Package: dataeuropa (via r-universe)

November 28, 2026

Title Access the 'data.europa.eu' European Open Data Portal

Version 0.1.1

Description Search, retrieve and download open data from the European data portal <https://data.europa.eu> through its Search API, Registry API and SPARQL endpoint. Returns data frames of datasets, distributions and facets.

License MIT + file LICENSE

URL https://github.com/randsley/dataeuropa, https://randsley.github.io/dataeuropa/r/

BugReports https://github.com/randsley/dataeuropa/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports httr2 (>= 1.1.0), jsonlite, openssl

Suggests testthat (>= 3.0.0), withr, callr, readxl, arrow, sf

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libssl-dev

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

Date/Publication 2026-11-27 23:57:07 UTC

RemoteUrl https://github.com/randsley/dataeuropa

RemoteRef v0.1.1

RemoteSha 71bc9905d5f7105acee62a6bdd2a37c32cdda349

RemoteSubdir r

Contents

de_cache_on . . . 2
de_catalogues . . . 3
de_count . . . 3
de_distributions . . . 4
de_download . . . 5
de_facet_values . . . 6
de_fetch_datasets . . . 6
de_get_dataset . . . 7
de_get_rdf . . . 8
de_load_distribution . . . 8
de_load_geodata . . . 9
de_pick_lang . . . 10
de_search . . . 11
de_search_datasets . . . 12
de_sparql . . . 13

Index **14**

`de_cache_on` *Turn the on-disk response cache on or off, or empty it*

Description

Cached: dataset records (`de_get_dataset()`, `de_distributions()`), the catalogue list (`de_catalogues()`), and searches with `limit = 0` (`de_facet_values()`, `de_count()`). Other searches, SPARQL and file downloads are never cached.

Usage

```
de_cache_on(dir = NULL, ttl = NULL)

de_cache_off()

de_clear_cache()
```

Arguments

`dir`	Cache folder. Defaults to `DATAEUROPA_CACHE_DIR`, else `tools::R_user_dir("dataeuropa", "cache")`.
`ttl`	How long a response stays valid, in seconds. Defaults to `DATAEUROPA_CACHE_TTL`, else one day.

Details

The cache is off unless `de_cache_on()` is called or the environment variable `DATAEUROPA_CACHE=1` is set. `de_cache_off()` turns it off for the session; cached files stay until `de_clear_cache()`.

Value

`de_cache_on()`: the cache folder, invisibly. `de_cache_off()`: `NULL`, invisibly. `de_clear_cache()`: the number of responses deleted.

Examples

```
de_cache_on(dir = file.path(tempdir(), "dataeuropa-cache"), ttl = 3600)
de_clear_cache()
de_cache_off()
```

---

de_catalogues *Ids of all harvested catalogues*

Description

Ids of all harvested catalogues

Usage

```
de_catalogues()
```

Value

A character vector of catalogue ids.

Examples

```
head(de_catalogues())
```

---

de_count *Number of hits for a query (no records transferred)*

Description

Number of hits for a query (no records transferred)

Usage

```
de_count(q = NULL, ...)
```

Arguments

`q`	Free-text query.
`...`	Passed to `de_search()`.

Value

An integer.

Examples

```
de_count(country = "pt")
de_count("water", format = "CSV")
```

`de_distributions` *Distributions (files, services, pages) of a dataset*

Description

One row per distribution. The `url` column is the download URL if present, otherwise the access URL. The `kind` column says what each one is: `"file"`, `"service"` (WMS, WFS, APIs, feeds), `"page"` (a web page), `"none"` (no usable URL, e.g. a localhost placeholder) or `"unknown"`.

Usage

```
de_distributions(id, lang = "en", files_only = FALSE)
```

Arguments

`id`	Dataset id as shown in search results.
`lang`	Preferred language.
`files_only`	Keep only distributions whose `kind` is `"file"`.

Value

A data frame with one row per distribution: `id`, `title`, `format`, `media_type`, `byte_size`, `checksum`, `checksum_algorithm`, `license`, `modified`, `download_url`, `access_url`, `url` and `kind`. An empty data frame when the dataset has none.

Examples

```
files <- de_distributions("6689f1686e4bd9bc98ed011b", files_only = TRUE)
files[, c("title", "format", "kind")]
```

de_download *Download a distribution to disk*

Description

If the row has a checksum (`checksum` and `checksum_algorithm`; MD5, SHA-1 or SHA-256), the file is checked against it. On a mismatch the file is deleted and an error is raised; `verify = FALSE` skips the check. `byte_size` is not checked: on the portal it is often a placeholder (0, 1, 1024).

Usage

```
de_download(x, dest, verify = TRUE, resume = TRUE, max_bytes = NULL)
```

Arguments

`x`	A URL, or a one-row data frame from `de_distributions()`.
`dest`	A file path, or an existing directory to save the file in under its own name. Required: nothing is written to the working directory unless you pass it (e.g. `"."`).
`verify`	Check the file against the row's checksum, if it has one.
`resume`	Continue an interrupted download of the same URL to the same place.
`max_bytes`	Stop a download that is (or is announced as) larger than this, delete the partial file and raise an error. Default: no limit.

Details

The file is written as `<file>.part` and renamed when complete. If a download is interrupted, calling `de_download()` again for the same URL and destination continues where it stopped, when the server supports ranges and identifies the file (ETag or Last-Modified); a file that changed meanwhile is downloaded again from the start. `resume = FALSE` always starts from the start.

Value

The local path, invisibly.

Examples

```
files <- de_distributions("6689f1686e4bd9bc98ed011b", files_only = TRUE)
path <- de_download(files[files$title == "QA_2024_Q1", ], tempdir())
file.size(path)
```

de_facet_values *Values and counts of a facet*

Description

Values and counts of a facet

Usage

```
de_facet_values(facet, q = NULL, top = 50, ...)
```

Arguments

facet	e.g. "country", "format", "categories", "catalog", "publisher".
q	Free-text query.
top	Keep the top most frequent values (NULL for all).
...	Passed to de_search().

Value

A data frame with columns id, label and count, most frequent first.

Examples

```
de_facet_values("format", country = "pt", top = 10)
```

de_fetch_datasets *Fetch many datasets with the scroll API*

Description

Fetch many datasets with the scroll API

Usage

```
de_fetch_datasets(
  q = NULL,
  ...,
  max_results = 1000,
  page_size = 500,
  lang = "en",
  raw = FALSE
)
```

Arguments

q	Free-text query.
...	Passed to de_search().
max_results	Stop after this many records.
page_size	Records per request (max 1000).
lang	Language for titles/descriptions.
raw	Return the raw list of records instead of a data frame.

Value

A data frame with the columns of de_search_datasets(), one row per dataset, or with raw = TRUE a list of the raw records. At most max_results either way.

Examples

```
ds <- de_fetch_datasets(country = "pt", max_results = 2500, page_size = 1000)  # three requests
nrow(ds)
```

---

de_get_dataset *Full metadata for one dataset*

Description

Full metadata for one dataset

Usage

```
de_get_dataset(id, raw = FALSE, lang = "en")
```

Arguments

id	Dataset id as shown in search results.
raw	If TRUE return the untouched nested list.
lang	Preferred language.

Value

A list with summary (a one-row data frame with the columns of de_search_datasets()), keywords (a character vector) and raw (the record as a nested list). With raw = TRUE, the record alone.

Examples

```
d <- de_get_dataset("6689f1686e4bd9bc98ed011b")
d$summary$title
d$summary[, c("publisher", "modified", "formats")]
```

de_get_rdf *Dataset metadata as RDF text*

Description

Dataset metadata as RDF text

Usage

```
de_get_rdf(id, fmt = c("ttl", "jsonld"))
```

Arguments

id	Dataset id.
fmt	"ttl" (Turtle) or "jsonld".

Value

The RDF document as a single string.

Examples

```
ttl <- de_get_rdf("6689f1686e4bd9bc98ed011b")
cat(substr(ttl, 1, 500))
```

de_load_distribution *Download a distribution and read it into a data frame*

Description

Reads CSV and JSON; XLSX with the readxl package and Parquet with the arrow package (both optional). The format comes from `format` if given (`"csv"`, `"json"`, `"xlsx"`, `"parquet"`), else from the row's format id, its media type, then the URL's file extension.

Usage

```
de_load_distribution(x, format = NULL, sheet = NULL)
```

Arguments

x	A URL, or a one-row data frame from `de_distributions()`.
format	Override the detected format: "csv", "json", "xlsx" or "parquet".
sheet	XLSX sheet name or 1-based index (default: the first sheet).

Details

CSV: the delimiter (`,` `;` tab `|`) is taken from the header line, text is read as UTF-8 with a Latin-1 fallback, only empty cells count as missing, and dates stay text. JSON: an array of objects, or an object holding exactly one such array; keys become columns.

Stops with an error for services (WMS, WFS, APIs), compressed archives, HTML pages served instead of data, formats it cannot tell or read, and files that fail their checksum (see `de_download()`).

Value

A data frame.

Examples

```
files <- de_distributions("6689f1686e4bd9bc98ed011b", files_only = TRUE)
qa <- de_load_distribution(files[files$title == "QA_2024_Q1", ])
head(qa)
```

`de_load_geodata` *Download a geodata distribution, or read a feature service, into an sf data frame*

Description

Needs the sf package. GDAL does the reading, so any vector format it knows works: GeoJSON, Shapefile, GeoPackage, KML/KMZ, GML, GPX, MapInfo TAB, FlatGeobuf, file geodatabase. The geometry column is always called `geometry`.

Usage

```
de_load_geodata(
  x,
  layer = NULL,
  crs = NULL,
  max_mb = 200,
  max_features = 10000
)
```

Arguments

`x`	A URL, or a one-row data frame from `de_distributions()`.
`layer`	A layer by name or 1-based position, in files with several (GeoPackage, file geodatabase). Default: the first, with a message if there are more.
`crs`	Reproject to this coordinate reference system, e.g. `"EPSG:4326"`.
`max_mb`	Do not download files larger than this many megabytes; `Inf` lifts the limit.
`max_features`	Read at most this many features from a service, with a warning when there are more; `Inf` reads all.

Details

ZIP archives (how shapefiles are published) are opened in place. Inside, the one geodata file is read (a .shp with its .dbf/.shx/.prj, .gpkg, .geojson, .kml, .gml, .gpx, .tab, .fgb, or a .gdb folder); if there are several, `layer` picks one by file name (without extension), otherwise an error lists them. Archives inside archives are not opened.

The source coordinate reference system is kept; a file without one (a shapefile without .prj) is read as stored, with a warning.

Feature services are read too: WFS (`service=WFS` URLs, WFS format ids) and OGC API Features (collection or landing-page URLs, OGCFEAT format ids). GDAL pages through the results. Each collection or feature type is a layer.

Raises an error for other services (WMS, ...), web pages, HTML served instead of data, files or services GDAL cannot read, and checksum mismatches.

Value

An sf data frame.

Examples

```
dists <- de_distributions("eac1e504-9d63-4670-95b3-3ef04e22aa6e~~1")
districts <- de_load_geodata(dists[dists$format == "OGCFEAT", ])
sf::st_crs(districts)$epsg
```

---

de_pick_lang *Collapse a multilingual* `{lang: text}` *list to one string*

Description

Collapse a multilingual `{lang: text}` list to one string

Usage

```
de_pick_lang(x, lang = "en")
```

Arguments

`x`	A named list of translations (or a string).
`lang`	Preferred language code; falls back to English, then any.

Value

A character scalar, or `NA_character_`.

Examples

```
de_pick_lang(list(de = "Wasser", en = "Water"), lang = "de")
de_pick_lang(list(de = "Wasser", en = "Water"), lang = "fr")   # falls back to English
de_pick_lang(list(pt = "Qualidade do ar"))                     # else any language
```

---

de_search *Raw Search API call*

Description

Returns the `result` object of the Search API as a nested list (`count`, `results`, `facets`, ...).

Usage

```
de_search(
  q = NULL,
  filters = "dataset",
  facets = NULL,
  limit = 10,
  page = 0,
  sort = NULL,
  country = NULL,
  catalog = NULL,
  format = NULL,
  license = NULL,
  category = NULL,
  publisher = NULL,
  keywords = NULL,
  hvd = NULL,
  min_date = NULL,
  max_date = NULL,
  date_type = NULL,
  bbox = NULL,
  aggregation = FALSE,
  extra = list()
)
```

Arguments

`q`	Free-text query.
`filters`	Document type: "dataset" (default), "catalogue", "dataservice", "resource"...
`facets`	Named list of facet filters, e.g. `list(country = "pt")`.
`limit`	Results per page (0-1000).
`page`	Zero-based page.
`sort`	e.g. `"modified+desc"`.

`country`, `catalog`, `format`, `license`, `category`, `publisher`, `keywords`, `hvd`
: Convenience facet filters. `country` takes ISO-2 codes, `format` ids like "CSV", `category` DCAT-AP theme codes ("ENVI").

`min_date`, `max_date`, `date_type`
: Date range; `date_type` one of issued, modified, temporal.

`bbox`
: Numeric `c(min_lon, min_lat, max_lon, max_lat)` in WGS84 degrees. A dataset matches when its spatial extent overlaps the box, so datasets covering all of Europe or the world match any box. Datasets without a spatial extent (about half the portal) never match.

`aggregation`
: Return facet counts?

`extra`
: Named list of additional raw query parameters.

Details

With `limit = 0` the result can come from the response cache (see `de_cache_on()`).

Value

The `result` object of the response, a nested list with `count`, `results` (the records), `facets` when `aggregation = TRUE`, and `scrollId` in scroll mode.

Examples

```
res <- de_search("air quality", country = "pt", limit = 5)
res$count
de_pick_lang(res$results[[1]]$title)
```

---

de_search_datasets *Search datasets and return a data frame*

Description

Accepts all arguments of `de_search()`. Total hits are in `attr(x, "count")`.

Usage

```
de_search_datasets(q = NULL, ..., lang = "en")
```

Arguments

`q`
: Free-text query.

`...`
: Passed to `de_search()`.

`lang`
: Language for titles/descriptions.

Value

A data frame with one row per dataset: `id`, `title`, `description`, `country`, `catalog`, `publisher`, `issued`, `modified`, `formats`, `n_distributions`, `is_hvd`, `quality_score`, `landing_page` and `portal_url`. The total number of hits is in `attr(x, "count")`.

Examples

```
ds <- de_search_datasets("air quality", country = "pt", format = "CSV", limit = 5)
ds[, c("id", "title", "formats")]
attr(ds, "count")
```

---

de_sparql *Run a SPARQL query against data.europa.eu*

Description

Run a SPARQL query against data.europa.eu

Usage

```
de_sparql(query)
```

Arguments

`query`	SPARQL SELECT or ASK text.

Value

A data frame of plain string values (SELECT) or a logical (ASK).

Examples

```
de_sparql("SELECT ?d WHERE { ?d a <http://www.w3.org/ns/dcat#Dataset> } LIMIT 3")
```

Index

de_cache_off *(*de_cache_on*)*, 2
de_cache_on, 2
de_cache_on(), *12*
de_catalogues, 3
de_catalogues(), *2*
de_clear_cache *(*de_cache_on*)*, 2
de_count, 3
de_count(), *2*
de_distributions, 4
de_distributions(), *2*, *5*, *8, 9*
de_download, 5
de_download(), *9*
de_facet_values, 6
de_facet_values(), *2*
de_fetch_datasets, 6
de_get_dataset, 7
de_get_dataset(), *2*
de_get_rdf, 8
de_load_distribution, 8
de_load_geodata, 9
de_pick_lang, 10
de_search, 11
de_search(), *3*, *6, 7*, *12*
de_search_datasets, 12
de_search_datasets(), *7*
de_sparql, 13