# Package 'terra'

August 1, 2020

**Type** Package

**Title** Spatial Data Analysis

**Version** 0.8-6

**Date** 2020-07-28

**Depends** R (>= 3.5.0)

**Suggests** parallel, tinytest

**LinkingTo** Rcpp

**Imports** methods, Rcpp, raster (>= 3.3-7)

**SystemRequirements** C++11, GDAL (>= 3.0.4), GEOS (>= 3.8.0), PROJ (>= 6.3.1)

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

**Description** Methods for spatial data analysis, especially raster data. 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. Processing of very large files is supported. See the manual and tutorials on <https://rspatial.org/terra/> to get started. The package is similar to the 'raster' package; but 'terra' is simpler and 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** 2020-08-01 00:30:02 UTC

# R topics documented:

## `terra-package` *The terra package*

### Description

The 'terra' package implements two main classes for spatial data handling: `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.

'terra' provides methods to manipulate geographic (spatial) data in "raster" and "vector" data types. Raster data divides space into rectangular cells (pixels). Such data are also referred to as "grid" data are often used to represent spatially continuous phenomena, such as elevation or the weather. Satellite images also have this format. Raster data can be contrasted with discrete "vector" spatial data such as points, lines, polygons.

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'. A major simplification is that 'terra' has a single class "SpatRaster" for which 'raster' has three (RasterLayer, RasterStack, RasterBrick). Likewise there is a single class for vector data ("SpatVector"). Like "raster" this package should be particularly useful when using very large datasets that can not be loaded into the computer's memory. Functions will work correctly, because they they process large files in chunks, i.e., they read, compute, and write blocks of data, without loading all values into memory at once.

Furthermore, 'terra' has a "SpatDataSet" class that represents a collection of sub-datasets for the same area. Each subdataset is a SpatRaster with possibly many layers, and may, for example, represent different weather variables.

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 (for combining Raster* objects) | c |
| extent | ext |
| calc | app |
| overlay | lapp |
| stackApply | tapp |
| extend | expand |
| nlayers | nlyr |
| stackSelect | selectRange |
| reclassify, subs, cut | classify |
| cellStats | global |
| projectRaster | project |
| dropLayer | subset |
| isLonLat, isGlobalLonLat, couldBeLonLat | isLonLat |
| shapefile | vect |
| gridDistance, distanceFromPoints | distance |
| drawExtent, drawPoly, drawLine | draw |
| compareRaster | compareGeom |
| sampleRandom, sampleRegular | spatSample |
| rasterToPoints | as.points |
| rasterToPolygons | as.polygons |
| cellFromLine, cellFromPolygon, cellsFromExtent | cells |
| clump | patches |

Also note that even if function names are the same in terra and raster, their output can be different. In most cases 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:

| | |
|---|---|
| area | By default, terra returns the summed area of the raster cells that are not NA. raster returns a RasterLa |
| - | |
| 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 |
| - | |
| 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 |

Below is a list of some of the most important methods grouped by theme. Some of these may not have been implemented yet.

## I. Creating SpatRaster objects

| | |
|---|---|
| rast | Create a SpatRaster from scratch, file, or another object |

## II. Combining and subsetting SpatRaster objects

| | |
|---|---|
| `c` | Combine SpatRasters (multiple layers) (like raster::stack) |
| `subset` or `[[`, or `$` | Select layers of a SpatRaster |
| `selectRange` | Select cell values from different layers using an index layer |

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

Also see the methods in section XI

| | |
|---|---|
| `merge` | Combine SpatRasters with different extents (but same origin and resolution) |
| `mosaic` | Combine SpatRasters with different extents and a function for the values in overlapping areas |
| `crop` | Select a geographic subset of a SpatRaster |
| `expand` | 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 |
| `disaggregate` | 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 |

## IV. Local (cell based) computation

| | |
|---|---|
| `Arith-methods` | Arith functions (`+`, `-`, `*`, `^`, `%%`, `%/%`, `/`) |
| `Math-methods` | Math functions like `abs`, `sqrt`, `trunc`, `log`, `log10`, `exp`, `sin`, `round` |
| `Logic-methods` | Logic functions (`!`, `&`, `\|`) |
| `Summary-methods` | Summary functions (`mean`, `max`, `min`, `median`, `sum`, `range`, `prod`, `any`, `all`, `stdev`, `which.min`, |
| `Compare-methods` | Compare functions (`==`, `!=`, `>`, `<`, `<=`, `>=`) |
| `app` | Apply a function to cells of all layers(as in `base::apply`) |
| `tapp` | Apply a function to groups of layers (as in `base::tapply`) |
| `lapp` | Apply a function using the layers as variables |
| `rapp` | Apply a function to a spatially variable range of layers |
| `cover` | First layer covers second layer except where the first layer is `NA` |
| `mask` | Use values from first SpatRaster except where cells of the mask SpatRaster are `NA` (or another value |
| `classify` | (Re-)classify values |
| `init` | Initialize cells with new values |
| `area` | Compute the area of cells |

## V. Zonal and global computation

| | |
|---|---|
| `zonal` | Summarize a SpatRaster by zones in another SpatRaster |
| `global` | Summarize SpatRaster cell values with a function |
| `unique` | Get the unique values in a SpatRaster |
| `freq` | Frequency table of SpatRaster cell values |
| `crosstab` | Cross-tabulate two SpatRasters |
| `quantile` | Quantiles |
| `summary` | Summary of the values of a SpatRaster (quartiles and mean) |
| `area` | Compute the total area covered by cells |
| `stretch` | Stretch values |

## VI. 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 |
| `terrain` | Compute slope, aspect and other terrain characteristics from elevation data |
| `autocor` | Compute global or local Moran or Geary indices of spatial autocorrelation |

## VII. Model predictions

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

## VIII. 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 |
| `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 |

## IX. Accessing cell values

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

| | |
|---|---|
| `values` | Get or set all cell values (fails with very large rasters) |
| `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 |

## X. Plotting

| | |
|---|---|
| **Maps** | |
| `plot` | Plot a SpatRaster or SpatVector. The main method to create a map |
| `plotRGB` | Combine three layers (red, green, blue channels) into a single "real color" image |
| `image` | Alternative way to plot a SpatRaster |
| `persp` | Perspective plot of a SpatRaster |
| `contour` | Contour plot or filled-contour plot of a SpatRaster |
| `text` | Plot the values of a SpatRaster or SpatVector on top of a map |
| . | |
| **Interacting with a map** | |
| `zoom` | Zoom in to a part of a map |
| `click` | Query values of SpatRaster or SpatVector by clicking on a map |
| `select` | Select a spatial subset of a SpatRaster or SpatVector |
| `draw` | Create a SpatExtent or SpatVector by drawing on a map |
| . | |
| **Other plots** | |
| `plot` | x-y scatter plot of the values of two SpatRaster objects |

| | |
|---|---|
| hist | Histogram of SpatRaster values |
| barplot | Barplot 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 |

## XI. 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 (a.k.a. "bounding box") |
| origin | The origin of a SpatRaster |
| crs | The coordinate reference system (map projection) |
| isLonLat | Test if an object has (or may have) a longitude/latitude coordinate reference system; and if it has glob |
| filename | Filename(s) to which a SpatRaster is linked |
| compareGeom | Compare the geometry of SpatRasters |
| NAvalue | Get or set the `NA` value (for reading from a file) |

## XII. 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 |

## XIII. Writing files

| | |
|---|---|
| **Basic** | |
| writeRaster | Write all values of SpatRaster to disk |
| . | |
| **Advanced** | |
| blockSize | Get suggested block size for reading and writing |
| writeStart | Open a file for writing |
| writeValues | Write some values |
| writeStop | Close the file after writing |

## XIV. SpatDataSet objects

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

| | |
|---|---|
| sds | Create a SpatDataSet |
| [ or $ | Extract a SpatRaster |
| names | Get the names of the sub-datasets |

## XV. Manipulation of SpatVector objects

The name in **bold** is the equivalent command in ArcGIS.

| | |
|---|---|
| c | **append** vector data of the same (vector) type ("rbind") |
| erase or "-" | **erase** parts of a polygons |
| intersect or "*" | **intersect** polygons |
| union or "+" | **union** polygons |
| cover | **update** and **identity** a polygons |
| symdif | **symmetrical difference** of two polygons |

| | |
|---|---|
| `aggregate` | **dissolve** smaller polygons into larger ones |
| `disaggregate` | **explode**: turn polygon parts into separate polygons |
| `crop` | **clip** vector data using a rectangle (SpatExtent ) |
| `select` | **select** - interactively select spatial features |
| `click` | **identify** attributes by clicking on a map |
| `merge` | **Join table** |
| `fill` | remove holes from polygons |
| `extract` | spatial queries between SpatVector and SpatRaster objects |
| `as.data.frame` | get attributes as a data.frame |

## XVI. SpatExtent objects

| | |
|---|---|
| `extent` | Create a SpatExtent object |
| `intersect` | Intersect two SpatExtent objects |
| `union` | Combine two SpatExtent objects |
| `Math-methods` | round/floor/ceiling of the coordinates of a SpatExtent |
| `align` | Align a SpatExtent with a SpatRaster |
| `draw` | Create a SpatExtent object by drawing it on top of a map (plot) |

## XVII. Miscellaneous

| | |
|---|---|
| `terraOptions` | Show, set, save or get session options |
| `sources` | Show the data sources of a SpatRaster |
| `tmpFiles` | Show or remove temporary files |
| `canProcessInMemory` | Test whether a file can be created in memory |
| `readStart` | Open file connections for efficient multi-chunck reading |
| `readStop` | Close file connections |
| `inMemory` | Are the cell values in memory? |
| `fromDisk` | Are the cell values read from a file? |

## Acknowledgments

This package is an attempt to stand on the shoulders of giants (GDAL, PROJ, GEOS, GeographicLib, Rcpp, R)

## Author(s)

Except where indicated otherwise, the methods and functions in this package were written by Robert Hijmans

---

`adjacent` *Adjacent cells*

---

## Description

Identify cells that are adjacent to a set of cells on a raster.

## Usage

```
## S4 method for signature 'SpatRaster'
adjacent(x, cells, directions, include, ...)
```

## 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` | the directions in which cells should be connected: "rook" (4 directions), "queen" (8 directions), "16" (knight and one-cell queen moves), or "bishop" to connect cells with one-cell diagonal moves. |
| `include` | logical. Should the focal cells be included in the result? |
| `...` | additional arguments. None implemented |

## Value

vector with adjacent cells.

## 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")
# global lat/lon wraps around
r <- rast(nrows=10, ncols=10, crs="+proj=longlat +datum=WGS84")
adjacent(r, cells=11, directions="rook")
```

## aggregate *Aggregate raster cells*

### 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", ..., nodes=1, filename="", overwrite=FALSE, wopt=list())

## S4 method for signature 'SpatVector'
aggregate(x, by=NULL, sums=NULL, dissolve=TRUE, vars=NULL, ...)
```

### 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 a character specifying a C++ level implemented function: "mean", "max", "min", "median", "sum" and "modal" |
| `...` | additional arguments passed to `fun`, such as `na.rm=TRUE` |
| `nodes` | positive integer. If `nodes` > 1, a 'parallel' package cluster with that many nodes is created. Ignored for C++ level implemented functions "mean", "max", "min", "median", "sum" and "modal" |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `by` | character |
| `sums` | list |
| `dissolve` | logical |
| `vars` | character |

## 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

`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)
```

---

`align` *Align a SpatExtent with a SpatRaster*

---

## Description

Align an SpatExtent object 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).

## Usage

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

## Arguments

| | |
|---|---|
| `x` | SpatExtent |
| `y` | SpatRaster |
| `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 |
| `...` | additional arguments. None implemented |

## 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
```

---

`app` *Apply a function to the cells of a SpatRaster*

---

## Description

Apply a function to values of each cell of a SpatRaster. This is similar to `apply` – think of each layer in a SpatRaster as a column (or row) in a matrix.

You can also use a SpatDataSet to apply `fun` across datasets by layer.

## Usage

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

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatDataSet |
| `fun` | function |
| `...` | additional arguments for `fun` |
| `nodes` | positive integer. If `nodes` > `1`, a 'parallel' package cluster with that many nodes is created. Ignored for C++ level implemented functions "max", "min", "mean", "range", "prod", "sum", "any", and "all" |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. 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)


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

## area *Area and perimeter*

### Description

Compute the area of polygons or for raster cells that are not NA. Computing the surface area of raster cells is particularly relevant for longitude/latitude rasters, as the size of the cells is constant in degrees, but not in meters.

The perimeter method works only on SpatVector objects, and computes the length of lines or the perimeter of polygons.

### Usage

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

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

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `sum` | logical. If `TRUE` the summed area of the cells that are not `NA` is returned. Otherwise, a SpatRaster with the area for each cell is returned |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

If the coordinate reference system is longitude/latitude the values returned are in square meter for area and meter for perimeter.

In other cases, the unit is set by coordinate reference system. In most cases it is meter too.

### Examples

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

```
# area for each raster cell
a <- area(r, sum=FALSE)

# summed area in km2
area(r) / 1000000

## you can use mask to remove the cells in r that are NA
## and compute the global sum to get the same result
#am <- mask(a, r)
#global(am, "sum", na.rm=TRUE) / 1000000

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

## 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 |
| `...` | additional arguments. None implemented |

### Value

character

### Examples

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

---

## as.data.frame *Get the attributes of a SpatVector*

---

### Description

Get a data.frame or list with the attribute values of a SpatVector

### Usage

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

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

### Arguments

| | |
|---|---|
| `x` | SpatVector |
| `row.names` | ignored |
| `optional` | ignored |
| `...` | additional arguments (none) |

### Value

data.frame

### Examples

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

## 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(nrow=10, ncol=10)
r2 <- rast(nrow=10, ncol=10)
values(r1) <- (runif(ncell(r1))-0.5) * 10
values(r2) <- (runif(ncell(r1))-0.5) * 10
atan2(r1, r2)
```

## 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, type="inner",
         directions=8, filename="", overwrite=FALSE, wopt=list(), ...)
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `type` | character. "inner" or "outer" |
| `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) |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

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

## See Also

`focal`, `clump`

## Examples

```
r <- rast(nrow=18, ncol=36, xmn=0)
v <- rep(NA, ncell(r))
v[150:250] <- 1
v[251:450] <- 2
values(r) <- v
bi <- boundaries(r, type="inner")
bo <- boundaries(r, type="outer")
bc <- boundaries(r, classes=TRUE)
plot(bc)
```

---

# `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 objcts in 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="", overwrite=FALSE, wopt=list(), ...)

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |
| `quadsegs` | postive integer. Number of line segments to use to draw a quart circle |
| `capstyle` | character. Style of cap to use at the ends of the geometry. Allowed values: "round", "flat", "square" (ignored for now) |

## Value

SpatRaster

## See Also

`distance`

## Examples

```
r <- rast(ncol=36,nrow=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)))
b <- buffer(v, 20)
plot(b)
points(v)

crs(v) <- "+proj=longlat +datum=WGS84"
b <- buffer(v, 1500000)
plot(b)
points(v)
```

## c *Combine SpatRasters*

### Description

Combine SpatRaster objects

### Usage

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

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatDataSet |
| `...` | SpatRaster objects to be combined with `x` |

### Value

SpatRaster

### Examples

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

## cells *Get cell numbers*

### Description

Get the cell numbers covered by a SpatVector or SpatExtent. Or all non `NA` values.

### Usage

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

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

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `y` | SpatVector, SpatExtent, 2-column matrix representing points, or missing |
| `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 |
| `method` | character. method for extracting values with points. The default is "simple", the alternative is "bilinear" |
| `...` | additional arguments. None implemented |

### Value

matrix if `y` is a `SpatVector`, otherwise a vector.

### Examples

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

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

v <- vect(m)
cells(r, v)

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

#z  <- cells(r,ext(v))
#xy <- xyFromCell(r, z)
#plot(v)
#points(xy)
```

## `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="", overwrite=FALSE, wopt=list(), ...)
```

## 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## 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.

Classification can be based on ranges "from-to-becomes" or on specific values "is-becomes", or on "cuts".

With "from-to-becomes" or "is-becomes", classification is done with matrix `rcl`, 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="", overwrite=FALSE, wopt=list(), ...)
```

## 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.<br>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`.<br>A single column matrix or a vector is interpreted as a set of cuts. In that case the values are classified based on their location inbetween the cut-values |
| `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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster

## Note

For model-based classification see `predict`

## Examples

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

```
# equivalent to
rc2 <- classify(r, c(0, 0.25, 0.5, 1), include.lowest=TRUE)

# is becomes
x <- round(r*5)
unique(x)
m <- rbind(c(1,100), c(2,200))
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 values of a SpatRaster at that location; and optionally the coordinates and cell number of the location. For SpatVectors you need to click twice (draw a box).

### Usage

```
## S4 method for signature 'SpatRaster'
click(x, n=Inf, id=FALSE, xy=FALSE, cell=FALSE, type="n", 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 spatialy 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 clicking the second mouse button and selecting 'Stop' from the menu, or from the 'Stop' menu on the graphics window.

## See Also

draw

## Examples

```
r <-rast(system.file("ex/test.tif", package="terra"))
plot(r)
click(r)
 # now click on the plot (map)
```

---

`coerce` *Coercion to other object types*

---

## Description

Coercion to other object types or other vector 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.data.frame(x, xy=FALSE, cells=FALSE, ...)

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

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

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

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

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

## S4 method for signature 'SpatVector'
as.points(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `wide` | logical |
| `xy` | logical |
| `cells` | logical |
| `mode` | this argument is ignored |
| `trunc` | logical; truncate values to integers. If `FALSE` the object returned can be very large |
| `dissolve` | logical; combine cells with the same values? |
| `values` | logical; include cell values as attributes? If `FALSE` the cells are not dissolved and the object returned can be very large |
| `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 |
| `crs` | character. The coordinate reference system |
| `...` | additional arguments. None implemented |

## Value

vector, matrix, array, data.frame or SpatVector

## Examples

```
r <- rast(ncol=2, nrow=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)
as.points(r)

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

---

`compareGeom` *Compare geometries of SpatRaster objects*

---

## 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=TRUE,
          crs=TRUE, warncrs=FALSE, ext=TRUE, rowcol=TRUE, res=FALSE)
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `y` | SpatRaster |
| `...` | Additional SpatRaster objects |
| `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) |

## 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/test.tif", package="terra"))
plot(r)
contour(r, add=TRUE)

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

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

## cover — *Cover (replace) NA values with values of another raster*

### Description

Replace NA values (or another value) in SpatRaster (x) with the values of SpatRaster (y)

### Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
cover(x, y, value=NA, filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| x | SpatRaster |
| y | SpatRaster |
| value | numeric. The cell values in x to be replaced by the values in y |
| filename | character. Output filename. Optional |
| overwrite | logical. If TRUE, filename is overwritten |
| wopt | list. Options for writing files as in writeRaster |
| ... | additional arguments. None implemented |

### 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)
```

## crop — *Cut out a geographic subset*

### Description

Cut out a part of a SpatRaster with a SpatExtent, or another object from which an extent can be obtained. You can only extract rectangular areas, but see mask for setting cell values within SpatRaster to NA.

## Usage

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

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `y` | SpatExtent or other object that has a SpatExtent; or SpatVector if `x` is a SpatVector |
| `snap` | character. One of "near", "in", or "out" |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster

## 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)
```

---

`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* spatial from one CRS to another.

## Usage

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

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

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

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `value` | character string describing a coordinate reference system. This can be in a WKT format, as a EPSG code, or a PROJ.4 "+" format (but see Note) |
| `...` | additional arguments (none implemented) |

## Value

character or modified SpatRaster/Vector

## Note

Because of changes in the PROJ library that is behind the coordinate transformations, when using the PROJ.4 format, the datum should be WGS84 – if you want to transform your data to a different coordinate reference system)

## Examples

```
r <- rast()
crs(r)
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"
```

---

`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 |
| `...` | additional arguments. None implemented |

## 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 a file with raster data. (using the "GDALinfo" tool)

## Usage

```
describe(filename, options="", print=TRUE, open_opt="", ...)
gdal_version()
```

## Arguments

| | |
|---|---|
| `filename` | character |
| `options` | character. A vector of valid options 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 of a sub-dataset. See `https://gdal.org/programs/gdalinfo.html` |
| `print` | logical. If `TRUE`, print the results |
| `open_opt` | character. Driver specific open options |
| `...` | additional arguments. None implemented |

## Value

character (invisibly, if `print=FALSE`)

## Examples

```
f <- system.file("ex/test.tif", package="terra")
describe(f)

#g <- describe(f, c("json", "nomd", "proj4"), print=FALSE)
#cat(g, "\n")
```

---

## diff *diff*

---

### Description

Returns lagged differences.

### Usage

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatRaster

### Examples

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

`dimensions` *Dimensions of a SpatRaster or SpatVector*

## Description

Get the number of rows (`nrow`), columns (`ncol`), cells (`ncell`), layers (`nlyr`), sources (`nsrc`) or the spatial resolution of a SpatRaster; or sub-datasets of a SpatDataSet (`length`).

The `size` of a SpatRaster `x` is `ncell(x) * nlyr(x)`. 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'
size(x)


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

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

ncol(x, ...) <- value
nrow(x, ...) <- value
nlyr(x, ...) <- value
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'
size(x, ...)

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `value` | For ncol and nrow: positive integer. For res: one or two positive numbers |
| `...` | additional arguments. None implemented |

## Value

integer

## See Also

ext

## Examples

```
r <- rast()
ncol(r)
nrow(r)
nlyr(r)
dim(r)
nsrc(r)
ncell(r)
size(r)

rr  <- c(r,r)
nlyr(rr)
nsrc(rr)
ncell(rr)
size(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)
```

## disaggregate *Disaggregate raster cells*

### 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'
disaggregate(x, fact, method="near", filename="", overwrite=FALSE, wopt=list(), ...)

## S4 method for signature 'SpatVector'
disaggregate(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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatRaster

### See Also

`aggregate`, `resample`

## Examples

```
r <- rast(ncols=10, nrows=10)
rd <- disaggregate(r, fact=c(10, 2))
ncol(rd)
nrow(rd)
values(r) <- 1:ncell(r)
rd <- disaggregate(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 (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="", overwrite=FALSE, wopt=list(), ...)

## S4 method for signature 'SpatRaster,SpatVector'
distance(x, y, filename="", overwrite=FALSE, wopt=list(), ...)

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

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `y` | missing or SpatVector |

| | |
|---|---|
| `grid` | logical. If `TRUE`, distance is computed using a path that goes through the centers of the 8 neighboring cells |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |
| `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 |

## Value

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

The unit is in meters if the CRS is `longlat` and in map units (typically also meters) when it is not.

## Examples

```
r <- rast(ncol=36,nrow=18)
v <- rep(NA, ncell(r))
v[500] <- 1
values(r) <- v
d <- distance(r)

plot(d / 100000)

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)
```

---

## `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. To finish, right-click and select "stop".

### Usage

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

### 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 |
| `...` | additional arguments passed to `locator` |

### Value

SpatVector or SpatExtent

### See Also

`locator`

---

## `expand` *Expand*

---

### Description

Expand the 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'
expand(x, y, filename="", overwrite=FALSE, wopt=list(), ...)

## S4 method for signature 'SpatExtent'
expand(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 a numeric vector of length 2 indicating the number of rows and columns that need to be added (or a single number when the number of rows and columns is equal)<br>If `x` is a SpatExtent, `y` should be a numeric vector of 1, 2, or 4 elements |

| | |
|---|---|
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster or SpatExtent

## See Also

`crop`, `merge`, `ext`

## Examples

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

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

# SpatExtent object
e <- ext(r)
e
expand(e, 10)
expand(e, 10, -10, 0, 20)
```

---

`extent` *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 replacement method for signature 'SpatRaster,SpatExtent'
ext(x)<-value

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

### Arguments

| | |
|---|---|
| x | SpatRaster |
| value | SpatExtent, or numeric vector of lenght four (xmin, xmax, ymin, ymax) |
| ... | additional arguments. None implemented |

### 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.

### Usage

```
## S4 method for signature 'SpatRaster,SpatVector'
extract(x, y, fun=NULL, ..., touches=is.lines(y), method="simple", list=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` | character. function to summarize the data. Currently ignored |
| `...` | additional arguments passed to the `SpatRaster,SpatVector` method if `y` is a matrix, and passed to ‘fun‘ if `y` is a SpatVector |
| `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 |
| `method` | character. method for extracting values with points. The default is "simple", the alternative is "bilinear" |
| `list` | logical. If `TRUE`, the output is simplified |

## Value

data.frame

## See Also

values

## Examples

```
r <- rast(ncol=5, nrow=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/test.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)
#z <- rast(v)
#v <- v[1:2,]
#values(z) <- 1:100
#e <- extract(z, v)
#e
#tapply(e[,2], e[,1], mean)

#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)
```

---

## `factors` *Factors*

---

### Description

These functions allow for defining a SpatRaster layer as a categorical variable. The cell values are an index (id), whereas the actual values are stored seperately, in a table (sometimes called "Raster Attribute Table"). This table is a data.frame. The first column in the RAT ("ID") has the unique cell values of the layer; this column should normally not be changed. The other columns can be of any basic type (factor, character, integer, numeric or logical).

Function 'levels' returns the RAT for inspection. It can be modified and set using `levels(x) <-value`.

`as.factor` and `ratify` create a layer with a RAT table. `deratify` creates a single layer for a (or each) variable in the RAT table.

### Usage

```
is.factor(x)
as.factor(x)
levels(x)
## S4 method for signature 'SpatRaster'
rats(x)
setRat(x, layer, rat)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `layer` | layer number |
| `rat` | data.frame |

### Value

SpatRaster; list (levels); boolean (is.factor)

### Examples

```
set.seed(0)
r <- rast(nrow=10, ncol=10)
values(r) <- sample(3, ncell(r), replace=TRUE) + 2
is.factor(r)
cls <- c("forest", "water", "urban")
levels(r) <- cls



f <- as.factor(r)
is.factor(f)
```

---

## `fill` *Remove holes from polygons*

---

### Description

Remove the holes in SpatVector polygons

### Usage

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

### Arguments

| | |
|---|---|
| `x` | SpatVector |
| `...` | additional arguments. None implemented |

### Value

character

### 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')
z <- data.frame(z)
p <- vect(z, "polygons")
p

f <- fill(p)

plot(f, col="light blue", lwd=8, border="blue")
lines(p, lwd=2, col="red")
```

---

## flip *Flip a raster*

---

### Description

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

### Usage

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `vertical` | logical. If `TRUE`, values are flipped by rows, if `FALSE`, values are flipped by columns |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatRaster

### See Also

transpose, rotate

### Examples

```
r <- rast(nrow=18, ncol=36)
m <- matrix(1:ncell(r), nrow=18)
values(r) <- as.vector(t(m))
rx <- flip(r, vertical=FALSE)
values(r) <- as.vector(m)
ry <- flip(r, vertical=TRUE)
```

---

## 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, na.rm=TRUE, na.only=FALSE, fillvalue=NA, fun="sum",
            filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `w` | window. The window can be defined as one (for a square) or two numbers (row, col); or with an odd-sized weights matrix . See Details. If `w` is a matrix, `na.rm=FALSE` and `fun=sum`, and cannot be changed |
| `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 |
| `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 `...` |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## 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.

Example weight matrices

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

Sobel filter: `filter=matrix(c(1,2,1,0,0,0,-1,-2,-1) / 4,nrow=3)`

## Value

SpatRaster

## 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=36, nrows=18, xmn=0)
# Gaussian filter for square cells
gf <- focalMat(r, 2, "Gauss")
```

---

## `freq` *Frequency table*

---

### Description

Frequency table of the values of a SpatRaster. `NA`s are not counted.

### Usage

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `bylayer` | logical. If `TRUE` tabulation is done by layer |
| `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 |
| `...` | additional arguments (none implemented) |

### Value

matrix with 2 columns (value, count) or, if `bylayer=TRUE` three layers (layer, value, count).

### See Also

`freq`

### Examples

```
r <- rast(nrow=10, ncol=10)
set.seed(2)
values(r) <- sample(5, ncell(r), replace=TRUE)

freq(r, FALSE)
freq(r)
```

```
x <- c(r, r/3)
freq(x, FALSE)
freq(x)
freq(x, digits=1)
freq(x, digits=-1)

freq(x, value=5)
```

---

`geom` *Get the geometry (coordinates) of a SpatVector*

---

## Description

Get the geometry of a SpatVector. This is a five-column matrix: 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).

## Usage

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

## Arguments

| | |
|---|---|
| `x` | SpatVector |
| `...` | additional arguments. None implemented |

## 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(-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))
x4 <- rbind(c(80,0), c(105,13), c(120,2), c(105,-13))
z <- rbind(cbind(object=0, part=0, x1), cbind(object=1, part=0, x2),
           cbind(object=2, part=0, x3), cbind(object=2, part=1,  x4))
colnames(z)[3:4] <- c('x', 'y')
z <- data.frame(z)
z$hole <- 0
z$hole[z$object==2 & z$part==1] <- 1
```

```
p <- vect(z)
geom(p)

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

## geomtype *Geometry type of a SpatVector*

### Description

Get the geometry type (points, lines, or polygons) of a SpatVector

### Usage

```
## S4 method for signature 'SpatVector'
geomtype(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 |
| `...` | additional arguments. None implemented |

### Value

character

| | |
|---|---|
| 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 character:"mean", "min", "max", "sum"; 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(nrow=25, ncol=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 SpatRaster objects*

---

## Description

This functions like `ifelse` but this one works for SpatRaster objects. This method allows for a concise approach to what can otherwise be achieved with a combination of `classify`, `mask`, and `cover`.

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

## Usage

```
## S4 method for signature 'SpatRaster'
ifel(test, yes, no, filename="", overwrite=FALSE, wopt=list(), ...)
```

## Arguments

| | |
|---|---|
| `test` | SpatRaster |
| `yes` | SpatRaster or numeric |
| `no` | SpatRaster or numeric |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster

## Examples

```
r <- rast(nrows=5, ncols=5)
values(r) <- -10:14

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

y <- ifel(r > 1, 1, ifel(r < -1, -1, r))

z <- ifel(r > -2 & r < 2, 100, 0)

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/test.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 Raster* objects with many cells.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `fun` | function to be applied. This must be a either a single number, a function, or one of a set of 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster

## Examples

```
r <- rast(ncol=10, nrow=5, xmn=0, xmx=10, ymn=0, ymx=5)
x <- init(r, fun="cell")
y <- init(r, fun=runif)

# initialize with a single value
z <- init(r, fun=8)
```

---

`interpolate` *Interpolate*

---

## Description

Make a RasterLayer with interpolated values using a fitted model object of classes such as "gstat" (gstat package) or "Krige" (fields package). That is, these are models that have location ("x" and "y", or "longitude" and "latitude") as independent variables. If x and y are the only independent variables 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 Raster* object 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 the model object) |
| `...` | 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 many models as the predict function will extract the levels from the `model` object |

| | |
|---|---|
| `const` | data.frame. Can be used to add a constant for which there is no Raster object 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |

## Value

SpatRaster

## See Also

`predict`

---

## `isLonLat` *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).

### Usage

```
## S4 method for signature 'SpatRaster'
isLonLat(x, perhaps=FALSE, warn=TRUE, global=FALSE, ...)

## S4 method for signature 'SpatVector'
isLonLat(x, perhaps=FALSE, warn=TRUE, ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector object |
| `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 |
| `...` | additional arguments. None implemented |

## Value

logical

## Examples

```
r <- rast()
isLonLat(r)
isLonLat(r, global=TRUE)

crs(r) <- ""
isLonLat(r)
isLonLat(r, perhaps=TRUE, warn=FALSE)

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

---

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

---

## Description

Apply a function to layers of a SpatRaster ("overlay").

The number of arguments in function `fun` must match the number of layers in the SpatRaster (or the number of sub-datasets in the SpatDataSet). 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 works for vectors (not only for single numbers). That is, it must return the same number of elements as its input vectors, or multiples of that. Make sure it also returns the same number of values when some or all input values are `NA`. Also, the function must return a vector or matrix, not a `data.frame`.

Use `app` for summarize functions such as `sum`, that take any number of arguments; and `tapp` to 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 'SpatDataSet'
lapp(x, fun, ..., recycle=FALSE, filename="", overwrite=FALSE, wopt=list())
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatDataSet |
| `fun` | function to be applied |
| `...` | 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |

## Value

SpatRaster

## See Also

`app`, `tapp`, `math`

## Examples

```
s <- rast(system.file("ex/logo.tif", package="terra"))
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 give an error because
# there are too many layers in s
# x5 <- lapp(s, fvi2, usenames=FALSE)

pairs(c(x1, x2, x3, x4))

## SpatDataSet
x <- sds(s, s[[1]]+50)
lapp(x, function(x, y) x/y, recycle=TRUE)
```

---

`local` *Local statistics*

---

## Description

Compute cell (pixel) level "local" statistics 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`

To compute statistics that are not included here see `app` to summarize across layers.

See `modal` to compute the mode.

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.

The `stdev` method returns 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). Function `f` above is equivalent to function `g` below

`g <-function(x) sqrt(sum((x-mean(x))^2) / length(x))`

## 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, ..., 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 |
| `trim` | ignored |
| `na.rm` | logical. If `TRUE`, `NA` values are ignored. If `FALSE`, `NA` is returned if `x` has any `NA` values |

## Value

SpatRaster

## See Also

`Math-methods`,`modal`

## Examples

```
set.seed(0)
r <- rast(nrow=10, ncol=10, nlyr=3)
values(r) <- runif(size(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)
```

## 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, maskvalue=NA,
   updatevalue=NA, filename="", overwrite=FALSE, wopt=list(), ...)

## S4 method for signature 'SpatRaster,SpatVector'
mask(x, mask, inverse=FALSE, updatevalue=NA,
touches=is.lines(mask), filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `mask` | SpatRaster |
| `inverse` | logical. If `TRUE`, areas on mask that are _not_ the `maskvalue` are masked |
| `maskvalue` | numeric. The value 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatRaster object

### See Also

crop

## Examples

```
r <- rast(ncol=10, nrow=10)
m <- rast(ncol=10, nrow=10)
values(r) <- 1:100
x <- runif(ncell(r))
x[x < 0.5] <- NA
values(m) <- x
mr <- mask(r, m)
```

---

## `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","log`

For **SpatExtent** the following methods have been implemented: `"round","floor","ceil","=="`

### Usage

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

### Arguments

|   |   |
|---|---|
| `x` | SpatRaster |

### Value

SpatRaster

### 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)
```

---

## 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.

### Usage

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

## S4 method for signature 'SpatExtent,SpatExtent'
merge(x, y, ...)

## S4 method for signature 'SpatVector,data.frame'
merge(x, y, ...)
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatExtent object |
| `y` | object of same class as `x` |
| `...` | if `x` is a SpatRaster or SpatVector: additional objects of the same class as `x`. If `x` is a SpatVector, the same arguments as in `merge` |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. 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

## 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)
mr <- merge(x, y)
plot(mr)
mr <- merge(y, x)

# if you have many SpatRaster objects in a list
# you can use do.call:
s <- list(x, y)
# add arguments such as filename
s$filename <- ""
m <- do.call(merge, s)

##
# 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. 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)
```

`names` *Names of Spat objects*

## Description

Get or set the names of the layers of a SpatRaster or the attributes of a SpatVector.

## Usage

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

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

## S4 method for signature 'SpatDataSet'
names(x)

## S4 replacement method for signature 'SpatDataSet'
names(x)<-value

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

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `value` | character (vector) |

## Value

character

## Examples

```
s <- rast(ncols=5, nrows=5, nlyr=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)
```

---

## options *Options*

---

### Description

Class and methods for showing and setting general options for terra.

### Usage

```
terraOptions(...)
```

### Arguments

| | |
|---|---|
| `...` | additional arguments. Can be use to set options (see examples). When empty, the current options are shown |

### Details

The following options are available.

memfrac - value between 0.1 and 0.8. 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`

filetype - default file type. See `writeRaster`

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

Origin returns the coordinates of the point of origin of a SpatRaster object. 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

```
origin(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `...` | additional arguments. None implemented |

### Value

A vector of two numbers (x and y coordinates)

### Examples

```
r <- rast(xmin=-0.5, xmax = 9.5, ncols=10)
origin(r)
```

## `pack` *pack a Spat* object*

### Description

Pack a Spat* object so 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'
pack(x, ...)

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

### Arguments

| | |
|---|---|
| `x` | SpatVector or SpatRaster |
| `...` | additional arguments. None implemented |

## Value

Packed* object

## Examples

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

---

`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 large Raster objects |
| `...` | additional arguments (graphical parameters) |

## See Also

`boxplot`,`hist`

## Examples

```
r <-rast(system.file("ex/test.tif", package="terra"))
s <- c(r*1, 1/r, sqrt(r))
pairs(s)

# to make indvidual histograms:
hist(r)
# or scatter plots:
plot(r, 1/r)
```

`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/test.tif", package="terra"))
persp(r)
```

`plot` *Plot a SpatRaster*

## Description

Plot (that is, make a map of) the values of a SpatRaster or SpatVector

## Usage

```
## S4 method for signature 'SpatRaster,numeric'
plot(x, y=1, col,
  type, mar=c(5.1, 4.1, 4.1, 7.1), legend=TRUE, axes=TRUE,
  pal=list(), pax=list(), maxcell=50000, ...)

## S4 method for signature 'SpatRaster,missing'
plot(x, y, maxcell=50000, nc, nr, main,
   maxnl=16, ...)
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `y` | missing or positive integer indicating the layer(s) to be plotted, or the name of the layer to be mapped |
| `col` | character. Colors |
| `type` | character. Type of map/legend. One of "continuous", "classes", or "internval" |
| `mar` | numeric vector of lenght 4 to set the margins of the plot |
| `legend` | logical. Draw a legend? |
| `axes` | logical. Draw axes? |
| `pal` | list. Parameters for the legend |
| `pax` | list. Parameters for drawing axes |
| `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) |
| `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) |
| `...` | additional graphical arguments |

## See Also

`image`, `plotVector`, `scatterplot`

## Examples

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

plot(r, type="interval")

e <- c(178200,178400,331600,333600)
plot(r, pal=list(ext=e, title="Title\n", title.cex=1.5))

d <- (r > 400) + (r > 600)
plot(d)
plot(d, type="classes")

plot(d, type="interval", levels=0:3)
plot(d, type="interval", levels=3, pal=list(legend=c("0-1", "1-2", "2-3")))
plot(d, type="classes", pal=list(legend=c("M", "X", "A")))

x <- trunc(r/600)
x <- as.factor(x)
```

```
levels(x) <- c("earth", "wind", "fire")
plot(x)


# two plots with the same legend
dev.new(width=6, height=4, noRStudioGD = TRUE)
par(mfrow=c(1,2))
plot(r, range=c(100,1800))
plot(r/2, range=c(100,1800))

# as you only need one legend:
par(mfrow=c(1,2))
plot(r, range=c(100,1800), mar=c(4, 3, 4, 3), pal=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(100,1800), mar=c(4, 2, 4, 4), legend=FALSE,
pax=list(sides=c(1,4), cex.axis=.6))
```

---

## 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, scale, maxcell=500000, stretch=NULL,
ext=NULL, interpolate=FALSE, colNA="white", alpha, bgalpha, addfun=NULL, zlim=NULL,
zlimcol=NULL, axes=FALSE, xlab="", ylab="", asp=NULL, add=FALSE, margins=FALSE, ...)
```

### 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) |
| `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 |
| `stretch` | character. Option to stretch the values to increase the contrast of the image: "lin" or "hist" |

| | |
|---|---|
| `ext` | An `Extent` object to zoom in to a region of interest (see `drawExtent`) |
| `interpolate` | logical. If `TRUE`, interpolate the image when drawing |
| `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 |
| `margins` | logical. If `TRUE` standard whitespace margins are used. If `FALSE`, graphics::par(plt=c(0,1,0,1)) is used |
| `...` | graphical parameters as in `plot` or `rasterImage` |

## See Also

`plot`

## Examples

```
b <- rast(system.file("ex/logo.tif", package="terra"))
plotRGB(b)
plotRGB(b, 3, 2, 1)
plotRGB(b, 3, 2, 1, stretch='hist')
```

---

## `plotVector` *Plot a SpatVector*

---

## Description

Plot (that is, make a map of) using a SpatVector

## Usage

```
## S4 method for signature 'SpatVector,missing'
plot(x, y, col,
   axes=TRUE, add=FALSE, ...)

## S4 method for signature 'SpatVector,character'
plot(x, y, col, type,
   mar=c(5.1, 4.1, 4.1, 7.1), axes=TRUE, add=FALSE, ...)

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

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

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

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

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

## Arguments

| | |
|---|---|
| `x` | SpatVector |
| `y` | missing or positive integer indicating the layer(s) to be plotted, or the name of the layer to be mapped |
| `col` | character. Colors |
| `type` | character. Type of map/legend. One of "continuous", "classes", or "internval" |
| `axes` | logical. Draw axes? |
| `mar` | numeric vector of lenght 4 to set the margins of the plot (to make space for the legend) |
| `add` | logical. If `TRUE` add the object to the current plot (for points, lines, and polygons this is an alternative to the `lines` or `points` methods |
| `...` | additional graphical arguments |

## Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
plot(v)
plot(v, "ID_1")
```

```
r <- rast(v)
values(r) <- 1:ncell(r)
plot(r)
lines(v)
points(v)
```

---

## 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, 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 |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. 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)
```

---

`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="bilinear", mask=FALSE,
            filename="", overwrite=FALSE, wopt=list(), ...)
```

## Arguments

| | |
|---|---|
| `x` | SpatVector |
| `y` | SpatRaster or character if `x` is a SpatRaster; character if `x` is a SpatVector. The character should be a PROJ.4 description of the output coordinate reference system (crs). If `x` is a SpatVector, you can also provide another object from which a crs can be extracted with `crs` |
| `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. bilinear: bilinear interpolation. Default. cubic: cubic interpolation. cubicspline: cubic spline interpolation. |
| `mask` | logical. If `TRUE`, mask out areas outside the input extent (see example with Robinson projection) |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatVector or SpatRaster

### See Also

`resample`

### Examples

```
## SpatRaster
a <- rast(ncol=40, nrow=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(ncol=94, nrow=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` *SpatRaster local quantiles*

---

### Description

Compute quantiles for each cell across the layers of a SpatRaster

### Usage

```
## S4 method for signature 'SpatRaster'
quantile(x, probs=seq(0, 1, 0.25), na.rm=FALSE,
    filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments, none implemented |

### Value

A vector of quantiles

### See Also

`app`, `local`

### Examples

```
r <- rast(system.file("ex/logo.tif", package="terra"))
r <- c(r/2, r, r*2)
q <- quantile(r)
q

# same as (but faster than)
qa <- app(r, quantile)
```

---

## `range` *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)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |

### 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/test.tif", package="terra"))
minmax(r)
```

rapp *Apply a function to a range of the layers of a SpatRaster*

## Description

Apply a function to a range of the layers of a SpatRaster. The range is specified for each cell seperately by a two-layer SpatRaster `index`.

The function used should return a single value.

See `app` or `Summary-methods` if you want to apply a function to all layers (or a subset of all layers) in a SpatRaster.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `index` | factor or numeric (integer). Vector of length `nlayers(x)` (shorter vectors are recycled) grouping the input layers |
| `fun` | function to be applied |
| `...` | additional arguments passed to `fun` |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |

## Value

SpatRaster

## See Also

`app`, `Summary-methods`, `lapp`, `tapp`

## Examples

```
r <- rast(ncol=9, nrow=9)
values(r) <- 1:ncell(r)
s <- c(r, r, r, r, r, r)
s <- s * 1:6

start <- end <- rast(r)
start[] <- 1:3
end[]   <- 4:6
index   <- c(start, end)
```

```
rapp(s, index, fun="mean")
```

## 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 (initially) not contain any cell (pixel) values in memory (RAM), it only has the parameters that describe the SpatRaster. You can access cell-values with `values`.

### Usage

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

## S4 method for signature 'missing'
rast(x, nrows=180, ncols=360, nlyrs=1, xmin=-180, xmax=180,
                                ymin=-90, ymax=90, crs, extent, resolution, ...)

## S4 method for signature 'SpatRaster'
rast(x, nlyrs=nlyr(x), ...)

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

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

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

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

## S4 method for signature 'SpatDataSet'
rast(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | filename (character), SpatRaster, SpatDataSet, SpatExtent, SpatVector, matrix, array, list of SpatRaster objects, or Raster* object (from the "raster" package). Can also be missing |

| | |
|---|---|
| `subds` | positive integer or character to select a subdataset. If zero or "", all subdatasets are returned (if possible) |
| `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) |
| `extent` | object of class SpatExtent. If present, the arguments xmin, xmax, ymin and ymax are ignored |
| `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 |
| `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 |
| `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. |
| `...` | additional arguments, in some cases passed on to the `rast,missing-method` |

## Value

SpatRaster

## Examples

```
# Create a SpatRaster from scratch
x <- rast(nrow=108, ncol=21, xmin=0, xmax=10)

# Create a SpatRaster from a file
f <- system.file("ex/test.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=1:nrow(x), background=NA, update=FALSE,
              touches=is.lines(x), filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatVector |
| `y` | SpatRaster |
| `field` | numeric. The values to be rasterized. Either a field name of `x`, a single number, or a vector with the same length as `x` |
| `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 will be updated, not just those on the line render path, or whose center point is within the polygon |
| `update` | logical. If `TRUE`, the values of the Raster* object are updated for the cells that overlap with the geometries of `x`. Default is `FALSE` |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatRaster

### See Also

`mask`

### Examples

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

#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, wopt=list(), ...)

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

## S4 method for signature 'SpatRaster,vector'
writeValues(x, v, start, nrows, ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `filename` | Character. Output filename. Optional |
| `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 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`) |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## 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", filename="", overwrite=FALSE, wopt=list(), ...)
```

## Arguments

| | |
|---|---|
| `x` | Raster* object to be rectified |
| `method` | character. Method used to for resampling. See `resample` |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster

## rep *Combine*

### 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

## Examples

```
r <- rast(ncol=5, nrow=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`, `disaggregate`, `expand` or `crop`.

### Usage

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster to be resampleed |
| `y` | SpatRaster that `x` should be resampled to |
| `method` | character. Method used for estimating the new cell values. One of:<br>`near`: nearest neighbour. This method is fast, and the preferred method if the cell values represent classes. It is not a good choice for continuous values.<br>`bilinear`: bilinear interpolation. Default.<br>`cubic`: cubic interpolation.<br>`cubicspline`: cubic spline interpolation.<br>`lanczos`: Lanczos windowed sinc resampling. |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatRaster

## See Also

aggregate, disaggregate, crop, project,

## Examples

```
r <- rast(nrow=3, ncol=3, xmin=0, xmax=10, ymin=0, ymax=10)
values(r) <- 1:ncell(r)
s <- rast(nrow=25, ncol=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)
```

---

`rotate` *Rotate a SpatRaster along longitude*

---

## Description

Rotate a SpatRaster that has x coordinates (longitude) 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="", overwrite=FALSE, wopt=list(), ...)
```

## Arguments

| | |
|---|---|
| `x` | Raster* object |
| `left` | logical. If `TRUE`, rotate to the left, else to the right |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster

## Examples

```
x <- rast(nrow=9, ncol=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)
```

---

## 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` | postive 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 SpatDataSet*

### Description

Methods to create a SpatDataSet. This is an object to hold "sub-datasets", each a SpatRaster that in most cases will have multiple layers.

sds_info provides information about sub-datasets in a file.

### Usage

```
## S4 method for signature 'character'
sds(x, ids=0, ...)

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

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

describe_sds(filename, ...)
```

### Arguments

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

### Value

SpatDataSet

### 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]
```

## `select` *Geometric subsetting*

### Description

Geometrically subset SpatRaster or SpatVector (to be done) by drawing on a plot (map).

### Usage

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `...` | additional arguments passed on to `draw` |

### Value

SpatRaster or SpatVector

### See Also

`click`,`crop`

### Examples

```
## Not run:
# select a subset of a RasterLayer
r <- rast(nrow=10, ncol=10)
values(r) <- 1:ncell(r)
plot(r)
s <- select(r) # now click on the map twice

# plot the selection on a new canvas:
x11()
plot(s)

## End(Not run)
```

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

## Description

Use a single layer SpatRaster object to select cell values from different layers in a multi-layer SpatRaster. The values of the SpatRaster to select layers (`y`) should be between `1` and `nlyr(x)` (values outside this range are ignored); they are also truncated to integers.

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="", overwrite=FALSE, wopt=list(), ...)
```

## 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

## Value

SpatRaster

## See Also

`rapp`, `tapp`, `extract`

## Examples

```
r <- rast(ncol=10, nrow=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)
```

---

## shift *Shift*

---

### Description

Shift the location of a SpatRaster or SpatVector object

### Usage

```
## S4 method for signature 'SpatRaster'
shift(x, dx=0, dy=0, filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `dx` | numeric. The shift in horizontal direction |
| `dy` | numeric. The shift in vertical direction |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

Same object type as `x`

### See Also

`flip`, `rotate`

### Examples

```
r <- rast(xmn=0, ymn=0, xmx=1, ymx=1)
r <- shift(r, dx=1, dy=-1)
```

## slope *Compute slopes*

### Description

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

### Usage

```
## S4 method for signature 'SpatRaster'
slope(x, neighbors=8, unit="degrees", filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | Single layer SpatRaster with elevation values. Values should have the same unit as the map units, or in meters when the crs is longitude/latitude |
| `unit` | Character. 'degrees' or 'tangent' |
| `neighbors` | Integer. Indicating how many neighboring cells to use to compute slope for any cell. Either 8 (queen case) or 4 (rook case) |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### 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 crs of SpatRaster `x` must be known (may not be `NA`), to be able to safely differentiate between planar and longitude/latitude data.

### 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 slope as a property of the DEM. Computers & Geosciences 24: 315-323

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

---

## 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 an use hasValues to check if a in-memory layer these actualy have values.

### Usage

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

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `...` | additional arguments. None implemented |

### Value

data.frame with the source names and the number of layers by source

### Examples

```
f <- system.file("ex/test.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
```

## `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 regular sample of a SpatRaster. Either get cell values, or (when `as.raster=TRUE`) get a new SpatRaster with the same extent, but fewer cells. Note that, in order to assure regularity when requesting a regular sample, the number of values returned may not be exactly the same as the `size` requested.

## Usage

```
## S4 method for signature 'SpatRaster'
spatSample(x, size, method="regular", replace=FALSE, as.raster=FALSE, cells=FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `size` | numeric. The sample size |
| `method` | character. Should be "regular" or "random" |
| `replace` | logical. If `TRUE`, sampling is with replacement (if `method="random"` |
| `as.raster` | logical. If `TRUE`, a SpatRaster is returned |
| `cells` | logical. If `TRUE`, cellnumbers are returned instead of values |
| `...` | additional arguments. None implemented |

## Details

In some cases you may want to know the location of the sampled cells. In that situation you can take a sample of the cell numbers and use extract. See examples.

In stead of `spatSample(x,size,method="random")` you can also use the equivalent base method `sample(x,size)`. The base method also works for `SpatVector`

## Value

numeric or SpatRaster

## Examples

```
f <- system.file("ex/test.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)
v <- r[cells]
xy <- xyFromCell(r, cells)
cbind(xy, v)

# regular
cells <- spatSample(r, 6, "regular", cells=TRUE)
v <- r[cells]
xy <- xyFromCell(r, cells)
cbind(xy, v)
```

---

## `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

---

## `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="", overwrite=FALSE, wopt=list(), ...)
```

## 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. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments as for `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, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `subset` | integer or character. Should indicate the layers (represented as integer or by their names) |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### 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
```

---

## `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 |
| `...` | additional arguments. None implemented |

## 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")
```

---

`summary` *Summary*

---

## Description

Summarize the values of SpatRaster. A sample is used for very large files.

## Usage

```
## S4 method for signature 'SpatRaster'
summary(object, size=100000, ...)
```

## Arguments

| | |
|---|---|
| `object` | SpatRaster |
| `size` | positive integer. Size of a regular sample used for large datasets |
| `...` | additional arguments. None implemented |

## 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

`global`,`quantile`

## Examples

```
set.seed(0)
r <- rast(nrow=10, ncol=10, nlyr=3)
values(r) <- runif(size(r))
summary(r)
```

## 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 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. index 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 `nlayers(x)` (shorter vectors are recycled) grouping the input layers |
| `fun` | function to be applied |
| `...` | additional arguments passed to `fun` |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |

### Value

SpatRaster

### See Also

`app`, `Summary-methods`

## Examples

```
r <- rast(ncol=10, nrow=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
```

---

`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)
```

## time *time of SpatRaster layers*

### Description

Get or set the time of the layers of a SpatRaster. Experimental. Currently only Date's allowed.

### Usage

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

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `value` | Date (vector) |
| `...` | additional arguments. None implemented |

### Value

Date

### See Also

`depth`

### Examples

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

time(s) <- as.Date("2001-05-04") + 0:2
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()
```

## `transpose` *Transpose*

### Description

Transpose a SpatRaster

### Usage

```
## S4 method for signature 'SpatRaster'
t(x)

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `filename` | character. Output filename. Optional |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files as in `writeRaster` |
| `...` | additional arguments. None implemented |

### Value

SpatRaster

### See Also

`flip`,`rotate`

### Examples

```
r <- rast(nrow=18, ncol=36)
values(r) <- 1:ncell(r)
tr1 <- t(r)
tr2 <- transpose(r)
ttr <- transpose(tr2)
```

## trim *Trim a SpatRaster*

### Description

Trim (shrink) a SpatRaster by removing outer rows and columns that are NA.

### Usage

```
## S4 method for signature 'SpatRaster'
trim(x, padding=0, filename="", overwrite=FALSE, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| x | SpatRaster |
| padding | integer. Number of outer rows/columns to keep |
| filename | character. Output filename. Optional |
| overwrite | logical. If TRUE, filename is overwritten |
| wopt | list. Options for writing files as in writeRaster |
| ... | additional arguments. None implemented |

### Value

SpatRaster

### Examples

```
r <- rast(ncol=10, nrow=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)
```

## unique *Unique values*

### Description

This function returns the unique values in a SpatRaster.

### Usage

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

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `incomparables` | logical. If `TRUE`, the unique values are determined for all layers together, and the result is a matrix. If `FALSE`, each layer is evaluated separately, and a list is returned. |
| `...` | additional arguments. none implemented |

### Value

vector or matrix

### Examples

```
r <- rast(ncol=5, nrow=5)
values(r) <- rep(1:5, each=5)
unique(r)
s <- c(r, round(r/3))
unique(s)
unique(s,TRUE)
```

---

## `values` *Get or set cell values*

---

### Description

values returns all cell values of a SpatRaster (a matrix), or all the attributes of a SpatVector (a data.frame).

### Usage

```
## S4 method for signature 'SpatRaster'
values(x, mat=TRUE, ...)

## S4 replacement method for signature 'SpatRaster,ANY'
values(x, ...)<-value

## S4 method for signature 'SpatRaster,ANY'
setValues(x, values, ...)

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

## S4 replacement method for signature 'SpatVector,data.frame'
values(x, ...)<-value
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster or SpatVector |
| `mat` | logical. If `TRUE`, values are returned as a matrix instead of as a vector |
| `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 |
| `values` | Same as for `value` |
| `...` | additional arguments. none implemented |

## Details

If `x` is a `SpatRaster`:

If `matrix=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` for an alternative matrix representation where the number of rows and columns matches that of `x`, if `x` has a single layer. If `matrix=FALSE`, the values are returned as a vector. In cell-order by layer.

If `x` is a `SpatVector`: a `data.frame`

## Value

matrix, vector, SpatRaster, or nothing

## Examples

```
r <- rast(system.file("ex/test.tif", package="terra"))
r
v <- values(r)
values(r) <- v * 10
r

f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
x <- values(v)
x
values(v) <- x[,1:2]
v
```

---

`vect` *Create SpatVector objects*

---

## Description

Create a new SpatVector

## Usage

```
## S4 method for signature 'character'
vect(x, ...)

## S4 method for signature 'data.frame'
vect(x, type="points", atts=NULL, crs=NA, ...)

## S4 method for signature 'sf'
vect(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | character (filename), data.frame, matrix, missing, or a vector object defined in packages `sf` or `sp`. If `x` is a filename, all other arguments are ignored |
| `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 (PROJ4 notation) |
| `...` | additional matrices and/or lists with matrices |

## Value

SpatVector

## Examples

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

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')
z <- data.frame(z)

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

z[z$hole==1, "object"] <- 4
lns <- vect(z[,1:4], "lines")
plot(p)
lines(lns, col='red')
```

## 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 |
| value | vector |

### Value

vector

### See Also

values

### Examples

```
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
v$NAME_1
v$ID_1 <- LETTERS[1:12]
v$new <- sample(12)
values(v)
```

## 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, wopt=list(), ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster |
| `filename` | character. Output filename |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `wopt` | list. Options for writing files. See Details |
| `...` | additional arguments. None implemented |

### Details

In many methods, options for writing raster files to disk can be provided with the `wopt` argument. `wopt` should be a named list. The following options are available:

| **name** | **description** |
|---|---|
| `filetype` | file format expresses as GDAL driver names. If this argument is not supplied, the driver is derived from the filena |
| `datatype` | values for `datatype` are "INT1U", "INT2U", "INT2S", "INT4U", "INT4S", "FLT4S", "FLT8S". The first three le |
| `gdal` | GDAL driver specific datasource creation options. See the GDAL documentation. For example: `gdal=c("COMPRE` |
| `tempdir` | the path where temporary files are to be written to |
| `progress` | postive 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. code to use for NA values |
| `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.

## writeVector *Write vector data to a file*

### Description

Write a SpatVector object to a file.

### Usage

```
## S4 method for signature 'SpatVector,character'
writeVector(x, filename, overwrite=FALSE, ...)
```

### Arguments

| | |
|---|---|
| `x` | SpatVector |
| `filename` | character. Output filename |
| `overwrite` | logical. If `TRUE`, `filename` is overwritten |
| `...` | additional arguments. None implemented |

### Value

SpatVector (invisibly). This function is used for the side-effect of writing values to a file.

### 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)

xmin(x, ...) <- value
```

```
xmax(x, ...) <- value
ymin(x, ...) <- value
ymax(x, ...) <- value
```

## Arguments

| | |
|---|---|
| `x` | SpatRaster, SpatExtent, or SpatVector |
| `value` | numeric |
| `...` | additional arguments. None implemented |

## 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 |
| `...` | additional arguments. None implemented |

## 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

## 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)
cellFromXY(r, cbind(c(10,5), c(15, 88)))
```

---

## 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, ...)
```

## 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 |

## 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(1:4, each=25)
zonal(r, z, "sum", na.rm=TRUE)

# multiple layers
r <- rast(system.file("ex/logo.tif", package = "terra"))
# zonal layer
z <- rast(r, 1)
values(z) <- rep(1:4, each=ncell(r)/4, len=ncell(r))
zonal(r, z, "mean", na.rm = 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=10000, layer=1, new=TRUE, ...)
```

## 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