

Package: OccurrenceCleaner (via r-universe)

October 13, 2026

Title Fast Quality Checks for Species Occurrence Records

Version 0.2.0

Description An extension of 'CoordinateCleaner' (Zizka et al. 2019 <[doi:10.1111/2041-210X.13152](https://doi.org/10.1111/2041-210X.13152)>) tailored to large-scale species occurrence data. Flags common problems in occurrence records, such as coordinates at country centroids, capitals or biodiversity institutions, zero or equal coordinates, records at sea, in the wrong country or outside the species range, duplicates, outliers within a species, and datasets with converted or gridded coordinates, with the test definitions and default buffers of 'CoordinateCleaner' apart from documented differences. It also reads the quality warnings GBIF attaches to each record, the basis of record and the establishment means, and checks records at sea or on land against the habitat of their genus from the Interim Register of Marine and Nonmarine Genera. Records are flagged and kept, and users choose afterwards which flags to filter on. Data frames of a few million records are flagged in seconds, and GBIF snapshots of billions of records are processed by reading Parquet files in chunks.

License MIT + file LICENSE

URL <https://gillescolling.com/OccurrenceCleaner/>,
<https://github.com/gcol33/OccurrenceCleaner>

BugReports <https://github.com/gcol33/OccurrenceCleaner/issues>

Depends R (>= 4.1.0)

Imports graphics, grDevices, Rcpp, tools, utils, vectra (>= 0.13.0)

LinkingTo Rcpp

Suggests CoordinateCleaner, geosphere, ggplot2, knitr, maps, rmarkdown, rnaturalearth, rnaturalearthdata, sf, svglite, terra, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Collate 'OccurrenceCleaner-package.R' 'RcppExports.R' 'group.R'
'flag.R' 'clean.R' 'columns.R' 'data.R' 'flag_tests.R'
'habitat.R' 'methods.R' 'progress.R' 'pyrate.R' 'reference.R'
'shapefile.R' 'spatialvalid.R' 'store.R' 'stream.R' 'tally.R'
'test-docs.R'

Repository <https://gcol33.r-universe.dev>

Date/Publication 2026-10-13 14:24:17 UTC

RemoteUrl <https://github.com/gcol33/OccurrenceCleaner>

RemoteRef HEAD

RemoteSha 0e801fd2e1383c8aec38276a07ac87c35b476b16

Contents

as_spatialvalid	3
clean_age_equal	4
clean_age_outlier	5
clean_age_range_outlier	7
clean_aohi	8
clean_basis_fossil	10
clean_capital	11
clean_centroid	12
clean_country_mismatch	14
clean_dataset_ddmm	15
clean_dataset_rounded	17
clean_duplicate	18
clean_equal	19
clean_establishment_introduced	20
clean_gbif_headquarters	22
clean_institution	23
clean_invalid	25
clean_issue_coordinate	26
clean_outlier	28
clean_range	30
clean_sea	31
clean_sea_nonmarine	32
clean_spatiotemporal_outlier	34
clean_urban	36
clean_zero	37
download_reference	38
export_pyrate	39
flag_mask	40

flag_occurrences	41
flag_reference	45
flag_tests	45
gentiana	46
keep_unflagged	47
occ_columns	48
plot.occ_summary	50
read_flagged	51
reference_habitat	52
reference_info	53
reference_points	53
summary.occ_flags	54
tally_flags	56
test_plan	57
Index	58

as_spatialvalid	<i>Convert flags to CoordinateCleaner's spatialvalid format</i>
-----------------	---

Description

Returns flagged records in the layout that **CoordinateCleaner**'s `clean_coordinates()` and `clean_fossils()` return with value = "spatialvalid", so code written for that object (its `summary()` and `plot()` methods, filters on `.summary`) runs unchanged. The record columns come first, then one logical column per test, TRUE when the record passes, then `.summary`, TRUE when it passes every test given.

Usage

```
as_spatialvalid(x, tests = NULL)
```

Arguments

- x

A flagged data frame (occ_flags) from `flag_occurrences()`, or flags written to disk (an occ_flagged object or its directory). Flags on disk are read into memory with their keep columns as the record columns.
- tests

Tests to include, from those x was flagged with; default all of them. `.summary` is then the complement of the rule made of these tests.

Details

Each test takes the column name **CoordinateCleaner** uses for it (`.cap` for capital, `.otl` for outlier, see the `spatialvalid` column of `flag_tests()`); a test without one there is written as `.<test>`, for example `.issue_coordinate`. The columns are decoded from the flags word, so a record whose flag word is NA is NA in every column.

Value

A data frame of class `c("spatialvalid", "data.frame")`, carrying the reference and layout version of the flags as attributes `occ_reference` and `occ_layout_version`.

See Also

[flag_occurrences\(\)](#), [has_flag\(\)](#)

Examples

```
x <- data.frame(species = c("a", "a", "b"),
               decimalLongitude = c(0, 16.37, 10),
               decimalLatitude = c(0, 48.21, 50))
sv <- as_spatialvalid(flag_occurrences(x))
sv
sv[sv$.summary, ]
```

clean_age_equal	<i>Test: fossil age range</i>
-----------------	-------------------------------

Description

Flags records whose minimum age is equal to or greater than their maximum age: a zero-length or inverted stratigraphic range, which in the Paleobiology Database usually means one age was copied into both fields or the two were swapped. Records with a missing age are not flagged.

Usage

```
clean_age_equal(x, ...)
```

Arguments

<code>x</code>	A data frame of occurrence records.
<code>...</code>	Further arguments to flag_occurrences() : <code>columns</code> , <code>reference</code> , <code>gazetteers</code> , <code>land_scale</code> .

Details

The width of the range (`max_age - min_age`) is a numeric field, not a test: it stays a raw column and is cut at tally time with bins, as the record year is.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

`min_age`, `max_age` (`min_ma`, `max_ma` by default; see [occ_columns\(\)](#)).

CoordinateCleaner

Equivalent to `cf_equal()`; identical flags. `cf_range(method = "time")` is a threshold on the range width, here a bin in `tally_flags()`. The per-taxon tests `cf_age()`, `cf_outl()` and `cf_range()` with `method = "quantile"` or `"mad"` are `test_age_outlier`, `test_spatiotemporal_outlier` and `test_age_range_outlier`.

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_introduced()`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
x <- data.frame(min_ma = c(10, 20, 30), max_ma = c(20, 20, 25))
flag_occurrences(x, tests = "age_equal")$flag_age_equal
clean_age_equal(x)
```

clean_age_outlier	<i>Test: fossil age outliers</i>
-------------------	----------------------------------

Description

Flags fossil records whose age is far from the ages of the other records of their taxon. One age is drawn uniformly from each record's range (the minimum age when the range has zero width), each record gets its mean absolute age difference to all records of the taxon, and records above the upper fence ($Q75 + \text{mltpl} \times \text{IQR}$, or $\text{median} + \text{mltpl} \times \text{MAD}$) are outliers. The draw is repeated replicates times and a record is flagged when it is an outlier in more than $1 - \text{flag_thresh}$ of them. Taxa with fewer than `size_thresh` records are not tested.

Usage

```
clean_age_outlier(
  x,
  method = "quantile",
  mltpl = 5,
  size_thresh = 7,
  replicates = 5,
  flag_thresh = 0.5,
  uniq_loc = FALSE,
  seed = 1L,
  ...
)
```

Arguments

`x` A data frame of occurrence records.

`method`, `mltpl`, `size_thresh`, `replicates`, `flag_thresh`, `uniq_loc`, `seed` Options of the test, as in `Settings`; the defaults are in the usage line.

`...` Further arguments to `flag_occurrences()`: `columns`, `reference`, `gazetteers`, `land_scale`.

Details

The draws are seeded per taxon from `seed` and the taxon name, and taken in an order fixed by the records' ages, coordinates and identifier, so the flags are reproducible and do not depend on the order or the splitting of the input. The caller's random-number stream is left unchanged.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See `keep_unflagged()`, `Cleaning functions`.

Fields

`min_age`, `max_age`, `species` (the taxon), and `lon`, `lat` for `uniq_loc`.

Settings

`settings = list(age_outlier = list(...))` with `method("quantile")`, `mltpl(5)`, `size_thresh(7)`, `replicates(5)`, `flag_thresh(0.5)`, `uniq_loc(FALSE)`: when `TRUE`, records sharing taxon, coordinates rounded to 0.1 degrees and ages are tested once) and `seed(1)`.

CoordinateCleaner

Equivalent to `cf_age()` at taxon level: the same draw, replicates and majority rule. The flags are identical when no age has to be drawn (every range of zero width). Otherwise the flags of a single run can differ record by record, because **CoordinateCleaner** draws from the session's random stream in input order and this test from a stream seeded per taxon.

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_equal()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_introduced()`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
x <- data.frame(accepted_name = "A", min_ma = c(rep(20, 20), 90), max_ma = c(rep(22, 20), 92))
flag_occurrences(x, tests = "age_outlier",
                  columns = occ_columns(species = "accepted_name"))$flag_age_outlier
clean_age_outlier(x, columns = occ_columns(species = "accepted_name"))
```

clean_age_range_outlier

Test: fossil age-range outliers

Description

Flags fossil records whose age range ($\text{max_age} - \text{min_age}$) lies above the upper fence of the ranges of the other records of their taxon ($Q75 + \text{mltpl} \times \text{IQR}$, or $\text{median} + \text{mltpl} \times \text{MAD}$). A record dated only to a long period among records dated to stages is the typical case. With `level = "all"` the fence is taken over the whole input instead of per taxon, as in **CoordinateCleaner**'s dataset-level `cf_range(taxon = "")`. A fixed limit in million years needs no test: the range stays a raw column and is cut in `tally_flags()`.

Usage

```
clean_age_range_outlier(
  x,
  method = "quantile",
  mltpl = 5,
  size_thresh = 7,
  uniq_loc = FALSE,
  level = "taxon",
  ...
)
```

Arguments

<code>x</code>	A data frame of occurrence records.
<code>method</code> , <code>mltpl</code> , <code>size_thresh</code> , <code>uniq_loc</code> , <code>level</code>	Options of the test, as in Settings; the defaults are in the usage line.
<code>...</code>	Further arguments to <code>flag_occurrences()</code> : <code>columns</code> , <code>reference</code> , <code>gazetteers</code> , <code>land_scale</code> .

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See `keep_unflagged()`, Cleaning functions.

Fields

`min_age`, `max_age`, `species`.

Settings

```
settings = list(age_range_outlier = list(...)) with method("quantile"), mltpl(5), size_thresh(7), uniq_loc (FALSE) and level ("taxon").
```

CoordinateCleaner

Equivalent to `cf_range()` with `method = "quantile"` or `"mad"`; identical flags. With `uniq_loc = TRUE`, `CoordinateCleaner` also flags records of other taxa that happen to share a flagged range width; here the flag goes to the records with the same taxon, coordinates and ages.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#), [test_age_equal](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(accepted_name = "A", min_ma = 20, max_ma = c(rep(21, 20), 60))
flag_occurrences(x, tests = "age_range_outlier",
                  columns = occ_columns(species = "accepted_name"))$flag_age_range_outlier
clean_age_range_outlier(x, columns = occ_columns(species = "accepted_name"))
```

clean_aohi

Test: artificial hotspots

Description

Flags records within 10 km of a point of the Artificial Hotspot Occurrence Inventory (AHOI; Park et al. 2023): the coordinates that recur most often in GBIF plant, bird, insect and mammal records, each with the reason it recurs (a georeferencing default, a gazetteer centroid, a popular observation site). WGS84 geodesic distance, closed ball.

Usage

```
clean_aohi(x, buffer = 10000, ...)
```

Arguments

x	A data frame of occurrence records.
buffer	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Details

The shipped inventory is built from the original deposit (Dryad, doi:10.5061/dryad.v41ns1s0p, CC0); [reference_points\(\)](#) returns it with the taxon group, the record count at each point and whether the point marks a real observation site (determination), so it can be filtered and passed back through [gazetteers](#).

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat.

Buffer

`buffers = list(aohi = 10000)`, in metres.

Custom gazetteer

`gazetteers = list(aohi = points)`, a data frame of longitude and latitude; see [flag_occurrences\(\)](#).

CoordinateCleaner

Equivalent to `cc_aohi()` with its default four taxa; with `reference = "CoordinateCleaner"` its bundled table is used. Its 72-gon buffer edge lies within 9.5 m of the 10 km circle. Its table differs from the Dryad deposit at a few points.

References

Park, D. S., Xie, Y., Thammavong, H. T., Tulaiha, R. and Feng, X. (2023). Artificial Hotspot Occurrence Inventory (AHOI). Journal of Biogeography 50:441-449. doi:10.1111/jbi.14543

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#), [reference_points\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
a <- reference_points("aohi")
x <- data.frame(decimalLongitude = a$lon[1] + 0.01, decimalLatitude = a$lat[1])
flag_occurrences(x, tests = "aohi")$flag_aohi
clean_aohi(x)
```

clean_basis_fossil	<i>Tests: basis of record</i>
--------------------	-------------------------------

Description

basis_fossil flags records with basisOfRecord FOSSIL_SPECIMEN, basis_living those with LIVING_SPECIMEN (botanic garden, zoo and culture-collection individuals). Matching ignores case and surrounding spaces.

Usage

```
clean_basis_fossil(x, ...)
```

```
clean_basis_living(x, ...)
```

Arguments

x	A data frame of occurrence records.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Value

clean_<test>(): the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

basis.

CoordinateCleaner

No equivalent; additions of this package. **CoordinateCleaner**'s fossil tests (cf_*) test fossil ages and ranges, which this package does not.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(basisOfRecord = c("HUMAN_OBSERVATION", "LIVING_SPECIMEN", "FOSSIL_SPECIMEN"))
f <- flag_occurrences(x, tests = c("basis_fossil", "basis_living"))
f[, c("flag_basis_fossil", "flag_basis_living")]
clean_basis_living(x)
```

clean_capital	<i>Test: country capitals</i>
---------------	-------------------------------

Description

Flags records within 10 km of a country capital, measured as WGS84 geodesic distance (closed ball: a record at exactly the buffer distance is flagged). Candidates are found with a kd-tree on the unit sphere and decided with the Vincenty inverse formula.

Usage

```
clean_capital(x, buffer = 10000, ...)
```

Arguments

x	A data frame of occurrence records.
buffer	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Details

The default gazetteer contains the capital "geographic coordinates" of the CIA World Factbook (public domain; 232 points, one per country profile, through the factbook.json mirror retrieved 2026-10-11), the source **CoordinateCleaner**'s capitals are taken from. The column extension marks the 243 Natural Earth populated places of the classes admin-0 capital, alternative capital and admin-0 region capital (dependent territories), which add the second capitals of countries that have several (Cape Town, Sucre, Naypyidaw); passing the points with extension FALSE as gazetteers = list(capital = ...) leaves them out. See [reference_info\(\)](#).

Value

clean_<test>(): the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat.

Buffer

```
buffers = list(capital = 10000), in metres.
```

CoordinateCleaner

Equivalent to `cc_cap()`. `CoordinateCleaner` draws the buffer as a 72-gon with vertices on the circle, whose edge lies at 99.9 % of the buffer between vertices; records in that band (within 9.5 m of the 10 km edge) are flagged here and may not be there. Of its 225 capital points, 221 are Factbook points of the default gazetteer rounded to 0.01 degree, which moves them by up to 523 m. The other four hold values the Factbook has since revised: Bujumbura for Burundi, whose capital it now gives as Gitega; Buenos Aires at 58 40 W, 27 km west of the city, now 58 22 W; Melekeok for Palau, now Ngerulmud; and Rangoon, 1.8 km from its present coordinates. With `reference = "CoordinateCleaner"` its countryref capitals are used.

Custom gazetteer

```
gazetteers = list(capital = points), a data frame of longitude and latitude; see flag\_occurrences\(\).
```

See Also

```
flag\_occurrences\(\), flag\_tests\(\), reference\_info\(\)
```

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(decimalLongitude = c(16.37, 16.51), decimalLatitude = c(48.21, 48.25))
flag_occurrences(x, tests = "capital")$flag_capital
flag_occurrences(x, tests = "capital", buffers = list(capital = 20000))$flag_capital
g <- reference_points("capital")
flag_occurrences(x, tests = "capital",
                  gazetteers = list(capital = g[!g$extension, ]))$flag_capital
clean_capital(x, buffer = 20000)
```

clean_centroid

Test: country and province centroids

Description

Flags records within 1 km of a country or province centroid (WGS84 geodesic distance, closed ball), where records georeferenced to a country or province name end up. The default gazetteer contains the country points of the three sources **CoordinateCleaner**'s table is compiled from, each taken from the source: the "Geographic coordinates" of the CIA World Factbook (public domain; 275 points,

through the factbook.json mirror retrieved 2026-10-11), the label points of the thematicmapping.org World Borders Dataset 0.3 (Bjorn Sandvik, CC BY-SA; 246 points), and every point the GEOLocate georeferencing service (geo-locate.org) returns for a country name, queried 2026-10-11 (548 points; a place of that name more than 100 km away inside another country is left out). Province centroids are the Natural Earth 5.1.1 admin-1 label points (4,594), the current release of the layer CoordinateCleaner's province centroids come from. The column extension marks the 516 Natural Earth country polygon centroids and label points, which CoordinateCleaner's table does not hold; passing the points with extension FALSE as `gazetteers = list(centroid = ...)` leaves them out.

Usage

```
clean_centroid(x, buffer = 1000, ...)
```

Arguments

<code>x</code>	A data frame of occurrence records.
<code>buffer</code>	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
<code>...</code>	Further arguments to <code>flag_occurrences()</code> : <code>columns</code> , <code>reference</code> , <code>gazetteers</code> , <code>land_scale</code> .

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See `keep_unflagged()`, Cleaning functions.

Fields

lon, lat.

Buffer

`buffers = list(centroid = 1000)`, in metres.

CoordinateCleaner

Equivalent to `cc_cen()` with `test = "both"`. Its 72-gon buffer edge lies within 0.95 m of the 1 km circle. Of its 660 country centroids, 645 are points of the default gazetteer at distance 0. Of the other 15, India, Italy and the United Kingdom hold World Borders values that version 0.3 of the dataset changed; three GEOLocate points for Macedonia and one for Guinea, at sea at 10 W 5 N, are not returned by GEOLocate today; eight (Aland, Caribbean Netherlands, DR Congo, Kuwait, Martinique, Monaco, Saint Martin, San Marino) lie 5 m to 68 km from a default point. Its province centroids are the admin-1 label points of older Natural Earth releases (4,566 of 4,645 rows match release 3.3.0, 4.0.0 or 4.1.0 exactly); 4,591 are default points at distance 0, and 50 lie more than 10 km from any, where Natural Earth has since changed the unit or the row matches no tagged release.

Custom gazetteer

`gazetteers = list(centroid = points)`, a data frame of longitude and latitude; see `flag_occurrences()`.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#), [reference_info\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(decimalLongitude = c(14.1307, 10), decimalLatitude = c(47.5857, 50))
flag_occurrences(x, tests = "centroid")$flag_centroid
g <- reference_points("centroid")
flag_occurrences(x, tests = "centroid",
                  gazetteers = list(centroid = g[!g$extension, ]))$flag_centroid
clean_centroid(x)
```

clean_country_mismatch

Test: country code mismatch

Description

Flags records whose coordinates fall outside the Natural Earth 1:50 million country polygon named by their country code. The code is compared with the polygon's ISO 3166-1 alpha-2 code, its alpha-3 code and the Natural Earth ADM0_A3 code, case-insensitively. Records without a code, and records at sea, are flagged.

Usage

```
clean_country_mismatch(x, ...)
```

Arguments

x	A data frame of occurrence records.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Value

`clean_<test>()`: the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat, country.

CoordinateCleaner

Equivalent to `cc_coun()`, which compares alpha-3 codes only; for alpha-3 codes present in Natural Earth the flags are identical. GBIF's `countryCode` is alpha-2 and works here without conversion.

Custom gazetteer

`gazetteers = list(country_mismatch = polygons)`, an `sf` or `sfc` object of country polygons with a code column; see [flag_occurrences\(\)](#).

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#), [occ_columns\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(decimalLongitude = 16.37, decimalLatitude = 48.21,
                 countryCode = c("AT", "AUT", "DE", NA))
flag_occurrences(x, tests = "country_mismatch")$flag_country_mismatch
clean_country_mismatch(x)
```

<code>clean_dataset_ddmm</code>	<i>Test: degree-minute conversion errors in a dataset</i>
---------------------------------	---

Description

Flags every record of a dataset whose coordinates look converted from degrees and minutes as if they were decimal degrees, which leaves no decimal part above .59. The decimal parts of each dataset's distinct coordinates are placed on a `mat_size` x `mat_size` grid; a binomial test compares the share of filled cells with both decimals below .6 with its expected share, and the dataset fails when the excess is significant ($p < pvalue$) and larger than `diff` times the expected share. Datasets spanning less than `min_span` degrees in longitude or latitude are not tested.

Usage

```
clean_dataset_ddmm(
  x,
  pvalue = 0.025,
  diff = 1,
  mat_size = 1000,
  min_span = 2,
  ...
)
```

Arguments

`x` A data frame of occurrence records.

`pvalue`, `diff`, `mat_size`, `min_span` Options of the test, as in Settings; the defaults are in the usage line.

`...` Further arguments to `flag_occurrences()`: `columns`, `reference`, `gazetteers`, `land_scale`.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See `keep_unflagged()`, Cleaning functions.

Fields

`lon`, `lat`, `dataset` (`datasetKey` by default).

Settings

`settings = list(dataset_ddmm = list(...))` with `pvalue` (0.025), `diff` (1), `mat_size` (1000) and `min_span` (2).

CoordinateCleaner

Equivalent to `cd_ddmm()`; identical flags. Records without a dataset key are not flagged here; `cd_ddmm()` flags them.

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_introduced()`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
set.seed(2)
ddmm <- function(n) floor(runif(n, 0, 20)) + sample(0:59, n, TRUE) / 100
x <- data.frame(datasetKey = rep(c("ok", "ddmm"), each = 300),
               decimalLongitude = c(runif(300, 0, 20), ddmm(300)),
               decimalLatitude = c(runif(300, 0, 20), ddmm(300)))
tapply(flag_occurrences(x, tests = "dataset_ddmm")$flag_dataset_ddmm, x$datasetKey, all)
table(clean_dataset_ddmm(x)$datasetKey)
```

clean_dataset_rounded *Test: rasterised datasets*

Description

Flags every record of a dataset whose coordinates lie on a regular grid, as when records were collected or published per grid cell. For each axis the absolute coordinates are binned into about 3000 classes, the autocorrelation of the bin counts is computed, and its outliers (values above $Q75 + T1 \times IQR$ in sliding windows) are located; regularly spaced outliers, at a common distance between `reg_dist_min` and `reg_dist_max` degrees found at least `reg_out_thresh` times, mark the axis as periodic. A dataset fails when both axes are periodic (`axis = "both"`) or the chosen one is. Datasets with fewer than `min_unique_ds_size` distinct coordinates are not tested.

Usage

```
clean_dataset_rounded(
  x,
  T1 = 7,
  reg_out_thresh = 2,
  reg_dist_min = 0.1,
  reg_dist_max = 2,
  min_unique_ds_size = 4,
  axis = "both",
  ...
)
```

Arguments

<code>x</code>	A data frame of occurrence records.
<code>T1</code> , <code>reg_out_thresh</code> , <code>reg_dist_min</code> , <code>reg_dist_max</code> , <code>min_unique_ds_size</code> , <code>axis</code>	Options of the test, as in Settings; the defaults are in the usage line.
<code>...</code>	Further arguments to flag_occurrences() : <code>columns</code> , <code>reference</code> , <code>gazetteers</code> , <code>land_scale</code> .

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

`lon`, `lat`, `dataset`.

Settings

`settings = list(dataset_rounded = list(...))` with `T1` (7), `reg_out_thresh` (2), `reg_dist_min` (0.1), `reg_dist_max` (2), `min_unique_ds_size` (4) and `axis` ("both").

CoordinateCleaner

Equivalent to `cd_round()`; identical flags. In **CoordinateCleaner** 3.0.1 an input with several datasets of which one is below `min_unique_ds_size` stops with an error; here that dataset is not tested. Records without a dataset key are not flagged.

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_introduced()`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
set.seed(2)
x <- data.frame(datasetKey = rep(c("ok", "grid"), each = 400),
  decimalLongitude = c(runif(400, 0, 20), round(runif(400, 0, 20) * 2) / 2 + 0.01),
  decimalLatitude = c(runif(400, 0, 20), round(runif(400, 30, 50) * 2) / 2 + 0.01))
tapply(flag_occurrences(x, tests = "dataset_rounded")$flag_dataset_rounded, x$datasetKey, all)
table(clean_dataset_rounded(x)$datasetKey)
```

clean_duplicate	<i>Test: duplicates</i>
-----------------	-------------------------

Description

Flags every record with the same longitude, latitude and species as an earlier record; the first of each set is not flagged. `duplicate_by` adds columns that must also match; `duplicate_cell` bins coordinates to a grid of that many degrees first. A missing species counts as one value, as in `duplicated()`, so records without a species at the same coordinates are duplicates of each other.

Usage

```
clean_duplicate(x, by = NULL, cell = NULL, ...)
```

Arguments

- `x` A data frame of occurrence records.
- `by` Additional column names a duplicate must share, as `duplicate_by` in `flag_occurrences()`.
- `cell` Cell size in degrees for comparing coordinates, as `duplicate_cell` in `flag_occurrences()`.
- `...` Further arguments to `flag_occurrences()`: `columns`, `reference`, `gazetteers`, `land_scale`.

Details

The test compares each record with all records before it in the input order: the rows of a data frame, or on streamed input the files in sorted order and the rows within each file. Streamed input therefore gets the same flags as the same records in one data frame. It is a grouped test: the records are bucketed by a 124-bit hash of their key in a pass before the record tests, so it needs a path rather than a single-run vectra_node. The probability that any two different keys share a hash is about 2e-18 for ten billion records.

Value

clean_<test>(): the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat, species, and any duplicate_by columns.

CoordinateCleaner

Equivalent to cc_dupl(), with duplicate_by as its additions. Records with invalid coordinates are not flagged here; cc_dupl() flags the second of two such records.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#)
Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_equal\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(species = "a", decimalLongitude = c(16.37, 16.37, 16.3701),
               decimalLatitude = 48.21, eventDate = c("2020", "2021", "2020"))
flag_occurrences(x, tests = "duplicate")$flag_duplicate
flag_occurrences(x, tests = "duplicate", duplicate_by = "eventDate")$flag_duplicate
flag_occurrences(x, tests = "duplicate", duplicate_cell = 0.01)$flag_duplicate
clean_duplicate(x, by = "eventDate")
```

clean_equal	<i>Test: equal coordinates</i>
-------------	--------------------------------

Description

Flags records whose absolute longitude equals their absolute latitude, the result of one value being entered in both fields.

Usage

```
clean_equal(x, ...)
```

Arguments

x A data frame of occurrence records.

... Further arguments to [flag_occurrences\(\)](#): columns, reference, gazetteers, land_scale.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See [keep_unflagged\(\)](#), [Cleaning functions](#).

Fields

lon, lat.

CoordinateCleaner

Equivalent to `cc_equ()` with its default `test = "absolute"`; identical flags.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_establishment_introduced\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(decimalLongitude = c(10, -10, 10.0001), decimalLatitude = 10)
flag_occurrences(x, tests = "equal")$flag_equal
clean_equal(x)
```

clean_establishment_introduced

Tests: establishment means

Description

`establishment_introduced` flags records with `establishmentMeans` introduced, naturalised (either spelling) or invasive; `establishment_managed` flags managed or cultivated individuals. Matching ignores case and surrounding spaces.

Usage

```
clean_establishment_introduced(x, ...)
```

```
clean_establishment_managed(x, ...)
```

Arguments

x A data frame of occurrence records.

... Further arguments to [flag_occurrences\(\)](#): columns, reference, gazetteers, land_scale.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

establishment.

CoordinateCleaner

No equivalent; additions of this package.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(establishmentMeans = c(NA, "MANAGED", "native", "Naturalised"))
f <- flag_occurrences(x, tests = c("establishment_introduced", "establishment_managed"))
f[, c("flag_establishment_introduced", "flag_establishment_managed")]
x |> clean_establishment_introduced() |> clean_establishment_managed()
```

clean_gbif_headquarters

Test: GBIF headquarters

Description

Flags records within 1 km of the GBIF secretariat in Copenhagen (12.58 E, 55.67 N), a location some records receive as a default.

Usage

```
clean_gbif_headquarters(x, buffer = 1000, ...)
```

Arguments

x	A data frame of occurrence records.
buffer	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Value

clean_<test>(): the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat.

Buffer

buffers = list(gbif_headquarters = 1000), in metres.

CoordinateCleaner

Equivalent to cc_gbif(); same point, 72-gon buffer within 0.95 m of the 1 km circle. The point lies outside the 1:50 million Natural Earth coast, so it also fails sea at land_scale = 50 and country_mismatch with code "DK", in both packages.

Custom gazetteer

gazetteers = list(gbif_headquarters = points), a data frame of longitude and latitude; see [flag_occurrences\(\)](#).

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_int()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
x <- data.frame(decimalLongitude = c(12.58, 12.6), decimalLatitude = c(55.67, 55.7))
flag_occurrences(x, tests = "gbif_headquarters")$flag_gbif_headquarters
clean_gbif_headquarters(x)
```

clean_institution	<i>Test: biodiversity institutions</i>
-------------------	--

Description

Flags records within 100 m of a biodiversity institution (WGS84 geodesic distance, closed ball): herbaria, museums, botanic gardens, zoos and universities holding collections, where specimens are often georeferenced to the institution. The default gazetteer contains 23,895 points: the 6,080 institutions with coordinates in GBIF's Global Registry of Scientific Collections (GRSciColl), less five whose coordinates the zero test flags (they lie near (0, 0) or have a longitude of exactly 0, and are geocoding errors); 7,810 points from Wikidata (CC0), retrieved 2026-10-10: every botanical garden, natural history museum, zoo, aquarium and herbarium with coordinates, every museum whose subject is a natural-history field, and the museums of Wikipedia's lists of natural history museums (English, Arabic, Dutch, Esperanto, French, German and Portuguese editions) that have their own Wikidata coordinates; 700 further museums of those lists placed at the place the list gives for them, by its Wikidata coordinates or the GeoNames populated place of that name (source "Wikipedia", master_source says which); the three GeoNames places (CC BY 4.0) of **CoordinateCleaner**'s table; 3,652 gardens of Botanic Gardens Conservation International (BGCI) GardenSearch, retrieved 2026-10-10, used under the BGCI terms (<https://www.bgci.org/terms-and-conditions/>); 1,062 herbaria of Index Herbariorum that GRSciColl does not place, and 2,468 publishing organisations of the GBIF registry, both retrieved 2026-10-13; and 2,120 points for the institutions of CoordinateCleaner's table that none of these holds under their name (source "CoordinateCleaner list"), placed from Wikidata, OpenStreetMap (through Nominatim) or the GeoNames populated place of the institution's city, by its name, city and country; master_source says which. The source column of `reference_points()` tells them apart.

Usage

```
clean_institution(x, buffer = 100, ...)
```

Arguments

<code>x</code>	A data frame of occurrence records.
<code>buffer</code>	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
<code>...</code>	Further arguments to <code>flag_occurrences()</code> : <code>columns</code> , <code>reference</code> , <code>gazetteers</code> , <code>land_scale</code> .

Details

Every institution of CoordinateCleaner's table with coordinates has a point here except one, an expert group listed in Antarctica. Where a source gives several candidates for an institution, the one nearest CoordinateCleaner's point is taken; the coordinates are always the source's. For 496 institutions no source places the institution itself, and the point is the populated place of its country nearest CoordinateCleaner's point.

The exception is "SCAR - Expert Group on Birds and Marine Mammals". CoordinateCleaner places it at 45.1439 N, 0.5558 E. Its entry in the GBIF registry was deleted on 2021-05-31 and has no coordinates. It has no point here because no independently verified physical location of the group could be established.

The column extension marks the 7,665 points from sources CoordinateCleaner's table does not draw on: Wikidata beyond the museums of the English list, and the lists of the other language editions. The other 16,230 points come from the registries CoordinateCleaner's table is compiled from (GRSciColl, which absorbed GRBio and mirrors Index Herbariorum, Index Herbariorum itself, the GBIF registry, the English list, GeoNames and BGCI) and from its own list of institutions. Passing the points with extension FALSE as `gazetteers = list(institution = ...)` restricts the test to those sources (see the examples).

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See `keep_unflagged()`, Cleaning functions.

Fields

`lon`, `lat`.

Buffer

`buffers = list(institution = 100)`, in metres.

CoordinateCleaner

Equivalent to `cc_inst()`. Its 40-gon buffer edge lies within 0.31 m of the 100 m circle. Its institutions table (9,773 points with coordinates) places nearly every institution by its own geocoding, so its points can lie apart from the source coordinates used here: for 32 percent of its points a point of the default gazetteer lies within 100 m, for 73 percent within 1 km and for 91 percent within 5 km (28, 70 and 90 percent for the points with extension FALSE). `reference = "CoordinateCleaner"` uses that table, and `rbind()` of `reference_points()` under both references, passed as `gazetteers = list(institution = ...)`, flags records near either.

Custom gazetteer

gazetteers = list(institution = points), a data frame of longitude and latitude; see [flag_occurrences\(\)](#).

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#), [reference_info\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_int\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(decimalLongitude = c(16.3598, 15), decimalLatitude = c(48.2052, 47))
flag_occurrences(x, tests = "institution")$flag_institution
g <- reference_points("institution")
flag_occurrences(x, tests = "institution",
                  gazetteers = list(institution = g[!g$extension, ]))$flag_institution
clean_institution(x)
```

clean_invalid	<i>Test: invalid coordinates</i>
---------------	----------------------------------

Description

Flags records whose longitude or latitude is missing, not a number, or outside [-180, 180] x [-90, 90]. Text columns are coerced with `as.numeric(as.character())` first. A record failing this test gets no other spatial flag.

Usage

```
clean_invalid(x, ...)
```

Arguments

- x A data frame of occurrence records.
- ... Further arguments to [flag_occurrences\(\)](#): columns, reference, gazetteers, land_scale.

Value

`clean_<test>()`: the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat (see `occ_columns()`).

CoordinateCleaner

Equivalent to `cc_val()`. Its other spatial tests return NA or "clean" for invalid rows; here those rows carry only the invalid bit.

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_int`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
x <- data.frame(decimalLongitude = c("16.37", "abc", NA, "200"),
                decimalLatitude = c("48.21", "10", "10", "10"))
flag_occurrences(x, tests = "invalid")$flag_invalid
clean_invalid(x)
```

clean_issue_coordinate
<i>Tests: GBIF issue flags</i>

Description

Four tests read the issue names GBIF attaches to a record during processing:

- `issue_coordinate`: `COORDINATE_INVALID`, `COORDINATE_OUT_OF_RANGE`, `COORDINATE_REPROJECTION_FAILED`, `COORDINATE_REPROJECTION_SUSPICIOUS`, `COORDINATE_UNCERTAINTY_METERS_INVALID`, `COORDINATE_PRECISION_INVALID`, `GEODETIC_DATUM_INVALID`, `CONTINENT_COORDINATE_MISMATCH`, `COUNTRY_COORDINATE_MISMATCH`, `PRESUMED_SWAPPED_COORDINATE`, `PRESUMED_NEGATED_LONGITUDE`, `PRESUMED_NEGATED_LATITUDE`, `ZERO_COORDINATE`;
- `issue_taxon`: `TAXON_MATCH_FUZZY`, `TAXON_MATCH_HIGHERRANK`, `TAXON_MATCH_NONE`, `TAXON_MATCH_AGGREGATE`;
- `issue_date`: `RECORDED_DATE_INVALID`, `RECORDED_DATE_UNLIKELY`, `RECORDED_DATE_MISMATCH`;
- `issue_rounded`: `COORDINATE_ROUNDED`, which GBIF sets when it rounds a coordinate to five decimals.

Usage

```
clean_issue_coordinate(x, ...)
```

```
clean_issue_taxon(x, ...)
```

```
clean_issue_date(x, ...)
```

```
clean_issue_rounded(x, ...)
```

Arguments

x A data frame of occurrence records.

... Further arguments to [flag_occurrences\(\)](#): columns, reference, gazetteers, land_scale.

Details

The issue column is a string of names separated by ;, ,, | or spaces, or a list column of character vectors; case is ignored.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

issue.

CoordinateCleaner

No equivalent; additions of this package.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_int\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(issue = c("COORDINATE_ROUNDED;GEODETIC_DATUM_ASSUMED_WGS84",
                          "TAXON_MATCH_FUZZY", "PRESUMED_SWAPPED_COORDINATE", ""))
f <- flag_occurrences(x, tests = c("issue_coordinate", "issue_taxon", "issue_rounded"))
f[, c("flag_issue_coordinate", "flag_issue_taxon", "flag_issue_rounded")]
x |> clean_issue_coordinate() |> clean_issue_taxon()
```

clean_outlier

Test: geographic outliers

Description

Flags records far from the other records of their species. For each record the haversine distances (km) to the species' other records are computed, leaving out zero distances (the record itself and records at the same point); a record is flagged when its mean distance lies above the upper fence of the species' mean distances: $Q75 + \text{mltpl} \times \text{IQR}$ (method = "quantile") or median + $\text{mltpl} \times \text{MAD}$ ("mad"). With method = "distance" a record is flagged when its nearest other record is more than tdi km away. Species with fewer than min_occs distinct points are not tested.

Usage

```
clean_outlier(
  x,
  method = "quantile",
  mltpl = 5,
  tdi = 1000,
  min_occs = 7,
  thinning = FALSE,
  thinning_res = 0.5,
  sampling_thresh = 0,
  ...
)
```

Arguments

x A data frame of occurrence records.

method, mltpl, tdi, min_occs, thinning, thinning_res, sampling_thresh Options of the test, as in Settings; the defaults are in the usage line.

... Further arguments to [flag_occurrences\(\)](#): columns, reference, gazetteers, land_scale.

Details

Species with 10,000 records or more, and every species when thinning = TRUE, are tested on a raster approximation: records are counted on a grid of thinning_res degrees spanning all valid records (widened by two cells), distances are taken between cell centres, and the records of the outlier cells are flagged. Without thinning, distances are weighted by the product of the cell counts and a cell's distance to itself counts as $100 \times \text{thinning_res} / 2$ km, as in [CoordinateCleaner](#).

The test needs every record of a species, so it runs in a group pass over the whole input before the record tests; on streamed input that pass reads all files, and a species spread over several files is tested as one.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat, species. Records with invalid coordinates are left out.

Settings

`settings = list(outlier = list(...))` with method ("quantile"), `mltpl` (5), `tdi` (1000 km), `min_occs` (7), `thinning` (FALSE), `thinning_res` (0.5 degrees) and `sampling_thresh` (0).

Sampling correction

With `sampling_thresh` above 0, an outlier stays flagged only in a country whose record density (log records per 100 km²) lies above that quantile of the densities of all countries, so that isolated records in poorly sampled countries are kept. The records per country come from `gazetteers = list(sampling = <data frame>)` with an ISO alpha-2 country column and a records column, for example from `rgbif::occ_count()` or a tally of the input itself; country areas are computed from the country polygons of the reference.

CoordinateCleaner

Equivalent to `cc_out1()`; identical flags for species below 10,000 records. In `CoordinateCleaner` 3.0.1 the raster approximation (10,000 records or more, or `thinning = TRUE`) flags no records; here the records of the outlier cells are flagged. `CoordinateCleaner`'s sampling correction queries GBIF record counts online at run time; here the counts are a table given with the call, so the flags can be reproduced.

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_int\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_urban\(\)](#), [clean_zero\(\)](#)

Examples

```
set.seed(1)
x <- data.frame(species = "a", decimalLongitude = c(rnorm(50, 10, 1), 60),
               decimalLatitude = c(rnorm(50, 50, 1), 10))
which(flag_occurrences(x, tests = "outlier")$flag_outlier)
nrow(clean_outlier(x, mltpl = 3))
```

clean_range	<i>Test: outside the species range</i>
-------------	--

Description

Flags records that fall outside every range polygon of their species, for example an IUCN range map. The polygons come from `gazetteers = list(range = <sf>)`: an sf object with a species column (named like the record's species column, species, binomial or sci_name). Species without a polygon are not tested. With `buffers = list(range = m)`, records within m metres of a polygon edge count as inside.

Usage

```
clean_range(x, ranges, buffer = 0, ...)
```

Arguments

x	A data frame of occurrence records.
ranges	Species range polygons, as <code>gazetteers = list(range = ...)</code> in flag_occurrences() .
buffer	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Value

`clean_<test>()`: the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat, species.

Buffer

`buffers = list(range = 0)`, in metres. The distance to an edge is measured on the ellipsoid from the closest point of the edge in a local equirectangular projection.

Custom gazetteer

required; there is no shipped range layer.

CoordinateCleaner

Equivalent to `cc_iucn()`; identical flags without a buffer. With a buffer **CoordinateCleaner** buffers the polygons with **terra**, so records very close to the buffer edge may differ.

See Also

```
flag_occurrences(), flag_tests()

Other tests: clean_age_equal(), clean_age_outlier(), clean_age_range_outlier(), clean_aohi(),
clean_basis_fossil(), clean_capital(), clean_centroid(), clean_country_mismatch(),
clean_dataset_ddmm(), clean_dataset_rounded(), clean_duplicate(), clean_equal(), clean_establishment_int,
clean_gbif_headquarters(), clean_institution(), clean_invalid(), clean_issue_coordinate(),
clean_outlier(), clean_sea(), clean_sea_nonmarine(), clean_spatiotemporal_outlier(),
clean_urban(), clean_zero()
```

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  rg <- sf::st_sf(species = "a", geometry = sf::st_sfc(sf::st_polygon(list(
    rbind(c(0, 40), c(20, 40), c(20, 55), c(0, 55), c(0, 40))))), crs = 4326))
  x <- data.frame(species = c("a", "a", "b"), decimalLongitude = c(10, 30, 30),
    decimalLatitude = 50)
  flag_occurrences(x, tests = "range", gazetteers = list(range = rg))$flag_range
  clean_range(x, ranges = rg)
}
```

clean_sea	<i>Test: records at sea</i>
-----------	-----------------------------

Description

Flags records outside the Natural Earth land polygons, by a planar point-in-polygon test in longitude and latitude through a grid index that resolves boundary cells exactly. The land layer is chosen with land_scale: 110 (default), 50, or 10. The 1:10 million layer is downloaded once by [download_reference\(\)](#).

Usage

```
clean_sea(x, ...)
```

Arguments

- x A data frame of occurrence records.
- ... Further arguments to [flag_occurrences\(\)](#): columns, reference, gazetteers, land_scale.

Value

clean_<test>(): the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat.

CoordinateCleaner

Equivalent to `cc_sea()` with its default `scale = 110` (`clean_coordinates()` uses `seas_scale = 50`, which is `land_scale = 50` here); identical flags on the same layer. **CoordinateCleaner** downloads the layer through **rnaturalearth** at run time; here it is bundled and pinned.

Custom gazetteer

`gazetteers = list(sea = polygons)`, an `sf` or `sfc` object of land polygons; see `flag_occurrences()`.

See Also

`flag_occurrences()`, `flag_tests()`, `reference_info()`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_int()`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
x <- data.frame(decimalLongitude = c(12.58, 16.37, -30), decimalLatitude = c(55.67, 48.21, 40))
flag_occurrences(x, tests = "sea")$flag_sea
flag_occurrences(x, tests = "sea", land_scale = 50)$flag_sea
clean_sea(x, land_scale = 50)
```

clean_sea_nonmarine	<i>Tests: habitat at sea and on land</i>
---------------------	--

Description

Two tests set the land test against the habitat of the record's genus:

- `sea_nonmarine` flags records outside the land polygons whose genus does not live in the sea (a tree or a toad in open water);
- `land_marine` flags records inside the land polygons whose genus lives only in the sea (a starfish or a coral inland).

Usage

```
clean_sea_nonmarine(x, buffer = 0, ...)
```

```
clean_land_marine(x, buffer = 0, ...)
```

Arguments

x	A data frame of occurrence records.
buffer	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Details

The genus habitats come from the Interim Register of Marine and Nonmarine Genera (IRMNG; Rees, T. (compiler) (2026). The Interim Register of Marine and Nonmarine Genera. Available from <https://www.irmng.org> at VLIZ; CC BY 4.0), see [reference_habitat\(\)](#) for how a genus is classed. Records of a genus IRMNG does not class (missing flags, or both habitats, as for gulls, seals, salmon or mangroves) are not tested. The genus is the first word of the genus field, so a binomial species column works too; a kingdom column, when present, resolves genus names used in two kingdoms. The land is the layer of sea: land_scale, or gazetteers\$sea.

A coarse coast puts coastal records on the wrong side: at the default land_scale = 110 the coastline is generalised by kilometres. The buffers leave records within that distance of the coast untested.

Marine fossils of a living marine genus lie on land too; combine with basis_fossil in a rule to keep them.

Value

clean_<test>(): the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat, genus; kingdom when present.

Buffer

buffers = list(sea_nonmarine = 0, land_marine = 0), in metres from the land polygon edge.

Custom gazetteer

gazetteers = list(habitat = <data frame>) with columns genus, marine (logical: TRUE only marine, FALSE not marine) and optionally kingdom; gazetteers = list(sea = <sf>) replaces the land.

CoordinateCleaner

Not in [CoordinateCleaner](#), whose cc_sea() flags every record off the land, marine or not.

See Also

`flag_occurrences()`, `flag_tests()`, `reference_habitat()`, `test_sea`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_int()`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_spatiotemporal_outlier()`, `clean_urban()`, `clean_zero()`

Examples

```
x <- data.frame(kingdom = c("Plantae", "Animalia", "Animalia", "Plantae"),
               genus = c("Quercus", "Asterias", "Asterias", "Quercus"),
               decimalLongitude = c(-30, 16.37, -30, 16.37),
               decimalLatitude = c(40, 48.21, 40, 48.21))
f <- flag_occurrences(x, tests = c("sea", "sea_nonmarine", "land_marine"))
f[, c("genus", "flag_sea", "flag_sea_nonmarine", "flag_land_marine")]
clean_sea_nonmarine(x)
```

`clean_spatiotemporal_outlier`

Test: fossil outliers in space and time

Description

Flags fossil records that are far from the other records of their taxon in space and time together. Ages are drawn as in [test_age_outlier](#); for each pair of records the age difference is rescaled so that the largest age difference equals the largest geographic distance, added to the haversine distance in km between coordinates rounded to 0.1 degrees (Euclidean degrees for taxa of 10,000 records or more) and rounded; each record's mean over all records of its taxon is compared with the upper fence. Replicates, majority and seeding are as in [test_age_outlier](#).

Usage

```
clean_spatiotemporal_outlier(
  x,
  method = "quantile",
  mltpl = 5,
  size_thresh = 7,
  replicates = 5,
  flag_thresh = 0.5,
  uniq_loc = FALSE,
  seed = 1L,
  ...
)
```

Arguments

`x` A data frame of occurrence records.

`method`, `mltpl`, `size_thresh`, `replicates`, `flag_thresh`, `uniq_loc`, `seed`
Options of the test, as in `Settings`; the defaults are in the usage line.

`...` Further arguments to `flag_occurrences()`: `columns`, `reference`, `gazetteers`, `land_scale`.

Value

`clean_<test>()`: the rows of `x` the test does not flag, with the columns of `x` unchanged. See `keep_unflagged()`, `Cleaning functions`.

Fields

`lon`, `lat`, `min_age`, `max_age`, `species`.

Settings

`settings = list(spatiotemporal_outlier = list(...))` with the same options and defaults as `age_outlier`.

CoordinateCleaner

Equivalent to `cf_outl()` at taxon level, the spatiotemp test of `clean_fossils()`; identical flags when every range has zero width, and otherwise a single run can differ record by record, as for `age_outlier`.

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_int`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_urban()`, `clean_zero()`

Examples

```
set.seed(1)
x <- data.frame(accepted_name = "A", lng = c(runif(20, 0, 5), 120), lat = c(runif(20, 40, 45), -30),
               min_ma = 20, max_ma = 20)
cols <- occ_columns(lon = "lng", lat = "lat", species = "accepted_name")
flag_occurrences(x, tests = "spatiotemporal_outlier", columns = cols)$flag_spatiotemporal_outlier
clean_spatiotemporal_outlier(x, columns = cols)
```

clean_urban

Test: urban areas

Description

Flags records inside the Natural Earth 1:50 million urban-area polygons. `CoordinateCleaner`'s `clean_coordinates()` leaves this test out by default, as urban records are often correct; here it runs whenever coordinates are present, and a cleaning rule can leave it out.

Usage

```
clean_urban(x, ...)
```

Arguments

x	A data frame of occurrence records.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Value

`clean_<test>()`: the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), [Cleaning functions](#).

Fields

lon, lat.

CoordinateCleaner

Equivalent to `cc_urb()` with its default 1:50 million urban areas; identical flags on the same layer. Not run by `clean_coordinates()` by default either.

Custom gazetteer

`gazetteers = list(urban = polygons)`, an sf or sfc object of urban-area polygons; see [flag_occurrences\(\)](#).

See Also

[flag_occurrences\(\)](#), [flag_tests\(\)](#)

Other tests: [clean_age_equal\(\)](#), [clean_age_outlier\(\)](#), [clean_age_range_outlier\(\)](#), [clean_aohi\(\)](#), [clean_basis_fossil\(\)](#), [clean_capital\(\)](#), [clean_centroid\(\)](#), [clean_country_mismatch\(\)](#), [clean_dataset_ddmm\(\)](#), [clean_dataset_rounded\(\)](#), [clean_duplicate\(\)](#), [clean_equal\(\)](#), [clean_establishment_int\(\)](#), [clean_gbif_headquarters\(\)](#), [clean_institution\(\)](#), [clean_invalid\(\)](#), [clean_issue_coordinate\(\)](#), [clean_outlier\(\)](#), [clean_range\(\)](#), [clean_sea\(\)](#), [clean_sea_nonmarine\(\)](#), [clean_spatiotemporal_outlier\(\)](#), [clean_zero\(\)](#)

Examples

```
x <- data.frame(decimalLongitude = c(16.37, 15.3), decimalLatitude = c(48.21, 47.5))
flag_occurrences(x, tests = "urban")$flag_urban
clean_urban(x)
```

clean_zero

Test: zero coordinates

Description

Flags records with a longitude or latitude of exactly zero, and records within a planar distance of 0.5 degrees of (0, 0).

Usage

```
clean_zero(x, buffer = 0.5, ...)
```

Arguments

x	A data frame of occurrence records.
buffer	Buffer distance of the test, in the unit of its Buffer section; the default is the test's default buffer.
...	Further arguments to flag_occurrences() : columns, reference, gazetteers, land_scale.

Value

clean_<test>(): the rows of x the test does not flag, with the columns of x unchanged. See [keep_unflagged\(\)](#), Cleaning functions.

Fields

lon, lat.

Buffer

buffers = list(zero = 0.5), in degrees. zero = 0 flags exact zeros only.

CoordinateCleaner

Equivalent to cc_zero(). **CoordinateCleaner** 3.0.1 documents the same 0.5 degree buffer but applies a radius of 0.5 metres, so it flags only exact zeros. buffers = list(zero = 0) gives the same flags.

See Also

`flag_occurrences()`, `flag_tests()`

Other tests: `clean_age_equal()`, `clean_age_outlier()`, `clean_age_range_outlier()`, `clean_aohi()`, `clean_basis_fossil()`, `clean_capital()`, `clean_centroid()`, `clean_country_mismatch()`, `clean_dataset_ddmm()`, `clean_dataset_rounded()`, `clean_duplicate()`, `clean_equal()`, `clean_establishment_int()`, `clean_gbif_headquarters()`, `clean_institution()`, `clean_invalid()`, `clean_issue_coordinate()`, `clean_outlier()`, `clean_range()`, `clean_sea()`, `clean_sea_nonmarine()`, `clean_spatiotemporal_outlier()`, `clean_urban()`

Examples

```
x <- data.frame(decimalLongitude = c(0, 0.3, 0.4), decimalLatitude = c(42, 0.3, 0.4))
flag_occurrences(x, tests = "zero")$flag_zero
flag_occurrences(x, tests = "zero", buffers = list(zero = 0))$flag_zero
clean_zero(x, buffer = 0)
```

download_reference	<i>Download the 1:10 million land layer</i>
--------------------	---

Description

The 1:10 million Natural Earth land polygons (2.3 MB compressed) are not bundled with the package. This fetches the pinned zip from Natural Earth, verifies its SHA-256 against [reference_info\(\)](#), converts it and stores the result in the user cache directory (`tools::R_user_dir("OccurrenceCleaner", "cache")`). `flag_occurrences()` calls it when `land_scale = 10` is requested and the layer is not cached yet.

Usage

```
download_reference(
  layer = "land_10",
  cache_dir = tools::R_user_dir("OccurrenceCleaner", "cache"),
  quiet = FALSE
)
```

Arguments

<code>layer</code>	Layer to download; only "land_10" is not bundled.
<code>cache_dir</code>	Directory for the converted layer.
<code>quiet</code>	Suppress the download message.

Value

The path of the cached layer, invisibly.

Examples

```
if (interactive()) {
  download_reference()
  x <- data.frame(decimalLongitude = c(12.58, -30), decimalLatitude = c(55.67, 40))
  flag_occurrences(x, tests = "sea", land_scale = 10)$flag_sea
}
```

export_pyrate

Export fossil records as PyRate input

Description

Writes fossil occurrences in the input format of PyRate (Silvestro et al. 2014), the Bayesian estimator of speciation and extinction rates from fossil occurrences: a Python file holding, per replicate, one array of occurrence ages per taxon, and a tab-separated list of the taxa with their status. An extant taxon gets an extra age of 0. Each record's age is drawn uniformly from its range (`random = TRUE`) or set to the range's midpoint; a minimum age of 0 is set to 0.001.

Usage

```
export_pyrate(
  x,
  file,
  status,
  columns = occ_columns(),
  trait = NULL,
  replicates = 1,
  random = TRUE,
  seed = NULL
)
```

Arguments

<code>x</code>	A data frame of fossil records, for example the kept records of a flagged table.
<code>file</code>	Path prefix of the output: <code><file>_PyRate.py</code> and <code><file>_TaxonList.txt</code> are written.
<code>status</code>	"extinct" or "extant": one value for all records, one per record, or the name of a column of <code>x</code> holding them. A taxon must have a single status.
<code>columns</code>	Field names, see occ_columns() ; the taxon is read from <code>species</code> and the ages from <code>min_age</code> and <code>max_age</code> .
<code>trait</code>	Optional numeric trait, one value per record or the name of a column; the taxon mean is written as a continuous trait.
<code>replicates</code>	Number of age replicates (data sets) in the file.
<code>random</code>	Draw ages from the ranges (<code>TRUE</code>) or use the midpoints. Replicates of midpoint ages are identical.
<code>seed</code>	Seed for the age draws; the caller's random-number stream is left unchanged. <code>NULL</code> draws from the current stream.

Details

The records are exported as given, so the cleaning rule is applied first: `x[!has_flag(x$flags, rule), ]`.

Value

The paths of the two files, invisibly.

References

Silvestro, D., Schnitzler, J., Liow, L. H., Antonelli, A. and Salamin, N. (2014). Bayesian estimation of speciation and extinction from incomplete fossil occurrence data. *Systematic Biology* 63:349-367. doi:10.1093/sysbio/syu006

See Also

[flag_occurrences\(\)](#), [test_age_equal](#)

Examples

```
x <- data.frame(accepted_name = c("Equus caballus", "Equus caballus", "Hipparion sp"),
  min_ma = c(0.5, 1.2, 5), max_ma = c(1, 2.6, 11))
paths <- export_pyrate(x, file.path(tempdir(), "horses"), status = c("extant", "extant", "extinct"),
  columns = occ_columns(species = "accepted_name"), random = FALSE)
cat(readLines(paths[1]), sep = "\n")
```

flag_mask	<i>Bit mask and membership for sets of tests</i>
-----------	--

Description

`flag_mask()` turns test names into the integer with those bits set; `has_flag()` tells which flag words carry at least one of the tests. A cleaning rule is a set of tests, so `has_flag(flags, rule)` marks the records that rule removes.

Usage

```
flag_mask(tests)

has_flag(flags, tests)
```

Arguments

- `tests` Character vector of test names from [flag_tests\(\)](#).
- `flags` Integer flag words, or a data frame returned by [flag_occurrences\(\)](#), whose `flags` column is then used.

Value

flag_mask(): a single integer. has_flag(): a logical vector, one element per flag word (per row of a data frame); NA flags give NA, so subset with which() or %in% TRUE when flags can be missing. For a flagged data frame, a test that was not run is an error, so a rule never silently removes nothing for a test it names.

Examples

```
flag_mask(c("capital", "centroid"))
has_flag(c(0L, 8L, 16L, 24L), c("capital", "centroid"))
x <- data.frame(decimalLongitude = c(16.37, 0), decimalLatitude = c(48.21, 0))
f <- flag_occurrences(x, tests = c("zero", "capital"))
f[!has_flag(f, "zero"), ]
```

flag_occurrences	<i>Flag occurrence records</i>
------------------	--------------------------------

Description

Runs record tests over species occurrence records and writes one flag per record per test. Nothing is removed and no threshold is baked in.

Usage

```
flag_occurrences(
  x,
  tests = NULL,
  columns = occ_columns(),
  reference = c("OccurrenceCleaner", "CoordinateCleaner"),
  gazetteers = NULL,
  land_scale = 110,
  buffers = NULL,
  settings = NULL,
  duplicate_by = NULL,
  duplicate_cell = NULL,
  out = NULL,
  keep = NULL,
  part_rows = 2e+06,
  resume = TRUE,
  verbose = TRUE,
  n_threads = 1L
)
```

Arguments

x A data frame, a vectra_node, or a path (file or directory).

tests	Character vector of test names, see flag_tests() . NULL (the default) runs every test whose columns are in x (and range when gazetteers has range polygons); see test_plan() . A named test whose columns are missing from x is an error.
columns	Field names, see occ_columns() .
reference	"OccurrenceCleaner" (own gazetteers, the default) or "CoordinateCleaner" (that package's tables, read at run time). See reference_info() .
gazetteers	Named list of user-supplied reference layers, each replacing the reference layer of one test. Point tests (capital, centroid, institution, gbif_headquarters) take a data frame with longitude and latitude columns (named lon/lat, longitude/latitude or decimalLongitude/decimalLatitude, any case). Polygon tests (sea, urban, country_mismatch) take an sf or sfc object of POLYGON or MULTIPOLYGON geometries in longitude and latitude; for sea they are the land, and for country_mismatch the features need at least one code column (iso_a2, iso_a3, adm0_a3 or code, any case). Each layer is indexed once per call, and its size and MD5 checksum are recorded in the reference label of the output. range takes species range polygons (see test_range), sampling a table of records per country for the sampling correction of test_outlier , and habitat a table of genus habitats for test_habitat (see reference_habitat()). The land layer given for sea is also the land of the habitat tests.
land_scale	Natural Earth scale of the land layer for sea and the habitat tests: 110 (the default of CoordinateCleaner's cc_sea()); its clean_coordinates() uses seas_scale = 50), 50, or 10 (downloaded on first use, see download_reference()).
buffers	Named list overriding buffer distances in metres for capital (10000), centroid (1000), institution (100), gbif_headquarters (1000), range (0), aohi (10000), sea_nonmarine (0) and land_marine (0), or in degrees for zero (0.5). Unnamed entries keep the defaults.
settings	Named list overriding the options of the grouped tests, one entry per test, each a named list; unnamed options keep their defaults, which are CoordinateCleaner's. See test_outlier , test_age_outlier , test_spatiotemporal_outlier , test_age_range_outlier , test_dataset_ddmm and test_dataset_rounded .
duplicate_by	Additional column names a duplicate must share (as CoordinateCleaner's additions), for example an event date.
duplicate_cell	Cell size in degrees. When given, coordinates are binned to this grid before comparing duplicates, so "same cell and species" counts.
out	Output directory for streamed input (required then, ignored for a data frame).
keep	Column names copied next to the flags in streamed output, for grouping in tally_flags() (taxon, year, uncertainty, ...). The id column of columns is kept when present.
part_rows	Target rows per output part file for streamed input: a part is written once at least this many rows have accumulated, and at the end of each input unit.
resume	Continue a run in out: skip input files whose .done marker exists and resume the group stage from its checkpoint. FALSE deletes the earlier output in out and starts over.
verbose	Report progress for streamed input.
n_threads	Number of threads for the outlier test and the spatial queries. 0 uses every thread OpenMP hands out; a request above that is reduced to it, with a message.

Details

Two choices are kept apart. Which tests run: by default every test whose inputs are available, so a screening sees every potential problem the columns allow; `test_plan()` lists the tests that ran and the reason each other test did not. Which records are excluded: a cleaning rule, a set of tests chosen afterwards and applied with `has_flag()` or compared with `tally_flags()`. A rule for species distribution modelling might exclude fossil and living specimens along with the coordinate problems; a study of fossil ranges would keep the fossils and exclude other flags.

The grouped tests (duplicate, outlier, the three fossil outlier tests and the two dataset tests) compare a record with the other records of its key, species, taxon or dataset; they run in a group stage before the record tests, which on streamed input reads every file once or twice more and keeps its buckets and results in `out/.group`.

The input is either a data frame, or an on-disk table streamed in chunks: a `vecra_node` (from `vecra::tbl_parquet()`, `vecra::tbl_csv()`, `vecra::tbl()`), a path to a Parquet file or directory of Parquet parts (such as a GBIF snapshot), or a path to a `.vtr` or delimited text file. For a directory each file is one resumable unit of work; the return follows the input, see Value.

Value

For a data frame: `x` with an integer column `flags` and one logical column `flag_<test>` per test, class `occ_flags` added in front, and attributes `occ_tests`, `occ_test_plan`, `occ_layout_version`, `occ_reference` and `occ_columns`. `summary()` reports the share of records each test flags, `plot()` draws it or maps the records. For streamed input: an object of class `occ_flagged` describing the output directory, which `tally_flags()` accepts; the parts are `.vtr` files holding the `id` and keep columns and flags, readable with `vecra::tbl()`.

Streamed runs

Before writing any output, a run records its configuration in `out/flags_config.rds`: the tests, columns, keep, reference, buffers, settings, duplicate options, package version, and each input file with its size and modification time. A later call into the same `out` must match the record, or it stops and names what differs; `resume = FALSE` clears the directory instead. Sizes and modification times detect a replaced, grown or touched file; they are not a hash of the contents.

The group stage checkpoints its state after a file or a bucket whenever five minutes have passed, and at the end of each pass, so an interruption repeats at most the work since the last checkpoint. The record pass writes a `.done` marker per input file; an interruption there repeats the file in progress.

A streamed run writes its progress to `out/progress.txt`, one line overwritten on every report: the step, items done and total, seconds elapsed, and the estimated seconds left in the step and in the run, and the share of the run done. The estimate for the steps ahead uses per-record costs measured on a part of the GBIF snapshot, scaled to the speed of the run so far, so it is rough until the slower steps have started. With `verbose`, the same is reported on the console every minute.

Threads

`n_threads` spreads the exact outlier test over species and the proximity and point-in-polygon queries of the record tests over records. Each species and each record is computed on one thread, so the flags do not depend on the number of threads. Reading the input, the other tests and the

bookkeeping of the group stage run on one thread. Without OpenMP support in the compiler every query runs on one thread.

Memory in the record pass is one input chunk, up to part_rows buffered output rows and the group-test hits of the current block of 2^{20} rows. The group stage holds two million bucketed rows before writing them (more above four billion records), the per-species record counts, per-dataset histograms and raster cell counts of large species, the duplicate keys of one bucket (at most four million records on average when the input has more than 256 million), and the records of one species (fewer than 10,000 for the exact outlier test) or one fossil taxon. Its disk use is at most 24 bytes per record for duplicate, plus the coordinates of the records of the outlier and fossil tests.

Tests

See `flag_tests()` for the layout. Tests equivalent to `CoordinateCleaner` use its default buffers: capital 10 km, centroid 1 km, institution 100 m, gbif_headquarters 1 km, zero 0.5 degrees. Proximity tests are closed balls in WGS84 geodesic distance. Records failing invalid are given no spatial flag.

Documented differences to `CoordinateCleaner` 3.0.1, each verified in the package tests:

- `CoordinateCleaner` approximates each buffer circle by an inscribed polygon (72 vertices for capital, centroid, gbif_headquarters; 40 for institution), so a record between 99.9 % (capital, centroid, gbif_headquarters) or 99.7 % (institution) of the buffer and the buffer itself is flagged here and may not be there. Everything `CoordinateCleaner` flags is flagged here.
- `cc_zero`'s 0.5 degree buffer is interpreted by `terra` as 0.5 metres, so `CoordinateCleaner` 3.0.1 flags only exact zeros; this package applies the documented 0.5 degree radius.
- `country_mismatch` compares the record's code with the ISO alpha-2 code, the ISO alpha-3 code and the Natural Earth ADM0_A3 code of the country polygon the point falls in, taking the ISO codes from Natural Earth's ISO_A2_EH and ISO_A3_EH columns, which are filled where ISO_A2 and ISO_A3 hold -99 (France, Norway, Kosovo); `CoordinateCleaner` compares ISO_A3 only, so it flags every record in France and Norway. Records without a code, or at sea, are flagged as in `CoordinateCleaner`.
- `outlier` flags the records of outlier cells in the raster approximation (species of 10,000 records or more, or `thinning = TRUE`); in `CoordinateCleaner` 3.0.1 that path never flags a record.
- `age_outlier` and `spatiotemporal_outlier` draw ages by the same procedure as `cf_age()` and `cf_outl()` but from a stream seeded per taxon, so a single run can differ record by record; when every age range has zero width the flags are identical.
- `dataset_rounded` leaves datasets below `min_unique_ds_size` untested, where `cd_round()` stops with an error; the dataset tests do not flag records without a dataset key.
- With the default reference the gazetteers are rebuilt from the original sources; `reference = "CoordinateCleaner"` uses that package's own tables, and gazetteers replaces the layer of single tests, as the `ref` arguments of `CoordinateCleaner`'s functions do.

Examples

```
x <- data.frame(species = c("a", "a", "b", "c"),
               decimalLongitude = c(12.58, 0, 16.37, 2.35),
               decimalLatitude = c(55.67, 0, 48.21, 48.85))
```

```
f <- flag_occurrences(x, tests = c("invalid", "zero", "gbif_headquarters", "capital"))
f[, c("species", "flags", "flag_zero", "flag_gbif_headquarters", "flag_capital")]
```

flag_reference	<i>Reference data behind a set of flags</i>
----------------	---

Description

Returns the label `flag_occurrences()` records with its flags: the package version, the land scale, the version of every reference layer, and for each user gazetteer passed through `gazetteers` its size and MD5 checksum. Store it next to any cleaned table that goes into an analysis.

Usage

```
flag_reference(x)
```

Arguments

x	A data frame returned by <code>flag_occurrences()</code> , an <code>occ_flagged</code> object, or the output directory of a streamed run.
---	---

Value

A character string.

Examples

```
x <- data.frame(decimalLongitude = c(16.37, 0), decimalLatitude = c(48.21, 0))
flag_reference(flag_occurrences(x, tests = c("zero", "capital")))
```

flag_tests	<i>The flag layout: one bit per test</i>
------------	--

Description

Every record flagged by `flag_occurrences()` carries an integer flag word in which bit bit is set when the record fails test test. The layout is versioned: `layout_version` changes whenever a bit is added, removed or redefined, and the flag output records which version produced it.

Usage

```
flag_tests()
```

Details

The word is an R integer, 32 bits of which the sign bit cannot be used (it carries NA): bits 0 to 29 hold the thirty tests and bit 30 is free. A thirty-second test needs a wider word, which will come with a new layout version.

Tests marked source = "CoordinateCleaner" are equivalent to the test of the same name in that package with its default buffer (see the equivalent column); the documented differences are listed in `flag_occurrences()`. Tests marked source = "OccurrenceCleaner" are additions.

Value

A data frame with columns bit (0-based), test, definition, source, equivalent (the **CoordinateCleaner** function, or NA), spatialvalid (the column `clean_coordinates()` or `clean_fossils()` writes for the test, used by `as_spatialvalid()`, or NA) and needs (the record fields the test reads, as names of `occ_columns()`). The attribute `layout_version` gives the layout version.

Examples

```
flag_tests()
```

gentiana	<i>GBIF records of gentians in Austria</i>
----------	--

Description

The occurrence records of the genus *Gentiana* that GBIF publishes with the country code AT (Austria) and a CC0 1.0 or CC BY 4.0 licence, taken from the GBIF snapshot of 1 October 2026. Specimens from herbaria and observations from surveys and citizen science, from the 18th century to 2026, with their coordinates, stated coordinate uncertainty and the fields GBIF fills during processing. Column names are those of the GBIF snapshot; `occ_columns()` matches them without renaming.

Usage

```
gentiana
```

Format

- A data frame with 33,025 rows and 15 columns:
- gbifid** GBIF record identifier.
 - datasetkey** GBIF dataset identifier; each record's licence and citation are those of its dataset.
 - kingdom, genus, species** Taxon, as matched by GBIF.
 - countrycode** ISO 3166-1 alpha-2 country code.
 - decimallongitude, decimallatitude** Coordinates in decimal degrees, NA where the record has none.
 - coordinateuncertaintyinmeters** Stated coordinate uncertainty in metres.

year Year the record was collected or observed.

basisofrecord Darwin Core basis of record, for example PRESERVED_SPECIMEN or HUMAN_OBSERVATION.

institutioncode Code of the institution holding the record.

establishmentmeans Darwin Core establishment means.

issue GBIF issue names, separated by ;.

license Licence of the record: CC0_1_0 or CC_BY_4_0.

Source

GBIF.org (1 October 2026) GBIF Occurrence Download [doi:10.15468/dl.tup2g6](https://doi.org/10.15468/dl.tup2g6). Records under CC BY 4.0 are attributed to their datasets, identified by datasetkey (<https://www.gbif.org/dataset/<datasetkey>>).

Examples

```
f <- flag_occurrences(gentiana, tests = c("invalid", "capital", "centroid", "institution"),
                      columns = occ_columns(id = "gbifid"))
summary(f, by = "basisofrecord")
```

keep_unflagged	<i>Keep the records no test flagged</i>
----------------	---

Description

Removes the records flagged by a cleaning rule from the output of `flag_occurrences()`. Without tests, the rule is every test that ran; with tests, only those tests exclude a record, so records flagged by other tests stay. The flag columns and the attributes of x are kept, so the result can be tallied, summarised or subset again.

Usage

```
keep_unflagged(x, tests = NULL)
```

Arguments

x	A data frame returned by <code>flag_occurrences()</code> .
tests	Character vector of test names, see <code>flag_tests()</code> ; NULL (the default) uses every test that ran. A test that was not run is an error.

Value

The rows of x with no flag from tests, with the columns and attributes of x.

Cleaning functions

Each test also has a cleaning function, `clean_<test>()`, documented on the test's help page (`test_capital`, `test_outlier`, ...). It runs that one test with `flag_occurrences()` and returns the rows of `x` the test does not flag, with the columns of `x` unchanged. Its arguments are the test's buffer or options, with the defaults in the usage line; further arguments (`columns`, `reference`, `gazetteers`, `land_scale`) go to `flag_occurrences()`.

A chain of cleaning functions runs each test on the records the previous steps kept. For record tests this gives the same records as flagging all tests once and calling `keep_unflagged()`. A grouped test (`duplicate`, `outlier`, the fossil outlier tests, the dataset tests) compares a record with the other records of its group, so its result depends on which records earlier steps removed; `flag_occurrences()` runs every test on the full input. The cleaning functions take a data frame; a table on disk is flagged with `flag_occurrences()` and counted with `tally_flags()`.

See Also

`has_flag()` for the logical vector behind the selection, `tally_flags()` to compare several rules.

Examples

```
x <- data.frame(species = c("a", "a", "b", "c"),
               decimalLongitude = c(12.58, 0, 16.37, 2.35),
               decimalLatitude = c(55.67, 0, 48.21, 48.85))
f <- flag_occurrences(x, tests = c("zero", "capital"))
keep_unflagged(f)
keep_unflagged(f, tests = "zero")
```

occ_columns

Name the record fields

Description

Tells `flag_occurrences()` which columns of the input hold each field. The defaults are the Darwin Core terms GBIF uses; matching is case-insensitive, so GBIF's lower-case snapshot columns (`decimallongitude`, ...) are found by the defaults too. Fields are resolved only when a requested test needs them.

Usage

```
occ_columns(
  lon = "decimalLongitude",
  lat = "decimalLatitude",
  species = "species",
  country = "countryCode",
  issue = "issue",
  basis = "basisOfRecord",
  establishment = "establishmentMeans",
  min_age = "min_ma",
```

```

    max_age = "max_ma",
    dataset = "datasetKey",
    genus = "genus",
    kingdom = "kingdom",
    id = "gbifID"
)

```

Arguments

lon, lat	Longitude and latitude in decimal degrees.
species	Taxon name or key used by the duplicate test.
country	Country code of the record (ISO 3166-1 alpha-2 or alpha-3), used by the country_mismatch test.
issue	GBIF issue flags, either a character column with the issue names separated by ; (as the GBIF snapshot read through vectra gives) or a list column of character vectors.
basis	Basis of record (basisOfRecord).
establishment	Establishment means (establishmentMeans).
min_age, max_age	Minimum and maximum age of a fossil record, in million years, as the Paleobiology Database's min_ma and max_ma.
dataset	Dataset the record belongs to, for the dataset-level tests (datasetKey in GBIF data).
genus	Genus of the record, for the habitat tests sea_nonmarine and land_marine. The first word is taken, so a binomial species column (occ_columns(genus = "species")) works as well as a genus column.
kingdom	Kingdom of the record, read by the habitat tests when the column is present, to tell apart genus names used in two kingdoms.
id	Record identifier, copied into on-disk flag output when present.

Value

A named list of column names with class `occ_columns`.

Examples

```
occ_columns(lon = "x", lat = "y")
```

plot.occ_summary	<i>Plot flag percentages, or map flagged records</i>
------------------	--

Description

Without grouping, one horizontal bar per test of a rule: the full bar is the share of records the test flags, the dark part the share it is the only reason for. With grouping (by or bins in [tally_flags\(\)](#)), a grid of tests by groups shaded and labelled with the share each test flags in each group. `type = "map"` on a flagged data frame draws the records over the Natural Earth land, coloured by whether the rule removes them.

Usage

```
## S3 method for class 'occ_summary'
plot(x, rule = NULL, ...)

## S3 method for class 'occ_tally'
plot(x, rule = NULL, ...)

## S3 method for class 'occ_flags'
plot(x, type = c("tests", "map"), rules = NULL, xlim = NULL, ylim = NULL, ...)

## S3 method for class 'occ_flagged'
plot(x, rules = NULL, ...)
```

Arguments

<code>x</code>	A flagged data frame (<code>occ_flags</code>), an <code>occ_flagged</code> object, an <code>occ_tally</code> or an <code>occ_summary</code> .
<code>rule</code>	Name of the rule to draw. Default: the first rule.
<code>...</code>	For <code>occ_flags</code> and <code>occ_flagged</code> : passed to tally_flags() (<code>by</code> , <code>bins</code> , <code>tests</code>). Otherwise unused.
<code>type</code>	For a flagged data frame: <code>"tests"</code> (the shares, as for an <code>occ_summary</code>) or <code>"map"</code> .
<code>rules</code>	Named list of rules, as in tally_flags() . For <code>type = "map"</code> the first rule decides which records are drawn as removed. Default: one rule holding every test run.
<code>xlim, ylim</code>	For <code>type = "map"</code> : longitude and latitude limits of the map. Default: the extent of the records. The title counts the records inside the limits.

Value

`x`, invisibly.

Examples

```
x <- data.frame(species = c("a", "a", "b", "c", "c"),
               kingdom = c("P", "P", "A", "A", "A"),
               decimalLongitude = c(12.58, 0, 16.37, 2.35, 2.35),
               decimalLatitude = c(55.67, 0, 48.21, 48.85, 48.85))
f <- flag_occurrences(x)
plot(f)
plot(f, by = "kingdom")
plot(f, type = "map")
plot(f, type = "map", xlim = c(0, 20), ylim = c(45, 60))
```

read_flagged

*Read flag output written by flag_occurrences()***Description**

Read flag output written by flag_occurrences()

Usage

```
read_flagged(x)

flagged_table(x)
```

Arguments

x An occ_flagged object or the output directory it describes.

Value

The occ_flagged metadata (read_flagged()), or a vectra_node over all parts (flagged_table()), to be used with **vectra** verbs and vectra::collect().

Examples

```
x <- data.frame(gbifID = 1:4, kingdom = c("Plantae", "Plantae", "Animalia", "Animalia"),
               decimalLongitude = c(12.58, 0, 16.37, 2.35),
               decimalLatitude = c(55.67, 0, 48.21, 48.85))
input <- file.path(tempdir(), "occurrence.csv")
utils::write.csv(x, input, row.names = FALSE)
out <- file.path(tempdir(), "flags")
flag_occurrences(input, tests = c("zero", "capital"), out = out, keep = "kingdom",
                 verbose = FALSE)
f <- read_flagged(out)
f
vectra::collect(flagged_table(f))
unlink(c(input, out), recursive = TRUE)
```

reference_habitat	<i>Genus habitats of the habitat tests</i>
-------------------	--

Description

Returns the genera the habitat tests sea_nonmarine and land_marine know, built from the Interim Register of Marine and Nonmarine Genera (IRMNG; Rees, T. (compiler) (2026). The Interim Register of Marine and Nonmarine Genera. Available from <https://www.irmng.org> at VLIZ; CC BY 4.0), Darwin Core archive of 2026-10-06. A name that is not accepted takes the habitat flags of its accepted name. A genus is

- not marine (marine = FALSE) when IRMNG sets isMarine to 0 and does not set isBrackish to 1;
- only marine (marine = TRUE) when IRMNG sets isMarine to 1 and both isTerrestrial and isFreshwater to 0, and the genus is not extinct (fossils of extinct marine genera lie on land).

Usage

```
reference_habitat()
```

Details

Genera with missing flags or with both habitats are not listed, and records of them are not tested. Several genera share a name across kingdoms; when they disagree the name is resolved only through the record's kingdom.

The table can be filtered or extended and passed back with `gazetteers = list(habitat = <data frame>)` in `flag_occurrences()`.

Value

A data frame with columns kingdom, genus and marine.

Examples

```
h <- reference_habitat()
table(h$kingdom, h$marine)
h[h$genus %in% c("Quercus", "Asterias", "Gadus"), ]
```

reference_info	<i>Reference data: sources and versions</i>
----------------	---

Description

Every gazetteer the package uses is pinned to a source file, a version and a retrieval date, and the flag output records which reference produced it.

Usage

```
reference_info()
```

Details

With `reference = "OccurrenceCleaner"` (the default of `flag_occurrences()`) the package uses its own gazetteers built in `data-raw/`: capitals from the CIA World Factbook and Natural Earth, province centroids, land, urban areas and country polygons from Natural Earth (public domain), country centroids from the CIA World Factbook, the thematicmapping.org World Borders Dataset and the GEOLocate service, and biodiversity institutions from GBIF's Global Registry of Scientific Collections (GRSciColl), Wikidata, Wikipedia, GeoNames, BGCI GardenSearch, Index Herbariorum and the GBIF registry, with every further institution of `CoordinateCleaner`'s table placed from Wikidata, OpenStreetMap or GeoNames. With `reference = "CoordinateCleaner"` the point gazetteers are read from the user's `CoordinateCleaner` installation (`countryref`, `institutions`) and the polygon layers are downloaded through `rnaturalearth` exactly as that package does, which gives its flags up to the differences documented in `flag_occurrences()`.

Value

A data frame with one row per layer: `layer`, `source`, `file`, `version`, `url`, `sha256`, `retrieved`, `licence`.

Examples

```
reference_info()
```

reference_points	<i>Reference points of a proximity test</i>
------------------	---

Description

Returns the points a proximity test measures distance to: capitals, country and province centroids, biodiversity institutions, the GBIF headquarters, or the points of the Artificial Hotspot Occurrence Inventory (Park et al. 2023). The table can be inspected, filtered or extended and passed back through the gazetteers argument of `flag_occurrences()`.

Usage

```
reference_points(  
  test = c("capital", "centroid", "institution", "gbif_headquarters", "aohi"),  
  reference = c("OccurrenceCleaner", "CoordinateCleaner")  
)
```

Arguments

test	One of "capital", "centroid", "institution", "gbif_headquarters", "aohi".
reference	"OccurrenceCleaner" (the package's own gazetteers) or "CoordinateCleaner" (that package's tables, which must be installed).

Value

A data frame with columns lon and lat and, for the package's own gazetteers, the attributes of each point: name, country, adm0_a3, iso_a2, class, source, key, extension for capitals (source is CIA World Factbook or Natural Earth, key the Factbook code or the Natural Earth country code and name with its prefix); name, adm0_a3, code, type, method, source, key, extension for centroids (source is CIA World Factbook, thematicmapping, GEOLocate or Natural Earth, key the Factbook code, ISO3 code or GEOLocate query with its prefix); key, code, name, country, city, type, active, master_source, source, extension for institutions (source is GRSciColl, Wikidata, Wikipedia, GeoNames, BGCI, Index Herbariorum, GBIF registry or CoordinateCleaner list, key the GRSciColl key, Wikidata item, the Wikipedia list language, revision and line, or the GeoNames, BGCI, Index Herbariorum, GBIF organisation or OpenStreetMap id with its prefix; for Wikipedia rows city is the place they are located by; for Wikipedia and CoordinateCleaner list rows master_source is where the coordinates come from; extension is TRUE for points from sources **CoordinateCleaner**'s table does not draw on, for capitals, centroids and institutions alike); id, taxon (the group the hotspot was found in: Aves, Insecta, Mammalia, Plantae), frequency (records at the point), category, determination (TRUE for a real observation site, FALSE for an artificial aggregate of records) and notes for the inventory. For reference = "CoordinateCleaner" the columns are lon and lat only.

Examples

```
head(reference_points("capital"))  
table(reference_points("centroid")$type)
```

summary.occ_flags	<i>Share of records flagged, per test and rule</i>
-------------------	--

Description

Turns flag counts into percentages: for every rule, how many records each of its tests flags, how many it is the only reason the rule removes, and how many the rule removes in total. A record can fail several tests, so the per-test shares of a rule can add up to more than the rule's total; the only shares add up to at most that total, and the rest is records removed for more than one reason.

Usage

```
## S3 method for class 'occ_flags'
summary(object, ...)

## S3 method for class 'occ_flagged'
summary(object, ...)

## S3 method for class 'occ_tally'
summary(object, ...)
```

Arguments

object	A data frame returned by flag_occurrences() (class <code>occ_flags</code>), an <code>occ_flagged</code> object, or a table from tally_flags() .
...	Passed to tally_flags() : rules, by, bins, tests.

Details

`summary()` on a flagged data frame or on on-disk flag output first calls [tally_flags\(\)](#) with the arguments in ..., so rules, by and bins work as there; on an `occ_tally` it reshapes the counts already made.

Value

A data frame of class `occ_summary`, one row per group, rule and test of the rule, plus one row per group and rule with `test = "(any)"` for the rule as a whole. Columns: the grouping columns, rule, test, n (records in the group), flagged (records carrying the test, or removed by the rule), pct (flagged as a percentage of n), only (records whose sole flag among the rule's tests is this one) and `pct_only`.

See Also

[plot.occ_summary\(\)](#) to draw it.

Examples

```
x <- data.frame(species = c("a", "a", "b", "c", "c"),
               kingdom = c("P", "P", "A", "A", "A"),
               decimalLongitude = c(12.58, 0, 16.37, 2.35, 2.35),
               decimalLatitude = c(55.67, 0, 48.21, 48.85, 48.85))
f <- flag_occurrences(x)
summary(f)
summary(f, rules = list(strict = c("zero", "capital", "duplicate")), by = "kingdom")
```

tally_flags

*Tally flags into a loss table***Description**

Counts, per group, how many records each test flags and how many a cleaning rule removes. A rule is a set of tests: a record is removed when it carries any of them. Works on a flagged data frame or on the on-disk output of `flag_occurrences()`, which is streamed part by part.

Usage

```
tally_flags(x, by = NULL, rules = NULL, bins = NULL, tests = NULL)
```

Arguments

<code>x</code>	A data frame returned by <code>flag_occurrences()</code> (or any data frame with an integer flags column), an <code>occ_flagged</code> object, or its output directory.
<code>by</code>	Character vector of grouping columns. For on-disk output these must have been passed as <code>keep</code> when flagging. Missing values form the group "missing".
<code>rules</code>	Named list of character vectors of test names. Default: one rule, "all", holding every test present in the flags.
<code>bins</code>	Named list of numeric break points. Each named column is cut at those breaks into a factor-like character column <code><name>_bin</code> (closed on the left, missing values labelled "missing") and added to <code>by</code> .
<code>tests</code>	Tests to count individually. Default: those recorded with the flags, or every test with a set bit.

Value

A data frame of class `occ_tally` with the `by` columns, `n` (records), `n_valid` (records without the invalid flag, when `invalid` was run), one column `flagged_<test>` per test, `per_rule_removed_<rule>` and `kept_<rule>`, and per rule and test of the rule `only_<rule>_<test>`: the records that test is the sole reason the rule removes. The attribute `occ_rules` lists the rules and `occ_by` the grouping columns. `summary()` turns the counts into percentages and `plot()` draws them.

Examples

```
x <- data.frame(species = c("a", "a", "b", "c"), kingdom = c("P", "P", "A", "A"),
               decimalLongitude = c(12.58, 0, 16.37, 2.35),
               decimalLatitude = c(55.67, 0, 48.21, 48.85))
f <- flag_occurrences(x, tests = c("invalid", "zero", "gbif_headquarters", "capital"))
tally_flags(f, by = "kingdom",
           rules = list(strict = c("zero", "gbif_headquarters", "capital"), mild = "zero"))
```

test_plan	<i>Which tests run on an input</i>
-----------	------------------------------------

Description

Without tests, `flag_occurrences()` runs every test whose inputs are available: the columns the test reads (the needs column of `flag_tests()`, named through `occ_columns()`) and, for range, species range polygons in gazetteers. Running a test only adds its flags, so which records to exclude is decided afterwards by a cleaning rule (see `has_flag()` and `tally_flags()`).

Usage

```
test_plan(x, columns = occ_columns(), gazetteers = NULL)
```

Arguments

`x` Input to `flag_occurrences()`: a data frame, or a path to a file or directory. Or its output: an `occ_flags` data frame, an `occ_flagged` object, or the output directory.

`columns, gazetteers` As in `flag_occurrences()`; used for input only.

Details

For input, `test_plan()` lists the tests that would run and the reason each other test would not. For flag output, it lists the tests that ran and the reason each other test did not: a missing input, or "not requested" when tests named the tests to run.

Value

A data frame with one row per test in layout order: test, run (logical), missing (the input the test lacks: column names, "range polygons", or "a path" for a grouped test given a single-run `vector_node`; NA when it runs or was not requested) and reason (why the test does not run, NA when it runs).

Examples

```
x <- data.frame(species = c("a", "a", "b"),
                decimalLongitude = c(12.58, 0, 16.37),
                decimalLatitude = c(55.67, 0, 48.21))
p <- test_plan(x)
p[p$run, "test"]
p[!p$run, ]
```

Index

- * **datasets**
 - gentiana, [46](#)
- * **tests**
 - clean_age_equal, [4](#)
 - clean_age_outlier, [5](#)
 - clean_age_range_outlier, [7](#)
 - clean_aohi, [8](#)
 - clean_basis_fossil, [10](#)
 - clean_capital, [11](#)
 - clean_centroid, [12](#)
 - clean_country_mismatch, [14](#)
 - clean_dataset_ddmm, [15](#)
 - clean_dataset_rounded, [17](#)
 - clean_duplicate, [18](#)
 - clean_equal, [19](#)
 - clean_establishment_introduced, [20](#)
 - clean_gbif_headquarters, [22](#)
 - clean_institution, [23](#)
 - clean_invalid, [25](#)
 - clean_issue_coordinate, [26](#)
 - clean_outlier, [28](#)
 - clean_range, [30](#)
 - clean_sea, [31](#)
 - clean_sea_nonmarine, [32](#)
 - clean_spatiotemporal_outlier, [34](#)
 - clean_urban, [36](#)
 - clean_zero, [37](#)
- as_spatialvalid, [3](#)
- as_spatialvalid(), [46](#)
- clean_age_equal, [4](#)
- clean_age_equal(), [6](#), [8–10](#), [12](#), [14–16](#),
[18–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#), [34–36](#),
[38](#)
- clean_age_outlier, [5](#)
- clean_age_outlier(), [5](#), [8–10](#), [12](#), [14–16](#),
[18–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#), [34–36](#),
[38](#)
- clean_age_range_outlier, [7](#)
- clean_age_range_outlier(), [5](#), [6](#), [9](#), [10](#), [12](#),
[14–16](#), [18–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#),
[34–36](#), [38](#)
- clean_aohi, [8](#)
- clean_aohi(), [5](#), [6](#), [8](#), [10](#), [12](#), [14–16](#), [18–21](#),
[23](#), [25–27](#), [29](#), [31](#), [32](#), [34–36](#), [38](#)
- clean_basis_fossil, [10](#)
- clean_basis_fossil(), [5](#), [6](#), [8](#), [9](#), [12](#), [14–16](#),
[18–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#), [34–36](#),
[38](#)
- clean_basis_living
(clean_basis_fossil), [10](#)
- clean_capital, [11](#)
- clean_capital(), [5](#), [6](#), [8–10](#), [14–16](#), [18–21](#),
[23](#), [25–27](#), [29](#), [31](#), [32](#), [34–36](#), [38](#)
- clean_centroid, [12](#)
- clean_centroid(), [5](#), [6](#), [8–10](#), [12](#), [15](#), [16](#),
[18–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#), [34–36](#),
[38](#)
- clean_country_mismatch, [14](#)
- clean_country_mismatch(), [5](#), [6](#), [8–10](#), [12](#),
[14](#), [16](#), [18–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#),
[34–36](#), [38](#)
- clean_dataset_ddmm, [15](#)
- clean_dataset_ddmm(), [5](#), [6](#), [8–10](#), [12](#), [14](#),
[15](#), [18–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#),
[34–36](#), [38](#)
- clean_dataset_rounded, [17](#)
- clean_dataset_rounded(), [5](#), [6](#), [8–10](#), [12](#),
[14–16](#), [19–21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#),
[34–36](#), [38](#)
- clean_duplicate, [18](#)
- clean_duplicate(), [5](#), [6](#), [8–10](#), [12](#), [14–16](#),
[18](#), [20](#), [21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#),
[34–36](#), [38](#)
- clean_equal, [19](#)
- clean_equal(), [5](#), [6](#), [8–10](#), [12](#), [14–16](#), [18](#), [19](#),
[21](#), [23](#), [25–27](#), [29](#), [31](#), [32](#), [34–36](#), [38](#)
- clean_establishment_introduced, [20](#)

- `clean_establishment_introduced()`, 5, 6, 8–10, 12, 14–16, 18–20, 23, 25–27, 29, 31, 32, 34–36, 38
- `clean_establishment_managed`
(`clean_establishment_introduced`), 20
- `clean_gbif_headquarters`, 22
- `clean_gbif_headquarters()`, 5, 6, 8–10, 12, 14–16, 18–21, 25–27, 29, 31, 32, 34–36, 38
- `clean_institution`, 23
- `clean_institution()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 26, 27, 29, 31, 32, 34–36, 38
- `clean_invalid`, 25
- `clean_invalid()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25, 27, 29, 31, 32, 34–36, 38
- `clean_issue_coordinate`, 26
- `clean_issue_coordinate()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25, 26, 29, 31, 32, 34–36, 38
- `clean_issue_date`
(`clean_issue_coordinate`), 26
- `clean_issue_rounded`
(`clean_issue_coordinate`), 26
- `clean_issue_taxon`
(`clean_issue_coordinate`), 26
- `clean_land_marine`
(`clean_sea_nonmarine`), 32
- `clean_outlier`, 28
- `clean_outlier()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 31, 32, 34–36, 38
- `clean_range`, 30
- `clean_range()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 29, 32, 34–36, 38
- `clean_sea`, 31
- `clean_sea()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 29, 31, 34–36, 38
- `clean_sea_nonmarine`, 32
- `clean_sea_nonmarine()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 29, 31, 32, 35, 36, 38
- `clean_spatiotemporal_outlier`, 34
- `clean_spatiotemporal_outlier()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 29, 31, 32, 34, 36, 38
- `clean_urban`, 36
- `clean_urban()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 29, 31, 32, 34, 35, 38
- `clean_zero`, 37
- `clean_zero()`, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 29, 31, 32, 34–36
- `download_reference`, 38
- `download_reference()`, 31, 42
- `export_pyrate`, 39
- `flag_mask`, 40
- `flag_occurrences`, 41
- `flag_occurrences()`, 3–38, 40, 45–48, 52, 53, 55–57
- `flag_reference`, 45
- `flag_tests`, 45
- `flag_tests()`, 3, 5, 6, 8–10, 12, 14–16, 18–21, 23, 25–27, 29, 31, 32, 34–36, 38, 40, 42, 44, 47, 57
- `flagged_table` (`read_flagged`), 51
- `gentiana`, 46
- `has_flag` (`flag_mask`), 40
- `has_flag()`, 4, 43, 48, 57
- `keep_unflagged`, 47
- `keep_unflagged()`, 4, 6, 7, 9–11, 13, 14, 16, 17, 19–22, 24, 25, 27, 29–31, 33, 35–37
- `occ_columns`, 48
- `occ_columns()`, 4, 15, 26, 39, 42, 46, 57
- `plot()`, 43, 56
- `plot.occ_flagged` (`plot.occ_summary`), 50
- `plot.occ_flags` (`plot.occ_summary`), 50
- `plot.occ_summary`, 50
- `plot.occ_summary()`, 55
- `plot.occ_tally` (`plot.occ_summary`), 50
- `read_flagged`, 51
- `reference_habitat`, 52
- `reference_habitat()`, 33, 34, 42
- `reference_info`, 53
- `reference_info()`, 11, 12, 14, 25, 32, 38, 42
- `reference_points`, 53
- `reference_points()`, 9, 23, 24
- `summary()`, 43, 56

summary.occ_flagged
 (summary.occ_flags), 54
 summary.occ_flags, 54
 summary.occ_tally (summary.occ_flags),
 54

 tally_flags, 56
 tally_flags(), 5, 7, 42, 43, 48, 50, 55, 57
 test_age_equal, 8, 40
 test_age_equal (clean_age_equal), 4
 test_age_outlier, 5, 34, 42
 test_age_outlier (clean_age_outlier), 5
 test_age_range_outlier, 5, 42
 test_age_range_outlier
 (clean_age_range_outlier), 7
 test_aohi (clean_aohi), 8
 test_basis (clean_basis_fossil), 10
 test_capital, 48
 test_capital (clean_capital), 11
 test_centroid (clean_centroid), 12
 test_country_mismatch
 (clean_country_mismatch), 14
 test_dataset_ddmm, 42
 test_dataset_ddmm (clean_dataset_ddmm),
 15
 test_dataset_rounded, 42
 test_dataset_rounded
 (clean_dataset_rounded), 17
 test_duplicate (clean_duplicate), 18
 test_equal (clean_equal), 19
 test_establishment
 (clean_establishment_introduced),
 20
 test_gbif_headquarters
 (clean_gbif_headquarters), 22
 test_gbif_issue
 (clean_issue_coordinate), 26
 test_habitat, 42
 test_habitat (clean_sea_nonmarine), 32
 test_institution (clean_institution), 23
 test_invalid (clean_invalid), 25
 test_outlier, 42, 48
 test_outlier (clean_outlier), 28
 test_plan, 57
 test_plan(), 42, 43
 test_range, 42
 test_range (clean_range), 30
 test_sea, 34
 test_sea (clean_sea), 31

 test_spatiotemporal_outlier, 5, 42
 test_spatiotemporal_outlier
 (clean_spatiotemporal_outlier),
 34
 test_urban (clean_urban), 36
 test_zero (clean_zero), 37