

# Package ‘wdpar’

June 29, 2019

**Type** Package

**Version** 1.0.0

**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 <<http://protectedplanet.net>>.

**Imports** utils, 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 (>= 0.1-4)

**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(>= 3.4.0), sf(>= 0.7-1)

**License** GPL-3

**LazyData** true

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

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

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**Collate** 'internal.R' 'geo.R' 'package.R' 'wdpa\_clean.R' 'wdpa\_url.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](mailto:jeffrey.hanson@uqconnect.edu.au)>

**Repository** CRAN

**Date/Publication** 2019-06-29 17:20:04 UTC

## R topics documented:

st_erase_overlaps . . . . .	2
wdpar . . . . .	3
wdpa_clean . . . . .	3
wdpa_fetch . . . . .	6
wdpa_read . . . . .	7
wdpa_url . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

---

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

---

### Description

Erase overlapping geometries in a [sf](#) object.

### Usage

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

### Arguments

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

### Details

This is a more robust—albeit slower—implementation for [st\\_difference](#) when `y` is missing.

### Value

[sf](#) object.

### See Also

[st\\_difference](#).

### Examples

```
# create data
p1 <- sf::st_polygon(list(matrix(c(0, 0, 2, 0, 1, 1, 0, 0), byrow = TRUE,
                                ncol = 2))) * 100
p2 <- sf::st_polygon(list(matrix(c(0, 0.5, 2, 0.5, 1, 1.5, 0, 0.5),
                                byrow = TRUE, ncol = 2))) * 100
p3 <- 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), main = "original", col = "white")
plot(sf::st_geometry(y), main = "no overlaps", col = "white")

```

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 = 1000, erase_overlaps = TRUE,
  verbose = interactive())

```

## Arguments

x	<code>sf</code> object containing protected area data.
crs	character or codeinteger object representing a coordinate reference system. Defaults to World Behrmann ( <i>ESRI:54017</i> ).
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 <code>st_set_precision</code> ). The default argument corresponds to the nearest millimeter (i.e. 1000).
erase_overlaps	logical should overlapping boundaries be removed erased? This can be useful when the protected area boundaries are going to be rasterized, and so processing time can be substantially reduced by skipping this step because overlapping boundaries will not be a problem. 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://protectedplanet.net/c/calculating-protected-area-coverage>). 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 `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 `st_wrap_dateline` with the options "WRAPDATELINE=YES" and "DATELINEOFFSET=180").
7. Reproject data to coordinate system specified in argument to `crs` (using `st_transform`).
8. Fix any invalid geometries that have manifested (using (using `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 `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 `st_snap_to_grid`).

11. Fix any invalid geometries that have manifested (using (using `st_make_valid`).
12. Simplify the protected area geometries to reduce computational burden (using argument to `simplify_tolerance` and `st_simplify`).
13. Fix any invalid geometries that have manifested (using (using `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 `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` 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://protectedplanet.net/c/calculating-protected-area-coverage>.

### Examples

```
# 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)
```

---

wdpa\_fetch

*Fetch data from the World Database on Protected Areas*


---

## Description

Download data from the World Database on Protected Areas (WDPA) (available at <http://protectedplanet.net>) and import it.

## 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/c/wdpa-manual>).

## Value

[sf](#) object.

## See Also

[wdpa\\_clean](#), [wdpa\\_read](#), [wdpa\\_url](#), [countrycode](#), <http://protectedplanet.net>, <https://www.protectedplanet.net/c/wdpa-manual>.

## Examples

```
# fetch data for Liechtenstein
# note that this does not include areas represented as point localities
lie_raw_data <- wdpa_fetch("Liechtenstein", wait = TRUE)

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

# plot data
plot(lie_raw_data)

## Not run:
# fetch data for Liechtenstein, including areas represented as points
# download global data
global_raw_data <- wdpa_fetch("global", wait = TRUE)

# subset data to include only Liechtenstein
lie_raw_data2 <- global_raw_data %>% filter(ISO == "LIE")

# plot data
plot(lie_raw_data2)

## 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)
```

## Arguments

x character file name for a zip archive file downloaded from <http://protectedplanet.net>.

**Value**

`sf` object.

**See Also**

`wdpa_fetch`, `wdpa_clean`, <http://protectedplanet.net>.

**Examples**

```
# 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)
```

---

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**

<code>x</code>	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.
<code>wait</code>	logical if data is not immediately available for download should the session be paused until it is ready for download? If argument to <code>wait</code> is <code>FALSE</code> and the data is not ready then an error will be thrown. Defaults to <code>FALSE</code> .



**Value**

character URL to download the data.

**See Also**

[wdpa\\_fetch](#), [countrycode](#).

**Examples**

```
# 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)
```

# Index

countrycode, [7](#), [9](#)

sf, [2](#), [4–6](#), [8](#)

st\_buffer, [4](#)

st\_difference, [2](#), [5](#)

st\_erase\_overlaps, [2](#)

st\_make\_valid, [4](#), [5](#)

st\_set\_precision, [4](#)

st\_simplify, [5](#)

st\_snap\_to\_grid, [4](#)

st\_transform, [4](#)

st\_wrap\_dateline, [4](#)

wdpa\_clean, [3](#), [3](#), [6–8](#)

wdpa\_fetch, [3](#), [5](#), [6](#), [7–9](#)

wdpa\_read, [7](#), [7](#)

wdpa\_url, [7](#), [8](#)

wdpar, [3](#)

wdpar-package (wdpar), [3](#)