Package: dataseries (via r-universe)

June 3, 2026

Type Package

Title Switzerland's Data Series in One Place

Version 1.0.0

Description Download and import open Swiss economic time series from 'dataseries.org' <https://dataseries.org>, a comprehensive and up-to-date collection of public data from Switzerland. Series are retrieved through the public 'dataseries.org' API and imported as a 'data.frame' or 'ts' object.

License GPL-3

URL https://dataseries.org, https://github.com/cynkra/dataseries

BugReports https://github.com/cynkra/dataseries/issues

Imports jsonlite, stats, utils

Suggests curl, testthat (>= 3.0.0)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

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

Date/Publication 2026-06-03 11:52:02 UTC

RemoteUrl https://github.com/cynkra/dataseries

RemoteRef HEAD

RemoteSha 1bb2af283427c43665381ec7ef2d0071d6088851

Contents

`cache_ls` *List or clear the in-memory cache*

Description

Everything downloaded from dataseries.org is cached in memory for the lifetime of the R session. `cache_ls()` lists the cached objects (keyed by request URL); `cache_rm()` empties the cache, which forces the next call to download fresh data.

Usage

```
cache_ls()

cache_rm()
```

Value

`cache_ls()` returns a character vector of cache keys; `cache_rm()` is called for its side effect and returns `NULL` invisibly.

Examples

```
ds_catalog()
cache_ls()
cache_rm()
```

`ds` *Download time series from dataseries.org*

Description

`ds()` downloads open Swiss economic time series from dataseries.org.

Usage

```
ds(dataset, ..., from = NULL, to = NULL, class = c("data.frame", "ts"))
```

Arguments

`dataset`	a single dataset `id`, as listed by `ds_catalog()`.
`...`	dimension filters, given as named arguments where each name is a dimension of `dataset` and each value is one or more codes, e.g. `type = "real"` or `structure = c("gdp", "gva")`. A single named list may be passed instead, which is convenient programmatically (e.g. from the output of `ds_search()`). See `ds_meta()` for the available dimensions and codes.
`from, to`	optional date bounds (a `Date` or an ISO `"YYYY-MM-DD"` string) that restrict the returned range, inclusive.
`class`	class of the return value: `"data.frame"` (the default, one row per observation in long format) or `"ts"`. For `"ts"` the selected series are laid out as columns, one per cell. (To obtain an **xts** object, wrap the result: `xts::as.xts(ds(..., class = "ts"))`.)

Details

Data on dataseries.org is organized into **datasets**. A dataset is a family of related series and is, in most cases, a multi-dimensional *cube*: a single time series is one cell of the cube, addressed by the dataset plus one code per dimension. Pass those codes as **named arguments** (the names are the dimension names, see `ds_meta()`):

```
ds("ch_seco_gdp", type = "real", structure = "gdp", seas_adj = "csa")
```

Dimension arguments are optional. Omit them and you get the whole dataset (all series, in long format). A few datasets are a single series and take no dimensions at all (e.g. `ds("ch_kof_barometer")`). Filtering happens on the server, so selecting one series does not download the whole cube.

Downloads are **cached in memory** for the session. Run `cache_rm()` to force a fresh download.

Value

A `data.frame` or `ts`/`mts` object, or `NULL` if the selection is empty.

See Also

`ds_catalog()` for the list of datasets and `ds_meta()` for a dataset's dimensions.

Examples

```
# whole dataset (long data.frame)
ds("ch_fso_cpi")

# one series, by dimension code
ds("ch_fso_cpi", item = "100_100")

# several series, restricted to a date range
ds("ch_fso_cpi", item = c("100_100", "100_1"), from = "2020-01-01")

# as a ts object
```

```
ds("ch_seco_gdp", type = "real", structure = "gdp", seas_adj = "csa",
   class = "ts")
```

`ds_catalog` *Catalog of available datasets*

Description

Lists every dataset available on dataseries.org, one row per dataset. Use the `id` column with `ds()` to download data and with `ds_meta()` to inspect a dataset's dimensions.

Usage

```
ds_catalog()
```

Value

A `data.frame` with one row per dataset and the columns `id`, `title`, `concept`, `topic`, `source`, `license`, `frequency`, `start`, `end` and `n_series`.

Examples

```
cat <- ds_catalog()
head(cat)

# search the catalog
cat[grepl("price", cat$title, ignore.case = TRUE), c("id", "title")]
```

`ds_meta` *Metadata for one dataset*

Description

Returns the full metadata for a single dataset: its dimensions, the codes (levels) available within each dimension, labels, source, license and date range. Use this to discover which dimension codes to pass to `ds()`.

Usage

```
ds_meta(dataset)
```

Arguments

`dataset`	a single dataset `id`, as listed by `ds_catalog()`.

Value

A named list (the parsed metadata). Notable elements are `dim_order` (the dataset's dimensions) and `dimensions` (each dimension's `levels`, keyed by code, with a `label`).

Examples

```
m <- ds_meta("ch_seco_gdp")
m$dim_order                        # "type", "structure", "seas_adj"
names(m$dimensions$type$levels)    # the codes you can pass as type = ...
```

---

ds_search *Search for series across all datasets*

Description

Returns a flat, searchable table of the individual series available across every dataset on dataseries.org — one row per series. This is the finest-grained way to discover what exists: grep it, or pass a `pattern` to filter. The `dataset`, `dim` and `code` columns are exactly what you feed back to `ds()`.

Usage

```
ds_search(pattern = NULL)
```

Arguments

`pattern`	optional search string, treated as a case-insensitive regular expression and matched against the series `label` and `dataset_title`. When `NULL` (the default) the full table is returned.

Value

A `data.frame` with the columns `dataset`, `dataset_title`, `frequency`, `dim`, `code`, `label` and `path`.

See Also

`ds_catalog()` for the dataset-level list and `ds_meta()` for one dataset's dimensions.

Examples

```
# everything
ds_search()

# find unemployment series, then download one
hits <- ds_search("unemployment")
head(hits)
ds(hits$dataset[1], setNames(list(hits$code[1]), hits$dim[1]))
```

Index