

Package ‘wdpar’

December 5, 2021

Type Package

Version 1.3.2

Title Interface to the World Database on Protected Areas

Description Fetch and clean data from the World Database on Protected Areas (WDPA) and the World Database on Other Effective Area-Based Conservation Measures (WDOECM). Data is obtained from Protected Planet <<https://www.protectedplanet.net/en>>. To augment data cleaning procedures, users can install the 'prepr' R package (available at <<https://github.com/dickoa/prepr>>).

Imports utils, sp, assertthat (>= 0.2.0), progress (>= 1.2.0), curl (>= 3.2), 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), rappdirs (>= 0.3.1)

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), prepr (>= 0.1.9000), dplyr (>= 1.0.7)

Depends R (>= 3.5.0), sf (>= 1.0-2)

License GPL-3

Encoding UTF-8

Language en-US

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

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

VignetteBuilder knitr

RoxygenNote 7.1.2

Collate 'internal.R' 'package.R' 'read_sf_n.R' 'st_erase_overlaps.R' 'st_repair_geometry.R' 'wdpa_clean.R' 'wdpa_dissolve.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 2021-12-05 08:00:02 UTC

R topics documented:

st_erase_overlaps	2
st_repair_geometry	3
wdpar	4
wdpa_clean	5
wdpa_dissolve	8
wdpa_fetch	9
wdpa_latest_version	12
wdpa_read	13
wdpa_url	14
Index	16

st_erase_overlaps	<i>Erase overlaps</i>
-------------------	-----------------------

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()`, `wdpa_dissolve()`.

Examples

```
# create data
p11 <- sf::st_polygon(list(matrix(c(0, 0, 2, 0, 1, 1, 0, 0), byrow = TRUE,
                                ncol = 2))) * 100
p12 <- sf::st_polygon(list(matrix(c(0, 0.5, 2, 0.5, 1, 1.5, 0, 0.5),
                                byrow = TRUE, ncol = 2))) * 100
p13 <- 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(p11, p12, p13), 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")
```

st_repair_geometry *Repair geometry*

Description

Repair the geometry of a `sf::st_sf()` object.

Usage

```
st_repair_geometry(x, geometry_precision = 1500)
```

Arguments

`x` `sf::sf()` object.

`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).

Details

This function works by first using the `sf::st_make_valid()` function to attempt to fix geometry issues. Since the `sf::st_make_valid()` function sometimes produce incorrect geometries in rare cases (e.g. when fixing invalid geometries that cross the dateline), this function then uses the `st_prepair()` function from the **prepr** package to fix those geometries instead (see <https://github.com/dickoa/prepr> for details).

Installation

This function uses the **prepr** package to help repair geometries in certain cases. Because the **prepr** package is not available on the Comprehensive R Archive Network (CRAN), it must be installed from its online code repository. To achieve this, please use the following code:

```
if (!require(remotes)) install.packages("remotes")
remotes::install_github("dickoa/prepr")
```

Note that the **prepr** package has system dependencies that need to be installed before the package itself can be installed (see package README file for platform-specific instructions).

Examples

```
# create sf object
p1 <- st_sf(
  id = 1,
  geometry = st_as_sfc("POLYGON((0 0, 0 10, 10 0, 10 10, 0 0))", crs = 3857)
)

# repair geometry
p2 <- st_repair_geometry(p1)

# print object
print(p2)
```

wdpar

wdpar: Interface to the World Database on Protected Areas

Description

The **wdpar** R package provides an interface to data provided by **Protected Planet**. Specifically, it can be used to automatically download data from the World Database on Protected Areas (WDPA) and the World Database on Other Effective Area-Based Conservation Measures (WDOECM). It also provides methods for cleaning data from these databases following best practices (outlined in Butchart *et al.* 2015; Protected Planet 2021; 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.

Protected Planet (2021) Calculating protected and OECM area coverage. Available at: <https://www.protectedplanet.net/en/resources/calculating-protected-area-coverage>.

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

Description

Clean data obtained from **Protected Planet**. Specifically, this function is designed to clean data obtained from the World Database on Protected Areas (WDPA) and the World Database on Other Effective Area-Based Conservation Measures (WDOECM). For recommended practices on cleaning large datasets (e.g. datasets that span multiple countries or a large geographic area), please see below.

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"),
  exclude_unesco = TRUE,
  retain_status = c("Designated", "Inscribed", "Established"),
  snap_tolerance = 1,
  simplify_tolerance = 0,
  geometry_precision = 1500,
  erase_overlaps = TRUE,
  verbose = interactive()
)
```

Arguments

<code>x</code>	<code>sf::sf()</code> object containing protected area data.
<code>crs</code>	character or codeinteger object representing a coordinate reference system. Defaults to World Behrmann (<i>ESRI:54017</i>).
<code>exclude_unesco</code>	logical should UNESCO Biosphere Reserves be excluded? Defaults to TRUE.
<code>retain_status</code>	character vector containing the statuses for protected areas that should be retained during the cleaning process. Available statuses include: "Proposed", "Inscribed", "Adopted", "Designated", and "Established". Additionally, a NULL argument can be specified to ensure that no protected areas are excluded according to their status. The default argument is a character vector containing "Designated", "Inscribed", and "Established". This default argument ensures that protected areas that are not currently implemented are excluded.
<code>snap_tolerance</code>	numeric tolerance for snapping geometry to a grid for resolving invalid geometries. Defaults to 1 meter.
<code>simplify_tolerance</code>	numeric simplification tolerance. Defaults to 0 meters.
<code>geometry_precision</code>	numeric level of precision for processing the spatial data (used with <code>sf::st_set_precision()</code>). 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 following best practices (Butchart *et al.* 2015; Protected Planet 2021; Runge *et al.* 2015). 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).

1. Repair invalid geometry (using `sf::st_make_valid()`).
2. Exclude protected areas according to their status (i.e. "STATUS" field). Specifically, protected areas that have a status not specified in the argument to `retain_status` are excluded. By default, only protected areas that have a "Designated", "Inscribed", or "Established" status are retained. This means that the default behavior is to exclude protected that are not currently implemented.
3. Exclude United Nations Educational, Scientific and Cultural Organization (UNESCO) Biosphere Reserves (Coetzer *et al.* 2014). This step is only performed if the argument to `exclude_unesco` is TRUE.
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. Repair any invalid geometries that have manifested (using `st_repair_geometry()`).
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. Repair any invalid geometries that have manifested (using `st_repair_geometry()`).
12. Simplify the protected area geometries to reduce computational burden (using argument to `simplify_tolerance` and `sf::st_simplify()`).
13. Repair any invalid geometries that have manifested (using `st_repair_geometry()`).

14. The "MARINE" field is converted from integer codes to descriptive names (i.e. 0 = "terrestrial", 1 = "partial", 2 = "marine").
15. The "PA_DEF" field is converted from integer codes to descriptive names (i.e. 0 = "OECM", and 1 = "PA").
16. Zeros in the "STATUS_YR" field are replaced with missing values (i.e. NA_real_ values).
17. 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).
18. 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()`).
19. Slivers are removed (geometries with areas less than 0.1 square meters).
20. The size of areas are calculated in square kilometers and stored in the field "AREA_KM2".

Value

`sf::sf()` object.

Recommended practices for large datasets

This function can be used to clean large datasets assuming that sufficient computational resources and time are available. Indeed, it can clean data spanning large countries, multiple countries, and even the full global dataset. When processing the full global dataset, it is recommended to use a computer system with at least 32 GB RAM available and to allow for at least one full day for the data cleaning procedures to complete. It is also recommended to avoid using the computer system for any other tasks while the data cleaning procedures are being completed, because they are very computationally intensive. Additionally, when processing large datasets – and especially for the global dataset – it is strongly recommended to disable the procedure for erasing overlapping areas. This is because the built-in procedure for erasing overlaps is very time consuming when processing many protected areas, so that information on each protected area can be output (e.g. IUCN category, year established). Instead, when cleaning large datasets, it is recommended to run the data cleaning procedures with the procedure for erasing overlapping areas disabled (i.e. with `erase_overlaps = FALSE`). After the data cleaning procedures have completed, the protected area data can be manually dissolved to remove overlapping areas (e.g. using `wdpa_dissolve()`). For an example of processing a large protected area dataset, please see the 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.
- 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.

Protected Planet (2021) Calculating protected and OECM area coverage. Available at: <https://www.protectedplanet.net/en/resources/calculating-protected-area-coverage>.

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\(\)](#), [wdpa_dissolve\(\)](#).

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_dissolve

Dissolve data

Description

Create a dataset of spatial boundaries that contains no overlapping geometries.

Usage

```
wdpa_dissolve(x, geometry_precision = 1500)
```

Arguments

x [sf::sf\(\)](#) object.

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).

Details

This function is basically a wrapper for `sf::st_union()`. It also contains additional parameters to assist with processing large and complex geometry data.

Value

`sf::sf()` object.

See Also

`sf::st_union()`, `st_erase_overlaps()`.

Examples

```
# create data
p11 <- sf::st_polygon(list(matrix(c(0, 0, 2, 0, 1, 1, 0, 0), byrow = TRUE,
                                ncol = 2))) * 100
p12 <- sf::st_polygon(list(matrix(c(0, 0.5, 2, 0.5, 1, 1.5, 0, 0.5),
                                byrow = TRUE, ncol = 2))) * 100
p13 <- 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(p11, p12, p13), crs = 3395))

# dissolve data
y <- wdpa_dissolve(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 = "dissolved", col = "transparent")
```

wdpa_fetch

Fetch data

Description

Fetch data from **Protected Planet**. Specifically, data are downloaded from the World Database on Protected Areas (WDPA) and the World Database on Other Effective Area-Based Conservation Measures (WDOECM). **Note that data are downloaded assuming non-commercial use.**

Usage

```
wdpa_fetch(
  x,
  wait = FALSE,
  download_dir = tempdir(),
```

```

force_download = FALSE,
check_version = TRUE,
n = NULL,
page_wait = 2,
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 temporary directory. To avoid downloading the same dataset multiple times, it is recommended to use a persistent directory (e.g. <code>rappdirs::user_data_dir("wdpar")</code>); see Examples below).
force_download	logical if the data has previously been downloaded and is available at argument to <code>download_dir</code> , should a fresh copy be downloaded? Defaults to FALSE.
check_version	logical if the data are being imported from from the argument to <code>download_dir</code> , should the data be checked to see if the version number matches the latest version available online? Defaults to TRUE.
n	integer number of records to import per data source. Defaults to NULL such that all data are imported.
page_wait	numeric number of seconds to wait for web pages to load when finding the download URL on Protected Planet . Defaults to 2. Since the process of finding a download URL requires navigating through multiple web pages, the default argument means that the function will take at least 8 seconds to complete. Users on slow internet connections may experience issues with the default argument (e.g. resulting in an error containing the message <code>Error: Summary: NoSuchElement</code>). To avoid this, users can try specifying a greater value (e.g. 5 seconds).
verbose	logical should a progress on downloading data be reported? Defaults to TRUE in an interactive session, otherwise FALSE.

Details

This function will check to see if the specified data have previously been downloaded, download the data if needed, and import the data. After importing the data, it is strongly recommended to clean the data prior to analysis (see `wdpa_clean()`).

Value

`sf::sf()` object.

Data source

The PA_DEF column indicates the data source for individual areas and sites that comprise the imported dataset. Specifically, data obtained through the World Database on Protected Areas (WDPA) are indicated with a value of 1 in the PA_DEF column. Additionally, data obtained through the World Database on Other Effective Area-Based Conservation Measures (WDOECM) are indicated with a value of 0 in the PA_DEF column. For more details on data conventions, please consult the official manual (UNEP-WCMC 2019).

Troubleshooting

This function will sometimes return the error message PhantomJS signals port = 4567 is already in use. This error message can occur when you have previously run the function and it threw an error, or it terminated early. It can also occur when attempting to run the the function in multiple sessions on the same computer. To address this issue, you will need to restart your computer.

References

UNEP-WCMC (2019). User Manual for the World Database on Protected Areas and world database on other effective area-based conservation measures: 1.6. UNEP-WCMC: Cambridge, UK. Available at: https://wcmc.io/WDPA_Manual.

See Also

`wdpa_clean()`, `wdpa_read()`, `wdpa_url()`, `countrycode::countrycode()`.

Examples

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

# print data
print(lie_raw_data)

# plot data
plot(lie_raw_data)

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

# since data are saved in a temporary directory by default,
# a persistent directory can be specified to avoid having to download the
# same dataset every time the R session is restarted
lie_raw_data <- wdpa_fetch("LIE", wait = TRUE,
                          download_dir = rappdirs::user_data_dir("wdpar"))

# 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*

Description

Find the latest version of the combined World Database on Protected Areas (WDPA) and World Database on Other Effective Area-Based Conservation Measures (WDOECM) 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	<i>Read data</i>
-----------	------------------

Description

Read data obtained from **Protected Planet**. Specifically, this function is designed to import data obtained from the World Database on Protected Areas (WDPA) and the World Database on Other Effective Area-Based Conservation Measures (WDOECM).

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.

Details

This function assumes that data have previously been downloaded to your computer, and need to import the data. After importing the data, it is strongly recommended to clean the data prior to analysis (see [wdpa_clean\(\)](#)).

Value

[sf::sf\(\)](#) object.

Data source

The PA_DEF column indicates the data source for individual areas and sites that comprise the imported dataset. Specifically, data obtained through the World Database on Protected Areas (WDPA) are indicated with a value of 1 in the PA_DEF column. Additionally, data obtained through the World Database on Other Effective Area-Based Conservation Measures (WDOECM) are indicated with a value of 0 in the PA_DEF column. For more details on data conventions, please consult the official manual (UNEP-WCMC 2019).

References

UNEP-WCMC (2019). User Manual for the World Database on Protected Areas and world database on other effective area-based conservation measures: 1.6. UNEP-WCMC: Cambridge, UK. Available at: https://wcmc.io/WDPA_Manual.

See Also

[wdpa_fetch\(\)](#), [wdpa_clean\(\)](#).

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	<i>Download URL</i>
----------	---------------------

Description

Obtain a URL to download data from **Protected Planet**. Specifically, the URL provides access to data available through the World Database on Protected Areas (WDPA) and the World Database on Other Effective Area-Based Conservation Measures (WDOECM). **Note that data are accessed assuming non-commercial use.**

Usage

```
wdpa_url(x, wait = FALSE, page_wait = 2)
```

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.
page_wait	numeric number of seconds to wait for web pages to load when finding the download URL on Protected Planet . Defaults to 2. Since the process of finding a download URL requires navigating through multiple web pages, the default argument means that the function will take at least 8 seconds to complete. Users on slow internet connections may experience issues with the default argument (e.g. resulting in an error containing the message Error: Summary:

NoSuchElement). To avoid this, users can try specifying a greater value (e.g. 5 seconds).

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

countrycode::countrycode(), *11, 15*

lwgeom::st_snap_to_grid(), *6*

sf::sf(), *2, 3, 5, 7–10, 13*

sf::st_buffer(), *6*

sf::st_difference(), *2, 7*

sf::st_make_valid(), *3, 6*

sf::st_set_precision(), *3, 5, 8*

sf::st_sf(), *3*

sf::st_simplify(), *6*

sf::st_transform(), *6*

sf::st_union(), *9*

sf::st_wrap_dateline(), *6*

st_erase_overlaps, *2*

st_erase_overlaps(), *9*

st_repair_geometry, *3*

st_repair_geometry(), *6*

wdpa_clean, *5*

wdpa_clean(), *4, 10, 11, 13*

wdpa_dissolve, *8*

wdpa_dissolve(), *2, 7, 8*

wdpa_fetch, *9*

wdpa_fetch(), *4, 8, 13, 15*

wdpa_latest_version, *12*

wdpa_read, *13*

wdpa_read(), *11*

wdpa_url, *14*

wdpa_url(), *11*

wdpar, *4*