Package: getdata (via r-universe)

August 31, 2024

Type Package

Title Get Easy Access to Tabular and Spatial Data

Version 0.1.0.9005

Description Download and format spatial and non-spatial data with simple filtering by location.

License MIT + file LICENSE

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

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

Depends R (>= 2.10)

Imports cli (>= 2.5.0), cliExtras (>= 0.1.0), DBI, dplyr, esri2sf (>= 0.2.0), glue, httr2, janitor, lifecycle, magrittr, rlang (>= 1.1.0), sf, sfext (>= 0.0.0.9001), tibble, tidyselect, utils, vctrs

Suggests bingmapr (>= 0.1.0), covr, elevatr, filenamr, FlickrAPI (>= 0.1.0.1), googlesheets4, knitr, labelled, lubridate, mapboxapi, naniar, osmdata (>= 0.1.10), rairtable, readr, roxygen2, RSocrata, smoothr, stringr, testthat (>= 3.0.0), tidygeocoder, tidyr, tigris (>= 2.0), units, withr

Remotes Chicago/RSocrata, elipousson/bingmapr, elipousson/cliExtras, elipousson/esri2sf, elipousson/filenamr, elipousson/rairtable@dev, elipousson/sfext

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

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

RemoteRef HEAD

RemoteSha 93b84190e920d91fc916d4616869a7f8f6d7a95c

Contents

`as_date_range` *Use lubridate to convert an object to a date range*

Description

Use `lubridate::as_date()` to convert an object to a length 2 list with a minimum and maximum date. By default the dates in the list will be named named "start" and "end". `date_range_query()` is a variation that returns a query string that can be passed to the "where" parameter of `get_esri_data()`. `between_date_range()` works the same way identical but uses the BETWEEN DATE syntax. `check_date_range()` validates whether a date or date range falls within supplied limits.

Usage

```
as_date_range(
  x = NULL,
  year = NULL,
  days = 90,
  ...,
  start_date = NULL,
  end_date = NULL,
  limits = NULL,
  nm = c("start", "end"),
  call = caller_env()
)

date_range_query(x = NULL, .col = "date", ..., nm = c("start", "end"))

between_date_range(x = NULL, .col = "date", ..., nm = c("start", "end"))

check_date_range(
  x = NULL,
  ...,
  limits = NULL,
  nm = c("start", "end"),
  call = caller_env()
)
```

Arguments

x	Date range as character vector in format of `c("<start date>", "<end date>")`. If length 1 and days is not `NULL`, return a range based on `c(date_range, date_range + lubridate::days(days))`. `as_date_range()` also allows a Date class object or a list of Date class objects. For all other functions, x can also be a named list of Date objects with names matching nm.
year	If date_range is `NULL` and year is provided, date range is set to `c("<year>-01-01", "<year>-12-31")`. year is ignored if date_range is provided.
days	Default range duration in days to use if date_range is length 1.
...	Arguments passed on to `lubridate::as_date`
start_date, end_date	Start and end date used if year and date_range are both `NULL`.
limits	Optional range of allowed dates. If dates supplied to `as_date_range()` falls outside limits range, abort function.
nm	Names to use for returned date range list. Defaults to `c("start", "end")`.
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.
.col	Name of date column to use for query. Defaults to "date".

Value

A length 2 list with min and max Date values.

Examples

```
as_date_range("2022-01-01", days = 10)

as_date_range(c("2022-01-01", "2022-01-31"))

as_date_range(year = 2022)

date_range_query(c("2022-01-01", "2022-01-31"))

# check_date_range("2022-09-01", limits = c("2022-07-01", "2022-09-30"))
```

---

cache_location_data *Cache location data with* `sf::write_sf()` *or* `readr::write_rds()`

Description

`cache_location_data()` is a variant on `get_location_data()` that saves the returned data to a cache directory using `sf::write_sf()` (if data is a sf object) or `readr::write_rds()` (if data is another class). Additional parameters are ignored if location is `NULL`.

Usage

```
cache_location_data(
  data = NULL,
  ...,
  location = NULL,
  name = NULL,
  label = NULL,
  fileext = "gpkg",
  filename = NULL,
  path = NULL,
  prefix = NULL,
  postfix = NULL,
  cache = TRUE,
  pkg = "getdata",
  create = TRUE,
  overwrite = FALSE,
  compress = c("none", "gz", "bz2", "xz"),
  version = 3,
  call = caller_env()
)
```

Arguments

data	Character string (e.g. url, file path, or name of data from package) for a spatial data or a sf, sfc, or bbox object with geometry overlapping the location. If data is NULL, all unnamed parameters are passed to sfext::read_sf_ext() with a bbox based on location. If data is not NULL and not a data.frame, url, file path, or bbox, conversion to a sf object will still always be attempted with sfext::as_sf().
...	Arguments passed on to get_location_data
	pkg,package Name of the package to search for data.
	fileext,filetype File extension or type to use if passing parameters to sfext::read_sf_download() or sfext::read_sf_pkg() (required for extdata and cached data).
	fn Function to apply to data after filtering by location but before returning from function.
	from_crs Coordinate reference system used to match the location CRS to the source data.
	crs Coordinate reference system to return.
	class Class of object to return.
	index A list of possible location, data, and (optionally) package values. List must be named and include a value named package and package must be NULL, to set package based on index. If list is not NULL and location and/or data as character or numeric values, the location and data are assumed to be index values for the index list. The index parameter supports nested lists created by make_location_data_list() (using only the default key names of "location" and "data"). This feature has not be fully tested and may result in errors or unexpected results.
	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"
	crop If TRUE, x is cropped to y using sf::st_crop().
	trim If TRUE, x is trimmed to y with st_trim().
	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.
	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.

var_names A named list following the format, list("New var name" = old_var_name), or a two column data frame with the first column being the new variable names and the second column being the old variable names; defaults to NULL.

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.

location sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()

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.

fileext File type or extension. Optional if filename or path include a file extension.

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

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.

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.

cache If TRUE, path is set to the package cache directory using get_data_dir(); defaults to FALSE.

pkg Package name passed to appname parameter of rappdirs::user_cache_dir()

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.

overwrite If TRUE, remove a file with the same name and path

compress Compression method to use: "none", "gz" ,"bz", or "xz".

version Serialization format version to be used. The default value is 2 as it's compatible for R versions prior to 3.5.0. See base::saveRDS() for more details.

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

Save data to file and invisibly return file path.

`format_address_data` *Format data frames and simple features with address data*

Description

getdata has two helpers for working with address data:

- `bind_address_col()` bind a provided value for city, county, and state to a data frame (to supplement address data with consistent values for these variables). This function is useful for converting partial street addresses with a consistent values for state, county, or city into full addresses
- `bind_block_col()` requires a data frame with columns named "bldg_num", "street_dir_prefix", "street_name", and "street_type" and binds derived values for whether a building is on the even or odd side of a block and create a block segment and a block face (including the even/odd identifier).

Usage

```
bind_block_col(
  x,
  bldg_num = "bldg_num",
  street_dir_prefix = "street_dir_prefix",
  street_name = "street_name",
  street_suffix = "street_type",
  replace_suffix = FALSE,
  street_col = NULL,
  block_col = NULL,
  .after = street_suffix,
  case = NULL
)

bind_address_col(x, ..., case = NULL, .cols = NULL, .after = NULL)

bind_location_text_col(
  x,
  text_col = "text",
  address_pattern = c("Ave.", "Avenue", "St.", "Street", "Rd.", "Road"),
 block_face_pattern = c("sides\\)", "side\\)", "[:space:]block", "-block", "blocks"),
  street_corridor_pattern = c("between(?=.+and)", "from(?=.+to)"),
  .cols = NULL
)
```

Arguments

x	A data.frame with a column name matching col and no column names matching the list passed to .cols (or the default values listed below).
bldg_num, street_dir_prefix, street_name, street_suffix	Column names to use for address information required to generate a block name and number.
replace_suffix	If TRUE, replace values in street_suffix column with abbreviations from street_suffixes.
street_col	String to use for street address column added based on component column values.
block_col	String to use as prefix for block identifier columns and separator between block number and street. Set to "block" when NULL (default). If length 2 (e.g. c("blk", "block")), the second value is used as the block separator and the first as the column identifier prefix.
.after	passed to dplyr::mutate() defaults to street_suffix for bind_block_col() and "address" for bind_address_col().
case	Case to use for text in new columns or in modified values. Options include "lower", "upper", "title", or "sentence". Defaults to NULL which leaves the case as is.
...	Additional parameters passed to dplyr::mutate() intended for use in filling missing values, e.g. state = "MD" to add a missing state column.
.cols	Column names to add. Defaults to is_address, is_block_face, is_street_corridor, and block_side. x must not have any column names matching the names found in .cols.
text_col	Column name containing the information to check for location details, Default: 'text'
address_pattern	A character vector of regex patterns to return TRUE for is_address.
block_face_pattern	A character vector of regex patterns to return TRUE for is_block_face.
street_corridor_pattern	A character vector of regex patterns to return TRUE for is_street_corridor.

Value

A data.frame with new indicator columns for address and block_face and a column indicating whether the text references a particular cardinal direction in describing a block.

Examples

```
address_df <- data.frame(
    "bldg_num" = c("100", "1415", "600"),
    "street_dir_prefix" = c(NA, NA, "N"),
    "street_name" = c("Holiday", "Key", "Charles"),
    "street_type" = c("Street", "Highway", "St")
  )
```

```
address_df <- bind_block_col(
    x = address_df,
    street_col = "street_address"
  )

address_df[1,]

address_df <- bind_address_col(
    address_df,
    city = "Baltimore",
    state = "MD"
  )

address_df[2,]

location_df <- data.frame(
    "text" = c(
      "100 Holiday St.",
      "1400 block Key Highway (north side)",
      "Charles St. from E. Centre St. to E. Madison St."
    )
)

location_df <- bind_location_text_col(location_df)

location_df
```

`format_data` *Format data frames and simple features using common approaches*

Description

This function can apply the following common data cleaning tasks:

- Applies stringr::str_squish and stringr::str_trim to all character columns
- Optionally replaces all character values of "" with NA values
- Optionally corrects UNIX formatted dates with 1970-01-01 origins
- Optionally renames variables by passing a named list of variables

The address functions previously included with `format_data()` are now documented at `format_address_data()`.

Usage

```
format_data(
  x,
  var_names = NULL,
  xwalk = NULL,
  clean_names = TRUE,
  .name_repair = "check_unique",
```

```
  replace_na_with = NULL,
  replace_with_na = NULL,
  replace_empty_char_with_na = FALSE,
  fix_date = FALSE,
  label = FALSE,
  remove_empty = NULL,
  remove_constant = FALSE,
  format_sf = FALSE,
  ...,
  call = caller_env()
)

rename_with_xwalk(
  x,
  xwalk = NULL,
  label = FALSE,
  .strict = TRUE,
  keep_all = TRUE,
  arg = caller_arg(x),
  call = caller_env()
)

label_with_xwalk(x, xwalk = NULL, label = "var", ...)

make_variable_dictionary(
  x,
  .labels = NULL,
  .definitions = NULL,
  details = c("basic", "none", "full")
)

fix_epoch_date(x, .cols = dplyr::contains("date"), tz = "")
```

Arguments

x	A tibble or data frame object
var_names	A named list following the format, `list("New var name" = old_var_name)`, or a two column data frame with the first column being the new variable names and the second column being the old variable names; defaults to NULL.
xwalk	A data frame with two columns using the first column as name and the second column as value; or a named list. The existing names of x must be the values and the new names must be the names.
clean_names	If TRUE, set .name_repair to `janitor::make_clean_names()`; defaults to TRUE.
.name_repair	Defaults to "check_unique"
replace_na_with	A named list to pass to `tidyr::replace_na()`; defaults to NULL.
replace_with_na	A named list to pass to `naniar::replace_with_na()`; defaults to NULL.

replace_empty_char_with_na
: If TRUE, replace "" with NA using naniar::replace_with_na_if(), Default: TRUE

fix_date
: If FALSE, fix UNIX epoch dates (common issue with dates from FeatureServer and MapServer sources) using the fix_epoch_date() function, Default: TRUE

label
: For label_with_xwalk() use label = "val" to use labelled::set_value_labels() or "var" (default) to use labelled::set_variable_labels(). For rename_with_xwalk(), if label is TRUE, xwalk is passed to label_with_xwalk() with label = "var" to label columns using the original names. Defaults to FALSE.

remove_empty
: If not NULL, pass values ("rows", "cols" or c("rows", "cols") (default)) to the which parameter of janitor::remove_empty()

remove_constant
: If TRUE, pass data to janitor::remove_constant() using default parameters.

format_sf
: If TRUE, pass x and additional parameters to format_sf_data().

...
: Additional parameters passed to format_sf_data()

.strict
: If TRUE (default), require that all values from the xwalk are found in the column names of the x data.frame. If FALSE, unmatched values from the xwalk are ignored.

keep_all
: If FALSE, columns that are not named in the xwalk are dropped. If TRUE (default), all columns are retained. If x is an sf object, the geometry column will not be dropped even it is not renamed.

arg, call
: Additional parameters used internally with cli::cli_abort() to improve error messages.

.labels
: Replaces labels column created by labelled::generate_dictionary() if column is all NA (no existing labels assigned); defaults to NULL.

.definitions
: Character vector of definitions appended to dictionary data frame. Must be in the same order as the variables in the provided data frame x.

details
: add details about each variable (full details could be time consuming for big data frames, FALSE is equivalent to "none" and TRUE to "full")

.cols
: tidyselect for columns to apply epoch date fixing function to. Defaults to dplyr::contains("date").

tz
: Time zone passed to as.POSIXct().

Value

The input data frame or simple feature object with formatting functions applied.

Examples

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

format_data(nc)
```

format_sf_data *Format simple feature data*

Description

The main format_sf_data function is a wrapper for the following common steps in transforming an sf object and preparing for mapping or analysis:

Usage

```
format_sf_data(
  x,
  crs = getOption("getdata.crs", default = 3857),
  erase_data = NULL,
  dTolerance = NULL,
  smooth = FALSE,
  sf_col = NULL,
  sf_req = TRUE,
  ...
)

erase_data(x, erase_data = NULL)
```

Arguments

x	A sf object or, if sf_req is FALSE, any object that can be converted to an sf object with sfext::as_sf.
crs	Coordinate reference system for returned data, Default: getOption("getdata.crs", default = 3857)
erase_data	A sf, sfc, or bbox object with geometry that should be erased from the data, Default: NULL
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.
smooth	If TRUE, smooth data with smoothr::smooth using default method and parameters, Default: FALSE.
sf_col	Name to use for output sf column, Default: 'geometry'.
sf_req	If TRUE, data must be a sf object. If FALSE, data is passed to sfext::as_sf to convert data to an sf object.
...	Additional parameters passed to format_data

Details

- Convert data to an sf object with sfext::as_sf if sf_req is FALSE
- Make data valid with sf::st_make_valid if needed
- Format data with format_data using the ... parameters
- Erase any data overlapping with erase_data (suggested for use with water or open space)
- Simplify geometry with sf::st_simplify if dTolerance is provided
- Smooth geometry with smoothr::smooth if smooth is TRUE
- Rename the sf column to match sf_col (defaults to "geometry")

The helper functions for format_sf_data and additional formatting functions for sf data are described in the details.

Helper functions for format_sf_data:

- erase_data: erase intersection of x and erase_data (validity of erase_data checked before sfext::st_erase and for x after completing the operation.
- rename_sf_col: Rename sf column.
- relocate_sf_col: Relocate sf column after selected columns (defaults to dplyr::everything()).

Value

A sf object with columns and geometry modified based parameters.

Examples

```
library(sf)

nc <- read_sf(system.file("shape/nc.shp", package = "sf"))
nc_county <- nc[2,]

# Transform coordinate reference system
st_crs(nc)$epsg
st_crs(format_sf_data(nc, crs = 3857))$epsg

# Simplify and smooth geometry
plot(nc_county, max.plot = 1)
nc_county_simple <- format_sf_data(nc_county, dTolerance = 5000, smooth = TRUE)
plot(nc_county_simple, max.plot = 1)

# Erase data
nc_co_water <- get_tigris_data(type = "area water", state = "NC", county = nc_county$NAME)
nc_county_erased <- format_sf_data(nc_county, erase_data = nc_co_water)
plot(nc_county_erased, max.plot = 1)

# If sf_req is set to FALSE, use any object that can be converted with sfext::as_sf
nc_bbox <- st_bbox(nc)
plot(format_sf_data(nc_bbox, erase_data = nc_county_simple, sf_req = FALSE))
```

get_airtable_data *Get data from an Airtable base and optionally convert to a sf object*

Description

[Experimental] Get data from an Airtable base using the Airtable API, a development version of the rairtable package, and the httr2 package. If the base includes coordinate fields/columns, optionally convert the data to a simple feature object using `sfext::df_to_sf()` if `geometry = TRUE`.

Usage

```
get_airtable_data(
  base,
  table = NULL,
  view = NULL,
  record = deprecated(),
  fields = NULL,
  filter = NULL,
  sort = NULL,
  direction = "asc",
  desc = deprecated(),
  max_records = 100,
  per_page = NULL,
  cell_format = "json",
  tz = NULL,
  locale = NULL,
  fields_by_id = FALSE,
  offset = NULL,
  geometry = FALSE,
  location = NULL,
  dist = getOption("getdata.dist"),
  diag_ratio = getOption("getdata.diag_ratio"),
  unit = getOption("getdata.unit", "meter"),
  asp = getOption("getdata.asp"),
  crs = getOption("getdata.crs", 3857),
  coords = getOption("getdata.coords", c("lon", "lat")),
  from_crs = getOption("getdata.from_crs", 4326),
  remove_coords = TRUE,
  address = getOption("getdata.address", "address"),
  geo = FALSE,
  name_repair = janitor::make_clean_names,
  token = NULL,
  type = "AIRTABLE_TOKEN",
  resp_type = deprecated(),
  ...
)
```

```
get_airtable_metadata(
  base,
  table = NULL,
  token = NULL,
  type = "AIRTABLE_TOKEN",
  resp_type = "tables",
  fields = FALSE
)
```

Arguments

base	Airtable base id starting with with "app". Optional if url or airtable are supplied. If base is an Airtable url, the table and view are replaced based on the values parsed from the url. Required.
table	Airtable table id or name. If table is a table ID it is a string starting with "viw". Optional only if base is a url.
view	Airtable view ID. View ID values starts with "viw". Optional if require_view is FALSE.
record	Airtable record identifier, Default: NULL Superseded by rairtable::list_records() function.
fields	For get_airtable_metadata(), if TRUE, return the fields column from the data.frame with the Airtable response. If only one table is provided, fields are returned as a data frame. Ignored if table is NULL
filter	Placeholder for filterByFormula API parameter allowing use of SQL style queries to filter data. Not yet implemented.
sort	Field names to sort by. Defaults to NULL.
direction	A string ("asc" for ascending (default) or "desc" for descending) or character vector matching length of sort parameter. Ignored if sort is NULL.
desc	Deprecated. Sort results in descending order. Replaced by direction parameter.
max_records	Maximum number of records to return. Must be 100 or less.
per_page	Passed to page_size parameter of rairtable::list_records()
cell_format	Cell format for "Link to another record" fields. Defaults to "json" which returns a unique record ID. A "string" cell_format returns the displayed character string.
tz, locale	Time zone and locale, Defaults to NULL. If cell_format is "string", tz defaults to Sys.timezone() and locale defaults to Sys.getlocale("LC_TIME").
fields_by_id	If TRUE, use fields IDs for column names in returned records. If FALSE (default), use field names.
offset	Offset parameter, Default: NULL
geometry	If TRUE, convert data into a simple feature object. Defaults to FALSE.
location	sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()
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"

`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.

`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").

`from_crs` For `df_to_sf()`, coordinate reference system used by coordinates or well known text in data frame.

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

`address` Address column name passed to `tidygeocoder::geocode()` or tidygeocoder::geo

`geo` If `TRUE`, use `address_to_sf()` to geocode address column; defaults to `FALSE`.

`name_repair` One of "unique" (default), "universal", "check_unique", "unique_quiet", or "universal_quiet" passed to `vctrs::vec_cbind()`. See vctrs::vec_as_names for the meaning of these options.

`token, type` API token and type, token defaults to `NULL` and type to `"AIRTABLE_TOKEN"` (same as `get_access_token(type = "AIRTABLE_TOKEN")`).

`resp_type` Response type to return, Reprecated. Previously, set resp_type to "resp" to return the API response without any additional formatting or conversion.

`...` Arguments passed on to `rairtable::list_records`

`airtable` An `airtable` class object. Optional for `read_airtable()` if url is supplied. For `list_records()` and `get_record()`, support the airtable, url, or a base *and* table parameter.

`airtable_id_col` Airtable record ID column name assigned to returned data frame. Defaults to `NULL` which is sets record ID column name to `getOption("rairtable.id_col", "airtable_record_id")`. For `list_records()` and `get_record()`, airtable_id_col is not used if metadata is `NULL` or does not include "id". The record ID column is dropped and converted to rownames if id_to_col is `FALSE`.

`model` Optional. A table model from `get_table_model()`. If supplied, model is used to validate fields and sort parameters and to arrange columns to match the order of the table model.

`.name_repair` One of "unique" (default), "universal", "check_unique", "unique_quiet", or "universal_quiet" passed to `vctrs::vec_cbind()`. See vctrs::vec_as_names for the meaning of these options.

`tz,locale` Time zone and locale, Defaults to `NULL`. If cell_format is "string", tz defaults to `Sys.timezone()` and locale defaults to `Sys.getlocale("LC_TIME")`.

`page_size` Maximum number of records to return per page.

`metadata` Record metadata columns to include with returned data frame. Options including "id", "createdTime", and "commentCount". Defaults to `c("id", "createdTime")`. If metadata is `NULL`, no additional fields are added to the returned data frame.

`simplifyVector` Passed to `httr2::req_body_json()`. If `FALSE`, `get_record()` and `list_records()` both return named lists of records.

Details

This function an Airtable personal access token which you can create at https://airtable.com/create/tokens and save to your local environment with `set_access_token(token = <YOUR_PERSONAL_ACCESS_TOKEN>`, The function previously required an Airtable API key which you can set using `set_access_token(token = <YOUR_API_KEY>`. However, Airtable is in the process of deprecating user API keys.

`get_airtable_data()` requires a scope that includes `data.records:read` and `get_airtable_metadata()` a scope including `schema.bases:read`.

As of May 2023, this function depends on the dev branch of my fork of the rairtable package. I expect this dependency to switch back to the rairtable package when the fork is merged.

Learn more about the Airtable API https://airtable.com/developers/web/api/introduction

---

get_elev_profile *Use* `elevatr::get_elev_point` *to get the elevation along a profile*

Description

[Experimental] `get_elev_profile()` is a wrapper for `elevatr::get_elev_point()` that takes a LINESTRING or POINT input for profile and returns a data frame of elevation. Optionally, create a series of points along a line with `sf::st_line_sample()` and/or include a column with the distance between each successive POINT in the data frame.

Usage

```
get_elev_profile(
  profile,
  units = NULL,
  dist = FALSE,
  n = NULL,
  density = NULL,
  type = "regular",
  sample = NULL,
  ...,
  drop_units = FALSE,
  cumulative = FALSE
)
```

Arguments

profile	A sfc or sf geometry with a LINESTRING or POINT geometry type.
units	If NULL, output elevation and distance is in meters. If a valid distance unit is supplied, elevation and distance are converted to match the supplied unit.
dist	If TRUE
n	integer; number of points to choose per geometry; if missing, n will be computed as round(density * st_length(geom)).
density	numeric; density (points per distance unit) of the sampling, possibly a vector of length equal to the number of features (otherwise recycled); density may be of class units.
type	character; indicate the sampling type, either "regular" or "random"
sample	numeric; a vector of numbers between 0 and 1 indicating the points to sample - if defined sample overrules n, density and type.
drop_units	If TRUE, return a plain numeric column for the elevation and distance columns. If FALSE (default) return a units class column.
cumulative	If TRUE, and dist is TRUE return distance as a cumulative sum.

Details

This function is proposed for addition to {elevatr}: https://github.com/elipousson/getdata/issues/4

---

get_esri_data — *Use esri2sf to get data from an ArcGIS FeatureServer or MapServer for a location*

Description

Wraps the esri2sf::esri2sf() and esri2sf::esri2df() functions to download an ArcGIS FeatureServer or MapServer. Supports spatial filtering with bounding box based on location and filtering by location name (if location name column is provided). As of fall 2022, this package suggests the elipousson/esri2sf fork using httr2.

Usage

```
get_esri_data(
  url,
  location = NULL,
  dist = getOption("getdata.dist"),
  diag_ratio = getOption("getdata.diag_ratio"),
  unit = getOption("getdata.unit"),
  asp = getOption("getdata.asp"),
  crs = getOption("getdata.crs", 3857),
  where = NULL,
```

```
  name = NULL,
  name_col = NULL,
  coords = NULL,
  from_crs = getOption("getdata.crs", 4326),
  clean_names = TRUE,
  token = NULL,
  progress = TRUE,
  quiet = FALSE,
  .name_repair = janitor::make_clean_names,
  ...
)

get_esri_layers(
  location = NULL,
  layers = NULL,
  url = NULL,
  nm = NULL,
  token = NULL,
  clean_names = TRUE,
  quiet = FALSE,
  .name_repair = janitor::make_clean_names,
  ...,
  call = caller_env()
)

get_esri_metadata(
  url,
  token = NULL,
  meta = NULL,
  clean_names = TRUE,
  .name_repair = janitor::make_clean_names,
  call = caller_env()
)
```

Arguments

url	FeatureServer or MapServer url to retrieve data from. Passed to `url` parameter of `esri2sf::esri2sf()` or `esri2sf::esri2df()` functions. For `get_esri_layers()`, the optional url must be a service url which is the base url for one or more layer urls.
location	`sf`, `sfc`, or bbox object (or other object convertible with `as_bbox()`. Optional.
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 "kilome-

	ter") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
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.
crs	Cordinate reference system to return, Default: 4326 for sf_to_df() and NULL for df_to_sf().
where	where query string passed to esri2sf, Default: NULL
name, name_col	Name value and name column found in the ArcGIS FeatureServer or MapServer data.
coords	Coordinate columns for input data.frame or output sf object (if geometry is 'centroid' or 'point') Default: c("lon", "lat").
from_crs	For df_to_sf(), coordinate reference system used by coordinates or well known text in data frame.
clean_names	If TRUE, set .name_repair to janitor::make_clean_names() Ignored when get_esri_metadata() is not returning a data.frame, e.g. meta = "id".
token	string for authentication token. defaults to NULL.
progress	Show progress bar from cli::cli_progress_along() if TRUE. Default FALSE.
quiet	If TRUE, use suppressMessages() to prevent the printing of messages about the requested layer. Defaults to FALSE.
.name_repair	Defaults to "check_unique"
...	Arguments passed on to esri2sf::esri2sf outFields vector of fields you want to include. default is NULL for all fields. replaceDomainInfo If TRUE, add domain information to the return data frame. Default FALSE.
layers	Either a vector with URLs, a named list of urls, or a numeric vector; defaults to NULL. Optional if url is a
nm	Name or vector of names to add to the layers; defaults to NULL.
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.
meta	Name of metadata list value to return from esri2sf::esrimeta, e.g. "name" to return layer name. Defaults to NULL.

`get_flickr_photos` *Use FlickrAPI to get geotagged photos for a location*

Description

get_flickr_photos() uses FlickrAPI::get_photo_search() to get a data frame or sf objects with photos from a specified location or matching other photo search parameters. Set API key using FlickrAPI::set_flickr_api_key() or pass to the api_key parameter.

Usage

```
get_flickr_photos(
  location = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  user_id = NULL,
  tags = NULL,
  license_id = "cc0",
  sort = "date-posted",
  desc = FALSE,
  img_size = "s",
  extras = c("description", "date_taken", "tags", "geo"),
  per_page = 100,
  page = 1,
  orientation = NULL,
  geometry = TRUE,
  crs = 4326,
  key = NULL
)
```

Arguments

location	A sf or bbox object to use in creating bounding box for getting photos from Flickr. Optional.
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"
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.
user_id	The NSID of the user with photos to search. If this parameter is NULL passed then all public photos will be searched.
tags	A vector of tags to search for.
license_id	The license id for photos. For possible values, see the Flickr API method flickr.photos.licenses.getInfo or see details for more information. If license_id is provided, "license" is added to extras.
sort	Supported options include "date-posted","date-taken", "interestingness", or "relevance"
desc	If TRUE return images in descending sort order, if FALSE, return in ascending sort order. Ignored if sort is set to "relevance".

img_size	Image size; defaults to "s" (small). Options ranging from smallest to largest size include "sq" (square), "t", "s", "q", "m", "n", "z", "c", "l", and "o" (original).
extras	Defaults to "description", "date_taken", "tags", and "geo".
per_page	Photos to return per page of search results, Default: 100. Maximum 250 if a location is provided or 500 otherwise.
page	Page to return. If page is greater than length 1, loop over all pages. This may cause issues with API access if a large page range is provided. Default: 1
orientation	If img_size is length 1, photos are filtered to one or more of the supported orientations ("portrait", "landscape", and "square"); defaults to NULL.
geometry	If TRUE, include "geo" in extras and convert photos data frame to sf object. Passed to geo parameter of FlickrAPI::get_photo_search()
crs	Coordinate reference system of sf object to return if geometry is TRUE.
key	Flickr API key. If api_key is NULL, the FlickrAPI::getPhotoSearch uses FlickrAPI::getFlickrAPIKey() to use the environment variable "FLICKR_API_KEY" as the key. Use set_access_token() w/ type = "FLICKR_API_KEY" or FlickrAPI::setFlickrAPIKey()

Details

License id options:

license_id can be an integer from 0 to 10 or a corresponding license code including:

- "c" (All Rights Reserved),
- "by-bc-sa" (Attribution-NonCommercial-ShareAlike),
- "by-nc" (Attribution-NonCommercial),
- "by-nc-nd" (Attribution-NonCommercial-NoDerivs),
- "by" (Attribution),
- "by-sa" (Attribution-ShareAlike),
- "by-nd" (Attribution-NoDerivs),
- "nkc" (No known copyright restrictions),
- "pd-us" (United States Government Work),
- "cc0" (Public Domain Dedication),
- or "pd" (Public Domain Mark).

Value

A data frame with photo information or sf object with geometry based on latitude and longitude of geocoded photos.

See Also

FlickrAPI::getPhotoSearch()

get_gsheet_data *Use googlesheets4 to get a data frame or simple feature data from a Google Sheet*

Description

Use googlesheets4 to get a data frame or simple feature data from a Google Sheet

Usage

```
get_gsheet_data(
  url,
  sheet = NULL,
  ss = NULL,
  ask = FALSE,
  geometry = FALSE,
  location = NULL,
  dist = getOption("getdata.dist"),
  diag_ratio = getOption("getdata.diag_ratio"),
  unit = getOption("getdata.unit", "meter"),
  asp = getOption("getdata.asp"),
  coords = getOption("getdata.coords", c("lon", "lat")),
  remove_coords = TRUE,
  address = getOption("getdata.address", "address"),
  geo = FALSE,
  from_crs = 4326,
  clean_names = TRUE,
  ...
)
```

Arguments

url	A Google Sheets url
sheet	Sheet to read, in the sense of "worksheet" or "tab". You can identify a sheet by name, with a string, or by position, with a number. Ignored if the sheet is specified via range. If neither argument specifies the sheet, defaults to the first visible sheet.
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 sfext::read_sf_gsheet.
geometry	Type of geometry to include in data frame. options include "drop", "wkt", "centroid", "point", Default: 'centroid'.
location	sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()
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"
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.
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.
address	Address column name passed to tidygeocoder::geocode() or tidygeocoder::geo
geo	If TRUE, use address_to_sf() to geocode address column; defaults to FALSE.
from_crs	For df_to_sf(), coordinate reference system used by coordinates or well known text in data frame.
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.
...	Other parameters passed onto methods.

`get_location` *Get location of a specified type based on name, id, or location*

Description

Filter by name or id or use a spatial filter based on an sf object or geocoded street address. Optionally you can use an index list to match the type to a named list of URLs or sf objects.

Usage

```
get_location(
  type,
  name = NULL,
  name_col = "name",
  id = NULL,
```

```
  id_col = "id",
  location = NULL,
  index = NULL,
  union = FALSE,
  crs = getOption("getdata.crs", 3857),
  label = NULL,
  class = "sf",
  ...
)
```

Arguments

type	Type of location to return. Type can be an sf object, e.g. a data frame with multiple neighborhoods or a character string that can be passed to `get_location_data()`. If index is provided, character can also be a character string to match the name of a list.
name	Location name to return.
name_col	Column name in type with name values, Default: 'name' Required if name provided.
id	Location id to return. id is coerced to character or numeric to match the class of the id_col for type.
id_col	Column name in type with id values, Default: 'id'. Required if id is provided.
location	An address, bounding box (bbox), or simple feature (sf) object passed to `sf::st_filter()`. Any valid address or addresses are geocoded with `tidygeocoder::geo()`, converted to a simple feature object, and then used as a spatial filter. bbox objects are converted using `sfext::sf_bbox_to_sf()`. Multiple addresses are supported.
index	Optional list used to match type to data, Default: NULL
union	If TRUE, the location geometry is unioned with `sf::st_union()` and the names are combined into a single value. Default: FALSE.
crs	Coordinate reference system to return; defaults to NULL which returns data using the same coordinate reference system as the provided type of location.
label	Label optionally added to "label" column; must be a length 1 or match the number of rows returned based on the other parameters. If union = TRUE, using label is recommended. Default: NULL
class	Class of object to return; defaults to "sf".
...	Additional parameters passed to `get_location_data()` if type is character and index is NULL.

Value

A simple feature object from data provided to type.

Examples

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

# get_location works with a type sf object and name and id values
get_location(type = nc, name = "Warren", name_col = "NAME")
get_location(type = nc, id = 37185, id_col = "FIPSNO")
# if name is named, the name of name is used as name_col
get_location(type = nc, name = c("NAME" = "Warren"))

# type can also be a file path
get_location(
  type = system.file("shape/nc.shp", package = "sf"),
  name = "Hertford",
  name_col = "NAME"
)

# type can also be an index name (if a named list of data sets, url values, or
# path values is passed to index)
get_location(
  type = "smaller",
  name = "Hertford",
  name_col = "NAME",
  index = list(
    "smaller" = dplyr::filter(nc, AREA <= 0.10),
    "larger" = dplyr::filter(nc, AREA > 0.15)
  )
)
```

---

get_location_data *Get data for a location*

Description

Returns data for a selected location or a list of locations (for `map_location_data()`). If data is a character string, the parameter is passed to `sfext::read_sf_url()`, `sfext::read_sf_path()`, or `sfext::read_sf_pkg()`. This function uses `sfext::st_filter_ext()` to filter, crop, or trim data to the provided location. location can also be an an address.

Usage

```
get_location_data(
  location = NULL,
  dist = getOption("getdata.dist"),
  diag_ratio = getOption("getdata.diag_ratio"),
  unit = getOption("getdata.unit", default = "meter"),
  asp = getOption("getdata.asp"),
  data = NULL,
  pkg = getOption("getdata.package"),
  package = getOption("getdata.package"),
```

```
  fileext = getOption("getdata.fileext", default = "gpkg"),
  filetype = getOption("getdata.filetype", default = "gpkg"),
  fn = NULL,
  crop = TRUE,
  trim = FALSE,
  from_crs = getOption("getdata.from_crs"),
  crs = getOption("getdata.crs", 3857),
  class = "sf",
  label = NULL,
  index = NULL,
  col = NULL,
  var_names = NULL,
  clean_names = FALSE,
  range = NULL,
  .name_repair = "check_unique",
  ...,
  call = caller_env()
)

map_location_data(
  location = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  data = NULL,
  package = NULL,
  fileext = "gpkg",
  filetype = "gpkg",
  fn = NULL,
  crop = TRUE,
  trim = FALSE,
  from_crs = NULL,
  crs = NULL,
  class = "list",
  label = NULL,
  load = FALSE,
  index = NULL,
  range = NULL,
  ...
)
```

Arguments

location	sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()
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"

`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.

`data` Character string (e.g. url, file path, or name of data from package) for a spatial data or a `sf`, `sfc`, or `bbox` object with geometry overlapping the location. If data is `NULL`, all unnamed parameters are passed to `sfext::read_sf_ext()` with a bbox based on location. If data is not `NULL` and not a data.frame, url, file path, or bbox, conversion to a sf object will still always be attempted with `sfext::as_sf()`.

`pkg, package` Name of the package to search for data.

`fileext, filetype` File extension or type to use if passing parameters to `sfext::read_sf_download()` or `sfext::read_sf_pkg()` (required for extdata and cached data).

`fn` Function to apply to data after filtering by location but before returning from function.

`crop` If `TRUE`, x is cropped to y using `sf::st_crop()`.

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

`from_crs` Coordinate reference system used to match the location CRS to the source data.

`crs` Coordinate reference system to return.

`class` Class of object to return.

`label` label is optionally used by `map_location_data()` to name the data objects in the list returned by the function.

`index` A list of possible location, data, and (optionally) package values. List must be named and include a value named package and package must be `NULL`, to set package based on index. If list is not `NULL` and location and/or data as character or numeric values, the location and data are assumed to be index values for the index list. The index parameter supports nested lists created by `make_location_data_list()` (using only the default key names of "location" and "data"). This feature has not be fully tested and may result in errors or unexpected results.

`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.

`var_names` A named list following the format, `list("New var name" = old_var_name)`, or a two column data frame with the first column being the new variable names and the second column being the old variable names; defaults to `NULL`.

`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.

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.
.name_repair	One of "unique", "universal", or "check_unique". See vctrs::vec_as_names() for the meaning of these options.
...	Additional parameters passed to sfext::read_sf_path(), sfext::read_sf_url(), sfext::read_sf_pkg(), sfext::as_sf() (with bbox), or sfext::read_sf_ext() (with no other parameters).
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.
load	If TRUE and class is "list", load data to local environment; defaults FALSE.

Details

This function previously supported county geoid, state name, abbreviation, or geoid as a location. Currently, recommend using get_tigris_data() and passing a sf object to location.

Working with sf lists for data and locations:

map_location_data() makes it easier to work with sf lists. It supports data as a character vector, data as an sf list when location is a single object, location as a character vector or sf list (including lists of bbox or sfc objects), or when both data and location are lists (such as a list created by make_location_data_list()).

---

get_open_data *Get data from an open data portal (Socrata) for a location*

Description

get_socrata_data is get_open_data with source_type set to "socrata" (the only currently supported option). get_open_data can return a selected dataset using Socrata Query Language (SoQL) parameters as a tibble or sf object. Details on SoQL queries are found in the Socrata API documentation https://dev.socrata.com/docs/queries/.

Usage

```
get_open_data(
  data = NULL,
  source_url = NULL,
  source_type = "socrata",
  select = NULL,
  where = NULL,
```

```
  query = NULL,
  location = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  name_col = NULL,
  name = NULL,
  location_col = NULL,
  coords = c("longitude", "latitude"),
  geometry = FALSE,
  token = NULL,
  type = NULL,
  from_crs = 4326,
  crs = NULL,
  clean_names = TRUE,
  quiet = FALSE,
  .name_repair = janitor::make_clean_names
)

get_socrata_data(
  data = NULL,
  source_url = NULL,
  select = NULL,
  where = NULL,
  query = NULL,
  location = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  name_col = NULL,
  name = NULL,
  location_col = NULL,
  coords = c("longitude", "latitude"),
  geometry = FALSE,
  token = NULL,
  type = NULL,
  from_crs = 4326,
  crs = NULL,
  clean_names = TRUE
)

get_socrata_metadata(source_url = NULL, data = NULL)

list_socrata_data(source_url)
```

Arguments

data	A data set identifier (known as a resource for Socrata) or a url for an individual dataset. If data is set to "list" and a valid source_url is provided, the function returns a list of all available resources. If data is a url, source_url must be NULL. get_socrata_metadata requires the data parameter.
source_url	A data source url. For Socrata, this should the base url for the open data portal.
source_type	Data source type; defaults to "socrata" which is currently the only supported option.
select	Names of of columns to return or transformed, equivalent to a SELECT in SQL. Passed to SODA $select parameter, see https://dev.socrata.com/docs/queries/select.html for more information.
where	Condition to filters the rows to return, equivalent to WHERE in SQL. Passed to the SODA $where parameter, see https://dev.socrata.com/docs/queries/where.html for more information.
query	A full SoQL query string, all as one parameter. Passed to the SODA $query parameter, see https://dev.socrata.com/docs/queries/query.html for more information.
location	sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()
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"
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.
name, name_col	Name of column in Socrata data resource with location names (e.g. County) and name of location to return.
location_col	Name of a "location" or "point" type column in a Socrata dataset.
coords	Coordinate columns for input data.frame or output sf object (if geometry is 'centroid' or 'point') Default: c("lon", "lat").
geometry	If TRUE and coords are provided, return a sf object. Default FALSE.
token, type	Access token or API Key and token type (name used to store token in .Renvironment). A token may be required to access data from Socrata and other open data portals but can be stored as an environment variable with set_access_token.
from_crs	Coordinate reference system used to match the location CRS to the source data.
crs	Coordinate reference system of bounding box to return; defaults to NULL which maintains the crs of the input object.
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.

quiet	If TRUE, suppress messages when downloading data. Defaults to FALSE.
.name_repair	One of "unique", "universal", or "check_unique". See vctrs::vec_as_names() for the meaning of these options.

Examples

```
## Get Q2 2020 vehicle crash data for Cecil County, Maryland
## Not run:
  get_open_data(
    source_url = "https://opendata.maryland.gov",
    data = "65du-s3qu",
    where = "(year = '2020') AND (quarter = 'Q2')",
    name_col = "county_desc",
    name = "Cecil",
    token = Sys.getenv("MARYLAND_OPEN_DATA_API_KEY")
  )

## End(Not run)
```

---

get_osm_data *Use osmdata to get Open Street Map data for a location*

Description

Use osmdata functions to query the overpass API and access OSM data by adjusted bounding box or by enclosing ways/relations around the center of a location. For more information on key and value options, refer to the osm_common_tags reference table or the OSM Wiki https://wiki.openstreetmap.org/wiki/Map_features. Use the osmdata package directly for more detailed control over queries https://docs.ropensci.org/osmdata/

Usage

```
get_osm_data(
  location = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  key,
  value = NULL,
  features = NULL,
  id = NULL,
  type = NULL,
  crs = NULL,
  geometry = NULL,
  osmdata = FALSE,
  enclosing = NULL,
  nodes_only = FALSE,
```

```
  key_exact = TRUE,
  value_exact = TRUE,
  match_case = TRUE
)

get_osm_id(id, type = NULL, crs = NULL, geometry = NULL, osmdata = FALSE)

get_osm_boundaries(
  location,
  level = NULL,
  lang = "en",
  crs = NULL,
  enclosing = "relation",
  geometry = NULL,
  osmdata = FALSE
)
```

Arguments

location	A sf, sfc, or bbox object converted to bounding box with sfext::st_bbox_ext() or a character object passed directly to the bbox parameter of osmdata::opq().
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"
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.
key	Feature key for overpass API query.
value	Value of the feature key; can be negated with an initial exclamation mark, value = "!this", and can also be a vector, value = c("this", "that"). If value = "all" or if key = "building" the values passed to the osmdata package are from a preset list extracted from osmdata::available_tags().
features	A named list with the format list("<key>" = "<value>") or a character vector of key-value pairs with keys and values enclosed in escape-formatted quotations (see osmdata::add_osm_features()) for examples of the latter option.
id	OpenStreetMap feature id with or without a type id prefix. If multiple id values are provided, they must use a single consistent value for geometry.
type	Type of feature for the id; "node", "way", or "relation". Optional if id includes a type prefix.
crs	Coordinate reference system for output data; if NULL, the data remains in the Open Street Map coordinate reference system 4326. Default: NULL.

geometry	Geometry type to output ("polygons", "points", "lines", "multilines", or "multipolygons"); if multiple geometry types are needed set osmdata to TRUE. Default NULL.
osmdata	If TRUE return a osmdata class object that includes the overpass API call, metadata including timestamp and version numbers, and all available geometry types; defaults to FALSE.
enclosing	If enclosing is "relation" or "way", this function uses the osmdata::opq_enclosing() to query the OSM data (instead of osmdata::add_osm_feature(). Defaults to NULL. When the enclosing parameter is provided, the dist, diag_ratio, asp, and unit parameters are ignored and the center of the provided location is used for the query. geometry is set automatically based enclosing with "relation" using "multipolygons" and "way" using "polygons" geometry.
nodes_only	WARNING: this parameter is equivalent to osm_types = "node" and will be DEPRECATED. If TRUE, query OSM nodes only. Some OSM structures such as place = "city" or highway = "traffic_signals" are represented by nodes only. Queries are built by default to return all nodes, ways, and relation, but this can be very inefficient for node-only queries. Setting this value to TRUE for such cases makes queries more efficient, with data returned in the osm_points list item.
key_exact	If FALSE, key is not interpreted exactly; see https://wiki.openstreetmap.org/wiki/Overpass_API
value_exact	If FALSE, value is not interpreted exactly
match_case	If FALSE, matching for both key and value is not sensitive to case
level	Numeric administrative level (admin_level) of boundary to return; defaults to NULL. If multiple levels are provided, the any admin levels between the min and max values of level is returned. See https://wiki.openstreetmap.org/wiki/Key:admin_level for more information. Only used for get_osm_boundaries().
lang	Language for boundary names to include in resulting data frame (e.g. "en" for English or "es" for Spanish). Default language names should always be included in results. Defaults to "en". See https://wiki.openstreetmap.org/wiki/Multilingual_names for more information.

Value

A simple feature object with features using selected geometry type or an osmdata object with features from all geometry types.

Examples

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

civic_buildings <- get_osm_data(
  location = nc[37,],
  features = c("building" = "civic"),
  geometry = "polygons"
)

civic_buildings
```

get_static_map *Use mapboxapi or bingmapr to get a static map image*

Description

Get a static map image using the Mapbox Static Maps API using mapboxapi::static_mapbox or the Bing Maps Static Map API using bingmapr::get_map_image. An API key or access token is required for both services. Set the bingmap API token using bingmapr::bing_maps_api_key and the Mapbox token with mapboxapi::mb_access_token or use `set_access_token` with `type = "BING_MAPS_API_KEY"` or `type = "MAPBOX_PUBLIC_TOKEN"`.

Usage

```
get_static_mapbox(
  location,
  dist = NULL,
  unit = "meter",
  style_url = "mapbox://styles/mapbox/light-v10",
  overlay_location = FALSE,
  overlay_sf = NULL,
  overlay_style = NULL,
  zoom = NULL,
  width = 600,
  height = 400,
  bearing = NULL,
  pitch = NULL,
  token = NULL,
  ...
)

get_osm_static_mapbox(
  id = NULL,
  key = NULL,
  level = NULL,
  location = NULL,
  dist = NULL,
  unit = "meter",
  overlay_location = TRUE,
  style_url = "mapbox://styles/mapbox/light-v10",
  overlay_sf = NULL,
  overlay_style = NULL,
  zoom = NULL,
  width = 600,
  height = 400,
  bearing = NULL,
  pitch = NULL,
  token = NULL,
```

```
  ...
)

get_location_static_mapbox(
  type,
  dist = NULL,
  unit = "meter",
  name = NULL,
  name_col = "name",
  id = NULL,
  id_col = "id",
  location = NULL,
  index = NULL,
  union = FALSE,
  overlay_location = TRUE,
  style_url = "mapbox://styles/mapbox/light-v10",
  overlay_sf = NULL,
  overlay_style = NULL,
  zoom = NULL,
  width = 600,
  height = 400,
  bearing = NULL,
  pitch = NULL,
  token = NULL,
  ...
)

get_static_bingmap(
  location = NULL,
  dist = NULL,
  unit = "m",
  imagery = "BirdsEye",
  zoom = NULL,
  width = 600,
  height = 400,
  bearing = NULL,
  token = NULL,
  ...
)
```

Arguments

`location`	An input location for which you would like to request tiles. Can be a length-4 vector representing a bounding box, or an `sf` object. If an input `sf` object is supplied, use the `buffer_dist` argument to control how much area you want to capture around the layer. While the input `sf` object can be in an arbitrary coordinate reference system, if a length-4 bounding box vector is supplied instead it must represent WGS84 longitude/latitude coordinates and be in the order `c(xmin, ymin, xmax, ymax)`.

dist	Buffer distance passed to buffer_dist parameter of mapboxapi::static_mapbox() or to sfext::st_buffer_ext() for get_static_bingmap().
unit	Unit of dist argument. Defaults to "meters".
style_url	Style URL; defaults to "mapbox://styles/mapbox/light-v10"
overlay_location	If TRUE, use the location (or OpenStreetMap feature) as the overlay_sf parameter. Default to FALSE. Ignored if overlay_sf is provided.
overlay_sf	The overlay sf object (optional). The function will convert the sf object to GeoJSON then plot over the basemap style. Spatial data that are too large will trigger an error, and should be added to the style in Mapbox Studio instead.
overlay_style	A named list of vectors specifying how to style the sf overlay. Possible names are "stroke", "stroke-width" (or "stroke_width"), "stroke-opacity" (or "stroke_opacity"), "fill", and "fill-opacity" (or "fill_opacity"). The fill and stroke color values can be specified as six-digit hex codes or color names, and the opacity and width values should be supplied as floating-point numbers. If overlay_style is NULL, the style values can be pulled from columns with the same names in overlay_sf.
zoom	The map zoom. The map will infer this from the overlay unless longitude, latitude, and zoom are all specified.
width, height	Map width and height; defaults to 600 px width and 400 px height.
pitch, bearing	The map pitch and bearing; defaults to NULL. pitch can range from 0 to 60, and bearing from -360 to 360.
token	Optional token or API key. Recommend setting the Bing Maps API key using bingmapr::bing_maps_api_key() and the Mapbox access token with mapboxapi::mb_access_token(
...	Additional parameters passed to get_location_data() if type is character and index is NULL.
id	OpenStreetMap feature id with or without a type id prefix. If multiple id values are provided, they must use a single consistent value for geometry.
key	Feature key for overpass API query.
level	Numeric administrative level (admin_level) of boundary to return; defaults to NULL. If multiple levels are provided, the any admin levels between the min and max values of level is returned. See https://wiki.openstreetmap.org/wiki/Key:admin_level for more information. Only used for get_osm_boundaries().
type	For get_osm_static_mapbox(), type of feature with id; ("node", "way", or "relation"); for get_location_static_mapbox(), type of location (see get_location() for details.
name	Location name to return.
name_col	Column name in type with name values, Default: 'name' Required if name provided.
id_col	Column name in type with id values, Default: 'id'. Required if id is provided.
index	Optional list used to match type to data, Default: NULL
union	If TRUE, the location geometry is unioned with sf::st_union() and the names are combined into a single value. Default: FALSE.
imagery	String with imagery type, Default: 'BirdsEye' Supported values include:

- Aerial: Aerial imagery.
- AerialWithLabels: Aerial imagery with a road overlay.
- AerialWithLabelsOnDemand: Aerial imagery with on-demand road overlay.
- Streetside: Street-level imagery.
- BirdsEye: Birds Eye (oblique-angle) imagery.
- BirdsEyeWithLabels: Birds Eye (oblique-angle) imagery with a road overlay.
- Road: Roads without additional imagery.
- CanvasDark: A dark version of the road maps.
- CanvasLight: A lighter version of the road maps which also has some of the details such as hill shading disabled.
- CanvasGray: A grayscale version of the road maps

Details

Variations on get_static_mapbox include

- get_location_static_mapbox wrapping `get_location()`
- get_osm_static_mapbox wrapping `get_osm_data()`, `get_osm_id()`, and `get_osm_boundaries()`

In those cases, the ... parameters are passed on the getdata functions rather than the static map function.

For `get_static_bingmap()`, parameter names are modified from `bingmapr::get_map_image()` for consistency, so the bearing parameter passed to orientation and token is passed to key.

Examples

```
## Not run:
  get_osm_static_mapbox(
    id = "way/49664223",
    dist = 0.5,
    unit = "mi",
    overlay_style = list(
      stroke = "darkgreen",
      fill = "green",
      fill_opacity = 0.25
    )
  )

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

  get_location_static_mapbox(
    type = nc,
    name = "Ashe",
    name_col = "NAME",
    dist = 50,
    unit = "mi"
  )
```

```
## End(Not run)
```

`get_tigris_data` *Use tigris to get state-level data from the U.S. Census Bureau*

Description

Use the tigris package to download state-level data from the U.S. Census Bureau API and optionally filter by name or GeoID.

Usage

```
get_tigris_data(
  type = NULL,
  state = getOption("getdata.state"),
  location = NULL,
  dist = getOption("getdata.dist"),
  diag_ratio = getOption("getdata.diag_ratio"),
  unit = getOption("getdata.unit", "meter"),
  asp = getOption("getdata.asp"),
  crs = getOption("getdata.crs", default = 3857),
  name = NULL,
  name_col = c("name", "namelsad", "geoid"),
  cb = TRUE,
  clean_names = TRUE,
  cache = TRUE,
  ...
)
```

Arguments

`type`	Type of data to return, Default: `NULL`; See details for supported options.
`state`	State name, abbreviation, or GeoID. Required. Defaults to getOption("getdata.state").
`location`	A sf, sfc, or bbox object passed to `sfext::st_bbox_ext()` and used to create the geometry passed to the `filter_by` parameter passed to the tigris functions.
`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"
`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.

crs	Coordinate reference system of bounding box to return; defaults to NULL which maintains the crs of the input object.
name, name_col	Name and columns to filter by name. name defaults to NULL, and name_col defaults to c("namelsad", "namelsad", "geoid") columns.
cb	If TRUE, download a generalized (1:500k) file. If FALSE, download the most detailed TIGER/Line file. Defaults to TRUE (reverse of the default for tigris functions). This parameter is *not* used when type is set to blocks, roads, primary secondary roads, area water, linear water, landmarks, or zctas.
clean_names	If TRUE, set .name_repair to janitor::make_clean_names(); defaults to TRUE.
cache	If TRUE, set options(tigris_use_cache = TRUE) to cache downloaded tigris data. Ignored if getOption("tigris_use_cache") is not NULL.
...	Additional parameters passed on to tigris functions.

Details

Supported data types:

Different type values corresponded to different tigris functions for downloading from the U.S. Census Bureau API include. Supported options include: "counties", "census places", "congressional districts", "legislative districts", "senate district", "tracts", "block groups", "blocks", "pumas", "voting districts", "zctas", "roads", "primary secondary roads", "area water", "linear water", and "landmarks".

tigris functions that do not use a "state" parameter (e.g. tigris::coastline or tigris::rails) are not supported by this function. Note that the default value of the cb parameter for get_tigris_data is TRUE and the default value of for the original tigris package is FALSE.

Value

A simple feature object matching the type provided.

---

get_wiki_data *Get Wikipedia articles for a location*

Description

Use the Wikipedia API geosearch API to get Wikipedia articles for a location. See https://www.mediawiki.org/wiki/Extension:GeoData for more information. Only returns Wikipedia articles with coordinates.

Usage

```
get_wiki_data(
  location,
  radius = FALSE,
  primary = NULL,
  details = NULL,
```

```
  limit = 50,
  list = "geosearch",
  lang = getOption("getdata.lang", default = "en"),
  geometry = TRUE,
  dist = getOption("getdata.dist"),
  diag_ratio = getOption("getdata.diag_ratio"),
  unit = getOption("getdata.unit", "meter"),
  asp = getOption("getdata.asp"),
  crs = getOption("getdata.unit", 3857),
  remove_coords = TRUE,
  clean_names = TRUE
)
```

Arguments

location	sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()
radius	If TRUE, use dist as a buffer around the center of the location; defaults to FALSE
primary	If NULL, search for primary coordinates. Set primary to "all" or "secondary" to search other coordinate types.
details	Additional detailed to return with results. Options include "type", "name", "country", "region"; defaults to NULL.
limit	Number of pages to return (max 500); defaults to 50
list	method to use for query; "geosearch" returns data, "resp" returns response
lang	Language to search on Wikipedia; defaults to "en".
geometry	If TRUE, return sf object. If FALSE, return data frame. Defaults to FALSE.
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"
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.
crs	Coordinate reference system of bounding box to return; defaults to NULL which maintains the crs of the input object.
remove_coords	For df_to_sf(), if TRUE, remove the coordinate columns after converting a data frame to simple feature object; defaults to FALSE.
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.

Details

For this function, `location` can be either an `sf`, `sfc`, or `bbox` object or the title of a Wikipedia article with a related location.

See Also

- `geonames::GNfindNearbyWikipedia()`, `geonames::GNwikipediaBoundingBox()`

`make_location_data_list`
Make a list of data and corresponding locations

Description

This function converts data and location into lists of sf objects using as_sf_list. If location_col, data_col, or col (which sets both to the same value), are provided the col is passed to as_sf_list to allow the creation of an sf list from a sf data frame using `dplyr::group_nest()`.

Usage

```
make_location_data_list(data, location, key = c("location", "data"), ...)
```

Arguments

`data`, `location`	A sf object or list of sf objects with data and corresponding locations.
`key`	Names for location and data in the returned list.
`...`	Pass location_col and/or data_col to group and nest the data provided to location and data. Use col to set both to the same value.

Details

If location and data are the same length are the same length, they are combined into a single list. If either one is length 1 when the other is not, the length 1 object is repeated to match the length of the longer object. Different length objects where neither are length 1 gives a warning.

make_location_grid *Make a grid over the bounding box of a location*

Description

If location is a single feature sf object, the original columns of the object are included in the output grid. If location has mutiple features, the values of name_col are combined with sfext::st_union_ext and other columns are dropped. The input sf object should not have columns named id, rows, or cols.

Usage

```
make_location_grid(location, name_col = "name", unit = NULL, ...)
```

Arguments

location	A sf, sfc, or bbox object passed to sfext::st_make_grid_ext
name_col	Column name to collapse into new name_col value, Default: 'name'
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"
...	Arguments passed on to sfext::st_make_grid_ext
	x A sf, sfc, or bbox object, Default: NULL. Required.
	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().

---

make_xwalk_list *Make a crosswalk list for use with label_with_xwalk() or rename_with_xwalk()*

Description

Make a crosswalk list for use with label_with_xwalk() or rename_with_xwalk()

Usage

```
make_xwalk_list(xwalk, cols = c("label", "name"), call = caller_env())
```

Arguments

xwalk	A data frame with two columns or a named list.
cols	Column names to use for crosswalk.
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 named list

---

osm_building_tags *OpenStreetMap building tags*

Description

Used by get_osm_data if key = "building".

Usage

```
osm_building_tags
```

Format

A character vector with length of 84

Details

More information on the building key https://wiki.openstreetmap.org/wiki/Key:building

osm_common_tags *Common OpenStreetMap tags*

Description

A subset of tags scraped from the OpenStreetMap Wiki page on Map features: https://wiki.openstreetmap.org/wiki/Map_features. Only those tags with a url are included in this reference table.

Usage

```
osm_common_tags
```

Format

A data frame with 272 rows and 5 variables:

`key` Key

`value` Value

`description` Description of tag/usage

`category` Category

`url` OSM Wiki url

replace_with_xwalk *Replace values in a character vector or data frame with a crosswalk*

Description

Use `stringr::str_replace_all()` to replace values in a character vector or (with `dplyr::across()`) in select columns from a data.frame. `replace_street_dir_prefixes()` and `replace_street_suffixes()` pass reference data (street_dir_prefixes and street_suffixes) to the dict parameter to support formatting addresses with `bind_block_col()`.

Usage

```
replace_with_xwalk(
  x,
  .cols = NULL,
  xwalk = NULL,
  dict = NULL,
  abb = TRUE,
  case = NULL,
  .strict = TRUE,
  ignore_case = TRUE
```

```
)

replace_street_suffixes(
  x,
  street_suffix = "street_type",
  xwalk = NULL,
  abb = TRUE,
  case = NULL
)

replace_street_dir_prefixes(
  x,
  street_dir_prefix = "street_dir_prefix",
  xwalk = NULL,
  abb = TRUE,
  case = NULL
)
```

Arguments

x	A data.frame or character vector. If x is a character vector, .cols is optional. If x is a data.frame, x is required.
.cols	<tidy-select> Columns to transform. You can't select grouping columns because they are already automatically handled by the verb (i.e. summarise() or mutate()).
xwalk, dict	Named list or data frame with a minimum of two columns where one column contains the replacement values and the other the values to replace. If xwalk is NULL, dict is used and vice-versa. If both are provided, the xwalk values take precedence so they can be used to override a dict or add new values.
abb	If abb is TRUE (default), the second column of the dict is assumed to be abbreviation that should be used as the replace for the values in x or the replacement column. Otherwise, the first column is assumed to hold the replacement values and the second column is assumed to hold the original values. For example, for replace_street_suffixes(), If TRUE, replace full suffix names with abbreviations. If FALSE, replace abbreviations with full street suffix names.
case	Case to use for text in new columns or in modified values. Options include "lower", "upper", "title", or "sentence". Defaults to NULL which leaves the case as is.
.strict	If TRUE (default), match whole strings by appending "^" to the front and "$" to the end of each pattern in the xwalk.
ignore_case	Passed to stringr::regex()
street_suffix	Street suffix column to apply replacement function to.
street_dir_prefix	Street direction prefix column to apply replacement function to.

Examples

```
address_df <-
  data.frame(
    "bldg_num" = c("100", "1415", "600", "10"),
    "street_dir_prefix" = c(NA, NA, "N", NA),
    "street_name" = c("Holiday", "Key", "Charles", "Art Museum"),
    "street_type" = c("St", "Hwy", "St", "Dr")
  )

replace_street_suffixes(
  c("Street", "Highway", "Avenue", "Drive")
)

replace_street_suffixes(
  address_df,
  abb = FALSE,
  case = "sentence"
)

replace_street_dir_prefixes(
  c("North", "East", "West")
)

replace_street_dir_prefixes(
  c("S", "W", "N"),
  abb = FALSE,
  case = "sentence"
)
```

---

set_access_token *Set or get an access token or API key to/from environment variables.*

Description

Based on the `mapboxapi::mb_access_token()` function from the mapboxapi package by Kyle Walker.

Usage

```
set_access_token(
  token,
  overwrite = FALSE,
  install = FALSE,
  type = NULL,
  quiet = FALSE,
  call = caller_env()
)

get_access_token(token = NULL, type = NULL, call = caller_env())
```

Arguments

token	An access token or API key; required for `set_access_token()`. If token is not provided; type is required for `get_access_token()`. Length 1 character vector or list. If named, the name of the token is used in place of type.
overwrite	If TRUE, overwrite any existing token in .Renviron using the same environment variable name. Defaults to FALSE.
install	If TRUE, this function adds your token to your .Renviron for use in future sessions. Defaults to FALSE.
type	Default name used for environment variable where the token is saved.
quiet	If TRUE, suppress messages when setting token by locally setting the cli.default_handler option to `suppressMessages()`. 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.

---

set_pkg_options *Set getdata or other package-specific options*

Description

This function can set named options for a package using the convention of "pkg.option". For examples `set_pkg_options(crs = 2804, .pkg = "getdata")` sets the option "getdata.crs" to 2804. If "getdata.crs" is already set, overwrite must be TRUE to replace the existing value.

Usage

```
set_pkg_options(..., overwrite = FALSE, .pkg = "getdata")
```

Arguments

...	Named list of options to set, e.g. "crs = 2804" with .pkg = "getdata" to set "getdata.crs" to 2804.
overwrite	If TRUE, overwrite any existing option value.
.pkg	Package name to append to option name. Defaults to "getdata".

Options for the getdata package

Implemented options (with defaults if used) for the getdata package include:

- dist
- diag_ratio
- unit ("meter")
- asp
- crs (3857)

- from_crs (4326)
- address ("address")
- package
- filetype ("gpkg")

A similar convention is used for the maplayer package. The use of options is not implemented across all functions and may be changed in the future.

`street_dir_prefixes` *Street directional prefixes*

Description

A data frame based on Appendix I4 Directionals from the U.S. Postal Service Publication 28 - Postal Addressing Standards. According to the U.S. Postal Service, "Directionals are not commonly used in Puerto Rican addresses because other descriptions, such as the urbanization name identify geographic areas. In the ZIP+4 file, the English equivalents are used. Note: Although the Spanish word for West is Oeste, the abbreviation W is used."

Usage

```
street_dir_prefixes
```

Format

A data frame with 8 rows and 3 variables:

`street_dir_abb` Street directional abbreviation

`street_dir_en` Street directional name (English)

`street_dir_es` Street directional name (Spanish)

Details

Source: https://pe.usps.com/text/pub28/28api_007.htm

street_suffixes *Street suffix abbreviations*

Description

A data frame based on Appendix C1 Street Suffix Abbreviations from the U.S. Postal Service Publication 28 - Postal Addressing Standards. This data includes examples of suffix forms that are primary street suffix names, common street suffixes or suffix abbreviations, and recommended official U.S. Postal Service standard suffix abbreviations.

Usage

```
street_suffixes
```

Format

A data frame with 206 rows and 3 variables:

`street_suffix_abb` U.S. Postal Service standard suffix abbreviation

`street_suffix` Street suffix name

`street_suffix_common` List column with commonly used street suffix or abbreviation

Details

Source: https://pe.usps.com/text/pub28/28apc_002.htm

str_trim_squish_across *Trim and squish across any character columns*

Description

Apply `stringr::str_squish()` and `stringr::str_trim()` to all character columns in a data.frame.

Usage

```
str_trim_squish_across(x)
```

Arguments

x	A data.frame with character columns.

Index