. If TRUE, each layer is evaluated separately, and a list is returned. If x is a SpatVector this argument is as for a data.frame.
...	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)

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 *units of SpatRaster or SpatRasterDataSet*

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)
```

values *Cell values and geometry attributes*

Description

Get the cell values of a SpatRaster or the attributes of a SpatVector

Usage

```
## S4 method for signature 'SpatRaster'
values(x, mat=TRUE, dataframe=FALSE, row=1, nrows=nrow(x), col=1, ncols=ncol(x))

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

Arguments

x	SpatRaster or SpatVector
mat	logical. If TRUE, values are returned as a matrix instead of as a vector, except when dataframe is TRUE
dataframe	logical. If TRUE, values are returned as a data.frame instead of as a vector (also if matrix is TRUE)
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

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="")

## 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="")

## 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)
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	the coordinate reference system in one of the following formats: WKT/WKT2, <authority>:<code>, or PROJ-string notation (seelink{crs})
geom	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)

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)
```

voronoi *Voronoi diagram and Delauny triangles*

Description

Get a voronoi diagram or delauny 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'
delauny(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 deldir is used instead of the GEOS C++ library method. It has been reported that deldir 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

plot(v, lwd=2, col=rainbow(15))
lines(x, col="gray", lwd=2)
points(x)
```

vrt *Virtual Raster Tiles*

Description

Create a Virtual Raster Tiles (VRT) dataset from file-based raster datasets.

Usage

```
## S4 method for signature 'character'
vrt(x, filename="", overwrite=FALSE)
```

Arguments

x	character. Filenames of raster "tiles". See tiles
filename	character. Output VRT filename
overwrite	logical. Should filename be overwritten if it exists?

Value

SpatRaster

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

vrtfile <- paste0(tempfile(), ".vrt")
v <- vrt(ff, vrtfile)
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	SpatRaster
w	A vector of weights (one number for each layer), or for spatially variable weights, a SpatRaster 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)
```

which.lyr *Which cells are TRUE?*

Description

This method returns a single layer SpatRaster with cell values that are the first layer in the input that has the value is not zero (FALSE), and, hence, is 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 that it can be saved as an R object to disk, or passed over a connection that serializes (e.g. using a computer cluster).

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 `ncvar_def`

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)", "precicpiation (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 `nlyr(x)` 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 0.1 and 0.9. The fraction of available RAM that terra is allowed to use.
`names`	output layer names.
`NAflag`	numeric. value to represent missing (`NA` or `NaN`) values. See note
`verbose`	logical. If `TRUE` debugging information is printed.
`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","of=COG"), datatype='INT1U')
```

```
## Or with a wopt argument:

writeRaster(r, f, overwrite=TRUE, wopt= list(gdal=c("COMPRESS=NONE", "of=COG"), 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,
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 gdal(drivers=TRUE). If NULL it is attempted to guess the filetype from the filename extension
layer	character. Output layer name. If NULL the filename is used
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)
row	integer row number(s)
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)
```

---

zonal *Zonal statistics*

Description

Compute zonal statistics, that is summarized values of a SpatRaster for each "zone" defined by another SpatRaster.

If `fun` is a true `function`, `zonal` may fail for very large SpatRaster objects, except for the functions ("mean", "min", "max", or "sum").

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
zonal(x, z, fun=mean, ..., as.raster=FALSE, filename="", wopt=list())
```

Arguments

x	SpatRaster
z	SpatRaster with values representing zones
fun	function to be applied to summarize the values by zone. Either as character: "mean", "min", "max", "sum", or, for relatively small SpatRasters, a proper function
...	additional arguments passed to fun
as.raster	logical. If TRUE, a SpatRaster is returned with the zonal statistic for each zone
filename	character. Output filename (ignored if as.raster=FALSE
wopt	list with additional arguments for writing files as in writeRaster

Value

A data.frame with a value for each zone (unique value in zones)

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)
```

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

x	SpatRaster
e	SpatExtent
maxcell	positive integer. Maximum number of cells used for the map
layer	positive integer to select the layer to be used
new	logical. If TRUE, the zoomed in map will appear on a new device (window)
...	additional arguments passed to plot

Value

SpatExtent (invisibly)

See Also

draw, plot

Index