Package 'wdpar'

December 18, 2020

Type Package

Version 1.0.6

Title Interface to the World Database on Protected Areas

Description Fetch and clean data from the World Database on Protected Areas (WDPA). Data is obtained from Protected Planet <https://www.protectedplanet.net/en>.

Imports utils, tools, sp, assertthat (>= 0.2.0), progress (>= 1.2.0), curl (>= 3.2), rappdirs (>= 0.3.1), httr (>= 1.3.1), countrycode (>= 1.1.0), wdman (>= 0.2.4), RSelenium (>= 1.7.4), xml2 (>= 1.2.0), cli (>= 1.0.1), lwgeom (>= 0.2-1), tibble (>= 2.1.3)

Suggests testthat (>= 2.0.1), knitr (>= 1.2.0), roxygen2 (>= 6.1.1), rmarkdown (>= 1.10), ggmap (>= 2.6.1), ggplot2 (>= 3.1.0), pingr (>= 1.1.2)

Depends R (>= 3.5.0), sf (>= 0.9-0)

License GPL-3

Encoding UTF-8

LazyData true

Language en-US

URL https://prioritizr.github.io/wdpar/, https://github.com/prioritizr/wdpar

BugReports https://github.com/prioritizr/wdpar/issues

VignetteBuilder knitr

RoxygenNote 7.1.1

Collate 'internal.R' 'geo.R' 'package.R' 'wdpa_clean.R' 'wdpa_url.R' 'wdpa_latest_version.R' 'wdpa_fetch.R' 'wdpa_read.R' 'zzz.R'

NeedsCompilation no

Author Jeffrey O Hanson [aut, cre]

Maintainer Jeffrey O Hanson <jeffrey.hanson@uqconnect.edu.au>

Repository CRAN

Date/Publication 2020-12-18 08:00:03 UTC

R topics documented:

`st_erase_overlaps` *Erase overlaps*

Description

Erase overlapping geometries in a `sf::sf()` object.

Usage

```
st_erase_overlaps(x, verbose = FALSE)
```

Arguments

`x`	`sf::sf()` object.
`verbose`	`logical` should progress be reported? Defaults to `FALSE`.

Details

This is a more robust—albeit slower—implementation for `sf::st_difference()` when `y` is missing.

Value

`sf::sf()` object.

See Also

`sf::st_difference()`.

Examples

```
# create data
pl1 <- sf::st_polygon(list(matrix(c(0, 0, 2, 0, 1, 1, 0, 0), byrow = TRUE,
                                  ncol = 2))) * 100
pl2 <- sf::st_polygon(list(matrix(c(0, 0.5, 2, 0.5, 1, 1.5, 0, 0.5),
                                  byrow = TRUE, ncol = 2))) * 100
pl3 <- sf::st_polygon(list(matrix(c(0, 1.25, 2, 1.25, 1, 2.5, 0, 1.25),
                                  byrow = TRUE, ncol = 2))) * 100
x <- sf::st_sf(order = c("A", "B", "C"),
               geometry = sf::st_sfc(list(pl1, pl2, pl3), crs = 3395))

# erase overlaps
y <- st_erase_overlaps(x)

# plot data for visual comparison
par(mfrow = c(1, 2))
plot(sf::st_geometry(x), xlim = c(0, 200), ylim = c(0, 250),
     main = "original", col = "transparent")
plot(sf::st_geometry(y), , xlim = c(0, 200), ylim = c(0, 250),
     main = "no overlaps", col = "transparent")
```

---

wdpar *wdpar: Interface to the World Database on Protected Areas*

Description

The **wdpar** R package provides an interface to the World Database on Protected Areas (WDPA). It provides functions for automatically downloading data (from Protected Planet) and cleaning data following best practices (outlined in Butchart *et al.* 2015; Runge *et al.* 2015). The main functions are `wdpa_fetch()` for downloading data and `wdpa_clean()` for cleaning data. For more information, please see the package vignette.

References

Butchart SH, Clarke M, Smith RJ, Sykes RE, Scharlemann JP, Harfoot M, ... & Brooks TM (2015) Shortfalls and solutions for meeting national and global conservation area targets. *Conservation Letters*, **8**: 329–337.

Runge CA, Watson JEM, Butchart HM, Hanson JO, Possingham HP & Fuller RA (2015) Protected areas and global conservation of migratory birds. *Science*, **350**: 1255–1258.

wdpa_clean *Clean data from the World Database on Protected Areas*

Description

Clean data obtained from the World Database on Protected Areas (WDPA).

Usage

```
wdpa_clean(
  x,
  crs = paste("+proj=cea +lon_0=0 +lat_ts=30 +x_0=0",
    "+y_0=0 +datum=WGS84 +ellps=WGS84 +units=m +no_defs"),
  snap_tolerance = 1,
  simplify_tolerance = 0,
  geometry_precision = 1500,
  erase_overlaps = TRUE,
  verbose = interactive()
)
```

Arguments

x	sf::sf() object containing protected area data.
crs	character or codeinteger object representing a coordinate reference system. Defaults to World Behrmann (*ESRI:54017*).
snap_tolerance	numeric tolerance for snapping geometry to a grid for resolving invalid geometries. Defaults to 1 meter.
simplify_tolerance	numeric simplification tolerance. Defaults to 0 meters.
geometry_precision	numeric level of precision for processing the spatial data (used with sf::st_set_precision()). The default argument is 1500 (higher values indicate higher precision). This level of precision is generally suitable for analyses at the national-scale. For analyses at finer-scale resolutions, please consider using a greater value (e.g. 10000).
erase_overlaps	logical should overlapping boundaries be erased? This is useful for making comparisons between individual protected areas and understanding their "effective" geographic coverage. On the other hand, this processing step may not be needed (e.g. if the protected area boundaries are going to be rasterized), and so processing time can be substantially by skipping this step and setting the argument to FALSE. Defaults to TRUE.
verbose	logical should progress on data cleaning be reported? Defaults to TRUE in an interactive session, otherwise FALSE.

Details

This function cleans data from World Database on Protected Areas following best practices (Butchart *et al.* 2015, Runge *et al.* 2015, https://www.protectedplanet.net/en/resources/calculating-protected-area-cov To obtain accurate protected area coverage statistics for a country, please note that you will need to manually clip the cleaned data to the countries' coastline and its Exclusive Economic Zone (EEZ). Although this function can *in theory* be used to clean the global dataset, this process can take several weeks to complete. Therefore, it is strongly recommended to use alternative methods for cleaning the global dataset.

1. Repair invalid geometry (using `sf::st_make_valid()`).
2. Exclude protected areas that are not currently implemented (i.e. exclude areas without the status `"Designated"`, `"Inscribed"`, `"Established"`).
3. Exclude United Nations Educational, Scientific and Cultural Organization (UNESCO) Biosphere Reserves (Coetzer *et al.* 2014).
4. Create a field (`"GEOMETRY_TYPE"`) indicating if areas are represented as point localities (`"POINT"`) or as polygons (`"POLYGON"`).
5. Exclude areas represented as point localities that do not have a reported spatial extent (i.e. missing data for the field
6. Geometries are wrapped to the dateline (using `sf::st_wrap_dateline()` with the options `"WRAPDATELINE=YES"` and `"DATELINEOFFSET=180"`).
7. Reproject data to coordinate system specified in argument to `crs` (using `sf::st_transform()`).
8. Fix any invalid geometries that have manifested (using `sf::st_make_valid()`).
9. Buffer areas represented as point localities to circular areas using their reported spatial extent (using data in the field `"REP_AREA"` and `sf::st_buffer()`; see Visconti *et al.* 2013).
10. Snap the geometries to a grid to fix any remaining geometry issues (using argument to `snap_tolerance` and `lwgeom::st_snap_to_grid()`).
11. Fix any invalid geometries that have manifested (using `sf::st_make_valid()`).
12. Simplify the protected area geometries to reduce computational burden (using argument to `simplify_tolerance` and `sf::st_simplify()`).
13. Fix any invalid geometries that have manifested (using `sf::st_make_valid()`).
14. The `"MARINE"` field is converted from integer codes to descriptive names (i.e. `0` = `"terrestrial"`, `1` = `"partial"`, `2` = `"marine"`).
15. Zeros in the `"STATUS_YR"` field are replaced with missing values (i.e. `NA_real_` values).
16. Zeros in the `"NO_TK_AREA"` field are replaced with `NA` values for areas where such data are not reported or applicable (i.e. areas with the values `"Not Applicable"` or `"Not Reported"` in the `"NO_TK_AREA"` field).
17. Overlapping geometries are erased from the protected area data (discussed in Deguignet *et al.* 2017). Geometries are erased such that areas associated with more effective management categories (`"IUCN_CAT"`) or have historical precedence are retained (using `sf::st_difference()`).
18. Slivers are removed (geometries with areas less than 0.1 square meters).
19. The size of areas are calculated in square kilometers and stored in the field `"AREA_KM2"`.

Value

`sf::sf()` object.

References

Butchart SH, Clarke M, Smith RJ, Sykes RE, Scharlemann JP, Harfoot M, ... & Brooks TM (2015) Shortfalls and solutions for meeting national and global conservation area targets. *Conservation Letters*, **8**: 329–337.

Coetzer KL, Witkowski ET, & Erasmus BF (2014) Reviewing Biosphere Reserves globally: Effective conservation action or bureaucratic label? *Biological Reviews*, **89**: 82–104.

Deguignet M, Arnell A, Juffe-Bignoli D, Shi Y, Bingham H, MacSharry B & Kingston N (2017) Measuring the extent of overlaps in protected area designations. *PloS One*, **12**: e0188681.

Runge CA, Watson JEM, Butchart HM, Hanson JO, Possingham HP & Fuller RA (2015) Protected areas and global conservation of migratory birds. *Science*, **350**: 1255–1258.

Visconti P, Di Marco M, Alvarez-Romero JG, Januchowski-Hartley SR, Pressey, RL, Weeks R & Rondinini C (2013) Effects of errors and gaps in spatial data sets on assessment of conservation progress. *Conservation Biology*, **27**: 1000–1010.

See Also

`wdpa_fetch()`, `https://www.protectedplanet.net/en/resources/calculating-protected-area-coverage`.

Examples

```
## Not run:
# fetch data for the Liechtenstein
lie_raw_data <- wdpa_fetch("LIE", wait = TRUE)

# clean data
lie_data <- wdpa_clean(lie_raw_data)

# plot cleaned dataset
plot(lie_data)

## End(Not run)
```

---

wdpa_fetch *Fetch data from the World Database on Protected Areas*

Description

Download data from the World Database on Protected Areas (WDPA) (available at `https://www.protectedplanet.net/en`) and import it. Note that data are downloaded assuming non-commercial use.

Usage

```
wdpa_fetch(
  x,
  wait = FALSE,
  download_dir = rappdirs::user_data_dir("wdpar"),
  force_download = FALSE,
  verbose = interactive()
)
```

Arguments

x	character country for which to download data. This argument can be the name of the country (e.g. "Liechtenstein") or the ISO-3 code for the country (e.g. "LIE"). This argument can also be set to "global" to download all of the protected areas available in the database (approximately 1.1 GB).
wait	logical if data is not immediately available for download should the session be paused until it is ready for download? If argument to wait is FALSE and the data is not ready then NA will be returned. Defaults to FALSE.
download_dir	character folder path to download the data. Defaults to a persistent data directory (rappdirs::user_data_dir("wdpar")).
force_download	logical if the data has previously been downloaded and is available at argument to download_dir, should a fresh copy be downloaded? Defaults to FALSE.
verbose	logical should a progress on downloading data be reported? Defaults to TRUE in an interactive session, otherwise FALSE.

Details

This function will download the specified protected area data and return it. **It is strongly recommended that the data be cleaned prior to analysis**. Check out the wdpa_clean() function to clean the data according to standard practices. For information on this database, prefer refer to the official manual (https://www.protectedplanet.net/en/resources/wdpa-manual).

Please note that this function will sometimes return the error PhantomJS signals port = 4567 is already in use. This can occur when you have previously run the function and terminated it early. To address this issue, you will need to restart your computer.

Value

sf::sf() object.

See Also

wdpa_clean(), wdpa_read(), wdpa_url(), countrycode::countrycode(), https://www.protectedplanet.net/en, https://www.protectedplanet.net/en/resources/wdpa-manual.

Examples

```
## Not run:
# fetch data for Liechtenstein
lie_raw_data <- wdpa_fetch("Liechtenstein", wait = TRUE)

# fetch data for Liechtenstein using the ISO3 code
lie_raw_data <- wdpa_fetch("LIE")

# print data
print(lie_raw_data)

# plot data
plot(lie_raw_data)

# data for multiple countries can be downloaded separately and combined,
# this is useful to avoid having to download the global dataset
## load packages to easily merge datasets
library(dplyr)
library(tibble)

## define country names to download
country_codes <- c("LIE", "MHL")

## download data for each country
mult_data <- lapply(country_codes, wdpa_fetch, wait = TRUE)

## merge datasets together
mult_dat <- st_as_sf(as_tibble(bind_rows(mult_data)))

## print data
print(mult_dat)

## End(Not run)
```

---

wdpa_latest_version *Query latest version of the World Database on Protected Areas*

Description

Find the latest version of the World Database on Protected Areas dataset. This is a character identifier representing the month and year (e.g. `Sep2020`) the data were released.

Usage

```
wdpa_latest_version()
```

Details

The version number is determined using a web address where the global dataset is available. For specific details, please refer to the source code for this function.

Value

`character` version of the dataset.

Examples

```
## Not run:
# find the latest version
wdpa_latest_version()

## End(Not run)
```

`wdpa_read` *Read data from the World Database on Protected Areas*

Description

Read data from the World Database on Protected Areas from a local file. This function assumes that the data has already been downloaded to your computer, see the `wdpa_fetch()` function for automatically downloading and importing the data into the current session.

Usage

```
wdpa_read(x, n = NULL)
```

Arguments

`x`	`character` file name for a zip archive file downloaded from https://www.protectedplanet.net/en.
`n`	`integer` number of records to import per data source. Defaults to `NULL` such that all data are imported.

Value

`sf::sf()` object.

See Also

`wdpa_fetch()`, `wdpa_clean()`, https://www.protectedplanet.net/en.

Examples

```
## Not run:
# find url for Liechtenstein dataset
download_url <- wdpa_url("LIE", wait = TRUE)

# path to save file zipfile with data
path <- tempfile(pattern = "WDPA_", fileext = ".zip")
```

```
# download zipfile
result <- httr::GET(download_url, httr::write_disk(path))

# load data
lie_raw_data <- wdpa_read(path)

# plot data
plot(lie_raw_data)

## End(Not run)
```

---

wdpa_url *Download URL for the World Database on Protected Areas*

Description

Obtain the URL for downloading data from the World Database on Protected Areas (WDPA).

Usage

```
wdpa_url(x, wait = FALSE)
```

Arguments

x	character country for desired data. This argument can be the name of the country (e.g. "Liechtenstein") or the ISO-3 code for the country (e.g. "LIE"). This argument can also be set to "global" to obtain the URL for the global dataset.
wait	logical if data is not immediately available for download should the session be paused until it is ready for download? If argument to wait is FALSE and the data is not ready then an error will be thrown. Defaults to FALSE.

Value

character URL to download the data.

See Also

wdpa_fetch(), countrycode::countrycode().

Examples

```
## Not run:
# obtain url for New Zealand data
nzl_url <- wdpa_url("New Zealand", wait = TRUE)
print(nzl_url)

# obtain url for New Zealand data using its ISO3 code
```

```
nzl_url <- wdpa_url("NZL", wait = TRUE)
print(nzl_url)

# obtain url for global data
global_url <- wdpa_url("global")
print(global_url)

## End(Not run)
```

Index