Package: sfext (via r-universe)

October 9, 2024

Type Package

Title Extra Functions for Simple Feature Data

Version 0.1.1.9000

Description Extra functions with additional options for reading, writing, and transforming spatial data. Includes a variety of utility functions for working with tabular data with coordinates and distance and area units.

License MIT + file LICENSE

URL https://github.com/elipousson/sfext, https://elipousson.github.io/sfext/

BugReports https://github.com/elipousson/sfext/issues

Depends R (>= 2.10)

Imports cli, cliExtras (>= 0.1.0), dplyr (>= 1.0.0), feltr (>= 0.1.1.9005), filenamr (>= 0.1.0.9002), glue, grDevices, lifecycle, purrr, rlang (>= 1.1.0), sf (>= 1.0-11), tibble, tidyselect, units (>= 0.8-2), vctrs

Suggests covr, esri2sf (>= 0.1.1), exiftoolr, geosphere, ggplot2, gistr, googlesheets4, httr2, janitor, knitr, leaflet, leafpop, lwgeom, mapview, naniar, openxlsx, osmdata, rappdirs, rdeck, readr, readxl, rmarkdown, spdep, testthat (>= 3.0.0), tidygeocoder, tidyr, withr

VignetteBuilder knitr

Remotes elipousson/cliExtras, elipousson/esri2sf, elipousson/feltr, elipousson/filenamr, qfes/rdeck

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Language en-US

Repository https://elipousson.r-universe.dev

RemoteUrl https://github.com/elipousson/sfext

RemoteRef HEAD

RemoteSha 6a3c1cedd34ae02a7fbfb1026e800a22d7996fb1

Contents

`address_to_sf` *Use tidygeocoder to convert an address or data frame with an address column to an sf object*

Description

Wraps `tidygeocoder::geo()` and `tidygeocoder::geocode()` to convert a character string or a data frame with an address column. Additional parameters passed to `tidygeocoder::geocode()` which passes `...` parameters to `tidygeocoder::geo()`.

Usage

```
address_to_sf(
  x,
  address = "address",
  method = "osm",
  coords = c("lon", "lat"),
  remove_coords = FALSE,
```

```
  crs = NULL,
  full_results = FALSE,
  ...,
  call = caller_env()
)
```

Arguments

x	Data frame with an address column. Multiple address columns are not currently supported.
address	Address column name, Default: 'address'
method	the geocoding service to be used. API keys are loaded from environmental variables. Run `usethis::edit_r_environ()` to open your .Renviron file and add an API key as an environmental variable. For example, add the line `GEOCODIO_API_KEY="YourAPIKeyHer` • `"osm"`: Nominatim. • `"census"`: US Census. Geographic coverage is limited to the United States. Batch geocoding is supported. • `"arcgis"`: ArcGIS. • `"geocodio"`: Geocodio. Geographic coverage is limited to the United States and Canada. An API key must be stored in the environmental variable "GEOCODIO_API_KEY". Batch geocoding is supported. • `"iq"`: Location IQ. An API key must be stored in the environmental variable "LOCATIONIQ_API_KEY". • `"google"`: Google. An API key must be stored in the environmental variable "GOOGLEGEOCODE_API_KEY". • `"opencage"`: OpenCage. An API key must be stored in the environmental variable "OPENCAGE_KEY". • `"mapbox"`: Mapbox. An API key must be stored in the environmental variable "MAPBOX_API_KEY". • `"here"`: HERE. An API key must be stored in the environmental variable "HERE_API_KEY". Batch geocoding is supported, but must be explicitly called with `mode = "batch"`. • `"tomtom"`: TomTom. An API key must be stored in the environmental variable "TOMTOM_API_KEY". Batch geocoding is supported. • `"mapquest"`: MapQuest. An API key must be stored in the environmental variable "MAPQUEST_API_KEY". Batch geocoding is supported. • `"bing"`: Bing. An API key must be stored in the environmental variable "BINGMAPS_API_KEY". Batch geocoding is supported, but must be explicitly called with `mode = "batch"`. • `"geoapify"`: Geoapify. An API key must be stored in the environmental variable "GEOAPIFY_KEY". • `"cascade"` **[Deprecated]** use geocode_combine or geo_combine instead. The "cascade" method first uses one geocoding service and then uses a second geocoding service if the first service didn't return results. The services and order is specified by the cascade_order argument. Note that this is not compatible with `full_results = TRUE` as geocoding services have different columns that they return.

`coords` Coordinate columns for input data.frame or output sf object (if geometry is 'centroid' or 'point') Default: c("lon", "lat").

`remove_coords` For `df_to_sf()`, if `TRUE`, remove the coordinate columns after converting a data frame to simple feature object; defaults to `FALSE`.

`crs` Cordinate reference system to return, Default: 4326 for `sf_to_df()` and `NULL` for `df_to_sf()`.

`full_results` returns all available data from the geocoding service if TRUE. If FALSE (default) then only latitude and longitude columns are returned from the geocoding service.

`...` Arguments passed on to `tidygeocoder::geocode`

`.tbl` dataframe containing addresses

`street` street address column name

`city` city column name

`county` county column name

`state` state column name

`postalcode` postalcode column name (zip code if in the United States)

`country` country column name

`lat` latitude column name. Can be quoted or unquoted (ie. lat or 'lat').

`long` longitude column name. Can be quoted or unquoted (ie. long or 'long').

`return_input` if TRUE then the input dataset will be combined with the geocoder query results and returned. If FALSE only the geocoder results will be returned.

`limit` maximum number of results to return per input address. For many geocoding services the maximum value of the limit parameter is 100. Pass `limit = NULL` to use the default `limit` value of the selected geocoding service. For batch geocoding, limit must be set to 1 (default) if `return_addresses = TRUE`. To use `limit > 1` or `limit = NULL` set return_input to FALSE. Refer to api_parameter_reference for more details.

`return_addresses` if TRUE return input addresses. Defaults to TRUE if `return_input` is FALSE and FALSE if `return_input` is TRUE. This argument is passed to the `geo()` function.

`unique_only` if TRUE then only unique results will be returned and return_input will be set to FALSE.

`call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

A `sf` object with POINT geometry for all geocoded addresses with valid coordinates.

See Also

`tidygeocoder::geo()`, `tidygeocoder::geocode()`

area_unit_options *Area units (vector)*

Description

A vector of supported area units derived from `dist_units` and `units::valid_udunits()`.

Usage

```
area_unit_options
```

Format

A character vector with 41 names, plural names, and aliases for area units.

as_crs *Convert object to coordinate reference system or check coordinate reference system*

Description

- as_crs: coerce x to a CRS object and (optionally) error if a NA value is returned.
- is_same_crs: do x and y have the same coordinate reference system?

Usage

```
as_crs(x = NULL, allow_na = TRUE, arg = caller_arg(x), call = caller_env())

is_same_crs(x, y)

is_wgs84(x)
```

Arguments

x	For `as_crs()`, object to convert to a coordinate reference system. For `is_same_crs()` and `is_wgs84()`, object to check. For `as_wgs84()`, object to convert to EPSG:4326.
allow_na	For `as_crs()`, if `TRUE`, return `NA_crs_` if x can't be converted to a valid coordinate reference system. If `FALSE`, error instead of returning an invalid CRS.
arg	An argument name as a string. This argument will be mentioned in error messages as the input that is at the origin of a problem.
call	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.
y	For `is_same_crs()`, object to compare to x.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
nc[["category"]] <- sample(c("A", "B", "C"), nrow(nc), replace = TRUE)

as_sf(nc$geometry)

nc_bbox <- as_bbox(nc)

nc_bbox

as_sfc(nc_bbox)

as_xy(nc[1,])

as_sf_list(nc, col = "category")
```

---

as_points *Convert an sf, numeric, or other object to a POINT (sfg) or POINT, MULTIPOINT, LINESTRING, or MULTILINESTRING (sfc) object*

Description

Works with sf, sfc, and bbox objects using `sf::st_centroid()`. Works with `sf_bbox_point()`

Usage

```
as_point(..., to = "POINT")

as_points(..., to = "POINT", call = caller_env())

as_startpoint(x)

as_endpoint(x)

as_line(..., to = "LINESTRING", call = caller_env())

as_lines(..., to = "LINESTRING")

as_polygons(..., to = "POLYGON")

as_centroid(x, ...)
```

Arguments

`...`	See details.
`to`	The geometry type to return, either POINT or MULTIPOINT or LINESTRING or MULTILINESTRING.

call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
x	A sf, sfc, or bbox object.

Details

Using as_point:

as_point() always returns a single point sfg object. The ... parameter is passed to sf::st_centroid() if ... is a sf, sfc, or bbox object, sf_bbox_point() includes a bbox object and a string indicating the requested point position, or sf::st_point() if ... includes a numeric vector.

Using as_points:

as_points() always returns an sfc object. The parameters are passed to as_point using purrr::map() and then converted to sfc using sf::st_as_sfc(). The ... parameters must include a crs, otherwise the crs will be NA for the resulting sfc object.

Using as_startpoint and as_endpoint:

as_startpoint() and as_endpoint() require a LINESTRING OR MULTILINESTRING geometry type sf or sfc object that is passed to lwgeom::st_startpoint() or lwgeom::st_endpoint() respectively. Both functions always return a sfc object matching the CRS of the input geometry.

Using as_lines:

If params do not have POINT or MULTIPOINT geometry, they are passed to as_points() to convert to an sfc object. If the parameters have POINT geometry, they are combined to create a MULTIPOINT geometry.

For as_lines() the ... parameters are passed to as_points() and/or sf::st_cast().

Both as_line and as_lines do not consistently retain the coordinate reference system of the original object but this should be improved in the future.

Using as_centroid()

as_centroid() always returns a sfc object with the same length and crs as the input object.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

as_point(nc)

as_point(c("xmax", "ymax"), bbox = as_bbox(nc))

as_points(nc)

as_points(nc[1, ], nc[2, ])

nc_line <- as_line(c(as_points(nc[1, ]), as_points(nc[10, ])))

as_startpoint(nc_line)

as_endpoint(nc_line)
```

```
as_centroid(nc)
```

---

as_sf *Convert an object to a simple feature or bounding box object*

Description

Both functions will pass a NULL value without returning an error. If a POINT or MULTIPOINT object is passed to as_bbox() a 0.00000001 meter buffer is applied. If a character object is passed to as_bbox() it is passed to osmdata::getbb() using format_out = "matrix" which is converted into a bounding box.

Usage

```
as_sf(
  x,
  crs = NULL,
  sf_col = "geometry",
  ext = TRUE,
  ...,
  as_tibble = TRUE,
  call = caller_env()
)

as_bbox(x, crs = NULL, ext = TRUE, ..., call = caller_env())

as_sfc(x, crs = NULL, ext = TRUE, ..., call = caller_env())

as_sf_class(x, class = NULL, allow_null = TRUE, ..., call = caller_env())
```

Arguments

x	Object to convert to an sf, sfc, bbox or a sf list object.
crs	Coordinate reference system for sf, bbox, sfc or sf list object to return.
sf_col	A column name to use for the geometry column created by as_sf; defaults to "geometry".
ext	If TRUE, as_sf will convert a data frame or character vector of addresses to an sf object using df_to_sf or address_to_sf. If FALSE, only spatial objects (bbox, sfg, sfc, sf list, raster, or sp objects) can be converted. Defaults to TRUE.
...	Additional parameters passed to as_sf, as_sfc, as_bbox, or as_sf_list
as_tibble	If TRUE, always return sf object as a tibble. If FALSE, some conversions may still return a tibble object.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

class	A class to convert data to; defaults to NULL (which returns "sf")
allow_null	For as_sf_class, if class is NULL and allow_null is TRUE, return x without any class conversion or checks. Defaults to TRUE.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
nc[["category"]] <- sample(c("A", "B", "C"), nrow(nc), replace = TRUE)

as_sf(nc$geometry)

nc_bbox <- as_bbox(nc)

nc_bbox

as_sfc(nc_bbox)

as_xy(nc[1,])

as_sf_list(nc, col = "category")
```

---

as_xy *Convert data to a data frame with X/Y coordinate pairs*

Description

Wraps as_points(), as_sfc(), and sf_bbox_point() to allow the conversion of sf, sfc, or sfg objects into a simple data frame with X and Y columns matching the provided nm parameter.

Usage

```
as_xy(x, bbox = NULL, crs = NULL, nm = c("x", "y"), ...)
```

Arguments

x	A length 2 character string or numeric coordinate pair or a sf, sfc, or a bbox object. If x is a character string (e.g. c("xmin", "ymax")), a sf, sfc, or bbox object must be provided to data argument.
bbox	A bbox object or object that can be converted with as_bbox() that is passed as bbox to sf_bbox_point() when x is passed to the point parameter.
crs	A character or numeric reference to a coordinate reference system supported by sf::st_crs() or another sf, sfc, or bbox object that is used to provide crs.
nm	Column names to use for X and Y columns.
...	Additional parameters passed to sf_bbox_point(), as_points(), or as_sfc().

bind_units_col *Bind units column to data frame*

Description

Utility function supporting get_area, get_dist, and get_bearing.

Usage

```
bind_units_col(
  x,
  y,
  units = NULL,
  drop = FALSE,
  keep_all = TRUE,
  .id = NULL,
  call = caller_env()
)
```

Arguments

x	Data frame or sf object.
y	Vector of numeric or units values to bind to x.
units	Units to use for y (if numeric) or convert to (if y is units class); defaults to NULL.
drop	If TRUE, apply the units::drop_units function to the column with units class values and return numeric values instead; defaults to FALSE.
keep_all	If FALSE, keep all columns. If FALSE, return only the named .id column.
.id	Name to use for vector of units provided to "y" parameter, when "y" is bound to the "x" data frame or tibble as a new column.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

check_sf *Check if x is an sf object*

Description

If x is an sf object invisibly return TRUE. If not, return an error with cli::cli_abort

Usage

```
check_sf(
  x,
  ...,
  ext = FALSE,
  allow_list = FALSE,
  allow_null = FALSE,
  arg = caller_arg(x),
  call = caller_env()
)
```

Arguments

x	An sf, sfc, or bbox object.
...	Additional parameters passed to rlang::abort().
ext	If TRUE, check if x is a sf, sfc, or bbox class object or not; defaults to FALSE. (used by is_sf)
allow_list	If TRUE, return TRUE if x is an sf list or, if ext is also TRUE, a list of sf, sfc, or bbox objects. Defaults to FALSE.
allow_null	If TRUE and x is NULL, return TRUE; defaults to FALSE.
arg	An argument name as a string. This argument will be mentioned in error messages as the input that is at the origin of a problem.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

---

cli_format.sf *cli_format style for sf objects*

Description

cli_format style for sf objects

Usage

```
## S3 method for class 'sf'
cli_format(x, style = NULL, ...)
```

Arguments

x	A sf object to format with the cli_format.sf method.
style	Not in use for the 'sf' class method.
...	Not in use for the 'sf' class method.

See Also

cliExtras::register_cli_format()

Examples

```
## Not run:
if (interactive()) {
  library(cli)
  library(sf)
  cliExtras::register_cli_format("sf", cli_format.sf)
  nc <- read_sf(system.file("shape/nc.shp", package = "sf"))
  cli_text("`nc` is a {.val {nc}}")
}

## End(Not run)
```

`compare_dist` *Compare a distance to the dimension of a bounding box or another distance value*

Description

Compare dist to a distance based on x. If to is "xdist" (default), "ydist", "diagdist", or any length 2 character vector supported by sf_bbox_dist() dist is compared to the resulting distance. If to is an sf or sfc object, the comparison distance is the results from sf::st_distance() with x as the first parameter and to as the second. x can also be a numeric value with or without a units class. Use the "how" argument to determine the type of comparison: "ratio" (dist divided by comparison distance) "fit" (how many lengths of dist can fit wholly within the comparison distance) "longer", "shorter", "same" (uses tolerance parameter).

Usage

```
compare_dist(
  dist,
  x,
  units = NULL,
  to = "xdist",
  how = "ratio",
  tolerance = 1.5e-08,
  ...
)
```

Arguments

dist	Distance to check as a units class or numeric vector
x	A sf, sfc, or bbox object or a numeric vector.

units	Units to compare. If dist or x are numeric they are assumed to be in the provided units or the units of the coordinate reference system (CRS) for x (if x is a sf, sfc, or bbox object).
to	Distance to compare dist with. Standard options include "xdist" (default), "ydist", or "diagdist" to compare to sf_bbox_xdist(), sf_bbox_ydist(), or sf_bbox_diagdist(). If to is a length 2 character vector, x, to, and any ... parameters are passed to sf_bbox_dist(). If to is an sf or sfc object, to and any additional ... parameters are passed to sf::st_distance() using x as the first argument and to as the second. If x is a numeric vector the value of to is ignored.
how	If "ratio", dist divided by the comparison distance. If what is "fit", return the number of times dist fits in the comparison distance without remainder. If "longer", "shorter", "same", return a logical vector.
tolerance	passed to all.equal() if how = "same"
...	Additional parameters passed to sf::st_distance() if to is an sf or sfc object or sf_bbox_dist() if to is a length 2 character vector.

Value

A logical or numeric vector.

See Also

Other dist: convert_dist_scale(), convert_dist_units(), get_measurements, is_dist_units(), sf_bbox_dist()

---

convert_dist_scale *Convert distance from scale to actual units*

Description

This function converts scale distances to actual units based on named standard_scales.

Usage

```
convert_dist_scale(
  dist = NULL,
  scale = NULL,
  scale_standard = NULL,
  scale_series = NULL,
  scale_unit = "in",
  scale_factor = NULL,
  actual_unit = NULL,
  dpi = 120,
  paper = NULL,
  orientation = NULL,
  ...
)
```

Arguments

dist	distance to convert. If paper is provided, dist is optional and paper width and height are used as dist.
scale	Scale name from standard_scales[["scale"]].
scale_standard, scale_series	Passed to standard and scale parameters of get_scale().
scale_unit	"mm" (converted to cm by dividing by 10), "cm", "px" (converted to inches by dividing by dpi), or "in".
scale_factor	factor for converting from scale_unit to actual_unit, e.g. if 1" = 1', the scale factor is 12. optional if scale if provided; defaults to NULL.
actual_unit	any unit supported by convert_dist_units()
dpi	dots per square inch (used as conversion factor for "px" to "in")
paper	Paper, Default: 'letter'.
orientation	Orientation "portrait", "landscape", or "square", Default: 'portrait'.
...	Arguments passed on to get_paper standard Size standard, "ANSI", "ISO", "British Imperial", "JIS", "USPS", "Facebook", "Instagram", or "Twitter". series Size series (e.g. A), Default: NULL size Size number (only used for "ISO" and "JIS" series). Standard, series, and size may all be required to return a single paper when using these parameters. width,height Width and height in units, Default: NULL. units Paper size units, either "in", "mm", or "px"; defaults to NULL (using "in" if width or height are provided). ncol,nrow Number of expected columns and rows in paper; used to determine row_height and section_asp in paper data frame returned by get_paper if nrow or ncol is greater than 1; defaults to NULL. gutter Gutter distance in units. Gutter is used as the spacing between nrow and columns (variable spacing is not currently supported); defaults to 0. margin A numeric vector or ggplot2 margin object. bbox A bounding box to use to get orientation using sf_bbox_asp() with orientation = TRUE.

Value

- If paper is not provided, return a vector of dist values converted from scale_unit to actual_unit based on scale_factor or information from standard_scales data.
- If paper is provided, return a data.frame with converted distances appends as columns named actual_width and actual_height.

See Also

Other dist: compare_dist(), convert_dist_units(), get_measurements, is_dist_units(), sf_bbox_dist()

convert_dist_units *Convert distance (and area) values between different units*

Description

Convert distance (and area) values between different units

Usage

```
convert_dist_units(
  dist,
  from = NULL,
  to = "meter",
  drop = FALSE,
  digits = NULL
)
```

Arguments

dist	Numeric or units object
from	Existing unit for dist, Default: NULL. If dist is a units object, the numerator is used as "from"
to	Unit to convert distance to, Default: 'meter'
drop	If TRUE, return numeric. If FALSE, return class units object.
digits	Number of digits to include in result; defaults to NULL.

Value

Object created by `units::set_units()`

See Also

Other dist: `compare_dist()`, `convert_dist_scale()`, `get_measurements`, `is_dist_units()`, `sf_bbox_dist()`

coords_to_sf *Convert a data.frame with one or more coordinate columns to an sf object*

Description

- coords_to_sf(): Convert a data frame with coordinates into a simple feature object
- check_coords(): Check if a provided vector with coordinate column names are valid for the provided data frame
- separate_coords(): Separate coordinates from a single combined column into two columns
- format_coords(): Format coordinates as numeric values and remove missing coordinates from a data.frame
- has_coords(): Suggests a coordinate pair by comparing common values to the column names for a provided data.frame
- rev_coords(): Reverse a vector of coordinate names if the text "lat" or "y" appears in the first position

Usage

```
coords_to_sf(
  x,
  coords = c("lon", "lat"),
  into = NULL,
  sep = ",",
  rev = FALSE,
  remove_coords = FALSE,
  crs = 4326,
  call = caller_env()
)

check_coords(
  x = NULL,
  coords = NULL,
  default = c("lon", "lat"),
  rev = FALSE,
  call = caller_env()
)

rev_coords(coords, pattern = c("lat", "^y"), ignore.case = TRUE)

has_coords(x, coords = NULL, value = TRUE)

format_coords(
  x,
  coords = c("lon", "lat"),
  keep_missing = FALSE,
  call = caller_env()
)

separate_coords(x, coords, into = c("lon", "lat"), sep = ",")
```

Arguments

x	A data.frame with one or more coordinate columns. If coordinates are contained within a single column, coord must be length 1 and a length 2 into parameter must be provided.
coords	Coordinate columns for input data.frame or output sf object (if geometry is 'centroid' or 'point') Default: c("lon", "lat").
into	If coords is a single column name with both longitude and latitude, into is used as the names of the new columns that coords is separated into. Passed to tidyr::separate().
sep	If coords is a single column name with both longitude and latitude, sep is used as the separator between coordinate values. Passed to tidyr::separate().
rev	If TRUE, reverse c("lat", "lon") coords to c("lon", "lat"). check_coords() only.
remove_coords	For df_to_sf(), if TRUE, remove the coordinate columns after converting a data frame to simple feature object; defaults to FALSE.
crs	Coordinate reference system used by the coordinates in the provided data frame.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
default	Default coordinate values; defaults to c("lon", "lat").
pattern	Pattern passed by rev_coords() to grepl() used to match vectors that are reversed. Defaults to c("lat", "^y").
ignore.case	If TRUE, pattern matching is not case sensitive.
value	If TRUE, return the value of the coordinate column names. Used by has_coords().
keep_missing	If TRUE, keep rows with missing coordinate values. Defaults to FALSE which filters out rows with missing coordinates.

See Also

is_geo_coords()

---

count_features *Count simple features based on relationship with a second simple feature object*

Description

Use st_join_ext() and dplyr::count() to count features in x based on their spatial relationship with y. Very similar to count_sf_ext() so they may be merged in the future.

Usage

```
count_features(
  x = NULL,
  y = NULL,
  nm = "data",
  join = NULL,
  .id = "name",
  by = NULL,
  count = NULL,
  sort = FALSE,
  name = NULL,
  geometry = "y",
  ...
)
```

Arguments

x	Data frame or sf object, Default: NULL
y	Length 1 named sf list (name of y is used as count if count is NULL) or a sf object if nm is not NULL. y must include a column name matching the value of .id. If y is NULL, count is required. Default: NULL
nm	Vector of names to use with as_sf_list() to convert y to an sf list (if y is not already an sf list). Defaults to "data".
join	geometry predicate function; defaults to NULL, set to sf::st_intersects() if y contains only POLYGON or MULTIPOLYGON objects or sf::st_nearest_feature() if y contains other types.
.id	Column name with the values that should be added as a column to the input sf object.
by	A character vector of variables to join by passed to dplyr::left_join().
count	Name of column to count. If NULL, count is set to names(y) (assuming y is a length 1 named sf list).
sort	If TRUE, will show the largest groups at the top.
name	The name of the new column in the output. If omitted, it will default to n. If there's already a column called n, it will use nn. If there's a column called n and nn, it'll use nnn, and so on, adding ns until it gets a new name.
geometry	If "y", replace x geometry with y geometry joining based on by. If by is NULL, by is set to the same value as count.
...	Additional parameters passed to st_join_ext()

count_sf_ext *Count extended for working with sf objects*

Description

An extended version of dplyr::count() that makes it easier to count the occurrences of features from data that intersect with features from a second sf object (set by y) or created by passing x or data to st_make_grid_ext(). Similar to count_features() and the two functions may be combined in the future.

Usage

```
count_sf_ext(
  data,
  x = NULL,
  y = NULL,
  join = sf::st_intersects,
  largest = TRUE,
  wt = NULL,
  sort = FALSE,
  replace_na = FALSE,
  keep_na = FALSE,
  lims = NULL,
  geometry = TRUE,
  .id = "id",
  name = NULL,
  ...
)
```

Arguments

data	Data to count in relationship to y
x	Optional sf object passed to st_make_grid_ext(). Defaults to NULL.
y	If NULL (default), y defaults to an sf object created by st_make_grid_ext() using x or data (if x is NULL) as the x parameter for st_make_grid_ext(). If not NULL, y must be an sf object that has a column with the same name as .id (defaults to "id").
join	geometry predicate function with the same profile as st_intersects; see details
largest	logical; if TRUE, return x features augmented with the fields of y that have the largest overlap with each of the features of x; see https://github.com/r-spatial/sf/issues/578
wt	<data-masking> Frequency weights. Can be NULL or a variable: • If NULL (the default), counts the number of rows in each group. • If a variable, computes sum(wt) for each group.
sort	If TRUE, will show the largest groups at the top.

`replace_na` If TRUE, replace NA values from count with 0.

`keep_na` If TRUE, filter NA values from count. Ignored if replace_na is TRUE.

`lims` Optional numeric vector with minimum or both minimum and maximum count values. If provided, any values below the minimum are set to that minimum and any values above the maximum as set to the maximum. If only one value is provided, it is assumed to be a minimum limit.

`geometry` If TRUE (default) return a sf object. If FALSE, return a data frame.

`.id` A name to use for the cell id column. Defaults to "id".

`name` The name of the new column in the output.

If omitted, it will default to n. If there's already a column called n, it will use nn. If there's a column called n and nn, it'll use nnn, and so on, adding ns until it gets a new name.

`...` Arguments passed on to st_make_grid_ext

`ncol,nrow` Used to set n if either are not NULL; defaults to NULL. row and id are added as columns to the grid if they are provided.

`gutter` Distance in units between each column cell; gutter effectively serves as a margin as the negative buffer is applied to all cells (including those at the edges of the grid).

`desc` If TRUE, reverse standard order of cell id numbering; defaults FALSE

`n` If n is NULL and square is TRUE, the grid is set automatically to be 10 cells wide, Default: NULL

`what` "polygons", "corners", "centers"; set to centers automatically if style is "circle", "circle_offset" but a buffer is applied to return circular polygons.

`style` Style of cell to return with options including "rect", "square", "hex", "flat_top_hex", "circle", "circle_offset"

`filter` If TRUE (or if trim is TRUE) filter grid geometry by x using st_filter_ext

`crs` Coordinate reference system of bounding box to return; defaults to NULL which maintains the crs of the input object.

`unit` Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"

`cellsize` numeric of length 1 or 2 with target cellsize: for square or rectangular cells the width and height, for hexagonal cells the distance between opposite edges (edge length is cellsize/sqrt(3)). A length units object can be passed, or an area unit object with area size of the square or hexagonal cell.

`trim` If TRUE, x is trimmed to y with st_trim().

Value

A sf object or a tibble (if geometry = FALSE) with a column counting occurrences of features from data.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
data <- sf::st_sample(nc, size = 75)

# Count data based on nc
count <- count_sf_ext(data = data, y = nc, .id = "FIPS")
plot(count[, "n"], reset = FALSE)
plot(data, col = "gray60", pch = 3, add = TRUE)

# Count data based grid created by passing nc to st_make_grid_ext
count_grid <- count_sf_ext(data = data, x = nc, .id = "FIPS")
plot(count_grid[, "n"], reset = FALSE)
plot(data, col = "gray60", pch = 3, add = TRUE)
```

---

dist_units *Distance units (data frame)*

Description

A subset of units supported by the units package accessible through the `units::valid_udunits()` function.

Usage

```
dist_units
```

Format

A data frame with 33 rows and 12 variables:

`symbol` symbols

`symbol_aliases` symbol aliases

`name_singular` singular names

`name_singular_aliases` singular name aliases

`name_plural` character plural names

`name_plural_aliases` plural name aliases

`def` short definition

`definition` definition

`comment` comment

`dimensionless` logical indicator for dimensionless units

`source_xml` source XML

`unit_opts` character vector with symbols, singular, and plural names for the unit

dist_unit_options *Distance units (vector)*

Description

A vector of supported distance units pulled from `dist_units`.

Usage

```
dist_unit_options
```

Format

A character vector with 86 names, plural names, aliases, and symbols for distance units.

get_asp *Get aspect ratio from string or based on specific paper and margins*

Description

Get aspect ratio from string or based on specific paper and margins

Usage

```
get_asp(
  asp = NULL,
  paper = NULL,
  orientation = NULL,
  bbox = NULL,
  margin = NULL,
  block_asp = FALSE,
  allow_null = TRUE,
  ...
)
```

Arguments

asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
paper	Paper, Default: 'letter'.
orientation	Orientation "portrait", "landscape", or "square", Default: 'portrait'.
bbox	A bounding box to use to get orientation using `sf_bbox_asp()` with orientation = TRUE.
margin	A numeric vector or ggplot2 margin object.

block_asp If TRUE, and margin is not NULL, return the aspect ratio of the text or content block inside the page margins.

allow_null If TRUE and asp and paper are both NULL, return NULL without an error.

... Arguments passed on to get_paper

standard Size standard, "ANSI", "ISO", "British Imperial", "JIS", "USPS", "Facebook", "Instagram", or "Twitter".

series Size series (e.g. A), Default: NULL

size Size number (only used for "ISO" and "JIS" series). Standard, series, and size may all be required to return a single paper when using these parameters.

width,height Width and height in units, Default: NULL.

units Paper size units, either "in", "mm", or "px"; defaults to NULL (using "in" if width or height are provided).

ncol,nrow Number of expected columns and rows in paper; used to determine row_height and section_asp in paper data frame returned by get_paper if nrow or ncol is greater than 1; defaults to NULL.

gutter Gutter distance in units. Gutter is used as the spacing between nrow and columns (variable spacing is not currently supported); defaults to 0.

Value

A numeric aspect ratio.

---

get_coords *Get coordinates for a simple feature or bounding box object*

Description

An extended version of sf::st_coordinates() that supports binding coordinates to the object, optionally dropping the geometry, and returning wkt or a point on surface (geometry = "surface point") instead of the centroid.

Usage

```
get_coords(
  x,
  coords = NULL,
  geometry = "centroid",
  crs = NULL,
  keep_all = TRUE,
  drop = TRUE,
  call = caller_env()
)

get_minmax(x, crs = NULL, keep_all = TRUE, drop = TRUE)
```

Arguments

x	A sf, bbox, or sfc object.
coords	Column names to use for coordinates in results, Default: NULL; which is set to c("lon", "lat") by check_coords().
geometry	geometry to use for coordinates "centroid", "surface point", or alternatively "wkt"; defaults to NULL ("centroid").
crs	Coordinate reference system to use for coordinates; defaults to NULL.
keep_all	If TRUE, bind the coordinates columns to the provided object x; defaults to TRUE.
drop	If TRUE and x is an sf object, drop the geometry Default: TRUE.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Details

get_minmax() get a bounding box for each feature (or group of features) appends the xmin, ymin, xmax, and ymax values for each feature to the simple feature object.

---

get_data_dir *Check if data directory exists and create a new directory if needed*

Description

[Superseded] Get the path for a package-specific cache directory with rappdirs::user_cache_dir(), check for the existence of a data directory, optionally create a new directory at the provided path location.

Usage

```
get_data_dir(
  path = NULL,
  cache = FALSE,
  create = TRUE,
  pkg = "sfext",
  allow_null = TRUE
)

list_data_files(
  path = NULL,
  pkg = "sfext",
  cache = FALSE,
  fileext = NULL,
  pattern = NULL,
  full.names = TRUE,
  ignore.case = TRUE,
  ...
)
```

Arguments

path	Path to directory for use as data directory.
cache	If TRUE, and path is NULL set path to rappdirs::user_cache_dir (using value of pkg as appname). If path is not NULL, the path is returned even if cache is TRUE.
create	If FALSE and path does not exist, return path with a warning. If TRUE and rlang::is_interactive is TRUE, ask user if directory should be created. If the session not interactive and create is TRUE, a new directory will be created.
pkg	Package name; defaults to "sfext"
allow_null	If TRUE, path is NULL, cache is FALSE, return the NULL path value; defaults to TRUE.
fileext	If pattern is NULL, fileext is used to set the pattern and filter listed files to those matching the file extension.
pattern	an optional regular expression. Only file names which match the regular expression will be returned.
full.names	a logical value. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned.
ignore.case	logical. Should pattern-matching be case-insensitive?
...	Additional parameters passed to list.files()

---

get_margin *Get margins for a ggplot2 plot or map based on style or distance*

Description

This function works in combination with the get_paper() function to make it easier to position a map on a page before saving to file. This is primarily useful when using a map or plot created with ggplot2 as part of a print document format that is composed outside of R using a page layout application such as Adobe InDesign.

Usage

```
get_margin(
  margin = NULL,
  paper = NULL,
  orientation = NULL,
  dist = NULL,
  unit = "in",
  block_width = NULL,
  header = 0,
  footer = 0
)
```

Arguments

margin	Margin style (options include "extrawide", "wide", "standard", "narrow", "none"), Additional "auto" option to generate margin based on line length is planned but not yet implemented. Default: NULL (equivalent to "none").
paper	Paper, Default: 'letter'.
orientation	Orientation "portrait", "landscape", or "square", Default: 'portrait'.
dist	Margin distance (single value used to all sides), Default: NULL
unit	Unit for margin distance, Default: 'in'.
block_width	Plot or map width in units. If paper and block_width are provided, margins are half the distance between the two evenly distributed. This sets the margin distance for height as well as width so does not work well with header and footers and should be improved in the future.
header, footer	Header and footer height in units; defaults to 0. Please note: headers and footers are not currently supported for "px" units.

Value

A ggplot2::margin() element intended for use with ggplot2::element_rect() and the plot.background theme element.

See Also

ggplot2::margin()

Examples

```
get_margin("standard")

get_margin("none")

get_margin(dist = 25, unit = "mm")

get_margin(paper = "letter", block_width = 5.5)
```

---

get_measurements *Get measurements for simple feature objects*

Description

Get measurements for simple feature objects

Usage

```
get_area(x, units = NULL, keep_all = TRUE, drop = FALSE, .id = "area")

st_area_ext(x, units = NULL, keep_all = TRUE, drop = FALSE, .id = "area")

get_length(x, units = NULL, keep_all = TRUE, drop = FALSE, .id = "length")

st_length_ext(x, units = NULL, keep_all = TRUE, drop = FALSE, .id = "length")

get_dist(
  x,
  to,
  by_element = TRUE,
  units = NULL,
  drop = FALSE,
  keep_all = TRUE,
  .id = "dist",
  ...
)

st_distance_ext(
  x,
  to,
  by_element = TRUE,
  units = NULL,
  drop = FALSE,
  keep_all = TRUE,
  .id = "dist",
  ...
)

get_bearing(x, to = NULL, dir = FALSE, keep_all = TRUE, .id = "bearing")

st_bearing(x, to = NULL, dir = FALSE, keep_all = TRUE, .id = "bearing")
```

Arguments

x	A sf or sfc object to measure.
units	Units to return for area, length, perimeter, or distance; Default: NULL
keep_all	If TRUE, return all columns from the original object, Default: TRUE
drop	If TRUE, drop units from the line lengths, Default: FALSE
.id	Column name to use for area, line length/perimeter, distance, or bearing.
to	A sf, sfc, or bbox object or a length 2 character vector. If "to" is an sf or sfc object, it must have either a single feature or the same number of features as x (if by_element is TRUE). If "to" is a character vector it must represent a valid xy pair using the following options: "xmin", "ymin", "xmax", "ymax", "xmid", "ymid".

by_element	logical; if TRUE, return a vector with distance between the first elements of x and y, the second, etc; an error is raised if x and y are not the same length. If FALSE, return the dense matrix with all pairwise distances.
...	passed on to s2_distance, s2_distance_matrix, or s2_perimeter
dir	Logical indicator whether to include direction in bearing; If FALSE, return the absolute (positive) bearing value. If TRUE, return negative and positive bearing values. Default: FALSE.

Details

Wrapper functions for sf::geos_measures:

- get_area(): Wraps on sf::st_area() but MULTIPOINT or MULTILINESTRING geometry is converted to a polygon using sf::st_polygonize() which is used to determine the coverage area.
- get_length(): Wraps to sf::st_length() but POINT and MULTIPOINT geometry is converted to LINESTRING using as_lines(). If x has POLYGON geometry, lwgeom::st_perimeter() is used to return the perimeter instead of the length.
- get_dist(): Wraps sf::st_distance() but x is converted to a POINT using st_center() and "to" can be a POINT, a sf object that can be converted to a POINT, or a character vector indicating a point on the overall bounding box for x.

Additional measurement functions:

- get_bearing(): Wraps geosphere::bearing().

See Also

Other dist: compare_dist(), convert_dist_scale(), convert_dist_units(), is_dist_units(), sf_bbox_dist()

Examples

```
nc <- read_sf_path(system.file("shape/nc.shp", package = "sf"))

# Get area for North Caroline counties
get_area(nc[1:2,])$area
get_area(nc[1:2,], units = "acres")$area
get_area(nc[1:2,], units = "acres", .id = "acreage")$acreage

# Get distances for North Caroline counties
get_dist(nc[1,], to = c("xmax", "ymax"), units = "mile")$dist
get_dist(nc[1,], to = nc[30,], units = "km")$dist

# Create a line between two counties
nc_line <- as_line(c(as_point(nc[1,]), as_point(nc[30,])), crs = sf::st_crs(nc))

# Get length and bearing of the line
get_length(nc_line)
get_bearing(nc_line)
```

get_paper *Get standard paper and image sizes*

Description

[Superseded]

Usage

```
get_paper(
  paper = "letter",
  orientation = "portrait",
  standard = NULL,
  series = NULL,
  size = NULL,
  width = NULL,
  height = NULL,
  units = NULL,
  ncol = 1,
  nrow = 1,
  gutter = 0,
  bbox = NULL,
  margin = NULL,
  ...
)
```

Arguments

paper	Paper, Default: 'letter'.
orientation	Orientation "portrait", "landscape", or "square", Default: 'portrait'.
standard	Size standard, "ANSI", "ISO", "British Imperial", "JIS", "USPS", "Facebook", "Instagram", or "Twitter".
series	Size series (e.g. A), Default: NULL
size	Size number (only used for "ISO" and "JIS" series). Standard, series, and size may all be required to return a single paper when using these parameters.
width, height	Width and height in units, Default: NULL.
units	Paper size units, either "in", "mm", or "px"; defaults to NULL (using "in" if width or height are provided).
ncol, nrow	Number of expected columns and rows in paper; used to determine row_height and section_asp in paper data frame returned by get_paper if nrow or ncol is greater than 1; defaults to NULL.
gutter	Gutter distance in units. Gutter is used as the spacing between nrow and columns (variable spacing is not currently supported); defaults to 0.
bbox	A bounding box to use to get orientation using sf_bbox_asp() with orientation = TRUE.

margin	A numeric vector or ggplot2 margin object.
...	Additional parameters passed to get_margin. plot_width can only be passed in these parameters if paper has only a single row. margin is returned as a list column.

Details

Use the "paper" parameter (matching name from paper_sizes), standard (optionally including series and size) parameter, or width, height and units. May return multiple paper sizes depending on parameters.

If margin is provided, a block_width, block_height, and block_asp are calculated and included as columns in the returned data frame.

Paper can also be a data frame with "width", "height", "orientation", and "units" columns.

Value

Data frame with one or more paper/image sizes.

Examples

```
get_paper("letter")

get_paper(paper = NULL, standard = "ISO", series = "A", size = 4)

get_paper(width = 11, height = 17)
```

---

get_scale *Get standard scales and convert to scale distances*

Description

This function returns a scale from standard_scales based on a provided name, standard, and/or series.

Usage

```
get_scale(scale = NULL, standard = NULL, series = NULL)
```

Arguments

scale	Scale name from standard_scales[["scale"]].
standard	Scale standard. Options include "USGS", "Engineering", or "Architectural".
series	Map series from standard_scales[["series"]]. Series is only available for USGS scales.

Value

A tibble based on standard_scales with rows filtered to values that match parameters.

get_social_image *Get social media image size to match platform and format*

Description

See `paper_sizes[paper_sizes$type == "social",]$name` for support image options.

Usage

```
get_social_image(
  image = NULL,
  platform = NULL,
  format = NULL,
  orientation = NULL
)
```

Arguments

`image`	Image size name, Default: `NULL`
`platform`	Social media platform, "Instagram", "Facebook", or "Twitter", Default: `NULL`
`format`	Image format, "post", "story", or "cover", Default: `NULL`
`orientation`	Image orientation, Default: `NULL`.

is_dist_units *General utility functions for working with distance units objects*

Description

- `is_dist_units()`: Is x a distance unit object?
- `is_diff_dist()`: What is the difference between x and y distance?
- `is_same_dist()`: Is x the same distance as y? or does the bbox of x and bbox of y have the same x, y, or diagonal distance?
- `is_shorter()`, `is_longer()`: Is x shorter or longer than y?
- `is_same_area()`: do x and y have the same area?
- `is_same_units()`: are x and y character strings that represent the same units or objects that use the same units?

Usage

```
is_dist_units(x)

is_diff_dist(x, y, units = NULL)

is_same_dist(x, y, dist = NULL, diff = FALSE, call = caller_env(), ...)

is_longer(x, y)

is_shorter(x, y)

get_dist_units(x, allow_null = TRUE, multiple = TRUE, quiet = FALSE)

as_dist_units(x, units = NULL, allow_null = FALSE, call = caller_env())

is_diff_area(x, y, units = NULL, combine = TRUE)

is_same_area(x, y, units = NULL, combine = TRUE, diff = FALSE, ...)

is_same_units(x, y = NULL)
```

Arguments

x, y	objects to check
units	For is_diff_dist, if x and y are both not units objects, use units; default to NULL.
dist	type of distance to compare if x and y are sf, sfc, or bbox objects; "diagdist", "xdist", "ydist". defaults to NULL.
diff	If TRUE, return results from is_diff_dist or is_diff_area; if FALSE, return logical indicator; defaults to FALSE
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
...	Additional parameters passed to all.equal()
allow_null	If allow_null is TRUE, allow x to return a NULL value; if FALSE, error on NULL values.
multiple	If TRUE and x is a character vector with distance/area units, get_dist_units may return multiple units. Passed to rlang::arg_match.
quiet	If TRUE, suppress warning messages.
combine	If TRUE, combine objects with sf::st_combine() before comparing area with is_diff_area() or is_same_area(), defaults to TRUE.

Details

There are two additional functions that support these utility functions:

- get_dist_units(): Get the distance units from x (if x is a sf or units objects or a character string from dist_unit_options)

- as_dist_units(): Convert x to units using units::as_units

See Also

Other dist: compare_dist(), convert_dist_scale(), convert_dist_units(), get_measurements, sf_bbox_dist()

---

is_geom_type *What geometry type is this feature?*

Description

is_geom_type() extends sf::st_geometry_type() to return a list with the standard output ("TYPES") and additional list values with logical vectors from sf::st_is() for "POINTS" (passing "POINT" and "MULTIPOINT" as type), "POLYGONS" (passing "POLYGON", "MULTIPOLYGON"), "LINESTRINGS" ("LINESTRING" and "MULTILINESTRING"), "GEOMETRYCOLLECTION" and "OTHER".

st_is_ext() adds a by_geometry argument that passes the results of sf::st_is() to all() if by_geometry is FALSE.

Usage

```
is_geom_type(x, type = NULL, by_geometry = FALSE, ext = TRUE)

is_point(x, by_geometry = FALSE)

is_multipoint(x, by_geometry = FALSE)

is_line(x, by_geometry = FALSE)

is_multiline(x, by_geometry = FALSE)

is_polygon(x, by_geometry = FALSE)

is_multipolygon(x, by_geometry = FALSE)

st_is_ext(x, type = NULL, by_geometry = FALSE)
```

Arguments

x	A sf or sfc object passed to sf::st_geometry_type()
type	If "POINT", check if geometry type is POINT. Same for all available geometry types; not case sensitive; Default: NULL
by_geometry	Passed to sf::st_geometry_type(); defaults to FALSE
ext	For is_geom_type(), if ext TRUE and check is NULL, return a list with checks for POINTS, POLYGONS, LINESTRING, and the returned types.

Value

- If ext is FALSE and type is NULL, returns vector with geometry types identical to `sf::st_geometry_type()`.
- If ext is TRUE, returns a list and, if type is not NULL, returns a logical vector.

`is_sf` *What is the class or spatial attributes of this feature?*

Description

What is the class or spatial attributes of this feature?

Usage

```
is_sf(x, ext = FALSE, allow_null = FALSE, allow_list = FALSE)

is_sfg(x, allow_null = FALSE)

is_sfc(x, allow_null = FALSE)

is_bbox(x, allow_null = FALSE, allow_na = FALSE)

is_raster(x, allow_null = FALSE)

is_sp(x, allow_null = FALSE)

is_geo_coords(x, allow_null = FALSE)
```

Arguments

x	An sf, sfc, or bbox object.
ext	If TRUE, check if x is a sf, sfc, or bbox class object or not; defaults to FALSE. (used by is_sf)
allow_null	If TRUE and x is NULL, return TRUE; defaults to FALSE.
allow_list	If TRUE, is_sf will return TRUE if x is a list of sf objects.
allow_na	If TRUE, `is_bbox()` ignores any NA values. Defaults to FALSE.

Details

- is_sf: is x a sf class object?
- is_sfc: is x is a sfc class object?
- is_bbox: is x is a bbox class object?
- is_sf_list: is x is a list of sf class objects (with or without names)?
- is_raster: is x a Raster class object?
- is_sp: is x a Spatial class object of any type?

- is_geo_coords: is x likely a geodetic coordinate pair (a length 2 numeric vector, with a max absolute value less than or equal to 180)?
- is_wgs84: is x using the WSG84 coordinate reference system?

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

is_sf(nc)

is_sfc(nc$geometry)

is_sf_list(list(nc[1, ], nc[2, ]))

is_bbox(sf::st_bbox(nc))

is_wgs84(sf::st_transform(nc, 4326))

is_same_crs(nc, 4267)
```

---

lonlat_to_sfc *Convert a lon/lat or lat/lon coordinate pair to a sfc object*

Description

[Experimental]

Usage

```
lonlat_to_sfc(
  x,
 range = getOption("sfext.coord_range", c(xmin = -180, ymin = -50, xmax = 180, ymax =
    60)),
  quiet = FALSE,
  call = parent.frame(),
  ...
)
```

Arguments

x	A length 2 numeric vector with geodetic coordinates in a EPSG:4326 coordinate reference system.
range	For lonlat_to_sfc(), an object that is coercible to a bbox object or a length 4 vector with names xmin, xmax, ymin, and ymax. If a coordinate pair falls outside the latitude/longitude range defined by the vector but inside the range if reversed, the coordinates are assumed to be in lat/lon order and are switched to lon/lat order before being converted to a point. Defaults to c("xmin" = -180, "ymin" = -50, "xmax" = 180, "ymax" = 60). Note that this default setting will reverse valid coordinates north of Anchorage, Alaska or south of New Zealand.

quiet	If TRUE, suppress alert messages when converting a lat/lon coordinate pair to a lon/lat pair. Defaults to FALSE.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
...	Arguments passed on to sf::st_sfc precision numeric; see st_as_binary check_ring_dir see st_read dim character; if this function is called without valid geometries, this argument may carry the right dimension to set empty geometries recompute_bbox logical; use TRUE to force recomputation of the bounding box

See Also

is_geo_coords()

---

make_sf_grid_list *Make a sf list by grid position*

Description

Create a grid with st_make_grid_ext() and

Usage

```
make_sf_grid_list(
  x,
  style = "rect",
  ncol = 2,
  nrow = 2,
  .id = "grid_id",
  crs = NULL,
  ...
)
```

Arguments

x	A sf, sfc, or bbox object, Default: NULL. Required.
style	Style of cell to return with options including "rect", "square", "hex", "flat_top_hex", "circle", "circle_offset"
ncol, nrow	Used to set n if either are not NULL; defaults to NULL. row and id are added as columns to the grid if they are provided.
.id	A name to use for the cell id column. Defaults to "id".
crs	Coordinate reference system of bounding box to return; defaults to NULL which maintains the crs of the input object.

... Arguments passed on to st_make_grid_ext

ncol,nrow Used to set n if either are not NULL; defaults to NULL. row and id are added as columns to the grid if they are provided.

gutter Distance in units between each column cell; gutter effectively serves as a margin as the negative buffer is applied to all cells (including those at the edges of the grid).

desc If TRUE, reverse standard order of cell id numbering; defaults FALSE

n If n is NULL and square is TRUE, the grid is set automatically to be 10 cells wide, Default: NULL

what "polygons", "corners", "centers"; set to centers automatically if style is "circle", "circle_offset" but a buffer is applied to return circular polygons.

filter If TRUE (or if trim is TRUE) filter grid geometry by x using st_filter_ext

unit Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"

cellsize numeric of length 1 or 2 with target cellsize: for square or rectangular cells the width and height, for hexagonal cells the distance between opposite edges (edge length is cellsize/sqrt(3)). A length units object can be passed, or an area unit object with area size of the square or hexagonal cell.

trim If TRUE, x is trimmed to y with st_trim().

---

mapview_ext *Use mapview to interactively explore spatial data*

Description

A wrapper for mapview::mapview() that drops list columns and makes it easier to quickly specify a zcol value.

Usage

```
mapview_ext(x, zcol = NULL, remove_na = FALSE, ...)

mapview_exif(
  path = NULL,
  fileext = "jpeg",
  popup = TRUE,
  tooltip = FALSE,
  images = NULL,
  width = 320,
  ...
)
```

```
mapview_popup_img(
  images,
  popup = TRUE,
  tooltip = FALSE,
  map = NULL,
  width = 320,
  ...,
  call = caller_env()
)
```

Arguments

x	a Raster* or Spatial* or Satellite or sf or stars object or a list of any combination of those. Furthermore, this can also be a data.frame, a numeric vector or a character string pointing to a tile image folder or file on disk. If missing, a blank map will be drawn. A value of NULL will return NULL.
zcol	attribute name(s) or column number(s) in attribute table of the column(s) to be rendered. See also Details.
remove_na	If TRUE and zcol is not NULL, filter NA values from the data frame column defined by zcol before passing to mapview::mapview()
...	Arguments passed on to mapview::mapview
path	A path to folder or file.
fileext	File extension. Defaults to "jpeg".
popup	If TRUE, add a popup image to a leaflet map; defaults TRUE.
tooltip	logical, whether to show image(s) as popup(s) (on click) or tooltip(s) (on hover).
images	A simple feature object with columns for the image path/url, image width, and image height.
width	the width of the image(s) in pixels.
map	an optional existing map to be updated/added to.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

See Also

mapview::mapview()

misc_sf *Additional utility functions for sf objects*

Description

- `transform_sf()` is similar to `sf::st_transform()` but supports sf, sfc, or bbox objects as the crs parameter, supports sfg objects (transformed to sfc), and uses `sf::st_set_crs()` if the CRS for the provided object is NA. This function does not support bounding box transformations.
- `relocate_sf_col()` relocates the sf geometry column after specified column (by default after everything).
- `rename_sf_col()` a wrapper for `sf::st_set_geometry()` that renames the sf column.
- `get_sf_col()` returns the "sf_column" attribute.
- `get_sf_colnames()` returns the column names of a file that can be read with `sf::read_sf()` to allow you to use column names to build a query (or provide a value for name_col) without reading the whole file.

Usage

```
transform_sf(x, crs = NULL, allow_null = TRUE, ...)

relocate_sf_col(x, .after = dplyr::everything())

rename_sf_col(x, sf_col = "geometry")

get_sf_col(x = NULL)

get_sf_colnames(x = NULL, dsn = NULL, layer = NULL, ...)
```

Arguments

x	A sf or sfc object. If x has a missing crs, the crs is set to the provided value.
crs	A coordinate reference system identifier (numeric or character) or a sf, sfc, bbox, or crs class object supported by `sf::st_crs()`.
allow_null	If TRUE and crs is NULL, return x.
...	Additional parameters passed to `sf::st_transform()` by `transform_sf()` or to `sf::read_sf()` by `get_sf_colnames()` if x is not NULL.
.after	The location to place sf column after; defaults to `dplyr::everything()`.
sf_col	Name to use for the sf column after renaming; defaults to "geometry".
dsn	data source name (interpretation varies by driver - for some drivers, dsn is a file name, but may also be a folder, or contain the name and access credentials of a database); in case of GeoJSON, dsn may be the character string holding the geojson data. It can also be an open database connection.

layer — layer name (varies by driver, may be a file name without extension); in case layer is missing, st_read will read the first layer of dsn, give a warning and (unless quiet = TRUE) print a message when there are multiple layers, or give an error if there are no layers in dsn. If dsn is a database connection, then layer can be a table name or a database identifier (see Id). It is also possible to omit layer and rather use the query argument.

---

number_features *Sort and number features by coordinates or distance*

Description

Used by layer_numbers() function from maplayer package. Supports multiple types of sorting including sorting:

- by centroid coordinates ("lon", "lat") appended with get_coords()
- by one or more bounding box min or max values ("xmin", "ymin", "xmax", "ymax") appended with get_minmax()
- by distance from the corner, side midpoint, or center of a bounding box ("dist_xmin_ymin", "dist_xmax_ymax", "dist_xmin_ymax", "dist_xmax_ymin", "dist_xmin_ymid", "dist_xmax_ymid", "dist_xmid_ymin", "dist_xmid_ymax", "dist_xmid_ymid")
- by distance to a point (or sf, sfc, or bbox object) passed to the to parameter

For example, in the eastern United States, you can sort and number features from the top-left corner of the map to the bottom right by setting sort to "dist_xmin_ymax" (default).

number_features also supports a range of different numbering styles designed to match the standard enumeration options available in LaTeX.

Usage

```
number_features(
  x,
  col = NULL,
  sort = "dist_xmin_ymax",
  to = NULL,
  desc = FALSE,
  crs = NULL,
  num_style = "arabic",
  num_start = 1,
  suffix = NULL,
  .id = "number"
)

number_sf(
  x,
  col = NULL,
```

```
  sort = "dist_xmin_ymax",
  to = NULL,
  desc = FALSE,
  crs = NULL,
  num_style = "arabic",
  num_start = 1,
  suffix = NULL,
  .id = "number"
)

sort_features(
  x,
  col = NULL,
  sort = c("lon", "lat"),
  to = NULL,
  desc = FALSE,
  crs = NULL
)

sort_sf(
  x,
  col = NULL,
  sort = c("lon", "lat"),
  to = NULL,
  desc = FALSE,
  crs = NULL
)
```

Arguments

x	A sf or sfc object.
col	Group column name, Default: NULL
sort	Sort column name, Default: "dist_xmin_ymax".
to	A sf object used to determine sort order based on distance from a feature to the center of the "to" object.
desc	If TRUE, sort descending; default FALSE.
crs	Coordinate reference to use with get_coords() or get_minmax() if "sort" any of the following: "lon", "lat", "longitude", "latitude", "xmin", "ymin", "xmax", "ymax"
num_style	Style of enumeration, either "arabic", "alph", "Alph", "roman", "Roman".
num_start	Starting number; defaults to 1.
suffix	Character to appended to "number" column. (e.g. "." for "1." or ":" for "1:"). Can also be a character vector with the same length as the number column.
.id	Name of the column to use for the feature numbers; defaults to "number".

Value

A `sf` object with a number column ordered by sort values.

`paper_sizes` *Standard paper and image sizes*

Description

Reference table of standard paper, postcard, photo print, social media image sizes, and playing card sizes for `get_paper()` and `get_social_image()` functions. Derived from visioguy/PaperSizes repo, Adobe UK guide to photo sizes and other sources.

Usage

```
paper_sizes
```

Format

A data frame with 123 rows and 9 variables:

`name` Name of paper

`series` Series

`standard` Standard

`size` Size in series

`units` Units ("in", "mm", or "px") for dimensions

`width` Width in units

`height` Height in units

`orientation` Portrait (width less than height), landscape, or square

`type` Type (paper, postcard, print, or social)

`rdeck_edit` *rdeck editor*

Description

A wrapper for `rdeck::editor_options()` that automatically converts features to WGS84.

Usage

```
rdeck_edit(features, mode = rdeck::cur_value(), initial_bounds = NULL, ...)

rdeck_select(features, ..., mode = "select")

editor_options(mode = rdeck::cur_value(), features = rdeck::cur_value())
```

Arguments

features	<wk-geometry> Features with which to initialise the editor. Requires CRS EPSG:4326.
mode	<editor-mode> The polygon editor mode. One of: • view: editor is in readonly mode • select: select/unselect features • modify: add/move/delete vertices • transform: move/scale/rotate selected features • point: draw points • linestring: draw linestrings by clicking each vertex • polygon: draw polygons by clicking each vertex • lasso: freehand polygon draw by click-dragging
initial_bounds	<rct/st_bbox/wk-geometry> Sets the initial bounds of the map if not NULL. Takes priority over initial_view_state. Accepts a bounding box, or a geometry from which a bounding box can be computed. Requires CRS EPSG:4326.
...	Arguments passed on to rdeck::rdeck map_style <string> The mapbox basemap style url. See https://docs.mapbox.com/api/maps/#mapbox-styles theme <"kepler" | "light"> The widget theme which alters the style of the legend and tooltips. initial_view_state <view_state> Defines the map position, zoom, bearing and pitch. controller <logical> If NULL or FALSE, the map is not interactive. picking_radius <number> Extra pixels around the pointer to include while picking; useful when rendering objects that are difficult to hover, e.g. thin lines, small points, etc. use_device_pixels <logical | number> Controls the resolution of drawing buffer used for rendering. • TRUE: Resolution is defined by window.devicePixelRatio. On Retina/HD displays, this resolution is usually twice as big as CSS pixels resolution. • FALSE: CSS pixels resolution is used for rendering. • number: Custom ratio (drawing buffer resolution to CSS pixel) to determine drawing buffer size. A value less than 1 uses resolution smaller than CSS pixels, improving rendering performance at the expense of image quality; a value greater than 1 improves image quality at the expense of rendering performance. blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes: • normal: Normal blending doesn't alter colours of overlapping objects. • additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps. • subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

layer_selector <boolean> If TRUE, the layer selector control will be enabled and layers with visibility_toggle = TRUE may be toggled. If FALSE, the layer selector control won't be rendered.

editor <boolean|editor_options> Whether to render the polygon editor. If TRUE, renders with the default editor_options(). If FALSE, the polygon editor is not rendered.

lazy_load **[Deprecated]**. Maps are always eagerly rendered.

width <number> The width of the map canvas.

height <number> The height of the map canvas.

id <string> The map element id. Not used in shiny applications.

---

read_sf_exif *Read EXIF metadata to create a simple feature object or write EXIF metadata to image files*

Description

read_sf_exif() read EXIF data from folder of files and, geometry is TRUE and coordinate metadata is available, convert the data to a sf object. This function also assigns a cardinal direction based on the direction metadata and recodes the orientation metadata.

For write_exif() the parameters are used to multiple tags with the same values:

- title: Title, IPTC:Headline, IPTC:ObjectName, XMP-dc:Title
- description: ImageDescription, XMP-dc:Description, and IPTC:Caption-Abstract
- keywords: Keywords, IPTC:Keywords, XMP-dc:Subject

Usage

```
read_sf_exif(
  path = NULL,
  fileext = NULL,
  filetype = NULL,
  bbox = NULL,
  sort = NULL,
  tags = NULL,
  geometry = TRUE,
  quiet = TRUE,
  ...
)
```

Arguments

path A path to folder or file.

fileext, filetype File extension or file type. filetype is used if fileext is NULL.

bbox	Bounding box to filter by.
sort	Column name for variable to sort by passed to sort_features(). Currently supports "lon", "lat", or "filename". Defaults to NULL.
tags	Optional list of EXIF tags to read from files. Must include GPS tags to create an sf object.
geometry	If TRUE (default), return a simple feature object. If FALSE, return a data.frame.
quiet	If TRUE (default), suppress function messages.
...	Additional parameters to pass to exiftoolr::exif_read()

See Also

Other read_write: read_sf_ext()

Examples

```
## Not run:
  read_sf_exif(
    path = system.file("extdata/photos", package = "overedge"),
    filetype = "jpeg"
  )

## End(Not run)
```

---

read_sf_ext *Read spatial data in a bounding box to a simple feature object from multiple sources*

Description

An extended version of sf::read_sf() that support reading spatial data based on a file path, URL, or the data name and associated package. A RDS, RDA, or RData file Optionally provide a bounding box to filter data (data is filtered before download or reading into memory where possible).

Usage

```
read_sf_ext(...)

read_sf_pkg(
  data,
  bbox = NULL,
  package = NULL,
  pkg = NULL,
  fileext = "gpkg",
  filetype = NULL,
  ...
)
```

```
read_sf_path(path, bbox = NULL, ...)

read_sf_zip(
  path,
  bbox = NULL,
  exdir = NULL,
  overwrite = TRUE,
  unzip = "internal",
  ...
)

read_sf_rdata(
  path,
  file = NULL,
  refhook = NULL,
  bbox = NULL,
  .name_repair = "check_unique",
  ...
)

read_sf_query(
  path,
  dsn = NULL,
  bbox = NULL,
  query = NULL,
  table = NULL,
  name = NULL,
  name_col = NULL,
  wkt_filter = NULL,
  zm_drop = FALSE,
  .name_repair = "check_unique",
  ...
)

read_sf_excel(
  path,
  sheet = NULL,
  combine_sheets = FALSE,
  bbox = NULL,
  coords = c("lon", "lat"),
  from_crs = 4326,
  geo = FALSE,
  address = "address",
  .name_repair = "check_unique",
  ...
)
```

```
read_sf_csv(
  path,
  url = NULL,
  bbox = NULL,
  coords = c("lon", "lat"),
  from_crs = 4326,
  geo = FALSE,
  address = "address",
  wkt = NULL,
  .name_repair = "check_unique",
  show_col_types = FALSE,
  ...
)

read_sf_url(url, bbox = NULL, coords = c("lon", "lat"), ...)

read_sf_esri(
  url,
  bbox = NULL,
  where = NULL,
  name = NULL,
  name_col = NULL,
  coords = c("lon", "lat"),
  from_crs = 4326,
  .name_repair = "check_unique",
  ...
)

read_sf_felt(
  url = NULL,
  bbox = NULL,
  map_id = NULL,
  .name_repair = "check_unique",
  ...
)

read_sf_gist(url, id = NULL, bbox = NULL, nth = 1, ...)

read_sf_gmap(
  url,
  bbox = NULL,
  layer = NULL,
  combine_layers = FALSE,
  zm_drop = TRUE,
  .name_repair = "check_unique"
)

read_sf_download(
```

```
  url,
  filename,
  bbox = NULL,
  path = NULL,
  filetype = "geojson",
  prefix = "date",
  method = "auto",
  unzip = FALSE,
  .name_repair = "check_unique",
  ...
)

read_sf_gsheet(
  url,
  sheet = NULL,
  ss = NULL,
  bbox = NULL,
  ask = FALSE,
  coords = c("lon", "lat"),
  from_crs = 4326,
  geo = FALSE,
  address = "address",
  .name_repair = "check_unique",
  ...
)
```

Arguments

...	Additional parameters passed to multiple functions; see details.
data	Name of a package dataset; used by read_sf_pkg() only.
bbox	A bounding box object; defaults to NULL. If "bbox" is provided, only returns features intersecting the bounding box.
package, pkg	Package name; used by read_sf_pkg() only. pkg is used if package is NULL.
fileext	File extension. If supplied to list_path_filenames() and pattern is NULL, only return file names matching this extension.
filetype	File type supported by sf::read_sf(); Default: 'gpkg'; used by read_sf_pkg() only and required only if the data is in the package cache directory or extdata system files.
path	A file path.
exdir	The directory to extract files to (the equivalent of unzip -d). It will be created if necessary.
overwrite	If TRUE, overwrite existing files (the equivalent of unzip -o), otherwise ignore such files (the equivalent of unzip -n).
unzip	If TRUE, url must be a zip file that is downloaded to a cache folder, unzipped into a temporary directory (created with tempdir()), and then read to a file using the specified file type.

file	The file path to read from/write to.
refhook	A function to handle reference objects.
.name_repair	Passed to repair parameter of vctrs::vec_as_names()
dsn	data source name (interpretation varies by driver - for some drivers, dsn is a file name, but may also be a folder, or contain the name and access credentials of a database); in case of GeoJSON, dsn may be the character string holding the geojson data. It can also be an open database connection.
query	SQL query to select records; see details
table	table can usually be inferred from basename of the data source. table is used to generate a custom query if both name and name_col are provided. Use sf::st_layers(dsn = dsn)[["name"]] to see a list of available table names.
name, name_col	Name value and name column to use in generated a query for sources read with read_sf_query() or read_sf_esri().
wkt_filter	character; WKT representation of a spatial filter (may be used as bounding box, selecting overlapping geometries); see examples
zm_drop	If TRUE, drop Z and/or M dimensions using sf::st_zm
sheet	Sheet to read. Either a string (the name of a sheet), or an integer (the position of the sheet). Ignored if the sheet is specified via range. If neither argument specifies the sheet, defaults to the first sheet.
coords	Character vector with coordinate values. Coordinates must use the same crs as the from_crs parameter.
from_crs	For df_to_sf(), coordinate reference system used by coordinates or well known text in data frame.
geo	If TRUE, use address_to_sf() to geocode address column; defaults to FALSE.
address	Address column name, Default: 'address'
url	A url for a spatial data file, tabular data with coordinates, or a ArcGIS FeatureServer or MapServer to access with esri2sf::esri2sf()
wkt	Name of column with well-known text for geometry. Used by read_sf_csv().
show_col_types	If FALSE, do not show the guessed column types. If TRUE always show the column types, even if they are supplied. If NULL (the default) only show the column types if they are not explicitly supplied by the col_types argument.
where	string for where condition. Default is NULL (equivalent to 1=1) to return all rows.
map_id	A Felt map URL, map ID string, or a named list with a id and type element. If map_id is a list, it must be equivalent to the output from get_felt_map() where the list includes a "id" string and a "type" string with the value "map".
id	The name of a column in which to store the file path. This is useful when reading multiple input files and there is data in the file paths, such as the data collection date. If NULL (the default) no extra column is created.
nth	For read_sf_gist(), the file to return from the gist, e.g. 1 for first, 2 for second. Defaults to 1.

`layer` layer name (varies by driver, may be a file name without extension); in case `layer` is missing, `st_read` will read the first layer of `dsn`, give a warning and (unless `quiet = TRUE`) print a message when there are multiple layers, or give an error if there are no layers in `dsn`. If `dsn` is a database connection, then `layer` can be a table name or a database identifier (see `Id`). It is also possible to omit `layer` and rather use the `query` argument.

`combine_layers`, `combine_sheets`
If `FALSE` (default), return a list with a sf object for each layer or sheet as a separate item. If `TRUE`, use `purrr::map_dfr()` to combine layers or sheets into a single sf object using the layer or sheet name as an additional column.

`filename` File name; if filename is `NULL` and path does not include a file extension, name and file extension are both required.

`prefix` File name prefix. "date" adds a date prefix, "time" adds a date/time prefix; defaults to `NULL`.

`method` Method to be used for downloading files. Current download methods are `"internal"`, `"libcurl"`, `"wget"`, `"curl"` and `"wininet"` (Windows only), and there is a value `"auto"`: see 'Details' and 'Note'.

The method can also be set through the option `"download.file.method"`: see `options()`.

`ss` Something that identifies a Google Sheet:

- its file id as a string or `drive_id`
- a URL from which we can recover the id
- a one-row `dribble`, which is how googledrive represents Drive files
- an instance of `googlesheets4_spreadsheet`, which is what `gs4_get()` returns

Processed through `as_sheets_id()`.

`ask` If `TRUE`, ask for the name of the Google Sheet to read if ss is not provided to read_sf_gsheet.

Details

Reading data from a url:

`read_sf_url()` supports multiple types of urls:

- A MapServer or FeatureServer URL
- A URL for a GitHub gist with a single spatial data file (first file used if gist contains multiple)
- A URL for a spatial data file, CSV file, Excel file, or RDS file (RDA and RData files supported by `read_sf_path()`)
- A Google Sheets URL
- A public Google Maps URL

Reading data from a package

`read_sf_pkg()` looks for three types of package data:

- Data loaded with the package
- External data in the `extdata` system files folder.
- Cached data in the cache directory returned by `rappdirs::user_cache_dir()`

Additional ... parameters

`read_sf_ext()` is a flexible function where ... are passed to one of the other read functions depending on the provided parameters. If you are using more than one parameter, all parameters *must* be named.

`read_sf_pkg()` and `read_sf_download()` both pass additional parameters to `read_sf_path()` which supports query, name_col, name, and table. name and name_col are ignored if a query parameter is provided. If table is not provided, a expected layer name is created based on the file path.

`read_sf_url()` pass the where, name_col, and name for any ArcGIS FeatureServer or MapServer url (passed to `read_sf_esri()`) or sheet if the url is for a Google Sheet (passed to `googlesheets4::read_sheet()`), or a query or wkt filter parameter if the url is some other type (passed to `sf::read_sf()`).

See Also

Other read_write: `read_sf_exif()`

`sf_bbox_corners` *Get bounding box corner points from a bbox, sfc, or sf object*

Description

If x is an sf object, the results include a column with a name matching .id containing the values "SW", "SE", "NE", and "NW" corresponding to the cardinal position of each corner.

Usage

```
sf_bbox_corners(x, .id = "corner", class = "sf")
```

Arguments

`x`	A `bbox`, length 1 `sfc` object, or a `sf` object with a single feature
`.id`	A name to use for added column with cardinal position of points, Default: 'corner'
`class`	Class of object to return. Must be "sf" (default) or "sfc".

Value

A `sf` with four features (duplicating input feature with new geometry derived from bounding box) or a length 4 `sfc` object with POINT geometry.

sf_bbox_dist *Measure a bounding box using x, y, or diagonal distance*

Description

- Measure distances (sf_bbox_dist(), sf_bbox_xdist(), sf_bbox_ydist(), and sf_bbox_diagdist())
- Convert a diag_ratio value to a distance value if a bbox and diag_ratio are provided (sf_bbox_diag_ratio_to_dist()) return NULL if bbox and diag_ratio are NULL
- Get an aspect ratio or orientation (sf_bbox_asp()) (counts asp between 0.9 and 1.1 as "square" when tolerance is 0.1) or (sf_bbox_orientation())

Usage

```
sf_bbox_dist(
  bbox,
  from,
  to,
  units = NULL,
  drop = TRUE,
  by_element = TRUE,
  call = caller_env(),
  ...
)

sf_bbox_xdist(bbox, units = NULL, drop = TRUE)

sf_bbox_ydist(bbox, units = NULL, drop = TRUE)

sf_bbox_diagdist(bbox, units = NULL, drop = TRUE)

sf_bbox_diag_ratio_to_dist(bbox, diag_ratio, units = NULL, drop = TRUE)

sf_bbox_asp(bbox)

sf_bbox_orientation(bbox, tolerance = 0.1, call = caller_env())

sf_bbox_check_fit(bbox, dist)
```

Arguments

bbox	A bbox object.
from, to	Length 2 character vectors with xy pairs (e.g. c("xmax", "ymax) defining points to measure distance from and to.
units	The units to return for sf_bbox_dist. Defaults to NULL.
drop	If FALSE, distance functions return with units. If FALSE (default), distance functions return numeric values.

by_element	logical; if TRUE, return a vector with distance between the first elements of x and y, the second, etc; an error is raised if x and y are not the same length. If FALSE, return the dense matrix with all pairwise distances.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
...	passed on to s2_distance, s2_distance_matrix, or s2_perimeter
diag_ratio	Proportion of the diagonal distance (ratio of distance to the full diagonal distance) across the bounding box.
tolerance	Numeric value above or below 1 used by sf_bbox_orientation() to describe an aspect ratio as landscape or portrait; defaults to 0.1.
dist	Distance to compare with bounding box for sf_bbox_check_fit(). If dist is length 1 compare to bounding box diagonal distance. If dist is length 2, compare to bounding box x and y distances. Return TRUE if dist fits within bounding box or FALSE if dist exceeds bounding box limits.

See Also

Other dist: compare_dist(), convert_dist_scale(), convert_dist_units(), get_measurements(), is_dist_units()

---

sf_bbox_misc *Convert and tranform bounding boxes*

Description

Simple bounding box functions that you can use to:

- Convert a bounding box to a simple feature object (sf_bbox_to_sf()) or simple feature collection (sf_bbox_to_sfc())
- Transform the coordinate reference system (sf_bbox_transform())
- Return a point from any of the corners, center, or midpoints (sf_bbox_point())
- Convert a bounding box to a SQL style query (sf_bbox_to_lonlat_query()), well known text (sf_bbox_to_wkt())
- Convert a point and a corresponding bounding box into into a npc (normalised parent coordinates) value with sf_bbox_to_npc()

If sf (>=1.0-17) is installed, sf_bbox_transform() uses sf::st_transform() *without* converting to a sfc object and back to bbox.

Usage

```
sf_bbox_to_sf(bbox, sf_col = "geometry")

sf_bbox_to_sfc(bbox)

sf_bbox_transform(bbox, crs = NULL)

sf_bbox_point(bbox, point = NULL, crs = NULL, call = caller_env())

sf_bbox_to_wkt(bbox, crs = NULL)

sf_bbox_to_lonlat_query(bbox, coords = c("longitude", "latitude"), crs = 4326)

sf_bbox_to_npc(bbox, point)
```

Arguments

bbox	A bbox object.
sf_col	name to use for geometry column after converting to simple feature object; defaults to "geometry".
crs	A coordinate reference system to use for sf_bbox_to_lonlat_query() (defaults to 4326) or for resulting bounding box (sf_bbox_transform()) or sfc object (sf_bbox_point()) (defaults to NULL).
point	point to find npc coords for center
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
coords	Column names with coordinates for query. e.g. c("X", "Y") or c("longitude", "latitude") (default). Used by sf_bbox_to_lonlat_query() only.

Value

sf object

See Also

- sf::st_as_sf(),sf::st_as_sfc()
- st_extent() in sfx package

sf_bbox_shift *Shift sides, contract, or expand a bounding box*

Description

- Shift, expand, or contract a bounding box (sf_bbox_shift(), sf_bbox_expand(), sf_bbox_contract())

Usage

```
sf_bbox_shift(
  bbox,
  nudge_x = 0,
  nudge_y = 0,
  side = c("all", "top", "bottom", "left", "right"),
  dir = NULL,
  call = caller_env()
)

sf_bbox_contract(bbox, nudge_x = 0, nudge_y = 0)

sf_bbox_expand(bbox, nudge_x = 0, nudge_y = 0)
```

Arguments

bbox	A bbox object.
nudge_x, nudge_y	Length 1 or 2 numeric vector; unitless. Used by or passed to sf_bbox_shift(). Required for sf_bbox_shift().
side	one or more sides to shift: "top", "bottom", "left", "right", or "all". Required for sf_bbox_shift().
dir	If "in", contract the bbox by nudge_x and nudge_y. If "out", expand the bbox by nudge_x and nudge_y. If dir is not NULL; absolute values are used for nudge_x and nudge_y. Defaults to NULL. Optional sf_bbox_shift().
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

sf_list *Creating and checking sf lists*

Description

Functions for coercing an object to an sf_list class objects created by vctrs::new_list_of() and checking lists of sf objects. Any function with the allow_list parameter supports this type of input.

Usage

```
as_sf_list(
  x,
  nm = "data",
  col = NULL,
  crs = NULL,
  clean_names = TRUE,
  .name_repair = "check_unique",
  call = caller_env()
)

new_sf_list(
  x,
  nm = "data",
  col = NULL,
  clean_names = TRUE,
  .name_repair = "check_unique",
  call = caller_env()
)

is_sf_list(x, ext = TRUE, allow_null = FALSE)

sf_list_rbind(x, ...)

map_as_sf_list(x, .f, ...)

map_as_sf(x, .f, ...)
```

Arguments

x	An sf, sfc, or bbox object.
nm	For as_sf_list, name(s) for sf list; defaults to "data". If col is provided, the values of the grouping column are used as names.
col	For as_sf_list, the name of the column used to group data if x is a sf object or used to group and nest data before passing to x.
crs	A character or numeric reference to a coordinate reference system supported by sf::st_crs() or another sf, sfc, or bbox object that is used to provide crs.
clean_names	If TRUE, clean names provided to nm or created based on value of col using janitor::clean_names. If FALSE, use names as provided.
.name_repair	One of "unique", "universal", or "check_unique". See vctrs::vec_as_names() for the meaning of these options.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
ext	If TRUE, check if x is a sf, sfc, or bbox class object or not; defaults to FALSE. (used by is_sf)

allow_null — If TRUE and x is NULL, return TRUE; defaults to FALSE.

... — For sf_list_rbind(), additional parameters passed to purrr::list_rbind(). For map_as_sf(), additional parameters passed to map.

.f — A function, specified in one of the following ways:

- A named function, e.g. mean.
- An anonymous function, e.g. \(x) x + 1 or function(x) x + 1.
- A formula, e.g. ~ .x + 1. You must use .x to refer to the first argument. Only recommended if you require backward compatibility with older versions of R.
- A string, integer, or list, e.g. "idx", 1, or list("idx", 1) which are shorthand for \(x) pluck(x, "idx"), \(x) pluck(x, 1), and \(x) pluck(x, "idx", 1) respectively. Optionally supply .default to set a default value if the indexed element is NULL or does not exist.

See Also

is_sf()

---

sf_to_df *Convert between simple feature and data frame objects*

Description

sf_to_df() converts a simple feature object to a data.frame by dropping geometry or, using get_coords(), converting geometry to well known text, or (if the geometry type is not POINT) getting coordinates for a centroid or point on surface. If an sfc object is provided,the "drop" geometry option is not supported.

df_to_sf() converts a data.frame to a simple feature object using a geometry column (named "geometry"), address column (name matching the address parameter), well known text column (named "wkt"), or coords (must have columns matching coords values). Related helper functions include check_coords() to suggest the appropriate coordinate column names based on the column names in the provided data frame; has_coords() to check if a data frame has the specified coordinates.

Usage

```
sf_to_df(
  x,
  crs = 4326,
  coords = c("lon", "lat"),
  geometry = "centroid",
  keep_all = TRUE
)

df_to_sf(
  x,
```

```
  crs = NULL,
  coords = c("lon", "lat"),
  from_crs = 4326,
  into = NULL,
  sep = ",",
  rev = TRUE,
  remove_coords = FALSE,
  geo = FALSE,
  address = "address",
  y = NULL,
  by = NULL,
  ...,
  as_tibble = TRUE,
  .name_repair = "unique",
  call = caller_env()
)
```

Arguments

x	A sf or sfc object or a data frame with lat/lon coordinates in a single column or two separated columns.
crs	Cordinate reference system to return, Default: 4326 for sf_to_df() and NULL for df_to_sf().
coords	Coordinate columns for input data.frame or output sf object (if geometry is 'centroid' or 'point') Default: c("lon", "lat").
geometry	Type of geometry to include in data frame. options include "drop", "wkt", "centroid", "point", Default: 'centroid'.
keep_all	If FALSE, drop all columns other than those named in coords, Default: TRUE.
from_crs	For df_to_sf(), coordinate reference system used by coordinates or well known text in data frame.
into	If coords is a single column name with both longitude and latitude, into is used as the names of the new columns that coords is separated into. Passed to tidyr::separate().
sep	If coords is a single column name with both longitude and latitude, sep is used as the separator between coordinate values. Passed to tidyr::separate().
rev	If TRUE, reverse c("lat", "lon") coords to c("lon", "lat"). check_coords() only.
remove_coords	For df_to_sf(), if TRUE, remove the coordinate columns after converting a data frame to simple feature object; defaults to FALSE.
geo	If TRUE, use address_to_sf() to geocode address column; defaults to FALSE.
address	Address column name passed to tidygeocoder::geocode() or tidygeocoder::geo
y	A sf object passed as y argument to dplyr::left_join().
by	A character vector of variables to join by passed to dplyr::left_join().
...	Other parameters passed onto methods.

as_tibble	If TRUE, use tibble::as_tibble() to make sure a sf tibble is returned.
.name_repair	Passed to tibble::as_tibble().
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

sf_to_df() returns a data frame with geometry dropped or converted to wkt or coordinates for the centroid or point on surface; df_to_sf() returns a simple feature object with POINT geometry.

See Also

sf::st_coordinates()

df_spatial() in the ggspatial package sf::st_as_sf()

Examples

```
nc <- read_sf_path(system.file("shape/nc.shp", package = "sf"))

# Convert a sf object to a data frame
nc_df <- sf_to_df(nc)

# Convert a data frame to a sf object
df_to_sf(nc_df, coords = c("lon", "lat"), remove_coords = TRUE)

# If lon and lat values are present in a single column, use the into parameter
# to split the values back into separate columns
nc_df$xy <- paste(nc_df$lon, nc_df$lat, sep = ",")

df_to_sf(nc_df, coords = "xy", into = c("lon", "lat"))
```

---

standard_scales *Standard map, architectural, and engineering scales*

Description

Standard map scales derived from USGS 2002 report on map scales https://pubs.usgs.gov/fs/2002/0015/report.pdf

Usage

```
standard_scales
```

Format

A data frame with 36 rows and 16 variables:

`scale` Scale name

`standard` Standard (USGS, architectural, or engineering)

`series` Series name (USGS map scales only)

`actual_ft` Scale distance for 1 ft actual.

`actual_ft_unit` Unit of scale for 1 ft actual.

`scale_in` Actual distance for 1 in scale.

`scale_in_unit` Unit of actual distance for 1 in scale.

`scale_in_accuracy` Accuracy of 1 in scale (approximate or exact)

`scale_cm` Actual distance for 1 cm scale.

`scale_cm_unit` Unit of actual distance for 1 cm scale.

`scale_cm_accuracy` Accuracy of 1 cm scale (approximate or exact)

`size_latlon` Standard size in latitude/longitude

`size_latlon_unit` Unit of latitude/longitude size (minutes or degrees)

`area_approx` Approximate actual area

`area_approx_unit` Approximate area unit

`series_status` Series status (select USGS map series are "abandoned")

Details

Common architectural and engineering scales derived from FEMA guide to using scales https://www.usfa.fema.gov/downloads/pdf/nfa/engineer-architect-scales.pdf

---

st_bbox_ext *Get a bounding box buffered a set distance or to match an aspect ratio*

Description

`st_bbox_ext()` converts the input to a bounding box with `as_bbox()`, applies a buffer (based on a specified distance or proportion of the diagonal distance across the bounding box) and adjusts the bounding box aspect ratio before returning a bounding box (or another class specified by the class parameter). If the input object is a list or `sf_list` object, the function always returns a `list` or `sf_list`.

Usage

```
st_bbox_ext(
  x,
  dist = NULL,
  diag_ratio = NULL,
  asp = NULL,
  unit = NULL,
  crs = NULL,
  class = "bbox",
  nudge = NULL,
  allow_null = TRUE,
  allow_list = TRUE
)

st_bbox_asp(
  x,
  asp = NULL,
  class = "bbox",
  allow_null = TRUE,
  allow_list = TRUE
)
```

Arguments

x	A sf, sfc, bbox, sfg, Raster, Spatial, Extent, numeric, or character object (a place name passed to osmdata::getbb()). See as_bbox() for more details.
dist	buffer distance in units. Optional.
diag_ratio	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when dist is provided.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, get_asp() returns the same value without modification.
unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
crs	Coordinate reference system of bounding box to return; defaults to NULL which maintains the crs of the input object.
class	Class of object to return passed to as_sf_class(); defaults to "bbox".
nudge	Passed as to parameter st_nudge() when not NULL. A numeric vector, a sf object, or any other object that can be converted to a simple feature collection with as_sfc()..
allow_null	If TRUE (default) and x is NULL, return NULL or, if FALSE, abort function.
allow_list	If TRUE (default), allow sf list objects as an input and use purrr::map() to apply the provided parameters to each object within the list to return as a new sf list object.

Details

st_bbox_asp() supports aspect ratio adjustments without applying a buffer. asp can be supplied as a number or a string matching the format of "width:height". Common aspect ratios include "1:1" (1), "4:6" (0.666), "8.5:11", "16:9" (1.777).

Value

A bbox object converted to match the class of the class parameter.

---

st_buffer_ext *Buffer a simple feature or bounding box object*

Description

Return an sf object with a buffer based on dist or a proportion of the diagonal distance defined by diag_ratio. If x uses geographic coordinates, the coordinate reference system is transformed into EPSG:3857 and then transformed back into the original CRS after the buffer has been applied.

Usage

```
st_buffer_ext(
  x,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  dist_limits = NULL,
  end_style = NULL,
  join_style = NULL,
  single_side = FALSE,
  allow_null = TRUE,
  allow_list = TRUE,
  ...
)

## Default S3 method:
st_buffer_ext(
  x,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  dist_limits = NULL,
  end_style = NULL,
  join_style = NULL,
  single_side = FALSE,
  allow_null = TRUE,
  ...
)
```

```
## S3 method for class 'bbox'
st_buffer_ext(
  x,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  dist_limits = NULL,
  end_style = NULL,
  join_style = NULL,
  single_side = FALSE,
  ...
)

## S3 method for class 'list'
st_buffer_ext(
  x,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  dist_limits = NULL,
  end_style = NULL,
  join_style = NULL,
  single_side = FALSE,
  allow_null = TRUE,
  allow_list = TRUE,
  ...
)

## S3 method for class 'sf_list'
st_buffer_ext(
  x,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  dist_limits = NULL,
  end_style = NULL,
  join_style = NULL,
  single_side = FALSE,
  allow_null = TRUE,
  allow_list = TRUE,
  ...
)

st_edge(x, dist = NULL, diag_ratio = NULL, unit = "meter", ...)
```

Arguments

x	A sf, sfc, or bbox object or a list of sf objects.

dist	buffer distance in units. Optional.
diag_ratio	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when dist is provided.
unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
dist_limits	Numeric vector of any length (minimum and maximum values used as lower and upper limits on distance buffer). Units must match the provided units; defaults to NULL.
end_style	"round" (default), "flat", or "square" passed to the endCapStyle parameter of sf::st_buffer().
join_style	"round" (default), "mitre", or "bevel" passed to the joinStyle parameter of sf::st_buffer().
single_side	If TRUE, single-sided buffers are returned for linear geometries, in which case negative dist values give buffers on the right-hand side, positive on the left.
allow_null	If TRUE (default) and x is NULL, a NULL value is returned with no error.
allow_list	If TRUE (default), allow sf list objects as an input and use purrr::map() to apply the provided parameters to each object within the list to return as a new sf list object.
...	Additional parameters passed to sf::st_buffer()

Details

st_edge() is a variation on st_buffer_ext() where dist or diag_ratio is used to define the width of the edge to return either outside the existing geometry (for positive dist values) or inside the existing geometry (for negative dist values).

---

st_cast_ext *Cast geometry of a simple feature object or simple feature collection to another type*

Description

Wrapper for sf::st_cast() that currently supports casting MULTIPOLYGON to POLYGON or MULTIPOLYGON or POLYGON to POINT or, if simplify = FALSE, can cast MULTIPOINT to LINESTRING. This is not very different than the basic functionality of st_cast but further development may improve the utility of this function.

Usage

```
st_cast_ext(x, to = "POINT", simplify = TRUE, ...)
```

Arguments

x	A sf or sfc object to cast to another type.
to	character; target type, if missing, simplification is tried; when x is of type sfg (i.e., a single geometry) then to needs to be specified.
simplify	If TRUE, simplify geometry type; defaults to TRUE.
...	Arguments passed on to sf::st_cast

---

st_clip *Clip the side or corner of a simple feature or bounding box object*

Description

Clip based on the corner of the object bounding box.

Usage

```
st_clip(
  x,
  clip = NULL,
  keep = NULL,
  flip = FALSE,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter"
)
```

Arguments

x	sf or bbox object to clip
clip	Character string describing the part of the area to clip or remove. Options include c("top", "right", "bottom", "left", "topright", "bottomright", "bottomleft", "topleft"). If NULL, the area is not clipped and a full edge can be returned.
keep	Alternate way of defining clip (by naming the section to keep).
flip	Logical. Default FALSE. If TRUE, than the clip area is kept instead of removed. If keep is provided, flip is automatically set to TRUE.
dist	Numeric. Distance to use for the edge. Default NULL meters. Use negative values for an inside edge or positive numbers for an outside edge.
diag_ratio	Alternate way to define edge distance.
unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"

Value

sf object clipped based on parameters

`st_concave_hull_ext` *Make a concave hull around simple feature object by attribute*

Description

Make a concave hull around simple feature object by attribute

Usage

```
st_concave_hull_ext(
  x,
  by = NULL,
  centroid = FALSE,
  ratio = 0.5,
  allow_holes = FALSE
)
```

Arguments

x	A sf object.
by	Column name to use for grouping and combining geometry, Default: NULL
centroid	If TRUE, use centroids for geometry of x, Default: FALSE
ratio	numeric; fraction convex: 1 returns the convex hulls, 0 maximally concave hulls
allow_holes	logical; if TRUE, the resulting concave hull may have holes

---

`st_dissolve` *Dissolve geometry preserving existing or supplied grouping variables*

Description

`st_dissolve()` dissolves sf and sfc objects. If the input object is an sf object, any existing grouping variables or added grouping variables passed to the `.by` parameter are included in the output sf data frame.

Usage

```
st_dissolve(
  x,
  ...,
  .by = NULL,
  .keep = "nest",
  do_union = TRUE,
  .data_key = "data",
  .dissolve_key = "group.comp.id",
  call = caller_env
)
```

Arguments

x	A sf or sfc object. If sf object, a grouped data frame is allowed.
...	Arguments passed on to spdep::poly2nb pl list of polygons of class extending SpatialPolygons, or an sf or sfc object containing non-empty (multi-)polygon objects row.names character vector of region ids to be added to the neighbours list as attribute region.id, default seq(1, nrow(x)); if pl has row.names, they are used instead of the default sequence. snap boundary points less than snap distance apart are considered to indicate contiguity; used both to find candidate and actual neighbours for planar geometries, but only actual neighbours for spherical geometries, as spherical spatial indexing itself injects some fuzzyness. If not set, for all SpatialPolygons objects, the default is as before sqrt(.Machine$double.eps), with this value also used for sf objects with no coordinate reference system. For sf objects with a defined coordinate reference system, the default value is 1e-7 for geographical coordinates (approximately 10mm), is 10mm where projected coordinates are in metre units, and is converted from 10mm to the same distance in the units of the coordinates. Should the conversion fail, snap reverts to sqrt(.Machine$double.eps). queen if TRUE, a single shared boundary point meets the contiguity condition, if FALSE, more than one shared point is required; note that more than one shared boundary point does not necessarily mean a shared boundary line useC default TRUE, doing the work loop in C, may be set to false to revert to R code calling two C functions in an n*k work loop, where k is the average number of candidate neighbours foundInBox default NULL using R code or st_intersects() to generate candidate neighbours (using snap= if the geometries are not spherical); if not NULL (for legacy purposes) a list of length (n-1) with integer vectors of candidate neighbours (j > i) (as created by the poly_findInBoxGEOS function in **rgeos** for clean polygons)
.by	**[Experimental]** <tidy-select> Optionally, a selection of columns to group by for just this operation, functioning as an alternative to group_by(). For details and examples, see ?dplyr_by.

.keep	Method for handling attributes for input data. By default ("nest"), input data is converted to a nested list column with the name from .data_key. Any other values
do_union	Use sf::st_union() to dissolve geometry if TRUE (default). Use sf::st_combine() if FALSE.
.data_key	Passed to .data argument of tidyr::nest()
.dissolve_key	Used as the column name for the dissolve grouping variable. x can't have any existing columns with this name.
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

---

st_erase *Erase or trim geometry of a sf or sfc object*

Description

st_erase() extends sf::st_difference() by unioning the second parameter by default, checking validity of inputs, and optionally (when flip = TRUE) using sf::st_intersection() instead of sf::st_difference. st_trim() is equivalent to st_erase() with flip set to TRUE.

Usage

```
st_erase(x, y, flip = FALSE, union = TRUE, combine = FALSE, ...)

st_trim(x, y, union = TRUE, combine = FALSE, ...)
```

Arguments

x	A sf, sfc, or bbox object to erase or trim.
y	A sf, sfc, or bbox object to use to erase or trim.
flip	If TRUE, use sf::st_intersection() to "erase" geometry of x that intersects y; if FALSE use sf::st_difference() to trim x to y geometry, Default: FALSE.
union	If TRUE, union y with sf::st_union() before applying difference/intersection; defaults to TRUE.
combine	If TRUE and union = TRUE, combine y with sf::st_combine() before unioning. Defaults to FALSE.
...	arguments passed on to s2_options

Examples

```
nc <- read_sf_ext(system.file("shape/nc.shp", package = "sf"))

nc <- st_transform_ext(nc, 3657)

plot(
  st_erase(
    sf::st_buffer(nc[1, ], 1000),
    nc[1, ]
  ),
  max.plot = 1
)

plot(
  st_trim(
    nc,
    sf::st_buffer(nc[1, ], 2000)
  ),
  max.plot = 1
)
```

`st_filter_ext` *Filter, crop, trim, or erase a simple feature object or list*

Description

Extend `sf::st_filter()` to filter a sf list or a `sf`, `sfc`, or `bbox` with options to crop, trim or erase the geometry of the input object based on a predicate function. Returns x transformed to match crs if y is `NULL`.

Usage

```
st_filter_ext(
  x,
  y = NULL,
  crop = FALSE,
  trim = FALSE,
  erase = FALSE,
  crs = NULL,
  .predicate = sf::st_intersects,
  type = NULL,
  allow_list = TRUE,
  ...
)

st_filter_geom_type(x, type = NULL)
```

Arguments

x, y	A sf, sfc, or bbox object. x may also be a sf list objects. If x is an sf list, additional parameters in ... will be ignored.
crop	If TRUE, x is cropped to y using sf::st_crop().
trim	If TRUE, x is trimmed to y with st_trim().
erase	If TRUE, x is erased by y with st_erase().
crs	Coordinate reference system to return.
.predicate	geometry predicate function with the same profile as sf::st_intersects(); see details for sf::st_filter() for more options.
type	Geometry type.
allow_list	If TRUE, x can be a list of sf, sfc, or bbox objects. If FALSE, only sf, sfc, or bbox objects are supported. Defaults to TRUE.
...	Arguments passed on to sf::st_filter

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

plot(
  st_filter_ext(
    nc,
    nc[c(1:10), ]
  ),
  max.plot = 1
)

plot(
  st_filter_ext(
    nc,
    nc[c(1:10), ],
    crop = TRUE
  ),
  max.plot = 1
)


plot(
  st_filter_ext(
    nc,
    nc[c(1:10), ],
    erase = TRUE
  ),
  max.plot = 1
)

plot(
  st_filter_ext(
```

```
    nc,
    sf::st_union(nc[c(1:10), ]),
    .predicate = sf::st_disjoint
  ),
  max.plot = 1
)

st_filter_geom_type(nc, "POINT")

st_filter_geom_type(nc, "MULTIPOLYGON")
```

`st_filter_pct` *Filter by share of length or area of one geometry overlapping with a second geometry*

Description

[Experimental]

Usage

```
st_filter_pct(x, y, pct = NULL, ...)

st_filter_pct_area(x, y, pct = NULL)

st_filter_pct_length(x, y, pct = NULL)
```

Arguments

`x`	A sf object to filter.
`y`	A sf or sfc object to filter by.
`pct`	Percent of length or area to use as a threshold value for filter. Numeric value of 1 or less.
`...`	Additional parameters. Not used currently.

Value

A filtered version of the input sf object.

st_join_ext — *Complete a spatial join using a simple feature objects or an object and list*

Description

Wrapper for `sf::st_join()` that works with sf lists.

Usage

```
st_join_ext(x, y, col = NULL, .id = "name", join = NULL, ...)
```

Arguments

x	A `sf`, `sfc`, or `bbox` object.
y	An `sf` object with a column named "name" or a list of sf objects where all items in the list have a "name" column.
col	Column name used to convert y into a `sf` list if it is not already.
.id	Column name with the values that should be added as a column to the input `sf` object.
join	geometry predicate function; defaults to `NULL`, set to `sf::st_intersects()` if y contains only POLYGON or MULTIPOLYGON objects or `sf::st_nearest_feature()` if y contains other types.
...	Additional parameters passed to `sf::st_join()`

st_make_grid_ext — *Make a grid over a simple feature bounding box*

Description

Create a grid with an id column and optionally a set number of columns and rows. This documentation is incomplete the function may change.

Usage

```
st_make_grid_ext(
  x,
  ...,
  unit = NULL,
  crs = NULL,
  ncol = NULL,
  nrow = NULL,
  n = NULL,
  gutter = 0,
```

```
  desc = FALSE,
  cellsize = NULL,
  what = NULL,
  style = "rect",
  .id = "id",
  filter = FALSE,
  trim = FALSE
)
```

Arguments

x	A sf, sfc, or bbox object, Default: NULL. Required.
...	Arguments passed on to st_bbox_ext nudge Passed as to parameter st_nudge() when not NULL. A numeric vector, a sf object, or any other object that can be converted to a simple feature collection with as_sfc().. dist buffer distance in units. Optional. diag_ratio ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when dist is provided. allow_list If TRUE (default), allow sf list objects as an input and use purrr::map() to apply the provided parameters to each object within the list to return as a new sf list object. asp Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, get_asp() returns the same value without modification.
unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
crs	Coordinate reference system of bounding box to return; defaults to NULL which maintains the crs of the input object.
ncol, nrow	Used to set n if either are not NULL; defaults to NULL. row and id are added as columns to the grid if they are provided.
n	If n is NULL and square is TRUE, the grid is set automatically to be 10 cells wide, Default: NULL
gutter	Distance in units between each column cell; gutter effectively serves as a margin as the negative buffer is applied to all cells (including those at the edges of the grid).
desc	If TRUE, reverse standard order of cell id numbering; defaults FALSE
cellsize	numeric of length 1 or 2 with target cellsize: for square or rectangular cells the width and height, for hexagonal cells the distance between opposite edges (edge length is cellsize/sqrt(3)). A length units object can be passed, or an area unit object with area size of the square or hexagonal cell.
what	"polygons", "corners", "centers"; set to centers automatically if style is "circle", "circle_offset" but a buffer is applied to return circular polygons.

style	Style of cell to return with options including "rect", "square", "hex", "flat_top_hex", "circle", "circle_offset"
.id	A name to use for the cell id column. Defaults to "id".
filter	If TRUE (or if trim is TRUE) filter grid geometry by x using st_filter_ext
trim	If TRUE, x is trimmed to y with st_trim().

See Also

sf::st_make_grid

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

# Make a 2 by 2 grid across a location with a 1000 meter gutter between each cell
plot(
  st_make_grid_ext(
    x = nc[24, ],
    dist = 500,
    unit = "meter",
    ncol = 2,
    nrow = 2,
    gutter = 1000
  )
)

# Make a 5 by 5 grid with a 8.5 by 11 aspect ratio filtered to x
plot(
  st_make_grid_ext(
    x = nc[24, ],
    asp = 8.5 / 11,
    ncol = 5,
    nrow = 5,
    filter = TRUE
  )
)

# Make a 4 by 5 grid of circles trimmed to x boundaries
plot(
  st_make_grid_ext(
    x = nc[24, ],
    ncol = 4,
    nrow = 5,
    style = "circle_offset",
    trim = TRUE
  ),
  max.plot = 1
)
```

st_make_valid_ext *Checks if all geometries are already valid and make valid if not*

Description

Checks if all geometries are already valid and make valid if not

Usage

```
st_make_valid_ext(x, ...)
```

Arguments

x	object of class sfg, sfc or sf
...	passed on to s2_options

st_misc *Modify the geometry of a simple feature or bounding box object*

Description

Support sf, sfc, and bbox and objects as inputs.

- Get the center point for a sf object
- Get a circumscribed circle or inscribed circle in a sf object
- Get a donut for a sf object (may not work if inscribed = TRUE)

st_inscribed_square wraps sf::st_inscribed_circle() but limits the circle to 1 segment per quadrant (nQuadSegs = 1) and then rotates the resulting geometry 45 degrees to provide a (mostly) inscribed square. A different rotation value can be provided to change the orientation of the shape, e.g. rotate = -45 to return a diamond shape. st_square() wraps st_bbox_ext() with asp = 1.

Usage

```
st_center(x, class = "list", ext = TRUE, ...)

st_circle(
  x,
  scale = 1,
  inscribed = TRUE,
  dTolerance = 0.01,
  by_feature = FALSE,
  use_hull = FALSE,
  use_lwgeom = FALSE
)
```

```
st_circumscribed_circle(x, scale = 1, dTolerance = 0, by_feature = FALSE)

st_donut(x, width = 0.4, scale = 1, inscribed = FALSE, by_feature = TRUE, ...)
```

Arguments

x	A sf, sfc, or bbox object
class	Class to return for st_center(): "sfc", "sf", "geometry" (original input geometry), "x" (original input object), "crs" (original input crs), or "list" (including all other class types).
ext	If TRUE, st_center returns a list with the centroid as a sfc object, as an sf object (with lon and lat values), the original geometry (x), and the original crs. objects; defaults TRUE. If FALSE, return an sf object.
...	Additional parameters passed to sf::st_centroid() by st_center() or st_circle() by st_donut().
scale	For st_donut(), scale to apply to donut using st_circle(). Defaults to 1 which keeps the donut as the same size as the input object.
inscribed	If TRUE, make circle, square, or donut inscribed within x, if FALSE, make it circumscribed.
dTolerance	numeric; tolerance parameter, specified for all or for each feature geometry. If you run st_simplify, the input data is specified with long-lat coordinates and sf_use_s2() returns TRUE, then the value of dTolerance must be specified in meters.
by_feature	For st_donut(), if TRUE the input object x is unioned before converting into a donut geometry. If FALSE, each feature in the input data remains a separate feature in the output.
use_hull	For st_circle(), if TRUE use the geometry from sf::st_convex_hull() (to address issues with MULTIPOLYGON objects).
use_lwgeom	If TRUE, by_feature = TRUE and inscribed = FALSE, use lwgeom::st_minimum_bounding_circle().
width	Donut width as proportion of outer size.

See Also

- sf::geos_unary

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
nc <- sf::st_transform(nc, crs = 3857)

plot(nc, max.plot = 1)

plot(st_circumscribed_circle(nc, by_feature = FALSE), max.plot = 1)
plot(st_circle(nc, by_feature = FALSE), max.plot = 1, add = TRUE)

plot(st_donut(nc[1:10,], by_feature = TRUE), max.plot = 1)
```

st_nudge *Nudge a simple feature to the center of another feature and/or a set distance*

Description

Nudge, move, or shift a `sf`, `sfc`, or `bbox` object to the center of another feature and/or by a set distance.

Usage

```
st_nudge(
  x,
  to = NULL,
  nudge_y = 0,
  nudge_x = 0,
  unit = NULL,
  scale = 1,
  rotate = 0,
  crs = NULL
)

## Default S3 method:
st_nudge(
  x,
  to = NULL,
  nudge_y = 0,
  nudge_x = 0,
  unit = NULL,
  scale = 1,
  rotate = 0,
  crs = NULL
)

## S3 method for class 'bbox'
st_nudge(x, ...)

## S3 method for class 'sf'
st_nudge(x, ...)
```

Arguments

`x`	Object to convert to an sf, sfc, bbox or a sf list object.
`to`	sf object to use as new center for x or length 2 numeric vector with the nudge_y and nudge_x distance (in that order).

nudge_y, nudge_x
: Distance to nudge geometry in unit. If unit is NULL, distance is assumed to be in the same units as the coordinate reference system of the input object.

unit
: Units for nudge_y and nudge_x distance (also used if to is numeric).

scale
: numeric; scale factor, Default: 1

rotate
: numeric; degrees to rotate (-360 to 360), Default: 0

crs
: Coordinate reference system for sf, bbox, sfc or sf list object to return.

...
: Additional parameters passed to as_sf, as_sfc, as_bbox, or as_sf_list

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
nc <- sf::st_transform(nc, crs = 3857)

plot(sf::st_union(st_nudge(nc, to = nc[1, ]), nc), max.plot = 1)
```

---

st_scale_rotate *Scale and rotate a simple feature object, simple feature collection, or bounding box*

Description

Scale or rotate a simple feature or bounding box object using affine transformations.

Usage

```
st_scale_rotate(x, scale = 1, rotate = 0, call = caller_env())
```

Arguments

x
: A sf, sfc, or bbox object or another object coercible to a simple feature collection with as_sfc().

scale
: numeric; scale factor, Default: 1

rotate
: numeric; degrees to rotate (-360 to 360), Default: 0

call
: The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
nc <- sf::st_transform(nc, crs = 3857)

plot(st_scale_rotate(nc, scale = 0.75, rotate = 15), max.plot = 1)
```

st_square *Create a square within or around a simple feature object or collection*

Description

Get a circumscribed square or approximate inscribed square in a sf object. If inscribed is TRUE, the square geometry returned may not be contained wholly within the original geometry. The inscribed square is created from the bounding box of an inscribed circle rotated 45 degrees.

Usage

```
st_square(
  x,
  scale = 1,
  rotate = 0,
  inscribed = FALSE,
  by_feature = FALSE,
  call = caller_env()
)

## Default S3 method:
st_square(x, ...)

## S3 method for class 'sfc'
st_square(
  x,
  scale = 1,
  rotate = 0,
  inscribed = FALSE,
  by_feature = FALSE,
  call = caller_env()
)

## S3 method for class 'sf'
st_square(x, ..., by_feature = FALSE)

## S3 method for class 'bbox'
st_square(x, ...)

st_inscribed_square(x, scale = 1, rotate = 0, by_feature = FALSE)
```

Arguments

x	A sf, sfc, or bbox object or another object coercible to a simple feature collection with as_sfc().
scale	numeric; scale factor, Default: 1

rotate	numeric; degrees to rotate (-360 to 360), Default: 0
inscribed	If TRUE, the returned geometry is inscribed within x, if FALSE (default), the geometry is circumscribed.
by_feature	If TRUE, create new geometry for each feature. If FALSE, create new geometry for all features combine with st_union_ext().
call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
...	Additional parameters passed to st_square() sfc method if x is a sf or bbox object.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
nc <- sf::st_transform(nc, crs = 3857)

plot(st_square(nc), max.plot = 1)
plot(st_square(nc[1:10, ], by_feature = TRUE), max.plot = 1)
```

---

st_transform_ext *Transform or convert coordinates of a simple feature or bounding box object*

Description

This function wraps sf::st_transform() but supports a wider range of input objects and, using the as_sf_class() function, returns a wider range of objects. Typically, takes a sf, sfc, or bbox object and transform to coordinate reference system to match the value of crs or the object provided to crs. If x is a data.frame or if x is NULL and allow_null is TRUE (defaults to FALSE) it is returned as is.

Usage

```
st_transform_ext(
  x,
  crs = NULL,
  class = NULL,
  rotate = 0,
  allow_null = FALSE,
  allow_list = TRUE
)

st_omerc(x, rotate = 0)

st_wgs84(x)
```

Arguments

x	An sf, sfc, or bbox object, a list of sf objects, or a data.frame object (always returned as is).
crs	A character or numeric reference to a coordinate reference system supported by sf::st_crs() or another sf, sfc, or bbox object that is used to provide crs.
class	Class of object to return (sf or bbox). If x is an sf list, the returned object remains a list but may be converted to bbox if class = "sf".
rotate	If rotate is greater or less than 0, st_transform_ext() calls st_omerc() and returns an object with the Oblique Mercator projection passing the value of rotate to the gamma parameter of the projection. rotate must be between -45 and 45 degrees.
allow_null	If TRUE and x is NULL return x without an error. Defaults to FALSE.
allow_list	If TRUE, x can be a list of sf, sfc, or bbox objects. If FALSE, only sf, sfc, or bbox objects are supported. Defaults to TRUE.

Value

An sf, sfc, or bbox object transformed to a new coordinate reference system.

See Also

sf::st_transform(),sf::st_crs()

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

nc_bbox <- sf::st_bbox(nc)

nc_3857 <- st_transform_ext(nc, 3857)

st_transform_ext(nc_bbox, crs = 4326)

sf::st_crs(st_transform_ext(nc_3857, crs = nc))$input

sf::st_crs(st_wgs84(nc_3857))$input
```

---

st_union_ext *Union simple feature objects and combine name column values*

Description

Wrapper for sf::st_union() supporting the additional feature of a combined name column collapsed into a single vector with cli::pluralize(). st_union_by() wraps dplyr::group_by() and dplyr::summarise() to allow quick unioning geometry by the passed parameters.

Usage

```
st_union_ext(
  x,
  y = NULL,
  name_col = "name",
  .sf_col = NULL,
  label = NULL,
  ext = TRUE,
  ...
)

st_union_by(x, ..., .sf_col = NULL)
```

Arguments

x	A sf, sfc, or bbox object or sf only for st_union_by() Required.
y	A sf or sfc object, defaults to NULL.
name_col	Column name to collapse into new name_col value, Default: 'name'
.sf_col	Geometry column name to use if x is an sf object, Default: NULL.
label	Length 1 character vector used if name_col is NULL.
ext	If FALSE, st_union_ext() functions the same as sf::st_union(), Default: TRUE
...	Additional parameters passed to sf::st_union() or dplyr::group_by().

Value

A sfc object if y is NULL and ext is FALSE. A tibble sf if x is a sf object and y is NULL. If y is provided, st_union_ext() is identical to sf::st_union()

Examples

```
nc <- read_sf_path(system.file("shape/nc.shp", package = "sf"))
nc_union <- st_union_ext(nc[10:15,], name_col = "NAME")
nc_union
plot(nc_union)
```

---

write_exif_from *Write EXIF data for photos on spatial join with a sf object or list of sf objects*

Description

Extends read_sf_exif() and filenamr::write_exif()

Usage

```
write_exif_from(
  path,
  fileext = NULL,
  filetype = NULL,
  from,
  .id = "name",
  tag = "keywords",
  join = NULL,
  overwrite = TRUE
)
```

Arguments

path	A path to folder or file.
fileext, filetype	File extension or file type. filetype is used if fileext is NULL.
from	A sf object or list of sf objects where each object has a column with a name matching the .id parameter. The attribute value in this column are used to assign the tag parameter to the file at the provided path based on the spatial relationship set by join. For example, from may be boundary data used to assign keywords based on photo locations.
.id	Column name in from with the values to use for tag values.
tag	EXIF tag to update, supported options include "keywords", "title", or "description".
join	geometry predicate function; defaults to NULL, set to sf::st_intersects if from contains only POLYGON or MULTIPOLYGON objects or sf::st_nearest_feature if from contains other types.
overwrite	If TRUE, overwrite any existing EXIF metadata present in the provided fields; defaults to TRUE

---

write_sf_ext *Write or cache a simple feature object to a file*

Description

The write_sf_ext and write_sf_cache helper functions wrap the `sf::write_sf()` function to provide some additional options including consistent file naming with `make_filename()` and features including:

Usage

```
write_sf_ext(
  data,
  name = NULL,
  label = NULL,
  prefix = NULL,
  postfix = NULL,
  filename = NULL,
  fileext = NULL,
  filetype = NULL,
  description = NULL,
  path = NULL,
  cache = FALSE,
  pkg = "sfext",
  overwrite = FALSE,
  onefile = FALSE,
  ...
)

write_sf_list(
  data,
  name = NULL,
  label = NULL,
  prefix = NULL,
  postfix = NULL,
  filename = NULL,
  fileext = NULL,
  filetype = NULL,
  path = NULL,
  overwrite = FALSE,
  onefile = FALSE,
  cache = FALSE,
  ...
)

write_sf_cache(
  data,
  name = NULL,
  label = NULL,
  prefix = NULL,
  postfix = NULL,
  filename = NULL,
  fileext = NULL,
  filetype = NULL,
  data_dir = NULL,
  pkg = "sfext",
  overwrite = FALSE,
  create = TRUE,
```

```
  ...
)

write_sf_gist(
  data,
  name = NULL,
  label = NULL,
  prefix = NULL,
  postfix = NULL,
  filename = NULL,
  fileext = "geojson",
  filetype = NULL,
  description = NULL,
  public = TRUE,
  browse = FALSE,
  token = Sys.getenv("GITHUB_PAT")
)

write_sf_gsheet(
  data,
  name = NULL,
  label = NULL,
  prefix = NULL,
  postfix = NULL,
  filename = NULL,
  sheet = 1,
  ask = FALSE,
  key = NULL,
  ...
)
```

Arguments

`data`	A `sf` object, data frame, or other object to write.
`name`	Name to make file name converted to snake case with `janitor::make_clean_names()`, e.g. "Residential zoning map" becomes "residential_zoning_map". If the name includes a file extension it is assumed that the filename has been provided as the name parameter.
`label`	Label to combine with name converted to snake case with `janitor::make_clean_names()`. The label is designed to identify the area or other shared characteristics across multiple data files, maps, or plots. label is ignored if name is NULL or if name includes a file extension.
`prefix`	File name prefix. "date" adds a date prefix, "time" adds a date/time prefix; defaults to `NULL`.
`postfix`	File name postfix; defaults to `NULL`.
`filename, fileext, filetype`	File name and/or file extension to write. filetype is superseded in favor of fileext. Both are optional if path includes filename and type, e.g. "~/Docu-

	ments/data.geojson". fileext can be provided as part of the filename, e.g. "data.geojson". If a filename includes a file extensions and a separate fileext is also provided, the separate fileext parameter is used. Supported file extensions include "csv", "xlsx", "gsheet" (writes a Google Sheet), "rda", or any fileext supported by the available drivers (use `sf::st_drivers()` to list drivers).
`description`	(character) Brief description of gist (optional)
`path`	Path to file or data directory. Optional. If path includes a file extension and filename and fileext are both `NULL`, the filename and extension included with path will be used instead. If multiple file extensions are provided to filename, path, or fileext, `make_filename()` will abort.
`cache`	If `TRUE`, write `sf` object to file in cache directory; defaults to `FALSE`.
`pkg`	The name of the package cache directory to use for write_sf_cache or write_sf_ext if `cache = TRUE`.
`overwrite`	Logical. Default `FALSE`. If `TRUE`, overwrite any existing cached files that use the same file name.
`onefile`	If `TRUE` and the fileext if "gpkg" (directly or from filename), save a sf list as a multilayer GeoPackage file where names for list items are used as layer names.
`...`	If data is an sf object and the fileext is "csv" or "xlsx", the ... parameters are passed to `sf_to_df()` or to `sf::write_sf()` otherwise. If fileext is "rda" ... parameters are passed to `readr::write_rds()`.
`data_dir`	cache data directory, defaults to `rappdirs::user_cache_dir()` when data_dir is `NULL`. (only used for `write_sf_cache()`; default is used when `cache = TRUE` for `write_sf_ext()`)
`create`	If `FALSE` and path does not exist, return path with a warning. If `TRUE` and `rlang::is_interactive()` is `TRUE`, ask user if directory should be created. If the session not interactive and create is `TRUE`, a new directory will be created.
`public`	(logical) Whether gist is public (default: TRUE)
`browse`	(logical) To open newly create gist in default browser (default: TRUE)
`token`	A personal access token on GitHub with permission to create gists; defaults to Sys.getenv("GITHUB_PAT")
`sheet`	Sheet to write into, in the sense of "worksheet" or "tab". You can identify a sheet by name, with a string, or by position, with a number.
`ask`	If `TRUE`, the user is prompted to make revisions to the created Google Sheet. When user responds to the prompt, the date is read back into the environment using read_sf_gsheet and joined to the provided data with the column name provided to key. Defaults to `FALSE`.
`key`	If ask is `TRUE`, a key is required to join the sheet data to the provided data.

Details

- If fileext is "csv", "xlsx", or "gsheet" the file is converted to a dataframe using `df_to_sf()`
- If the data is not an `sf` object and none of these filenames are provided, the user is prompted to save the file as an rda file with `readr::write_rds()`.
- If cache is `TRUE` use `write_sf_cache()` to cache file after writing a copy to the path provided.

- If data is a named sf list, pass the name of each sf object in the list to the name parameter and keep all other parameters consistent to write a file for each object in the list. No ... parameters are passed if data is an sf list.

See Also

sf::st_write()

`write_sf_svg` *Write an sf object to an svg file*

Description

write_sf_svg() uses plot() and svg() to create a simple plot of an sf object geometry. This function is convenient for working with designers or other collaborators interested in using spatial data outside of R or a desktop GIS application.

Usage

```
write_sf_svg(data, filename = NULL, path = NULL, ..., width = 10, height = 10)
```

Arguments

data	A sf object to save as a svg file.
filename	the file path of the output file(s). The page number is substituted if a C integer format is included in the character string, as in the default. (Depending on the platform, the result must be less than PATH_MAX characters long, and may be truncated if not. See pdf for further details.) Tilde expansion is performed where supported by the platform.
path	File path to combine with filename. Optional if filename is provided. filename is optional if path includes a svg file extension.
...	Arguments passed on to grDevices::svg pointsize the default pointsize of plotted text (in big points). onefile should all plots appear in one file or in separate files? family one of the device-independent font families, "sans", "serif" and "mono", or a character string specify a font family to be searched for in a system-dependent way. On unix-alikes (incl.\ macOS), see the 'Cairo fonts' section in the help for X11. bg the initial background colour: can be overridden by setting par("bg"). antialias string, the type of anti-aliasing (if any) to be used; defaults to "default". symbolfamily a length-one character string that specifies the font family to be used as the "symbol" font (e.g., for plotmath output).
width	the width of the device in inches.
height	the height of the device in inches.

Index