Package 'scomps'

October 4, 2023

Title Scalable R geospatial computation

Version 0.0.3.10042023

Description A package for scalable geospatial computation for environmental health research

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports covr, dplyr, future, future.apply, methods, rlang, sf, stars, terra, testthat, units

Suggests future.batchtools, igraph, knitr, logr, rmarkdown, withr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Insang Song [aut, cre] (<https://orcid.org/0000-0001-8732-3256>)

Maintainer Insang Song <geoissong@gmail.com>

R topics documented:

aw_covariates 2
calculate_sedc 3
check_bbox 3
check_crs 4
check_crs2 5
check_packbound 5
check_within_reference 6
clip_as_extent 6
clip_as_extent_ras 7
clip_as_extent_ras2 7
distribute_process 8
estimate_demands 8
extent_to_polygon 9
extract_with 9
extract_with_buffer 10
extract_with_polygons 11

get_computational_regions 11
grid_merge 12
initate_log 13
rast_short 14
set_clip_extent 14
sp_indexing 15
sp_index_grid 15
switch_packbound 16
validate_and_repair_vectors 16

Index **17**

`aw_covariates` *Computing area weighted covariates using two polygon sf or SpatVector objects*

Description

When poly_in and poly_weight are different classes, poly_weight will be converted to the class of poly_in.

Usage

```
aw_covariates(poly_in, poly_weight, id_poly_in = "ID")
```

Arguments

`poly_in`	A sf/SpatVector object at weighted means will be calculated.
`poly_weight`	A sf/SpatVector object from which weighted means will be calculated.
`id_poly_in`	character(1). The unique identifier of each polygon in poly_in

Value

A data.frame with all numeric fields of area-weighted means.

Author(s)

Insang Song <geoissong@gmail.com>

Examples

```
# package
library(sf)

# run
nc = sf::st_read(system.file("shape/nc.shp", package="sf"))
nc = sf::st_transform(nc, 5070)
pp = sf::st_sample(nc, size = 300)
pp = sf::st_as_sf(pp)
pp[["id"]] = seq(1, nrow(pp))
sf::st_crs(pp) = "EPSG:5070"
ppb = sf::st_buffer(pp, nQuadSegs=180, dist = units::set_units(20, 'km'))
```

```
system.time({ppb_nc_aw = aw_covariates(ppb, nc, 'id')})
summary(ppb_nc_aw)
#### Example of aw_covariates ends ####
```

`calculate_sedc` *Calculate SEDC covariates*

Description

NOTE: sf implementation is pending. Only available for terra.

Usage

```
calculate_sedc(
  point_from,
  point_to,
  id,
  sedc_bandwidth,
  threshold,
  target_fields
)
```

Arguments

`point_from`	SpatVector object. Locations where the sum of SEDCs are calculated.
`point_to`	SpatVector object. Locations where each SEDC is calculated.
`id`	character(1). Name of the unique id field in point_to.
`sedc_bandwidth`	numeric(1). Distance at which the source concentration is reduced to exp(-3) (approximately 95 %)
`threshold`	numeric(1). For computational efficiency, the nearest points in threshold will be selected
`target_fields`	character(varying). Field names in characters.

Author(s)

Insang Song

`check_bbox` *Check if the data extent is inside the reference bounding box*

Description

One of the most common errors in spatial computation is rooted in the entirely or partly incomparable spatial extents of input datasets. This function returns whether your data is inside the target computational extent. It is assumed that you know and have the exact computational region. This function will return TRUE if the reference region completely contains your data's extent and FALSE otherwise.

Usage

```
check_bbox(data_query, reference, reference_crs = NULL)
```

Arguments

data_query	sf*/stars/SpatVector/SpatRaster object.
reference	sf*/stars/SpatVector/SpatRaster object or a named numeric vector with four names (xmin, ymin, xmax, and ymax).
reference_crs	Well-known-text-formatted or EPSG code of the reference's coordinate system. Only required when a named numeric vector is passed to reference.

Value

TRUE (the queried data extent is completely within the reference bounding box) or FALSE

Author(s)

Insang Song <geoissong@gmail.com>

---

check_crs *Check Coordinate Reference System*

Description

Check Coordinate Reference System

Usage

```
check_crs(x)
```

Arguments

x	sf/stars/SpatVector/SpatRaster object.

Value

A st_crs or crs object.

Author(s)

Insang Song <geoissong@gmail.com>

Examples

```
# data
library(sf)
ncpath = system.file("shape/nc.shp", package = "sf")
nc = read_sf(ncpath)
check_crs(nc)
```

check_crs2 *check_crs2: Coordinate system checker*

Description

The input is checked whether its coordinate system is present. If not, it is reprojected to EPSG:5179.

Usage

```
check_crs2(input, crs_standard = "EPSG:4326")
```

Arguments

input	Input object one of sf or terra::Spat* object
crs_standard	character(1). A standard definition of coordinate reference system. Default is "EPSG:4326" Consult epsg.io for details of other CRS.

Value

A (reprojected) sf or SpatVector object.

check_packbound *Return the package the input object is based on*

Description

Detect whether the input object is sf or Spat* object.

Usage

```
check_packbound(input)
```

Arguments

input	Spat* in terra or sf object.

Value

A character object; one of 'terra' and 'sf'

Author(s)

Insang Song

check_within_reference *Check if the boundary of the vector/raster object is inside the reference*

Description

Check if the boundary of the vector/raster object is inside the reference

Usage

```
check_within_reference(input_object, reference)
```

Arguments

input_object	sf/stars/SpatVector/SpatRaster object.
reference	sf/stars/SpatVector/SpatRaster object.

Value

logical

Author(s)

Insang Song <geoissong@gmail.com>

clip_as_extent *Extent clipping*

Description

Clip input vector by the expected maximum extent of computation.

Usage

```
clip_as_extent(pnts, buffer_r, nqsegs = NULL, target_input)
```

Arguments

pnts	sf or SpatVector object
buffer_r	numeric(1). buffer radius. this value will be automatically multiplied by 1.25
nqsegs	integer(1). the number of points per a quarter circle; SOON TO BE DEPRECATED
target_input	sf or SpatVector object to be clipped

Value

A clipped sf or SpatVector object.

Author(s)

Insang Song

`clip_as_extent_ras` *clip_as_extent_ras: Clip input raster.*

Description

Clip input raster by the expected maximum extent of computation.

Usage

```
clip_as_extent_ras(pnts, buffer_r, nqsegs = 180, ras)
```

Arguments

`pnts`	sf or SpatVector object
`buffer_r`	numeric(1). buffer radius. this value will be automatically multiplied by 1.25
`nqsegs`	integer(1). the number of points per a quarter circle
`ras`	SpatRaster object to be clipped

Author(s)

Insang Song

`clip_as_extent_ras2` *clip_as_extent_ras2: Clip input raster (version 2).*

Description

Clip input raster by the expected maximum extent of computation.

Usage

```
clip_as_extent_ras2(points_in, buffer_r, nqsegs = 180, ras)
```

Arguments

`points_in`	sf or SpatVector object
`buffer_r`	numeric(1). buffer radius. this value will be automatically multiplied by 1.25
`nqsegs`	integer(1). the number of points per a quarter circle
`ras`	SpatRaster object to be clipped

Author(s)

Insang Song

distribute_process *Process a given function in the entire or partial computational grids (under construction)*

Description

Should

Usage

```
distribute_process(grids, grid_id = NULL, fun, ...)
```

Arguments

grids	sf/SpatVector object. Computational grids.
grid_id	character(1) or numeric(2). Default is NULL. If NULL, all grid_ids are used. "id_from:id_to" format or c(unique(grid_id)[id_from], unique(grid_id)[id_to])
fun	function supported in scomps.
...	Arguments passed to fun.

Value

a data.frame object with mean value

Author(s)

Insang Song <geoissong@gmail.com>

estimate_demands *Estimate computational demands from inputs (to be written)*

Description

Estimate computational demands from inputs (to be written)

Usage

```
estimate_demands(inputs, nx, ny, padding)
```

Arguments

inputs	a list of sf/Spat* objects or file paths
nx	integer(1).
ny	integer(1).
padding	numeric(1). Extrusion factor

Author(s)

Insang Song

extent_to_polygon *Generate a rectangular polygon from extent*

Description

Generate a rectangular polygon from extent

Usage

```
extent_to_polygon(extent, output_class = "terra", crs = "EPSG:4326")
```

Arguments

extent	input extent. A numeric vector with xmin/xmax/ymin/ymax, sf::st_bbox() or terra::ext() outputs.
output_class	character(1). Class of the output polygon. One of "sf" or "terra"
crs	character(1). Coordinate reference system definition.

Author(s)

Insang Song

extract_with *Extract raster values with point buffers or polygons*

Description

Extract raster values with point buffers or polygons

Usage

```
extract_with(raster, vector, id, func = mean, mode = "polygon", ...)
```

Arguments

raster	SpatRaster object.
vector	SpatVector object.
id	character(1). Unique identifier of each point.
func	function taking one numeric vector argument.
mode	one of "polygon" (generic polygons to extract raster values with) or "buffer" (point with buffer radius)
...	various. Passed to extract_with_buffer. See ?extract_with_buffer for details.

Author(s)

Insang Song <geoissong@gmail.com>

`extract_with_buffer` *Extract summarized values from raster with points and a buffer radius (to be written)*

Description

For simplicity, it is assumed that the coordinate systems of the points and the raster are the same. Kernel function is not yet implemented.

Usage

```
extract_with_buffer(
  points,
  surf,
  radius,
  id,
  qsegs = 90,
  func = mean,
  kernel = NULL,
  bandwidth = NULL
)
```

Arguments

`points`	SpatVector object. Coordinates where buffers will be generated
`surf`	SpatRaster object. A raster of whatnot a summary will be calculated
`radius`	numeric(1). Buffer radius. here we assume circular buffers only
`id`	character(1). Unique identifier of each point.
`qsegs`	integer(1). Number of vertices at a quarter of a circle. Default is 90.
`func`	a function taking a numeric vector argument.
`kernel`	character(1). Name of a kernel function (yet to be implemented)
`bandwidth`	numeric(1). Kernel bandwidth.

Value

a data.frame object with mean value

Author(s)

Insang Song <geoissong@gmail.com>

extract_with_polygons *Extract summarized values from raster with generic polygons*

Description

For simplicity, it is assumed that the coordinate systems of the points and the raster are the same. Kernel function is not yet implemented.

Usage

```
extract_with_polygons(polys, surf, id, func = mean, na.rm = TRUE)
```

Arguments

polys	sf/SpatVector object. Polygons.
surf	stars/SpatRaster object. A raster of whatnot a summary will be calculated
id	character(1). Unique identifier of each point.
func	a function taking one argument. For example, function(x) mean(x, na.rm = TRUE) or \(x) mode(x, na.rm = TRUE)
na.rm	logical(1). NA values are omitted when summary is calculated.

Value

a data.frame object with function value

Author(s)

Insang Song <geoissong@gmail.com>

get_computational_regions *Get a set of computational regions*

Description

TODO. Using input points, the bounding box is split to the predefined numbers of columns and rows. Each grid will be buffered by the radius.

Usage

```
get_computational_regions(
  input,
  mode = "grid",
  nx = 10,
  ny = 10,
  grid_min_features = 30,
  padding = NULL,
  unit = NULL,
  ...
)
```

Arguments

input	sf or Spat* object.
mode	character(1). Mode of region construction. One of "grid" (simple grid regardless of the number of features in each grid), "density" (clustering-based varying grids), "grid_advanced" (merging adjacent grids with smaller number of features than grid_min_features).
nx	integer(1). The number of grids along x-axis.
ny	integer(1). The number of grids along y-axis.
grid_min_features	integer(1). A threshold to merging adjacent grids
padding	numeric(1). A extrusion factor to make buffer to clip actual datasets. Depending on the length unit of the CRS of input.
unit	character(1). The length unit for padding (optional). units::set_units is used for padding when sf object is used. See units package vignette (web) for the list of acceptable unit forms.
...	arguments passed to the internal function

Value

A set of polygons in the input class

Author(s)

Insang Song

Examples

```
# data
library(sf)
ncpath = system.file("shape/nc.shp", package = "sf")
nc = read_sf(ncpath)
nc = st_transform(nc, "EPSG:5070")
# run
# nc_comp_region = get_computational_regions(nc, nx = 12, ny = 8)
```

---

grid_merge *grid_merge: Merge grid polygons with given rules*

Description

Merge boundary-sharing (in "Rook" contiguity) grids with fewer target features than the threshold. This function strongly assumes that the input is returned from the sp_index_grid, which has 'CGRIDID' as the unique id field.

Usage

```
grid_merge(points_in, grid_in, grid_min_features)
```

Arguments

points_in	sf or SpatVector object. Target points of computation.
grid_in	sf or SpatVector object. The grid generated by sp_index_grid
grid_min_features	integer(1). Threshold to merge adjacent grids.

Value

A sf or SpatVector object of computation grids.

Author(s)

Insang Song

Examples

```
# library(sf)
# library(igraph)
# ligrary(dplyr)
# dg = sf::st_as_sfc(st_bbox(c(xmin = 0, ymin = 0, xmax = 8e5, ymax = 6e5)))
# sf::st_crs(dg) = 5070
# dgs = sf::st_as_sf(st_make_grid(dg, n = c(20, 15)))
# dgs$CGRIDID = seq(1, nrow(dgs))
#
# dg_sample = st_sample(dg, kappa = 5e-9, mu = 15, scale = 20000, type = "Thomas")
# sf::st_crs(dg_sample) = sf::st_crs(dg)
# dg_merged = grid_merge(sf::st_as_sf(sss), dgs, 100)
#### NOT RUN ####
```

---

initate_log *Turn on logging*

Description

Turn on logging

Usage

```
initate_log(expr, dolog = FALSE, logpath)
```

Arguments

expr	expression. Any function call to be logged.
dolog	logical(1). Will the messages be logged.
logpath	character(1). Log file path with the full log file name.

Value

Nothing. It will export a log file in the specified path as logpath.

Author(s)

Insang Song

rast_short *Quick call for SpatRaster with a window*

Description

Quick call for SpatRaster with a window

Usage

```
rast_short(rasterpath, win)
```

Arguments

rasterpath	character(1). Path to the raster file.
win	Named integer vector (4) or terra::ext() results.

Value

SpatRaster object.

Author(s)

Insang Song

set_clip_extent *Setting the clipping extent*

Description

Return clipping extent with buffer radius. It assumes the input CRS is projected and linear unit is meters.

Usage

```
set_clip_extent(pnts, buffer_r)
```

Arguments

pnts	One of sf or vect class. Target points of computation.
buffer_r	numeric(1). Buffer radius. It is assumed in metres

Value

A terra::ext or sfc_POLYGON object of the computation extent.

Author(s)

Insang Song

sp_indexing *Create integer indices for grid*

Description

Returns a tibble object that includes x- and y- index by using two inputs ncutsx and ncutsy, which are x- and y-directional splits, respectively.

Usage

```
sp_indexing(points_in, ncutsx, ncutsy)
```

Arguments

points_in	sf object.
ncutsx	integer(1). The number of splits along x-axis.
ncutsy	integer(1). The number of splits along y-axis.

Author(s)

Insang Song

sp_index_grid *sp_index_grid: Generate grid polygons*

Description

Returns a sf object that includes x- and y- index by using two inputs ncutsx and ncutsy, which are x- and y-directional splits, respectively.

Usage

```
sp_index_grid(points_in, ncutsx, ncutsy)
```

Arguments

points_in	sf or SpatVector object. Target points of computation.
ncutsx	integer(1). The number of splits along x-axis.
ncutsy	integer(1). The number of splits along y-axis.

Value

A sf or SpatVector object of computation grids with unique grid id (CGRIDID).

Author(s)

Insang Song

switch_packbound *Switch spatial data class*

Description

Convert stars into SpatRaster and vice versa; sf into SpatVector and vice versa.

Usage

```
switch_packbound(input)
```

Arguments

input	Spat* in terra or sf object.

Value

Data converted to the other package class (if sf, terra; if terra, sf)

Author(s)

Insang Song

validate_and_repair_vectors *Validate and repair input vector data*

Description

It tries repairing input vector data. Vector validity violation usually appears in polygon data with self-crossing or hole orders. This function will pass the input_vector object to sf::st_make_valid() (if input_vector is sf) or terra::makeValid() (if input_vector is SpatVector). May take some time depending on the geometry complexity.

Usage

```
validate_and_repair_vectors(input_vector)
```

Arguments

input_vector	One of sf or vect class. Target points of computation.

Value

A repaired sf or SpatVector object depending on the class of input_vector.

Author(s)

Insang Song

Index

aw_covariates, 2

calculate_sedc, 3
check_bbox, 3
check_crs, 4
check_crs2, 5
check_packbound, 5
check_within_reference, 6
clip_as_extent, 6
clip_as_extent_ras, 7
clip_as_extent_ras2, 7

distribute_process, 8

estimate_demands, 8
extent_to_polygon, 9
extract_with, 9
extract_with_buffer, 10
extract_with_polygons, 11

get_computational_regions, 11
grid_merge, 12

initate_log, 13

rast_short, 14

set_clip_extent, 14
sp_index_grid, 15
sp_indexing, 15
switch_packbound, 16

validate_and_repair_vectors, 16