Package 'epidatr'

September 21, 2026

Type Package

Title Client for Delphi's 'Epidata' API

Version 1.4.0

Description The Delphi 'Epidata' API provides real-time access to epidemiological surveillance data for influenza, 'COVID-19', and other diseases for the USA at various geographical resolutions, both from official government sources such as the Center for Disease Control (CDC) and Google Trends and private partners such as Facebook and Change 'Healthcare'. It is built and maintained by the Carnegie Mellon University Delphi research group. To cite this API: David C. Farrow, Logan C. Brooks, Aaron 'Rumack', Ryan J. 'Tibshirani', 'Roni' 'Rosenfeld' (2015). Delphi 'Epidata' API. <https://github.com/cmu-delphi/delphi-epidata>.

License MIT + file LICENSE

URL https://cmu-delphi.github.io/epidatr/, https://cmu-delphi.github.io/delphi-epidata/, https://github.com/cmu-delphi/epidatr

BugReports https://github.com/cmu-delphi/epidatr/issues

Depends R (>= 3.5.0)

Imports cachem, checkmate, cli, glue, httr2, jsonlite, magrittr, MMWRweek, openssl, purrr, rappdirs, readr, rlang, tibble, utils, vctrs, lifecycle, xml2

Suggests knitr, rmarkdown, testthat (>= 3.1.5), withr, curl

VignetteBuilder knitr

Config/Needs/precompile dplyr, ggplot2, mapproj, maps, ragg

Config/Needs/website cmu-delphi/delphidocs

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Collate 'auth.R' 'cache.R' 'check.R' 'constants.R' 'covidcast.R' 'endpoints.R' 'epidatacall.R' 'epidatr-package.R' 'model.R' 'request.R' 'utils-pipe.R' 'utils.R'

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Logan Brooks [aut], Dmitry Shemetov [aut], Samuel Gratzl [aut], David Weber [ctb, cre], Nat DeFries [ctb], Alex Reinhart [ctb], Daniel J. McDonald [ctb], Kean Ming Tan [ctb], Will Townes [ctb], George Haff [ctb], Kathryn Mazaitis [ctb]

Maintainer David Weber <davidweb@andrew.cmu.edu>

Repository CRAN

Date/Publication 2026-09-21 10:00:19 UTC

Contents

avail_endpoints . . . 3
cache_info . . . 4
cast_api_queries . . . 4
clear_cache . . . 7
covidcast_epidata . . . 8
create_epidata_call . . . 9
disable_cache . . . 11
epidata_aux . . . 11
epidata_meta . . . 13
epirange . . . 14
fetch_args_list . . . 14
get_api_key . . . 16
pub_covidcast . . . 17
pub_covidcast_meta . . . 19
pub_covid_hosp_facility . . . 20
pub_covid_hosp_facility_lookup . . . 22
pub_covid_hosp_state_timeseries . . . 23
pub_delphi . . . 24
pub_dengue_nowcast . . . 25
pub_ecdc_ili . . . 26
pub_flusurv . . . 27
pub_fluview . . . 29
pub_fluview_clinical . . . 31
pub_fluview_meta . . . 32

pub_gft . . . 33
pub_kcdc_ili . . . 34
pub_meta . . . 35
pub_nidss_dengue . . . 36
pub_nidss_flu . . . 37
pub_nowcast . . . 38
pub_paho_dengue . . . 39
pub_wiki . . . 40
pvt_cdc . . . 41
pvt_dengue_sensors . . . 42
pvt_ght . . . 43
pvt_meta_norostat . . . 44
pvt_norostat . . . 45
pvt_quidel . . . 46
pvt_sensors . . . 47
pvt_twitter . . . 48
set_cache . . . 50
timeset . . . 52

Index **53**

`avail_endpoints` *List all available Epidata API endpoints*

Description

Fetches a data frame of all Epidata API endpoints that can be accessed using this package, with a brief description.

Usage

```
avail_endpoints()
```

Value

A `tibble::tibble` of endpoints, with two columns:

`Endpoint`	Name of the function for accessing this API endpoint.
`Description`	One-sentence description of the data available at the endpoint.

Examples

```
avail_endpoints()
```

cache_info *Describe current cache*

Description

Print out the information about the cache (as would be returned by cachem's `info()` method).

Usage

```
cache_info()
```

Value

`list` containing the info result as created by cachem

See Also

`set_cache` to start a new cache (and general caching info), `clear_cache` to delete the cache and set a new one, and `disable_cache` to disable without deleting

cast_api_queries *cast-API snapshot and archive queries*

Description

- `epidata_snapshot` fetches a snapshot of signals as they appeared at a specific date (or the latest available if `snapshot_date` is omitted).
- `epidata_archive` fetches the full version history of signals across all available issues.
- `epidata` is a wrapper that routes to one of the above based on which versioning argument is supplied.

For a source's signals, geo types, and extra key columns, see its API docs, e.g. NWSS: https://cmu-delphi.github.io/delphi-epidata/api/v5-signals/nwss.html.

Usage

```
epidata_snapshot(
  source,
  signals,
  geo_type,
  geo_values = "*",
  reference_time = "*",
  time_values = lifecycle::deprecated(),
  ...,
  fill_method = NULL,
  snapshot_date = NULL,
```

```
  as_of = lifecycle::deprecated(),
  fetch_args = fetch_args_list()
)

epidata_archive(
  source,
  signals,
  geo_type,
  geo_values = "*",
  reference_time = "*",
  time_values = lifecycle::deprecated(),
  ...,
  fill_method = NULL,
  report_time = "*",
  issues = lifecycle::deprecated(),
  fetch_args = fetch_args_list()
)

epidata(
  source,
  signals,
  geo_type,
  geo_values = "*",
  reference_time = "*",
  time_values = lifecycle::deprecated(),
  ...,
  fill_method = NULL,
  snapshot_date = NULL,
  as_of = lifecycle::deprecated(),
  report_time = NULL,
  issues = lifecycle::deprecated(),
  fetch_args = fetch_args_list()
)
```

Arguments

source	string. The data source to query (e.g., "nssp", "nhsn"). Use epidata_meta() to discover available sources.
signals	character vector. One or more signals to query for the given source; comma-joined strings (e.g., "sig1,sig2") are also accepted. Use epidata_meta() to discover available signals. All signals are sent comma-joined in a single request per geo type (the cast-API only accepts one geo type per request) and the results are combined.
geo_type	character vector. One or more geography types to query (e.g., "state", "nation", "county"); comma-joined strings are also accepted. Use epidata_meta() to discover available geo types for a given source and signal.
geo_values	character. The geographies to return. Defaults to all ("*") geographies within requested geographic resolution (see: https://cmu-delphi.github.io/delphi-epidata/

	api/covidcast_geography.html.).
reference_time	timeset. Reference time to return (filters on the reference_time column). Supports individual dates or epirange(). Defaults to all ("*"). Filtered locally after the API call.
time_values	**[Deprecated]** Use reference_time instead.
...	Named filters on extra key columns beyond geo_value, such as pcr_target = "sars-cov-2" or sample_index = c("a", "b"). Each key accepts one or more values (matched as OR) and is sent server-side via the extra_keys API parameter to shrink the download. Passing more than 10 values for a key warns. Unlike
fill_method	string. Optional filter to an imputation method. The API provides alternatives of the same signal differing in how nulls were handled during geographic aggregation: "source" means no imputation or aggregation (raw source data), "fill_ave" fills nulls with the average of neighboring values, and "fill_zero" fills nulls with zero. NULL (default) returns all fill methods.
snapshot_date	Date, POSIXt, UTC timestamp string (e.g. "2025-10-16T13:45:00Z"), or NULL. The point in time to snapshot: returns the version of the data that was current then. NULL returns the latest available version.
as_of	**[Deprecated]** Use snapshot_date instead.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.
report_time	String or epirange(). A filter on the report_time column. Accepts comparison operators (e.g., "<2025-10-16", ">=2025-10-16", or "<=2025-10-16T13:45:00Z" for a UTC timestamp bound) or an epirange() for an inclusive date range. Bare dates and the "=" operator are not accepted: use snapshot_date for point-in-time data. Internally maps to the report_time_query API parameter.
issues	**[Deprecated]** Use report_time instead.

Value

tibble::tibble

Data Versioning

epidata supports two mutually exclusive versioning arguments. Pass snapshot_date to retrieve data as it appeared on a specific date, or report_time to query the archive by when data was reported. If neither is supplied, epidata returns the latest available snapshot.

Empty results

An invalid geo_type or signals value for the given source raises an error (class epidatr__epidata__invalid_geo_type or epidatr__epidata__invalid_signals respectively), looked up via epidata_meta(). Otherwise, an empty result warns rather than failing silently: a warning of class epidatr__empty_signals if only some of the requested signals returned no data, or epidatr__empty_result if the whole query came back empty (whether because the server had no matching rows, or because the local geo_values/reference_time filters dropped everything the server returned). Pass fetch_args_list(return_empty = TRUE) to suppress these errors and warnings and get an empty tibble back instead.

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

See Also

`epidata_meta()`, `epirange()`

`clear_cache` *Manually reset the cache, deleting all currently saved data and starting afresh*

Description

Deletes the current cache and resets a new cache. Deletes local data! If you are using a session unique cache, the previous settings will be reused. If you pass in new `set_cache` arguments, they will take precedence over the previous settings.

Usage

```
clear_cache(..., disable = FALSE)
```

Arguments

`...`	Arguments passed on to `set_cache` `cache_dir` the directory in which the cache is stored. By default, this is `rappdirs::user_cache_dir("R version = "epidatr")`. The path can be either relative or absolute. The environmental variable is `EPIDATR_CACHE_DIR`. `days` the maximum length of time in days to keep any particular cached call. By default this is `1`. The environmental variable is `EPIDATR_CACHE_MAX_AGE_DAYS`. `max_size` the size of the entire cache, in MB, at which to start pruning entries. By default this is `1024`, or 1GB. The environmental variable is `EPIDATR_CACHE_MAX_SIZE_MB`. `logfile` where cachem's log of transactions is stored, relative to the cache directory. By default, it is `"logfile.txt"`. The environmental variable is `EPIDATR_CACHE_LOGFILE`. `confirm` whether to confirm directory creation. default is `TRUE`; should only be set in non-interactive scripts `startup` indicates whether the function is being called on startup. Affects suppressability of the messages. Default is `FALSE`.
`disable`	instead of setting a new cache, disable caching entirely; defaults to `FALSE`

Value

`NULL` no return value, all effects are stored in the package environment

See Also

set_cache to start a new cache (and general caching info), disable_cache to only disable without deleting, and cache_info

---

covidcast_epidata *Creates the COVIDcast Epidata autocomplete helper*

Description

Creates a helper object that can use auto-complete to help find COVIDcast sources and signals. The COVIDcast endpoint of the Epidata API contains many separate data sources and signals. It can be difficult to find the name of the signal you're looking for, so you can use covidcast_epidata to get help with finding sources and functions without leaving R.

The covidcast_epidata() function fetches a list of all signals, and returns an object containing fields for every signal:

```
epidata <- covidcast_epidata()
epidata$signals
#> # A tibble: 520 x 3
#>    source        signal                        short_description
#>    <chr>         <chr>                         <chr>
#>  1 chng         smoothed_outpatient_cli      Estimated percentage of outpatie~
#>  2 chng         smoothed_adj_outpatient_cli  Estimated percentage of outpatie~
#>  3 chng           smoothed_outpatient_covid     COVID-Confirmed Doctor Visits
#>  4 chng           smoothed_adj_outpatient_covid COVID-Confirmed Doctor Visits
#>  5 chng         smoothed_outpatient_flu      Estimated percentage of outpatie~
#>  6 chng         smoothed_adj_outpatient_flu  Estimated percentage of outpatie~
#>  7 chng          7dav_inpatient_covid        Ratio of inpatient hospitalizati~
#>  8 chng          7dav_outpatient_covid       Ratio of outpatient doctor visit~
#>  9 covid-act-now pcr_specimen_positivity_rate Proportion of PCR specimens test~
#> 10 covid-act-now pcr_specimen_total_tests    Total number of PCR specimens te~
#> # i 510 more rows
```

If you use an editor that supports tab completion, such as RStudio, type epidata$signals$ and wait for the tab completion popup. You will be able to type the name of signals and have the autocomplete feature select them from the list for you. Note that some signal names have dashes in them, so to access them we rely on the backtick operator:

```
epidata$signals$`fb-survey:smoothed_cli`
#> [1] "COVID-Like Symptoms (Unweighted 7-day average)"
#> [1] "fb-survey:smoothed_cli"
#> [1] "Estimated percentage of people with COVID-like illness "
```

These objects can be used directly to fetch data, without requiring us to use the pub_covidcast() function. Simply use the $call attribute of the object:

```
epidata$signals$`fb-survey:smoothed_cli`$call("state", "pa",
                                              epirange(20210405, 20210410))
#> Warning: `pub_covidcast()` uses the V4 Epidata API.
#> i Starting in October 2026, V4 is tentatively deprecated in favor of the V5
#>   API.
#> i See `vignette("migration-guide")` (or
#>  <https://cmu-delphi.github.io/epidatr/articles/migration-guide.html>) for the
#>   V5 endpoints and how to move to them. Old data will remain available for at
#>   least a year, but new ingestion will end.
#> This warning is displayed once every 8 hours.
#> # A tibble: 6 x 15
#>   geo_value signal     source geo_type time_type time_value direction issue
#>   <chr>     <chr>      <chr>  <fct>    <fct>     <date>         <dbl> <date>
#> 1 pa        smoothed_~ fb-su~ state    day       2021-04-05        NA 2021-04-10
#> 2 pa        smoothed_~ fb-su~ state    day       2021-04-06        NA 2021-04-11
#> 3 pa        smoothed_~ fb-su~ state    day       2021-04-07        NA 2021-04-12
#> 4 pa        smoothed_~ fb-su~ state    day       2021-04-08        NA 2021-04-13
#> 5 pa        smoothed_~ fb-su~ state    day       2021-04-09        NA 2021-04-14
#> 6 pa        smoothed_~ fb-su~ state    day       2021-04-10        NA 2021-04-15
#> # i 7 more variables: lag <dbl>, missing_value <dbl>, missing_stderr <dbl>,
#> #   missing_sample_size <dbl>, value <dbl>, stderr <dbl>, sample_size <dbl>
```

Usage

```
covidcast_epidata(base_url = global_base_url, timeout_seconds = 30)
```

Arguments

`base_url`	optional alternative API base url
`timeout_seconds`	the maximum amount of time to wait for a response

Value

An instance of `covidcast_epidata`

`create_epidata_call` *An abstraction that holds information needed to make an epidata request*

Description

`epidata_call` objects are generated internally by endpoint functions like `pub_covidcast`; by default, they are piped directly into the `fetch` function to fetch and format the data. For most endpoints this will return a tibble, but a few non-COVIDCAST endpoints will return a JSON-like list instead.

Usage

```
create_epidata_call(
  endpoint,
  params,
  meta = NULL,
  api_version = c("classic", "cast"),
  response_format = c("classic", "json", "csv")
)

fetch(epidata_call, fetch_args = fetch_args_list())
```

Arguments

endpoint	the epidata endpoint to call
params	the parameters to pass to the epidata endpoint
meta	meta data to attach to the epidata call
api_version	string. The API version to use. One of "classic" or "cast".
response_format	string. The expected format of the response. One of "classic", "json", or "csv".
epidata_call	an instance of epidata_call
fetch_args	a fetch_args object

Details

create_epidata_call is the constructor for epidata_call objects, but you should not need to use it directly; instead, use an endpoint function, e.g., pub_covidcast, to generate an epidata_call for the data of interest.

Value

- For create_epidata_call: an epidata_call object
- For fetch: a tibble

Examples

```
library(magrittr)

call <- pub_covidcast(
  source = "jhu-csse",
  signals = "confirmed_7dav_incidence_prop",
  time_type = "day",
  geo_type = "state",
  time_values = epirange(20200601, 20200801),
  geo_values = c("ca", "fl"),
  fetch_args = fetch_args_list(dry_run = TRUE)
)
call %>% fetch()
```

disable_cache *Turn off the caching for this session*

Description

Disable caching until you call `set_cache` or restart R. The files defining the cache are untouched. If you are looking to disable the caching more permanently, set `EPIDATR_USE_CACHE=FALSE` as environmental variable in your `.Renviron`.

Usage

```
disable_cache()
```

Value

`NULL` no return value, all effects are stored in the package environment

See Also

`set_cache` to start a new cache (and general caching info), `clear_cache` to delete the cache and set a new one, and `cache_info`

epidata_aux *Fetch V5 auxiliary data*

Description

Fetch auxiliary data associated with a cast signal.

You can pass a source string to fetch the auxiliary data directly. Alternatively, you can pass the output of `epidata_snapshot()` or `epidata_archive()`. In this case, `epidata_aux` automatically retrieves the source from the object, fetches the matching auxiliary data, and performs a version-aware left join onto the base data.

For the auxiliary key columns and their allowed values, see the source's API docs, e.g. NWSS: https://cmu-delphi.github.io/delphi-epidata/api/v5-signals/nwss.html.

Usage

```
epidata_aux(source, ...)

## Default S3 method:
epidata_aux(
  source,
  ...,
  reference_time = "*",
  time_values = lifecycle::deprecated(),
```

```
  snapshot_date = NULL,
  report_time = "*",
  issues = lifecycle::deprecated(),
  columns = NULL,
  fetch_args = fetch_args_list()
)

## S3 method for class 'data.frame'
epidata_aux(source, ..., columns = NULL, fetch_args = fetch_args_list())
```

Arguments

source	A source string to retrieve auxiliary data directly, or a tibble returned by epidata_snapshot() or epidata_archive() to merge the data onto (its source is recovered automatically).
...	Named filters on the auxiliary key columns, such as pcr_target = "sars-cov-2" or geo_value = c("ca", "ny"). Each key accepts one or more values (matched as OR); they are serialized as repeated key:value terms server-side to keep the aux pull small. Passing more than 10 values for a key warns, since the request URL may get too long. When source is a tibble and no filters are given, they are inferred from the base: each key it narrows to at most 10 distinct values is filtered to those.
reference_time	timeset. Reference time to return (filters on the reference_time column). Supports individual dates or epirange(). Only used when source is a string.
time_values	**[Deprecated]** Use reference_time instead.
snapshot_date	Date, POSIXt, UTC timestamp string (e.g. "2025-10-16T13:45:00Z"), "latest", or NULL. Return auxiliary data as it appeared at this date or instant (one row per key, the most recent version active then). "latest" uses today's date. Use NULL (default) to return the full version history filtered by report_time. Mutually exclusive with report_time. Only used when source is a string.
report_time	String or epirange() specifying the version of the auxiliary data to retrieve. Accepts comparison operators (e.g., "<2025-10-16", ">=2025-10-16", or "<=2025-10-16T13:45:00Z" for a UTC timestamp bound) or an epirange() for an inclusive date range. Bare dates and the "=" operator are not accepted: use snapshot_date for point-in-time data. Mutually exclusive with snapshot_date. Only used when source is a string.
issues	**[Deprecated]** Use report_time instead.
columns	A character vector of columns to return. By default, all columns are returned.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

A tibble::tibble.

See Also

epidata_snapshot(), epidata_archive(), epidata_meta()

---

epidata_meta *Get cast-API source metadata*

Description

epidata_meta returns source-level metadata from the cast-API, including report_time ranges, reference_time ranges, and lists of available signals and geo types.

Usage

```
epidata_meta(source = NULL, fetch_args = fetch_args_list())
```

Arguments

source	string. The data source to query. If NULL (default), returns metadata for all available sources. If specified, returns metadata for the given source.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

list. If source is NULL, a named list of source metadata objects. If source is specified, the metadata list for that source.

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

See Also

epidata_snapshot(), epidata_archive(), epidata(), epirange()

epirange *Specify a range of days or weeks for API requests*

Description

Specify a date range (in days or epiweeks) for an API request.

Usage

```
epirange(from, to)
```

Arguments

from	The first date to request. Can be specified as a Date or as an integer or integer-like string in the format YYYYMMDD for dates or YYYYWW for epiweeks.
to	The final date to request (inclusive), specified the same way as from.

Details

Epiweeks, also known as MMWR weeks number the weeks of the year from 1 to 53, each week spanning from Sunday to Saturday. The numbering is defined by the CDC.

Value

An EpiRange object.

Examples

```
# Represents 2021-01-01 to 2021-01-07, inclusive
epirange(20210101, 20210107)

# The same, but using Date objects
epirange(as.Date("2021-01-01"), as.Date("2021-01-07"))

# Represents epiweeks 2 through 4 of 2022, inclusive
epirange(202202, 202204)
```

fetch_args_list *Set custom API request parameters*

Description

Used to specify custom options when making API requests, such as to set timeouts or change data formats. These options are used by fetch() when it makes calls to the Epidata API.

Usage

```
fetch_args_list(
  ...,
  fields = NULL,
  disable_date_parsing = FALSE,
  disable_data_frame_parsing = FALSE,
  disable_missing_meta_warning = FALSE,
  return_empty = FALSE,
  timeout_seconds = 15 * 60,
  base_url = NULL,
  dry_run = FALSE,
  debug = lifecycle::deprecated(),
  format_type = lifecycle::deprecated(),
  refresh_cache = FALSE,
  reference_week_day = 1,
  limit = NULL
)
```

Arguments

`...`	not used for values, forces later arguments to bind by name
`fields`	a list of epidata fields to return, or `NULL` to return all fields (default). e.g. `c("time_value", "value")` to return only the `time_value` and `value` fields or `c("-direction")` to return everything except the direction field
`disable_date_parsing`	disable automatic date parsing
`disable_data_frame_parsing`	disable automatic conversion to data frame; this is only supported by endpoints that only support the 'classic' format (non-tabular)
`disable_missing_meta_warning`	if `TRUE`, suppress the warning emitted when the response contains columns not declared as expected epidata fields. Useful for endpoints/signals that intentionally return extra, untyped columns.
`return_empty`	boolean that allows returning an empty tibble if there is no data
`timeout_seconds`	the maximum amount of time (in seconds) to wait for a response from the API server
`base_url`	base URL to use; by default `NULL`, which means the global base URL `"https://api.delphi.cmu.edu/e`
`dry_run`	if `TRUE`, skip the call to the API and instead return the `epidata_call` object (useful for debugging)
`debug`	**[Deprecated]** No longer supported. Use `dry_run = TRUE` instead.
`format_type`	**[Deprecated]** Now managed internally.
`refresh_cache`	if `TRUE`, ignore the cache, fetch the data from the API, and update the cache, if it is enabled

`reference_week_day`
: the day of the week to use as the reference day when parsing epiweeks to dates (happens if `disable_date_parsing` is `FALSE`) Defaults to 1 Sunday (the first day of the week).

`limit`
: integer or `NULL`. cast-API only (`epidata_snapshot()`, `epidata_archive()`, `epidata_aux()`, `epidata()`): maximum number of rows to return. `NULL` (default) or `-1` requests no limit. The underlying query has no stable sort order, so `limit` does not guarantee the same rows (or even the same count) across repeated calls; use it only to preview or debug a query, never as a substitute for filtering.

Value

A `fetch_args` object containing all the specified options

`get_api_key` *Get and set API keys*

Description

Get and set the API key used to make requests to the Epidata API. Without a key, requests may be subject to rate limits and other limitations.

Usage

```
get_api_key()

save_api_key()
```

Details

We recommend you register for an API key. While most endpoints are available without one, there are limits on API usage for anonymous users, including a rate limit. If you regularly request large amounts of data, please consider registering for an API key.

API keys are strings read from the environment variable `DELPHI_EPIDATA_KEY`. We recommend setting your key with `save_api_key()`, which will open your `.Renviron` file in a text editor. You will need to write `DELPHI_EPIDATA_KEY=yourkeyhere` (without quotes) in the file and save it. Once the `.Renviron` file has been saved as instructed, it will be read automatically when you start future R sessions (see `?Startup` for details on `.Renviron` files). Alternatively, you can modify the environment variable at the command line before/while launching R, or inside an R session with `Sys.setenv()`, but these will not persist across sessions.

Once an API key is set, it is automatically used for all requests made by functions in this package.

Value

For `get_api_key()`, returns the current API key as a string, or `""` if none is set.

References

- Delphi Epidata API Keys documentation.
- Delphi Epidata API Registration Form.

`pub_covidcast` *Various COVID and flu signals via the COVIDcast endpoint*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the `epidata_snapshot()`, `epidata_archive()`, and `epidata_meta()` functions. For more details on the changes, refer to `vignette("migration-guide")`, and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html

The primary endpoint for fetching COVID-19 data, providing access to a wide variety of signals from a wide variety of sources. See the API documentation link above for more. Delphi's COVIDcast public dashboard is powered by this endpoint.

Usage

```
pub_covidcast(
  source,
  signals,
  geo_type,
  time_type,
  geo_values = "*",
  time_values = "*",
  ...,
  as_of = NULL,
  issues = NULL,
  lag = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

`source`	string. The data source to query (see: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html).
`signals`	string. The signals to query from a specific source (see: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html).
`geo_type`	string. The geographic resolution of the data (see: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_geography.html).
`time_type`	string. The temporal resolution of the data (either "day" or "week", depending on signal).

geo_values	character. The geographies to return. Defaults to all ("*") geographies within requested geographic resolution (see: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_geography.html.).
time_values	timeset. Dates or epiweeks to fetch. Supports epirange() and defaults to all ("*") dates.
...	not used for values, forces later arguments to bind by name
as_of	Date. Optionally, the as-of date for the issues to fetch. See the "Data Versioning" section for details.
issues	timeset. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
lag	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- as_of: (Date) Retrieve the data as it was on this date.
- issues: timeset Retrieve data from a specific issue date or range of dates.
- lag: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See vignette("versioned-data") for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

See Also

pub_covidcast_meta(), covidcast_epidata(), epirange()

Examples

```
pub_covidcast(
  source = "jhu-csse",
  signals = "confirmed_7dav_incidence_prop",
  geo_type = "state",
  time_type = "day",
```

```
  geo_values = c("ca", "fl"),
  time_values = epirange(20200601, 20200801)
)
pub_covidcast(
  source = "jhu-csse",
  signals = "confirmed_7dav_incidence_prop",
  geo_type = "state",
  time_type = "day",
  geo_values = "*",
  time_values = epirange(20200601, 20200801)
)
```

`pub_covidcast_meta` *Metadata for the COVIDcast endpoint*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the `epidata_snapshot()`, `epidata_archive()`, and `epidata_meta()` functions. For more details on the changes, refer to `vignette("migration-guide")`, and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_meta.html.

Fetch a summary of metadata for all sources and signals that are available in the API, along with basic summary statistics such as the dates they are available, the geographic levels at which they are reported, and etc.

The result can be filtered server-side by passing `signals`, `time_type`, and/or `geo_type`. Omitted filters (the default) return metadata for everything.

Usage

```
pub_covidcast_meta(
  signals = NULL,
  time_type = NULL,
  geo_type = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

`signals`	character. Optionally, the signals to return metadata for, each formatted as `"source:signal"` (e.g. `"fb-survey:smoothed_cli"`). Defaults to all signals.
`time_type`	string. The temporal resolution of the data (either "day" or "week", depending on signal).
`geo_type`	string. Optionally, a single geographic resolution to return metadata for (see: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_geography.html). Defaults to all geographic resolutions.

fetch_args fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

See Also

pub_covidcast(),covidcast_epidata()

Examples

```
pub_covidcast_meta()
# All signals from the Facebook survey data source
pub_covidcast_meta(
  signals = "fb-survey:*"
)
# All signals with time_type "day".
pub_covidcast_meta(
  time_type = "day",
)
# All signals with geo_type "state".
pub_covidcast_meta(
  geo_type = "state",
)
```

---

pub_covid_hosp_facility

COVID hospitalizations by facility

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/covid_hosp_facility.html

Obtains the COVID-19 reported patient impact and hospital capacity data by facility. This dataset is provided by the US Department of Health & Human Services. The companion function pub_covid_hosp_facility_lookup can be used to look up facility identifiers in a variety of ways.

Usage

```
pub_covid_hosp_facility(
  hospital_pks,
  collection_weeks = "*",
  ...,
  publication_dates = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

hospital_pks	character. Facility identifiers.
collection_weeks	timeset. Dates (corresponding to epiweeks) to fetch. Defaults to all ("*") dates.
...	not used for values, forces later arguments to bind by name
publication_dates	timeset. Publication dates to fetch.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Details

Starting October 1, 2022, some facilities are only required to report annually.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

See Also

pub_covid_hosp_facility(), epirange()

Examples

```
pub_covid_hosp_facility(
  hospital_pks = "100075",
  collection_weeks = epirange(20200101, 20200501)
)

pub_covid_hosp_facility(
  hospital_pks = "050063",
  collection_weeks = epirange(20240101, 20240301)
)
```

pub_covid_hosp_facility_lookup
Helper for finding COVID hospitalization facilities

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/covid_hosp_facility_lookup.html

Obtains unique identifiers and other metadata for COVID hospitalization facilities of interest. This is a companion endpoint to the pub_covid_hosp_facility() endpoint.

Usage

```
pub_covid_hosp_facility_lookup(
  ...,
  state = NULL,
  ccn = NULL,
  city = NULL,
  zip = NULL,
  fips_code = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

...	not used for values, forces later arguments to bind by name
state	string. A two-letter character state abbreviation. See US states codes for details.
ccn	string. A facility CMS certification number.
city	string. A city name.
zip	string. A 5-digit zip code.
fips_code	string. A 5-digit fips county code, zero-padded.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Details

Only one location argument needs to be specified. Combinations of the arguments are not currently supported.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

See Also

`pub_covid_hosp_facility()`

Examples

```
pub_covid_hosp_facility_lookup(state = "fl")
pub_covid_hosp_facility_lookup(city = "southlake")
```

`pub_covid_hosp_state_timeseries`

COVID hospitalizations by state

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/covid_hosp.html.

Obtains the COVID-19 reported patient impact and hospital capacity data by state. This dataset is provided by the US Department of Health & Human Services.

Usage

```
pub_covid_hosp_state_timeseries(
  states,
  dates = "*",
  ...,
  as_of = NULL,
  issues = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

`states`	character. Two-letter state abbreviations. See US states codes for details.
`dates`	`timeset`. Dates to fetch. Supports `epirange()` and defaults to all ("*") dates.
`...`	not used for values, forces later arguments to bind by name
`as_of`	Date. Optionally, the as-of date for the issues to fetch. See the "Data Versioning" section for details.
`issues`	`timeset`. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
`fetch_args`	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Details

Starting October 1, 2022, some facilities are only required to report annually.

Value

`tibble::tibble`

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- `as_of`: (Date) Retrieve the data as it was on this date.
- `issues`: `timeset` Retrieve data from a specific issue date or range of dates.
- `lag`: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See `vignette("versioned-data")` for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_covid_hosp_state_timeseries(
  states = "fl",
  dates = epirange(20200101, 20200501)
)
```

---

pub_delphi *Delphi's ILINet outpatient doctor visits forecasts*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/delphi.html

Usage

```
pub_delphi(system, epiweek, fetch_args = fetch_args_list())
```

Arguments

system	character. System name to fetch. See the available forecasting systems # nolint for details.
epiweek	timeset. Epiweek to fetch. Does not support multiple dates. Make separate calls to fetch data for multiple epiweeks.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

list

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
pub_delphi(system = "ec", epiweek = 201501)
```

---

pub_dengue_nowcast *Delphi's PAHO dengue nowcasts (North and South America)*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/dengue_nowcast.html

Usage

```
pub_dengue_nowcast(locations, epiweeks = "*", fetch_args = fetch_args_list())
```

Arguments

locations	character. List of locations to fetch. See the codes for countries and territories in the Americas. # nolint
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_dengue_nowcast(
  locations = "pr",
  epiweeks = epirange(201401, 202301)
)
```

---

pub_ecdc_ili *ECDC ILI incidence (Europe)*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/ecdc_ili.html.

Obtain information on influenza-like-illness from the European Centre for Disease Prevention and Control.

Usage

```
pub_ecdc_ili(
  regions,
  epiweeks = "*",
  ...,
  issues = NULL,
  lag = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

regions	character. List of regions to fetch. See the codes for European countries. # nolint
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
...	not used for values, forces later arguments to bind by name
issues	timeset. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
lag	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Details

The list of location argument can be found in https://github.com/cmu-delphi/delphi-epidata/blob/main/labels/ecdc_regions.txt.

Value

tibble::tibble

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- as_of: (Date) Retrieve the data as it was on this date.
- issues: timeset Retrieve data from a specific issue date or range of dates.
- lag: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See vignette("versioned-data") for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
pub_ecdc_ili(regions = "austria", epiweeks = epirange(201901, 202001))
```

---

pub_flusurv *CDC FluSurv flu hospitalizations*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the epidata_snapshot(), epidata_archive(), and epidata_meta() functions. For more details on the changes, refer to vignette("migration-guide"), and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/flusurv.html.

Obtain information on influenza hospitalization rates from the Center of Disease Control.

See also https://gis.cdc.gov/GRASP/Fluview/FluHospRates.html.

Usage

```
pub_flusurv(
  locations,
  epiweeks = "*",
  ...,
  issues = NULL,
  lag = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

locations	character. List of locations to fetch. See geographic codes for details.
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
...	not used for values, forces later arguments to bind by name
issues	timeset. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
lag	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Details

The list of location argument can be found in https://github.com/cmu-delphi/delphi-epidata/blob/main/labels/flusurv_locations.txt.

Value

tibble::tibble

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- as_of: (Date) Retrieve the data as it was on this date.
- issues: timeset Retrieve data from a specific issue date or range of dates.
- lag: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See vignette("versioned-data") for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_flusurv(locations = "ca", epiweeks = epirange(201701, 201801))
```

`pub_fluview` *CDC FluView ILINet outpatient doctor visits*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the `epidata_snapshot()`, `epidata_archive()`, and `epidata_meta()` functions. For more details on the changes, refer to `vignette("migration-guide")`, and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/fluview.html. For

Obtains information on outpatient inluenza-like-illness (ILI) from U.S. Outpatient Influenza-like Illness Surveillance Network (ILINet).

more information on ILINet, see https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

Usage

```
pub_fluview(
  regions,
  epiweeks = "*",
  ...,
  issues = NULL,
  lag = NULL,
  auth = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

`regions`	character. Vector of location IDs to fetch. Can be "nat" for national, "hhs1"–"hhs10" for HHS Regions, "cen1"–"cen9" for census divisions, lowercase two-letter state or territory abbreviations for most states and territories,"jfk" for New York City, or "ny_minus_jfk" for upstate New York. Full list of locations is available here and here.

epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
...	not used for values, forces later arguments to bind by name
issues	timeset. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
lag	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
auth	string. Your restricted access key (not the same as API key).
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Details

The full list of location inputs can be accessed at https://github.com/cmu-delphi/delphi-epidata/blob/main/src/acquisition/fluview/fluview_locations.py.

Value

tibble::tibble

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- as_of: (Date) Retrieve the data as it was on this date.
- issues: timeset Retrieve data from a specific issue date or range of dates.
- lag: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See vignette("versioned-data") for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
pub_fluview(regions = "nat", epiweeks = epirange(201201, 202005))
```

pub_fluview_clinical *CDC FluView flu tests from clinical labs*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the `epidata_snapshot()`, `epidata_archive()`, and `epidata_meta()` functions. For more details on the changes, refer to `vignette("migration-guide")`, and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/fluview_clinical.html

Usage

```
pub_fluview_clinical(
  regions,
  epiweeks = "*",
  ...,
  issues = NULL,
  lag = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

regions	character. Vector of location IDs to fetch. Can be "nat" for national, "hhs1"–"hhs10" for HHS Regions, "cen1"–"cen9" for census divisions, lowercase two-letter state or territory abbreviations for most states and territories,"jfk" for New York City, or "ny_minus_jfk" for upstate New York. Full list of locations is available here and here.
epiweeks	`timeset`. Epiweeks to fetch. Supports `epirange()` and defaults to all ("*") dates. Format as `epirange(startweek, endweek)`, where startweek and endweek are of the form YYYYWW (string or numeric).
...	not used for values, forces later arguments to bind by name
issues	`timeset`. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
lag	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
fetch_args	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Value

`tibble::tibble`

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- `as_of`: (Date) Retrieve the data as it was on this date.
- `issues`: `timeset` Retrieve data from a specific issue date or range of dates.
- `lag`: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See `vignette("versioned-data")` for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_fluview_clinical(regions = "nat", epiweeks = epirange(201601, 201701))
```

---

pub_fluview_meta *Metadata for the FluView endpoint*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the `epidata_snapshot()`, `epidata_archive()`, and `epidata_meta()` functions. For more details on the changes, refer to `vignette("migration-guide")`, and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/fluview_meta.html

Usage

```
pub_fluview_meta(fetch_args = fetch_args_list())
```

Arguments

`fetch_args`	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Value

`tibble::tibble`

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

See Also

`pub_fluview()`

Examples

```
pub_fluview_meta()
```

---

pub_gft *Google Flu Trends flu search volume*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/gft.html

Obtains estimates of inluenza activity based on volume of certain search queries from Google.

Usage

```
pub_gft(locations, epiweeks = "*", fetch_args = fetch_args_list())
```

Arguments

`locations`	character. List of locations to fetch.
`epiweeks`	`timeset`. Epiweeks to fetch. Supports `epirange()` and defaults to all ("*") dates. Format as `epirange(startweek, endweek)`, where startweek and endweek are of the form YYYYWW (string or numeric).
`fetch_args`	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Details

Google has discontinued Flu Trends and this is now a static endpoint. Possibile input for locations can be found in https://cmu-delphi.github.io/delphi-epidata/api/geographic_codes.html#hhs-regions, https://cmu-delphi.github.io/delphi-epidata/api/geographic_codes.html#us-states, and https://cmu-delphi.github.io/delphi-epidata/api/geographic_codes.html#selected-us-cities.

Value

`tibble::tibble`

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_gft(locations = "hhs1", epiweeks = epirange(201201, 202001))
```

`pub_kcdc_ili` *KCDC ILI incidence (Korea)*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/kcdc_ili.html

Usage

```
pub_kcdc_ili(
  regions,
  epiweeks = "*",
  ...,
  issues = NULL,
  lag = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

`regions`	character. List of regions to fetch. See South Korea's geographic codes # nolint for details.
`epiweeks`	`timeset`. Epiweeks to fetch. Supports `epirange()` and defaults to all ("*") dates. Format as `epirange(startweek, endweek)`, where startweek and endweek are of the form YYYYWW (string or numeric).
`...`	not used for values, forces later arguments to bind by name
`issues`	`timeset`. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
`lag`	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
`fetch_args`	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Value

`tibble::tibble`

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- `as_of`: (Date) Retrieve the data as it was on this date.
- `issues`: `timeset` Retrieve data from a specific issue date or range of dates.
- `lag`: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See `vignette("versioned-data")` for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_kcdc_ili(regions = "ROK", epiweeks = 200436)
```

---

pub_meta *Metadata for the Delphi Epidata API*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the `epidata_snapshot()`, `epidata_archive()`, and `epidata_meta()` functions. For more details on the changes, refer to `vignette("migration-guide")`, and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/meta.html

Usage

```
pub_meta(fetch_args = fetch_args_list())
```

Arguments

`fetch_args`	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Value

`list`

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

`pub_nidss_dengue` *NIDSS dengue cases (Taiwan)*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/nidss_dengue.html

Obtains counts of confirmed dengue cases in Taiwan from Taiwan National Infectious Disease Statistical System.

Usage

```
pub_nidss_dengue(locations, epiweeks = "*", fetch_args = fetch_args_list())
```

Arguments

`locations`	character. List of locations to fetch. See Taiwan's geographic codes for details.
`epiweeks`	`timeset`. Epiweeks to fetch. Supports `epirange()` and defaults to all ("*") dates. Format as `epirange(startweek, endweek)`, where startweek and endweek are of the form YYYYWW (string or numeric).
`fetch_args`	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Details

Possible location inputs can be found in https://github.com/cmu-delphi/delphi-epidata/blob/main/labels/nidss_regions.txt and https://github.com/cmu-delphi/delphi-epidata/blob/main/labels/nidss_locations.txt.

Value

`tibble::tibble`

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_nidss_dengue(locations = "taipei", epiweeks = epirange(201201, 201301))
```

pub_nidss_flu *NIDSS flu doctor visits (Taiwan)*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/nidss_flu.html

Obtains information on outpatient inluenza-like-illness from Taiwan National Infectious Disease Statistical System.

Usage

```
pub_nidss_flu(
  regions,
  epiweeks = "*",
  ...,
  issues = NULL,
  lag = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

regions	character. List of regions to fetch. See Taiwan's geographic codes for details.
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
...	not used for values, forces later arguments to bind by name
issues	timeset. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
lag	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- as_of: (Date) Retrieve the data as it was on this date.
- issues: timeset Retrieve data from a specific issue date or range of dates.

- lag: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See vignette("versioned-data") for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
pub_nidss_flu(regions = "taipei", epiweeks = epirange(201501, 201601))
```

---

pub_nowcast *Delphi's ILI Nearby nowcasts*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/nowcast.html.

Obtains information on outpatient inluenza-like-illness (ILI) from Delphi's

Usage

```
pub_nowcast(locations, epiweeks = "*", fetch_args = fetch_args_list())
```

Arguments

locations	character. List of locations to fetch.
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Details

The full list of location inputs can be accessed at https://cmu-delphi.github.io/delphi-epidata/api/geographic_codes.html#us-regions-and-states and https://cmu-delphi.github.io/delphi-epidata/api/geographic_codes.html#fluview-cities.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_nowcast(locations = "ca", epiweeks = epirange(201201, 201301))
```

---

pub_paho_dengue *PAHO dengue data (North and South America)*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/paho_dengue.html

Usage

```
pub_paho_dengue(
  regions,
  epiweeks = "*",
  ...,
  issues = NULL,
  lag = NULL,
  fetch_args = fetch_args_list()
)
```

Arguments

regions	character. List of regions to fetch. See Americas' geographic codes # nolint for details.
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
...	not used for values, forces later arguments to bind by name
issues	timeset. Optionally, the issue(s) of the data to fetch. See the "Data Versioning" section for details.
lag	integer. Optionally, the lag of the issues to fetch. See the "Data Versioning" section for details.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

Data Versioning

Several endpoints support retrieving historical versions of the data. The following parameters control this and are mutually exclusive (only one can be provided at a time).

- `as_of`: (Date) Retrieve the data as it was on this date.
- `issues`: `timeset` Retrieve data from a specific issue date or range of dates.
- `lag`: (integer) Retrieve data with a specific lag from its issue date.

If none of these is specified, the most recent version of the data is returned.

See `vignette("versioned-data")` for details and more ways to specify versioned data.

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
pub_paho_dengue(regions = "ca", epiweeks = epirange(201401, 201501))
```

---

pub_wiki *Wikipedia webpage counts by article*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/wiki.html Number of page visits for selected English, Influenza-related wikipedia articles.

- Source: Wikimedia
- Temporal Resolution: Hourly, daily, and weekly from 2007-12-09 (2007w50)
- Spatial Resolution: N/A
- Other resolution: By article (54)
- Open access

Usage

```
pub_wiki(
  articles,
  ...,
  time_type = c("day", "week"),
  time_values = "*",
  hours = NULL,
  language = "en",
  fetch_args = fetch_args_list()
)
```

Arguments

articles	character. Articles to fetch. See available articles for details.
...	not used for values, forces later arguments to bind by name
time_type	string. The temporal resolution of the data (either "day" or "week", depending on signal).
time_values	timeset. Dates or epiweeks to fetch. Supports epirange() and defaults to all ("*") dates.
hours	integer. Optionally, the hours to fetch.
language	string. Language to fetch.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
pub_wiki(
  articles = "avian_influenza",
  time_type = "week",
  time_values = epirange(201501, 201601)
)
```

---

pvt_cdc *CDC total and by topic webpage visits*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/cdc.html

Usage

```
pvt_cdc(auth, locations, epiweeks = "*", fetch_args = fetch_args_list())
```

Arguments

auth	string. Your restricted access key (not the same as API key).
locations	character. List of locations to fetch. See US Regions and States codes # nolint for details.
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
## Not run:
pvt_cdc(
  auth = Sys.getenv("DELPHI_EPIDATA_KEY"),
  locations = "fl,ca",
  epirange(201501, 201601)
)

## End(Not run)
```

---

pvt_dengue_sensors *PAHO dengue digital surveillance sensors (North and South America)*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/dengue_sensors.html

Usage

```
pvt_dengue_sensors(
  auth,
  names,
  locations,
  epiweeks = "*",
  fetch_args = fetch_args_list()
)
```

Arguments

auth	string. Your restricted access key (not the same as API key).
names	character. List of sensor names to fetch. See the available sensors.
locations	character. List of locations to fetch. See the codes for countries and territories in the Americas. # nolint
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
## Not run:
pvt_dengue_sensors(
  auth = Sys.getenv("DELPHI_EPIDATA_KEY"),
  names = "ght",
  locations = "ag",
  epiweeks = epirange(201501, 202001)
)

## End(Not run)
```

---

pvt_ght *Google Health Trends health topics search volume*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/ght.html

Estimate of influenza activity based on volume of certain search queries. ...

Usage

```
pvt_ght(auth, locations, epiweeks = "*", query, fetch_args = fetch_args_list())
```

Arguments

auth	string. Your restricted access key (not the same as API key).
locations	character. List of locations to fetch. See geographic codes # nolint for details.
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
query	string. The query to be fetched.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
## Not run:
pvt_ght(
  auth = Sys.getenv("DELPHI_EPIDATA_KEY"),
  locations = "ma",
  epiweeks = epirange(199301, 202304),
  query = "how to get over the flu"
)

## End(Not run)
```

---

pvt_meta_norostat *Metadata for the NoroSTAT endpoint*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/meta_norostat.html

Usage

```
pvt_meta_norostat(auth, fetch_args = fetch_args_list())
```

Arguments

auth	string. Your restricted access key (not the same as API key).
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

list

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

See Also

`pvt_norostat()`

Examples

```
## Not run:
pvt_meta_norostat(auth = Sys.getenv("DELPHI_EPIDATA_KEY"))

## End(Not run)
```

---

pvt_norostat *CDC NoroSTAT norovirus outbreaks*

Description

This is point data only, and does not include minima or maxima.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/norostat.html

This is the documentation of the API for accessing the NoroSTAT endpoint of the Delphi's epidemiological data.

Usage

```
pvt_norostat(auth, locations, epiweeks = "*", fetch_args = fetch_args_list())
```

Arguments

auth	string. Your restricted access key (not the same as API key).
locations	character. Locations to fetch. Only a specific list of full state names are permitted. See the `locations` column in the output of `pvt_meta_norostat()` for the allowed values.
epiweeks	timeset. Epiweeks to fetch. Supports `epirange()` and defaults to all ("*") dates. Format as `epirange(startweek, endweek)`, where startweek and endweek are of the form YYYYWW (string or numeric).
fetch_args	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
## Not run:
pvt_norostat(
  auth = Sys.getenv("DELPHI_EPIDATA_KEY"),
  locations = "Minnesota, Ohio, Oregon, Tennessee, and Wisconsin",
  epiweeks = 201233
)

## End(Not run)
```

---

pvt_quidel *Quidel COVID-19 and influenza testing data*

Description

This is a V4 endpoint. Starting in October 2026, it is tentatively deprecated in favor of the V5 API. The new API can be accessed via the `epidata_snapshot()`, `epidata_archive()`, and `epidata_meta()` functions. For more details on the changes, refer to `vignette("migration-guide")`, and visit the V5 signals documentation to see which sources are currently available.

API docs: https://cmu-delphi.github.io/delphi-epidata/api/quidel.html

Data provided by Quidel Corp., which contains flu lab test results.

Usage

```
pvt_quidel(auth, locations, epiweeks = "*", fetch_args = fetch_args_list())
```

Arguments

`auth`	string. Your restricted access key (not the same as API key).
`locations`	character. List of locations to fetch. See HHS regions' codes for details.
`epiweeks`	`timeset`. Epiweeks to fetch. Supports `epirange()` and defaults to all ("*") dates. Format as `epirange(startweek, endweek)`, where startweek and endweek are of the form YYYYWW (string or numeric).
`fetch_args`	`fetch_args_list()`. Additional arguments to pass to `fetch()`. See `fetch_args_list()` for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see `vignette("signal-discovery", package = "epidatr")`.

Examples

```
## Not run:
pvt_quidel(
  auth = Sys.getenv("DELPHI_EPIDATA_KEY"),
  epiweeks = epirange(201201, 202001),
  locations = "hhs1"
)

## End(Not run)
```

---

pvt_sensors *Influenza and dengue digital surveillance sensors*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/sensors.html

This is the documentation of the API for accessing the Digital Surveillance Sensors endpoint of the Delphi's epidemiological. Note: this repository was built to support modeling and forecasting efforts surrounding seasonal influenza (and dengue). In the current COVID-19 pandemic, syndromic surveillance data, like ILI data (influenza-like illness) through FluView, will likely prove very useful. However, we urge caution to users examining the digital surveillance sensors, like ILI Nearby, Google Flu Trends, etc., during the COVID-19 pandemic, because these were designed to track ILI as driven by seasonal influenza, and were NOT designed to track ILI during the COVID-19 pandemic.

Usage

```
pvt_sensors(
  auth,
  names,
  locations,
  epiweeks = "*",
  fetch_args = fetch_args_list()
)
```

Arguments

auth	string. Your restricted access key (not the same as API key).
names	character. List of sensor names to fetch. See the data sources available for details.
locations	character. List of locations to fetch. See the codes of the US regions and states # nolint for details.
epiweeks	timeset. Epiweeks to fetch. Supports epirange() and defaults to all ("*") dates. Format as epirange(startweek, endweek), where startweek and endweek are of the form YYYYWW (string or numeric).
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
## Not run:
pvt_sensors(
  auth = Sys.getenv("DELPHI_EPIDATA_KEY"),
  names = "sar3",
  locations = "nat",
  epiweeks = epirange(201501, 202001)
)

## End(Not run)
```

---

pvt_twitter *HealthTweets total and influenza-related tweets*

Description

API docs: https://cmu-delphi.github.io/delphi-epidata/api/twitter.html

This is the API documentation for accessing the Twitter Stream endpoint of Delphi's epidemiological data. Sourced from Healthtweets

Usage

```
pvt_twitter(
  auth,
  locations,
  ...,
  time_type = c("day", "week"),
  time_values = "*",
  fetch_args = fetch_args_list()
)
```

Arguments

auth	string. Your restricted access key (not the same as API key).
locations	character. List of locations to fetch. See the codes of the US regions and states # nolint for details.
...	not used for values, forces later arguments to bind by name
time_type	string. The temporal resolution of the data (either "day" or "week", depending on signal).
time_values	timeset. Dates or epiweeks to fetch. Supports epirange() and defaults to all ("*") dates.
fetch_args	fetch_args_list(). Additional arguments to pass to fetch(). See fetch_args_list() for details.

Value

tibble::tibble

See also

For example queries showing how to discover signals and build calls, see vignette("signal-discovery", package = "epidatr").

Examples

```
## Not run:
pvt_twitter(
  auth = Sys.getenv("DELPHI_EPIDATA_KEY"),
  locations = "CA",
  time_type = "week",
  time_values = epirange(201501, 202001)
)

## End(Not run)
```

set_cache *Create or renew a cache for this session*

Description

By default, epidatr re-requests data from the API on every call of `fetch`. In case you find yourself repeatedly calling the same data, you can enable the cache using either this function for a given session, or environmental variables for a persistent cache. The typical recommended workflow for using the cache is to set the environmental variables `EPIDATR_USE_CACHE=TRUE` and `EPIDATR_CACHE_DIRECTORY="/your/directory/here"`in your `.Renviron`, for example by calling `file.edit("~/.Renviron")`. See the parameters below for some more configurables if you're so inclined.

`set_cache` (re)defines the cache to use in a particular R session. This does not clear existing data at any previous location, but instead creates a handle to the new cache using cachem that seamlessly handles caching for you. Say your cache is normally stored in some default directory, but for the current session you want to save your results in `~/my/temporary/savedirectory`, then you would call `set_cache(dir = "~/my/temporary/savedirectory")`. Or if you know the data from 2 days ago is wrong, you could call `set_cache(days = 1)` to clear older data whenever the cache is referenced. In both cases, these changes would only last for a single session (though the deleted data would be gone permanently!).

An important feature of the caching in this package is that only calls which specify either `issues` before a certain date, or `as_of` before a certain date will actually cache. For example the call

```
pub_covidcast(
  source = "jhu-csse",
  signals = "confirmed_7dav_incidence_prop",
  geo_type = "state",
  time_type = "day",
  geo_values = "ca,fl",
  time_values = epirange(20200601, 20230801)
)
```

won't cache, since it is possible for the cache to be invalidated by new releases with no warning. On the other hand, the call

```
pub_covidcast(
  source = "jhu-csse",
  signals = "confirmed_7dav_incidence_prop",
  geo_type = "state",
  time_type = "day",
  geo_values = "ca,fl",
  time_values = epirange(20200601, 20230801),
  as_of = "2023-08-01"
)
```

will cache, since normal new versions of data can't invalidate it (since they would be `as_of` a later date). It is still possible that Delphi may patch such data, but the frequency is on the order of months rather than days. We are working on creating a public channel to communicate such updates. While specifying `issues` will usually cache, a call with `issues="*"` won't cache, since its subject to cache invalidation by normal versioning.

On the backend, the cache uses cachem, with filenames generated using an md5 encoding of the call url. Each file corresponds to a unique epidata-API call.

Usage

```
set_cache(
  cache_dir = NULL,
  days = NULL,
  max_size = NULL,
  logfile = NULL,
  confirm = TRUE,
  startup = FALSE
)
```

Arguments

`cache_dir`	the directory in which the cache is stored. By default, this is `rappdirs::user_cache_dir("R", version = "epidatr")`. The path can be either relative or absolute. The environmental variable is `EPIDATR_CACHE_DIR`.
`days`	the maximum length of time in days to keep any particular cached call. By default this is `1`. The environmental variable is `EPIDATR_CACHE_MAX_AGE_DAYS`.
`max_size`	the size of the entire cache, in MB, at which to start pruning entries. By default this is `1024`, or 1GB. The environmental variable is `EPIDATR_CACHE_MAX_SIZE_MB`.
`logfile`	where cachem's log of transactions is stored, relative to the cache directory. By default, it is `"logfile.txt"`. The environmental variable is `EPIDATR_CACHE_LOGFILE`.
`confirm`	whether to confirm directory creation. default is `TRUE`; should only be set in non-interactive scripts
`startup`	indicates whether the function is being called on startup. Affects suppressability of the messages. Default is `FALSE`.

Value

`NULL` no return value, all effects are stored in the package environment

See Also

`clear_cache` to delete the old cache while making a new one, `disable_cache` to disable without deleting, and `cache_info`

Examples

```
set_cache(
  cache_dir = tempdir(),
  days = 14,
  max_size = 512,
  logfile = "logs.txt"
)
```

`timeset` *Timeset formats for specifying dates*

Description

Many API calls accept timesets to specify the time ranges of data being requested. Timesets can be specified with `epirange()`, as `Date` objects, or with wildcards.

Details

Timesets are not special R types; the term simply describes any value that is accepted by epidatr to specify the time value of an epidata query:

- Dates: `Date` instances.
- Date strings or integers: Strings or integers in the format YYYYMMDD.
- Epiweeks: Strings or integers in the format YYYYWW, where WW is the epiweek number.
- EpiRanges: A range returned by `epirange()`, or a list of multiple ranges.
- Wildcard: The string `"*"`, which requests all available time values.

Refer to the specific endpoint documentation for guidance on using dates vs weeks. Most endpoints support only one or the other. Some (less commonly used) endpoints may not accept the `"*"` wildcard, but this can be simulated with a large `epirange()`.

Index

∗ **endpoint**
  cast_api_queries, 4
  epidata_aux, 11
  epidata_meta, 13
  pub_covid_hosp_facility, 20
  pub_covid_hosp_facility_lookup, 22
  pub_covid_hosp_state_timeseries, 23
  pub_covidcast, 17
  pub_covidcast_meta, 19
  pub_delphi, 24
  pub_dengue_nowcast, 25
  pub_ecdc_ili, 26
  pub_flusurv, 27
  pub_fluview, 29
  pub_fluview_clinical, 31
  pub_fluview_meta, 32
  pub_gft, 33
  pub_kcdc_ili, 34
  pub_meta, 35
  pub_nidss_dengue, 36
  pub_nidss_flu, 37
  pub_nowcast, 38
  pub_paho_dengue, 39
  pub_wiki, 40
  pvt_cdc, 41
  pvt_dengue_sensors, 42
  pvt_ght, 43
  pvt_meta_norostat, 44
  pvt_norostat, 45
  pvt_quidel, 46
  pvt_sensors, 47
  pvt_twitter, 48
?Startup, *16*

avail_endpoints, 3

cache_info, 4, *8*, *11*, *51*
cast_api_queries, 4
clear_cache, *4*, 7, *11*, *51*
covidcast_epidata, 8
covidcast_epidata(), *18*, *20*
create_epidata_call, 9

disable_cache, *4*, *8*, 11, *51*

epidata *(*cast_api_queries*)*, 4
epidata(), *13*
epidata_archive *(*cast_api_queries*)*, 4
epidata_archive(), *11–13*, *17*, *19*, *27*, *29*, *31*, *32*, *35*, *46*
epidata_aux, 11
epidata_call *(*create_epidata_call*)*, 9
epidata_meta, 13
epidata_meta(), *5–7*, *13*, *17*, *19*, *27*, *29*, *31*, *32*, *35*, *46*
epidata_snapshot *(*cast_api_queries*)*, 4
epidata_snapshot(), *11–13*, *17*, *19*, *27*, *29*, *31*, *32*, *35*, *46*
epirange, 14
epirange(), *6*, *7*, *12*, *13*, *18*, *21*, *23*, *25*, *26*, *28*, *30*, *31*, *33*, *34*, *36–39*, *41–46*, *48*, *49*

fetch *(*create_epidata_call*)*, 9
fetch_args *(*fetch_args_list*)*, 14
fetch_args_list, 14
fetch_args_list(), *6*, *12*, *13*, *18*, *20–23*, *25*, *26*, *28*, *30–39*, *41–46*, *48*, *49*

get_api_key, 16

list, *4*, *25*, *35*, *45*

NULL, *7*, *11*, *51*

pub_covid_hosp_facility, 20
pub_covid_hosp_facility(), *21–23*
pub_covid_hosp_facility_lookup, 22
pub_covid_hosp_facility_lookup(), *20*
pub_covid_hosp_state_timeseries, 23

pub_covidcast, *9, 10*, 17
pub_covidcast(), *20*
pub_covidcast_meta, 19
pub_covidcast_meta(), *18*
pub_delphi, 24
pub_dengue_nowcast, 25
pub_ecdc_ili, 26
pub_flusurv, 27
pub_fluview, 29
pub_fluview(), *33*
pub_fluview_clinical, 31
pub_fluview_meta, 32
pub_gft, 33
pub_kcdc_ili, 34
pub_meta, 35
pub_nidss_dengue, 36
pub_nidss_flu, 37
pub_nowcast, 38
pub_paho_dengue, 39
pub_wiki, 40
pvt_cdc, 41
pvt_dengue_sensors, 42
pvt_ght, 43
pvt_meta_norostat, 44
pvt_norostat, 45
pvt_norostat(), *45*
pvt_quidel, 46
pvt_sensors, 47
pvt_twitter, 48

save_api_key *(*get_api_key*)*, 16
set_cache, *4*, *7, 8*, *11*, 50
Sys.setenv(), *16*

tibble::tibble, *3*, *6*, *12*, *18*, *20–22*, *24, 25*, *27, 28*, *30–34*, *36–39*, *41–44*, *46–49*
timeset, *6*, *12*, *18*, *21*, *23–28*, *30–46*, *48, 49*, 52