Package 'covidcast'

July 13, 2023

Type Package

Title Client for Delphi's 'COVIDcast Epidata' API

Version 0.5.2

Date 2023-07-11

URL https://cmu-delphi.github.io/covidcast/covidcastR/, https://github.com/cmu-delphi/covidcast

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

Description Tools for Delphi's 'COVIDcast Epidata' API: data access, maps and time series plotting, and basic signal processing. The API includes a collection of numerous indicators relevant to the COVID-19 pandemic in the United States, including official reports, de-identified aggregated medical claims data, large-scale surveys of symptoms and public behavior, and mobility data, typically updated daily and at the county level. All data sources are documented at <https://cmu-delphi.github.io/delphi-epidata/api/covidcast.html>.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr, ggplot2, grDevices, httr, MMWRweek, purrr, rlang, sf, tidyr, xml2

RoxygenNote 7.2.3

Suggests gridExtra, httptest, knitr, mockery, rmarkdown, testthat, tibble, vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Taylor Arnold [aut], Jacob Bien [aut], Logan Brooks [aut],

Sarah Colquhoun [aut],
David Farrow [aut],
Jed Grabman [ctb],
Pedrito Maynard-Zhang [ctb],
Kathryn Mazaitis [aut],
Alex Reinhart [aut, cre] (<https://orcid.org/0000-0002-6658-514X>),
Ryan Tibshirani [aut]

Maintainer Alex Reinhart <areinhar@stat.cmu.edu>

Repository CRAN

Date/Publication 2023-07-12 23:40:06 UTC

R topics documented:

`abbr_to_fips` *Get FIPS codes from state abbreviations*

Description

Look up FIPS codes by state abbreviations (including District of Columbia and Puerto Rico); this function is based on `grep()`, and hence allows for regular expressions.

Usage

```
abbr_to_fips(
  abbr,
  ignore.case = TRUE,
  perl = FALSE,
  fixed = FALSE,
  ties_method = c("first", "all")
)
```

Arguments

abbr	Vector of state abbreviations to look up.
ignore.case, perl, fixed	Arguments to pass to grep(), with the same defaults as in the latter function, except for ignore.case = TRUE. Hence, by default, regular expressions are used; to match against a fixed string (no regular expressions), set fixed = TRUE.
ties_method	If "first", then only the first match for each name is returned. If "all", then all matches for each name are returned.

Value

A vector of FIPS codes if ties_method equals "first", and a list of FIPS codes otherwise. These FIPS codes have five digits (ending in "000").

See Also

abbr_to_name()

Examples

```
abbr_to_fips("PA")
abbr_to_fips(c("PA", "PR", "DC"))

# Note that name_to_fips() works for state names too:
name_to_fips("^Pennsylvania$")
```

---

abbr_to_name *Get state names from state abbreviations*

Description

Look up state names by state abbreviations (including District of Columbia and Puerto Rico); this function is based on grep(), and hence allows for regular expressions.

Usage

```
abbr_to_name(
  abbr,
  ignore.case = FALSE,
  perl = FALSE,
  fixed = FALSE,
  ties_method = c("first", "all")
)
```

Arguments

abbr	Vector of state abbreviations to look up.
ignore.case, perl, fixed	Arguments to pass to grep(), with the same defaults as in the latter function. Hence, by default, regular expressions are used; to match against a fixed string (no regular expressions), set fixed = TRUE.
ties_method	If "first", then only the first match for each name is returned. If "all", then all matches for each name are returned.

Value

A vector of state names if ties_method equals "first", and a list of state names otherwise.

See Also

name_to_abbr()

Examples

```
abbr_to_name("PA")
abbr_to_name(c("PA", "PR", "DC"))
```

---

aggregate_signals *Aggregate* covidcast_signal *objects into one data frame*

Description

Aggregates covidcast_signal objects into one data frame, in either "wide" or "long" format. (In "wide" aggregation, only the latest issue from each data frame is retained, and several columns, including data_source and signal are dropped; see details below). See vignette("multi-signals", package = "covidcast") for examples.

Usage

```
aggregate_signals(x, dt = NULL, format = c("wide", "long"))
```

Arguments

x	Single covidcast_signal data frame, or a list of such data frames, such as is returned by covidcast_signals().
dt	Vector of shifts to apply to the values in the data frame x. Negative shifts translate into in a lag value and positive shifts into a lead value; for example, if dt = -1, then the value on June 2 that gets reported is the original value on June 1; if dt = 0, then the values are left as is. When x is a list of data frames, dt can either be a single vector of shifts or a list of vectors of shifts, this list having the same length as x (in order to apply, respectively, the same shifts or a different set of shifts to each data frame in x).
format	One of either "wide" or "long". The default is "wide".

Details

This function can be thought of having three use cases. In all three cases, the result will be a new data frame in either "wide" or "long" format, depending on format.

The first use case is to apply time-shifts to the values in a given covidcast_signal object. In this use case, x is a covidcast_signal data frame and dt is a vector of shifts.

The second use case is to bind together, into one data frame, signals that are returned by covidcast_signals(). In this use case, x is a list of covidcast_signal data frames, and dt is NULL.

The third use case is a combination of the first two: to bind together signals returned by covidcast_signals(), and simultaneously, apply time-shifts to their values. In this use case, x is a list of covidcast_signal data frames, and dt is either a vector of shifts—to apply the same shifts for each signal in x, or a list of vector of shifts—to apply different shifts for each signal in x.

Value

Data frame of aggregated signals in "wide" or "long" form, depending on format. In "long" form, an extra column dt is appended to indicate the value of the time-shift. In "wide" form, only the latest issue of data is retained; the returned data frame is formed via full joins of the input data frames (on geo_value and time_value as the join key), and the columns data_source, signal, issue, lag, stderr, sample_size are all dropped from the output. Each unique signal—defined by a combination of data source name, signal name, and time-shift—is given its own column, whose name indicates its defining quantities. For example, the column name "value+2:usa-facts_confirmed_incidence_num" corresponds to a signal defined by data_source = "usa-facts", signal = "confirmed_incidence_num", and dt = 2.

See Also

covidcast_wider(), covidcast_longer()

`as.covidcast_signal` *Convert data from an external source into a form compatible with* `covidcast_signal`.

Description

Several methods are provided to convert common objects (such as data frames) into `covidcast_signal` objects, which can be used with the various `covidcast_signal` methods (such as `plot.covidcast_signal()` or `covidcast_cor()`). See `vignette("external-data")` for examples.

Usage

```
as.covidcast_signal(x, ...)

## S3 method for class 'covidcast_signal'
as.covidcast_signal(x, ...)

## S3 method for class 'data.frame'
as.covidcast_signal(
  x,
  signal = NULL,
  geo_type = c("county", "msa", "hrr", "dma", "state", "hhs", "nation"),
  time_type = c("day", "week"),
  data_source = "user",
  issue = NULL,
  metadata = list(),
  ...
)
```

Arguments

`x`	Object to be converted. See Methods section below for details on formatting of each input type.
`...`	Additional arguments passed to methods.
`signal`	The signal name to use for this data.
`geo_type`	The geography type stored in this object.
`time_type`	The time resolution stored in this object. If "day", the default, each observation covers one day. If "week", each time value is assumed to be the start date of the epiweek (MMWR week) that the data represents.
`data_source`	The name of the data source to use as a label for this data.
`issue`	Issue date to use for this data, if not present in `x`, as a `Date` object. If no issue date is present in `x` and `issue` is `NULL`, today's date will be used.
`metadata`	List of metadata to attach to the `covidcast_signal` object. See the "Metadata" section of `covidcast_signal()`. All objects will have `geo_type`, `data_source`, and `signal` columns included in their metadata; named entries in this list are added as additional columns.

Value

covidcast_signal object; see covidcast_signal() for documentation of fields and structure.

Methods (by class)

- as.covidcast_signal(covidcast_signal): Simply returns the covidcast_signal object unchanged.
- as.covidcast_signal(data.frame): The input data frame x must contain the columns time_value, value, and geo_value. If an issue column is present in x, it will be used as the issue date for each observation; if not, the issue argument will be used. Other columns will be preserved as-is.

See Also

covidcast_signal()

---

county_census *County census population data*

Description

Data set on county populations, from the 2019 US Census.

Usage

```
county_census
```

Format

A data frame with 3193 rows, one for each county (along with the 50 states and DC). Columns include:

SUMLEV Geographic summary level. Either 40 (state) or 50 (county).

REGION Census Region code

DIVISION Census Division code

STATE State FIPS code.

COUNTY County FIPS

STNAME Name of the state in which this county belongs.

CTYNAME County name, to help find counties by name.

POPESTIMATE2019 Estimate of the county's resident population as of July 1, 2019.

FIPS Five-digit county FIPS codes. These are unique identifiers used, for example, as the geo_values argument to covidcast_signal() to request data from a specific county.

Source

United States Census Bureau, at https://www2.census.gov/programs-surveys/popest/datasets/2010-2019/counties/totals/co-est2019-alldata.csv

References

Census Bureau documentation of all columns and their meaning: https://www2.census.gov/programs-surveys/popest/datasets/2010-2019/counties/totals/co-est2019-alldata.pdf, https://www.census.gov/data/tables/time-series/demo/popest/2010s-total-puerto-rico-municipios.html, and https://www.census.gov/data/tables/2010/dec/2010-island-areas.html

See Also

county_fips_to_name(), name_to_fips()

---

covidcast_cor *Compute correlations between two* covidcast_signal *data frames*

Description

Computes correlations between two covidcast_signal data frames, allowing for slicing by geo location, or by time. (Only the latest issue from each data frame is used for correlations.) See the correlations vignette for examples: vignette("correlation-utils", package = "covidcast").

Usage

```
covidcast_cor(
  x,
  y,
  dt_x = 0,
  dt_y = 0,
  by = c("geo_value", "time_value"),
  use = "na.or.complete",
  method = c("pearson", "kendall", "spearman")
)
```

Arguments

x, y	The covidcast_signal data frames to correlate.
dt_x, dt_y	Time shifts (in days) to consider for x and y, respectively, before computing correlations. Default is 0. Negative shifts translate into in a lag value and positive shifts into a lead value; for example, setting dt_y = 2 results in values of y being shifted earlier (leading) by 2 days before correlation, so values of x are correlated with values of y from two days later.

by	If "geo_value", then correlations are computed for each geo location, over all time. Each correlation is measured between two time series at the same location. If "time_value", then correlations are computed for each time, over all geo locations. Each correlation is measured between all locations at one time. Default is "geo_value".
use, method	Arguments to pass to cor(), with "na.or.complete" the default for use (different than cor()) and "pearson" the default for method (same as cor()).

Value

A data frame with first column geo_value or time_value (matching by), and second column value, which gives the correlation.

Examples

```
## Not run:
# For all these examples, let x and y be two signals measured at the county
# level over several months.

## `by = "geo_value"`
# Correlate each county's time series together, returning one correlation per
# county:
covidcast_cor(x, y, by = "geo_value")

# Correlate x in each county with values of y 14 days later
covidcast_cor(x, y, dt_y = 14, by = "geo_value")

# Equivalently, x can be shifted -14 days:
covidcast_cor(x, y, dt_x = -14, by = "geo_value")

## `by = "time_value"`
# For each date, correlate x's values in every county against y's values in
# the same counties. Returns one correlation per date:
covidcast_cor(x, y, by = "time_value")

# Correlate x values across counties against y values 7 days later
covidcast_cor(x, y, dt_y = 7, by = "time_value")

## End(Not run)
```

---

covidcast_longer *Pivot aggregated signals between "wide" and "long" formats*

Description

These functions take signals returned from aggregate_signals() and convert between formats. covidcast_longer() takes the output of aggregate_signals(..., format = "wide") and converts it to "long" format, while covidcast_wider() takes the output of aggregate_signals(..., format = "long") and converts it to "wide" format.

Usage

```
covidcast_longer(x)

covidcast_wider(x)
```

Arguments

x	A covidcast_signal_wide or covidcast_signal_long object, as returned from aggregate_signals() with the respective format argument.

Value

The object pivoted into the opposite form, i.e. as if aggregate_signals() had been called in the first place with that format argument.

See Also

covidcast_signals()

---

covidcast_meta *Obtain COVIDcast metadata*

Description

Obtains a data frame of metadata describing all publicly available data streams from the COVIDcast API.

Usage

```
covidcast_meta()
```

Value

Data frame containing one row per signal, with the following columns:

data_source	Data source name.
signal	Signal name.
min_time	First day for which this signal is available.
max_time	Most recent day for which this signal is available.
geo_type	Geographic level for which this signal is available, such as county, state, msa, or hrr. Most signals are available at multiple geographic levels and will hence be listed in multiple rows with their own metadata.
time_type	Temporal resolution at which this signal is reported. "day", for example, means the signal is reported daily.

num_locations	Number of distinct geographic locations available for this signal. For example, if geo_type is county, the number of counties for which this signal has ever been reported.
min_value	Smallest value that has ever been reported.
max_value	Largest value that has ever been reported.
mean_value	Arithmetic mean of all reported values.
stdev_value	Sample standard deviation of all reported values.
max_issue	Most recent issue date for this signal.
min_lag	Smallest lag from observation to issue, in time_type units
max_lag	Largest lag from observation to issue, in time_type units

References

COVIDcast API sources and signals documentation: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html

See Also

summary.covidcast_meta()

---

covidcast_signal *Obtain a data frame for one COVIDcast signal*

Description

Obtains data for selected date ranges for all geographic regions of the United States. Available data sources and signals are documented in the COVIDcast signal documentation. Most (but not all) data sources are available at the county level, but the API can also return data aggregated to metropolitan statistical areas, hospital referral regions, or states, as desired, by using the geo_type argument.

Usage

```
covidcast_signal(
  data_source,
  signal,
  start_day = NULL,
  end_day = NULL,
  geo_type = c("county", "hrr", "msa", "dma", "state", "hhs", "nation"),
  geo_values = "*",
  as_of = NULL,
  issues = NULL,
  lag = NULL,
  time_type = c("day", "week")
)
```

Arguments

data_source	String identifying the data source to query. See https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html for a list of available data sources.
signal	String identifying the signal from that source to query. Again, see https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html for a list of available signals.
start_day	Query data beginning on this date. Date object, or string in the form "YYYY-MM-DD". If start_day is NULL, defaults to first day data is available for this signal.
end_day	Query data up to this date, inclusive. Date object or string in the form "YYYY-MM-DD". If end_day is NULL, defaults to the most recent day data is available for this signal.
geo_type	The geography type for which to request this data, such as "county" or "state". Defaults to "county". See https://cmu-delphi.github.io/delphi-epidata/api/covidcast_geography.html for details on which types are available.
geo_values	Which geographies to return. The default, "*", fetches all geographies. To fetch specific geographies, specify their IDs as a vector or list of strings. See https://cmu-delphi.github.io/delphi-epidata/api/covidcast_geography.html for details on how to specify these IDs.
as_of	Fetch only data that was available on or before this date, provided as a Date object or string in the form "YYYY-MM-DD". If NULL, the default, return the most recent available data. Note that only one of as_of, issues, and lag should be provided; it does not make sense to specify more than one. For more on data revisions, see "Issue dates and revisions" below.
issues	Fetch only data that was published or updated ("issued") on these dates. Provided as either a single Date object (or string in the form "YYYY-MM-DD"), indicating a single date to fetch data issued on, or a vector specifying two dates, start and end. In this case, return all data issued in this range. There may be multiple rows for each observation, indicating several updates to its value. If NULL, the default, return the most recently issued data.
lag	Integer. If, for example, lag = 3, then we fetch only data that was published or updated exactly 3 days after the date. For example, a row with time_value of June 3 will only be included in the results if its data was issued or updated on June 6. If NULL, the default, return the most recently issued data regardless of its lag.
time_type	The temporal resolution to request this data. Most signals are available at the "day" resolution (the default); some are only available at the "week" resolution, representing an MMWR week ("epiweek").

Details

For data on counties, metropolitan statistical areas, and states, this package provides the county_census, msa_census, and state_census datasets. These include each area's unique identifier, used in the geo_values argument to select specific areas, and basic information on population and other Census data.

Downloading large amounts of data may be slow, so this function prints messages for each chunk of data it downloads. To suppress these, use `base::suppressMessages()`, as in `suppressMessages(covidcast_signal("fb-...))`.

Value

`covidcast_signal` object with matching data. The object is a data frame with additional metadata attached. Each row is one observation of one signal on one day in one geographic location. Contains the following columns:

`data_source`	Data source from which this observation was obtained.
`signal`	Signal from which this observation was obtained.
`geo_value`	String identifying the location, such as a state name or county FIPS code.
`time_value`	Date object identifying the date of this observation. For data with `time_type = "week"`, this is the first day of the corresponding epiweek.
`issue`	Date object identifying the date this estimate was issued. For example, an estimate with a `time_value` of June 3 might have been issued on June 5, after the data for June 3rd was collected and ingested into the API.
`lag`	Integer giving the difference between `issue` and `time_value`, in days.
`value`	Signal value being requested. For example, in a query for the "confirmed_cumulative_num" signal from the "usa-facts" source, this would be the cumulative number of confirmed cases in the area, as of the given `time_value`.
`stderr`	Associated standard error of the signal value, if available.
`sample_size`	Integer indicating the sample size available in that geography on that day; sample size may not be available for all signals, due to privacy or other constraints, in which case it will be `NA`.

Consult the signal documentation for more details on how values and standard errors are calculated for specific signals.

The returned data frame has a `metadata` attribute containing metadata about the signal contained within; see "Metadata" below for details.

Metadata

The returned object has a `metadata` attribute attached containing basic information about the signal. Use `attributes(x)$metadata` to access this metadata. The metadata is stored as a data frame of one row, and contains the same information that `covidcast_meta()` would return for a given signal.

Note that not all `covidcast_signal` objects may have all fields of metadata attached; for example, an object created with `as.covidcast_signal()` using data from another source may only contain the `geo_type` variable, along with `data_source` and `signal`. Before using the metadata of a `covidcast_signal` object, always check for the presence of the attributes you need.

Issue dates and revisions

The COVIDcast API tracks updates and changes to its underlying data, and records the first date each observation became available. For example, a data source may report its estimate for a specific state on June 3rd on June 5th, once records become available. This data is considered "issued" on June 5th. Later, the data source may update its estimate for June 3rd based on revised data, creating a new issue on June 8th. By default, `covidcast_signal()` returns the most recent issue available for every observation. The `as_of`, `issues`, and `lag` parameters allow the user to select specific issues instead, or to see all updates to observations. These options are mutually exclusive, and you should only specify one; if you specify more than one, you may get an error or confusing results.

Note that the API only tracks the initial value of an estimate and *changes* to that value. If a value was first issued on June 5th and never updated, asking for data issued on June 6th (using `issues` or `lag`) would *not* return that value, though asking for data `as_of` June 6th would. See `vignette("covidcast")` for examples.

Note also that the API enforces a maximum result row limit; results beyond the maximum limit are truncated. This limit is sufficient to fetch observations in all counties in the United States on one day. This client automatically splits queries for multiple days across multiple API calls. However, if data for one day has been issued many times, using the `issues` argument may return more results than the query limit. A warning will be issued in this case. To see all results, split your query across multiple calls with different `issues` arguments.

API keys

By default, `covidcast_signal()` submits queries to the API anonymously. All the examples in the package documentation are compatible with anonymous use of the API, but there are some limits on anonymous queries, including a rate limit. If you regularly query large amounts of data, please consider registering for a free API key, which lifts these limits. Even if your usage falls within the anonymous usage limits, registration helps us understand who and how others are using the Delphi Epidata API, which may in turn inform future research, data partnerships, and funding.

If you have an API key, you can use it by setting the `covidcast.auth` option once before calling `covidcast_signal()` or `covidcast_signals()`:

```
options(covidcast.auth = "your_api_key")

cli <- covidcast_signal(data_source = "fb-survey", signal = "smoothed_cli",
                        start_day = "2020-05-01", end_day = "2020-05-07",
                        geo_type = "state")
```

References

COVIDcast API documentation: https://cmu-delphi.github.io/delphi-epidata/api/covidcast.html

Documentation of all COVIDcast sources and signals: https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html

COVIDcast public dashboard: https://delphi.cmu.edu/covidcast/

See Also

plot.covidcast_signal(), covidcast_signals(), as.covidcast_signal(), county_census, msa_census, state_census

Examples

```
## Not run:
## Fetch all counties from 2020-05-10 to the most recent available data
covidcast_signal("fb-survey", "smoothed_cli", start_day = "2020-05-10")
## Fetch all counties on just 2020-05-10 and no other days
covidcast_signal("fb-survey", "smoothed_cli", start_day = "2020-05-10",
                 end_day = "2020-05-10")
## Fetch all states on 2020-05-10, 2020-05-11, 2020-05-12
covidcast_signal("fb-survey", "smoothed_cli", start_day = "2020-05-10",
                 end_day = "2020-05-12", geo_type = "state")
## Fetch all available data for just Pennsylvania and New Jersey
covidcast_signal("fb-survey", "smoothed_cli", geo_type = "state",
                 geo_values = c("pa", "nj"))
## Fetch all available data in the Pittsburgh metropolitan area
covidcast_signal("fb-survey", "smoothed_cli", geo_type = "msa",
                 geo_values = name_to_cbsa("Pittsburgh"))

## End(Not run)
```

---

covidcast_signals *Obtain multiple COVIDcast signals at once*

Description

This convenience function uses `covidcast_signal()` to obtain multiple signals, potentially from multiple data sources.

Usage

```
covidcast_signals(
  data_source,
  signal,
  start_day = NULL,
  end_day = NULL,
  geo_type = c("county", "hrr", "msa", "dma", "state", "hhs", "nation"),
  time_type = c("day", "week"),
  geo_values = "*",
  as_of = NULL,
  issues = NULL,
  lag = NULL
)
```

Arguments

data_source	String identifying the data source to query. See https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html for a list of available data sources.
signal	String identifying the signal from that source to query. Again, see https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html for a list of available signals.
start_day	Query data beginning on this date. Date object, or string in the form "YYYY-MM-DD". If start_day is NULL, defaults to first day data is available for this signal.
end_day	Query data up to this date, inclusive. Date object or string in the form "YYYY-MM-DD". If end_day is NULL, defaults to the most recent day data is available for this signal.
geo_type	The geography type for which to request this data, such as "county" or "state". Defaults to "county". See https://cmu-delphi.github.io/delphi-epidata/api/covidcast_geography.html for details on which types are available.
time_type	The temporal resolution to request this data. Most signals are available at the "day" resolution (the default); some are only available at the "week" resolution, representing an MMWR week ("epiweek").
geo_values	Which geographies to return. The default, "*", fetches all geographies. To fetch specific geographies, specify their IDs as a vector or list of strings. See https://cmu-delphi.github.io/delphi-epidata/api/covidcast_geography.html for details on how to specify these IDs.
as_of	Fetch only data that was available on or before this date, provided as a Date object or string in the form "YYYY-MM-DD". If NULL, the default, return the most recent available data. Note that only one of as_of, issues, and lag should be provided; it does not make sense to specify more than one. For more on data revisions, see "Issue dates and revisions" below.
issues	Fetch only data that was published or updated ("issued") on these dates. Provided as either a single Date object (or string in the form "YYYY-MM-DD"), indicating a single date to fetch data issued on, or a vector specifying two dates, start and end. In this case, return all data issued in this range. There may be multiple rows for each observation, indicating several updates to its value. If NULL, the default, return the most recently issued data.
lag	Integer. If, for example, lag = 3, then we fetch only data that was published or updated exactly 3 days after the date. For example, a row with time_value of June 3 will only be included in the results if its data was issued or updated on June 6. If NULL, the default, return the most recently issued data regardless of its lag.

Details

The argument structure is just as in covidcast_signal(), except the first four arguments data_source, signal, start_day, end_day are permitted to be vectors. The first two arguments data_source and signal are recycled appropriately in the calls to covidcast_signal(); see example below. The next two arguments start_day, end_day, unless NULL, must be either length 1 or N.

See `vignette("multi-signals")` for additional examples.

Value

A list of `covidcast_signal` data frames, of length `N = max(length(data_source), length(signal))`. This list can be aggregated into a single data frame of either "wide" or "long" format using `aggregate_signals()`.

See Also

`covidcast_signal()`, `aggregate_signals()`

Examples

```
## Not run:
## Fetch USAFacts confirmed cases and deaths over the same time period
covidcast_signals("usa-facts", signal = c("confirmed_incidence_num",
                                          "deaths_incidence_num"),
                  start_day = "2020-08-15", end_day = "2020-10-01")

## End(Not run)
```

---

fips_to_abbr *Get state abbreviations from FIPS codes*

Description

Look up state abbreviations by FIPS codes (including District of Columbia and Puerto Rico). Will match the first two digits of the input codes, so should work for 5-digit county codes, or even longer tract and census block FIPS codes.

Usage

```
fips_to_abbr(code)
```

Arguments

code	Vector of FIPS codes to look up; will match the first two digits of the code. Note that these are treated as strings; the number 1 will not match "01".

Value

A vector of state abbreviations.

See Also

`abbr_to_fips()`

Examples

```
fips_to_abbr("42000")
fips_to_abbr(c("42", "72", "11"))
```

`latest_issue` *Fetch the latest or earliest issue for each observation*

Description

The data returned from `covidcast_signal()` or `covidcast_signals()` can, if called with the `issues` argument, contain multiple issues for a single observation in a single location. These functions filter the data frame to contain only the earliest issue or only the latest issue.

Usage

```
latest_issue(df)

earliest_issue(df)
```

Arguments

`df`	A `covidcast_signal` or `covidcast_signal_long` data frame, such as returned from `covidcast_signal()` or the "long" format of `aggregate_signals()`.

Value

A data frame in the same form, but with only the earliest or latest issue of every observation. Note that these functions sort the data frame as part of their filtering, so the output data frame rows may be in a different order.

`msa_census` *Metro area population data*

Description

Data set on metropolitan area populations, from the 2019 US Census. This includes metropolitan and micropolitan statistical areas, although the COVIDcast API only supports fetching data from metropolitan statistical areas.

Usage

```
msa_census
```

Format

A data frame with 2797 rows, each representing one core-based statistical area (including metropolitan and micropolitan statistical areas, county or county equivalents, and metropolitan divisions). There are many columns. The most crucial are:

CBSA Core Based Statistical Area code. These are unique identifiers used, for example, as the `geo_values` argument to `covidcast_signal()` when requesting data from specific metro areas (with `geo_type = 'msa'`).

MDIV Metropolitan Division code

STCOU State and county code

NAME Name or title of the area.

LSAD Legal/Statistical Area Description, identifying if this is a metropolitan or micropolitan area, a metropolitan division, or a county.

STATE State FIPS code.

POPESTIMATE2019 Estimate of the area's resident population as of July 1, 2019.

Source

United States Census Bureau, at https://www2.census.gov/programs-surveys/popest/datasets/2010-2019/metro/totals/cbsa-est2019-alldata.csv

References

Census Bureau documentation of all columns and their meaning: https://www2.census.gov/programs-surveys/popest/datasets/2010-2019/metro/totals/cbsa-est2019-alldata.pdf

See Also

`cbsa_to_name()`, `name_to_cbsa()`

---

name_to_abbr *Get state abbreviations from state names*

Description

Look up state abbreviations by state names (including District of Columbia and Puerto Rico); this function is based on `grep()`, and hence allows for regular expressions.

Usage

```
name_to_abbr(
  name,
  ignore.case = FALSE,
  perl = FALSE,
  fixed = FALSE,
  ties_method = c("first", "all")
)
```

Arguments

`name`	Vector of state names to look up.
`ignore.case, perl, fixed`	Arguments to pass to `grep()`, with the same defaults as in the latter function. Hence, by default, regular expressions are used; to match against a fixed string (no regular expressions), set `fixed = TRUE`.
`ties_method`	If "first", then only the first match for each name is returned. If "all", then all matches for each name are returned.

Value

A vector of state abbreviations if `ties_method` equals "first", and a list of state abbreviations otherwise.

See Also

`abbr_to_name()`

Examples

```
name_to_abbr("Penn")
name_to_abbr(c("Penn", "New"), ties_method = "all")
```

---

name_to_fips *Get FIPS or CBSA codes from county or metropolitan area names*

Description

Look up FIPS or CBSA codes by county or metropolitan area names, respectively; these functions are based on `grep()`, and hence allow for regular expressions.

Usage

```
name_to_fips(
  name,
  ignore.case = FALSE,
  perl = FALSE,
  fixed = FALSE,
  ties_method = c("first", "all"),
  state = NULL
)

name_to_cbsa(
  name,
  ignore.case = FALSE,
  perl = FALSE,
```

```
  fixed = FALSE,
  ties_method = c("first", "all"),
  state = NULL
)
```

Arguments

`name`	Vector of county or metropolitan area names to look up.
`ignore.case, perl, fixed`	Arguments to pass to `grep()`, with the same defaults as in the latter function. Hence, by default, regular expressions are used; to match against a fixed string (no regular expressions), set `fixed = TRUE`.
`ties_method`	If "first", then only the first match for each name is returned. If "all", then all matches for each name are returned.
`state`	Two letter state abbreviation (case insensitive) indicating a parent state used to restrict the search. For example, when `state = "NY"`, then `name_to_fips()` searches only over only counties lying in New York state, whereas `name_to_cbsa()` searches over the metropolitan areas lying, either fully or partially (as a metropolitan area can span several states), in New York state. If `NULL`, the default, then the search is performed US-wide (not restricted to any state in particular).

Value

A vector of FIPS or CBSA codes if `ties_method` equals "first", and a list of FIPS or CBSA codes otherwise.

See Also

`state_fips_to_name()`, `cbsa_to_name()`

Examples

```
name_to_fips("Allegheny")
name_to_cbsa("Pittsburgh")
name_to_fips("Miami")
name_to_fips("Miami", ties_method = "all")
name_to_fips(c("Allegheny", "Miami", "New "), ties_method = "all")
```

---

plot.covidcast_signal *Plot* covidcast_signal *object as choropleths, bubbles, or time series*

Description

Several plot types are provided, including choropleth plots (maps), bubble plots, and time series plots showing the change of signals over time, for a data frame returned by `covidcast_signal()`. (Only the latest issue from the data frame is used for plotting.) See `vignette("plotting-signals", package = "covidcast")` for examples.

Usage

```
## S3 method for class 'covidcast_signal'
plot(
  x,
  plot_type = c("choro", "bubble", "line"),
  time_value = NULL,
  include = c(),
  range = NULL,
  choro_col = c("#FFFFCC", "#FD893C", "#800026"),
  alpha = 0.5,
  bubble_col = "purple",
  num_bins = 8,
  title = NULL,
  choro_params = list(),
  bubble_params = list(),
  line_params = list(),
  ...
)
```

Arguments

x	The covidcast_signal object to map or plot. If the object contains multiple issues of the same observation, only the most recent issue is mapped or plotted.
plot_type	One of "choro", "bubble", "line" indicating whether to plot a choropleth map, bubble map, or line (time series) graph, respectively. The default is "choro".
time_value	Date object (or string in the form "YYYY-MM-DD") specifying the day to map, for choropleth and bubble maps. If NULL, the default, then the last date in x is used for the maps. Time series plots always include all available time values in x.
include	Vector of state abbreviations (case insensitive, so "pa" and "PA" both denote Pennsylvania) indicating which states to include in the choropleth and bubble maps. Default is c(), which is interpreted to mean all states.
range	Vector of two values: min and max, in this order, to use when defining the color scale for choropleth maps and the size scale for bubble maps, or the range of the y-axis for the time series plot. If NULL, the default, then for the maps, the min and max are set to be the mean +/- 3 standard deviations, where this mean and standard deviation are as provided in the metadata for the given data source and signal; and for the time series plot, they are set to be the observed min and max of the values over the given time period.
choro_col	Vector of colors, as specified in hex code, to use for the choropleth color scale. Can be arbitrary in length. Default is similar to that from https://delphi.cmu.edu/covidcast/.
alpha	Number between 0 and 1, indicating the transparency level to be used in the maps. For choropleth maps, this determines the transparency level for the mega counties. For bubble maps, this determines the transparency level for the bubbles. Default is 0.5.

bubble_col	Bubble color for the bubble map. Default is "purple".
num_bins	Number of bins for determining the bubble sizes for the bubble map (here and throughout, to be precise, by bubble size we mean bubble area). Default is 8. These bins are evenly-spaced in between the min and max as specified through the range parameter. Each bin is assigned the same bubble size. Also, values of zero special: it has its own separate (small) bin, and values mapped to the zero bin are not drawn.
title	Title for the plot. If NULL, the default, then a simple title is used based on the given data source, signal, and time values.
choro_params, bubble_params, line_params	Additional parameter lists for the different plot types, for further customization. See details below.
...	Additional arguments, for compatibility with plot(). Currently unused.

Details

The following named arguments are supported through the lists choro_params, bubble_params, and line_params.

For both choropleth and bubble maps:

subtitle Subtitle for the map.

missing_col Color assigned to missing or NA geo locations.

border_col Border color for geo locations.

border_size Border size for geo locations.

legend_position Position for legend; use "none" to hide legend.

legend_height, legend_width Height and width of the legend.

breaks Breaks for a custom (discrete) color or size scale. Note that we must set breaks to be a vector of the same length as choro_col for choropleth maps. This works as follows: we assign the ith color for choropleth maps, or the ith size for bubble maps, if and only if the given value satisfies breaks[i] <= value < breaks[i+1], where we take by convention breaks[0] = -Inf and breaks[N+1] = Inf for N = length(breaks).

legend_digits Number of decimal places to show for the legend labels.

For choropleth maps only:

legend_n Number of values to label on the legend color bar. Ignored for discrete color scales (when breaks is set manually).

For bubble maps only:

remove_zero Should zeros be excluded from the size scale (hence effectively drawn as bubbles of zero size)?

min_size, max_size Min size for the size scale.

For line graphs:

xlab, ylab Labels for the x-axis and y-axis.

stderr_bands Should standard error bands be drawn?

stderr_alpha Transparency level for the standard error bands.

Value

A ggplot object that can be customized and styled using standard ggplot2 functions.

---

print.covidcast_meta *Print* covidcast_meta *object*

Description

Prints a brief summary of the metadata, and then prints the underlying data frame, for an object returned by covidcast_meta().

Usage

```
## S3 method for class 'covidcast_meta'
print(x, ...)
```

Arguments

x	The covidcast_meta object.
...	Additional arguments passed to print.data.frame() to print the data.

Value

The covidcast_meta object, unchanged.

---

print.covidcast_signal *Print* covidcast_signal *object*

Description

Prints a brief summary of the data source, signal, and geographic level, and then prints the underlying data frame, for an object returned by covidcast_signal().

Usage

```
## S3 method for class 'covidcast_signal'
print(x, ...)
```

Arguments

x	The covidcast_signal object.
...	Additional arguments passed to print.data.frame() to print the data.

Value

The covidcast_signal object, unchanged.

`state_census` *State population data*

Description

Data set on state populations, from the 2019 US Census.

Usage

```
state_census
```

Format

Data frame with 53 rows (including one for the United States as a whole, plus the District of Columbia and the Puerto Rico Commonwealth). Important columns:

SUMLEV Geographic summary level.

REGION Census Region code

DIVISION Census Division code

STATE State FIPS code

NAME Name of the state

POPESTIMATE2019 Estimate of the state's resident population in 2019.

POPEST18PLUS2019 Estimate of the state's resident population in 2019 that is over 18 years old.

PCNT_POPEST18PLUS Estimate of the percent of a state's resident population in 2019 that is over 18.

ABBR Postal abbreviation of the state

Source

United States Census Bureau, at https://www2.census.gov/programs-surveys/popest/datasets/2010-2019/counties/totals/co-est2019-alldata.pdf, https://www.census.gov/data/tables/time-series/demo/popest/2010s-total-puerto-rico-municipios.html, and https://www.census.gov/data/tables/2010/dec/2010-island-areas.html

See Also

`abbr_to_name()`, `name_to_abbr()`, `abbr_to_fips()`, `fips_to_abbr()`

`state_fips_to_name` *Get state, county or metropolitan area names from FIPS or CBSA codes*

Description

Look up county or metropolitan area names by FIPS or CBSA codes. Looking up FIPS code is done with the first 2 numbers (state) or 5 numbers (county) and therefore can be called with longer FIPS codes.

Usage

```
state_fips_to_name(code)

county_fips_to_name(code)

cbsa_to_name(code)
```

Arguments

code	Vector of FIPS or CBSA codes to look up.

Value

A vector of state, county or metro names.

See Also

`name_to_fips()`, `name_to_cbsa()`

Examples

```
state_fips_to_name("42")
state_fips_to_name("42003") # same as previous
county_fips_to_name("42003")
county_fips_to_name("42000") # the county "000" returns the state name
cbsa_to_name("38300")
```

summary.covidcast_meta *Summarize* covidcast_meta *object*

Description

Prints a tabular summary of the object returned by covidcast_meta(), containing each source and signal and a summary of the geographic levels it is available at.

Usage

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

Arguments

object	The covidcast_meta object.
...	Additional arguments, for compatibility with summary(). Currently unused.

Value

A data frame with one row per unique signal in the metadata, with the following columns:

data_source	Data source name
signal	Signal name
county	"*" if this signal is available at the county level, "" otherwise
msa	"*" if this signal is available at the Metropolitan Statistical Area level, "" otherwise
dma	"*" if this signal is available at the Designated Marketing Area level, "" otherwise
hrr	"*" if this signal is available at the Hospital Referral Region level, "" otherwise
state	"*" if this signal is available at the state level, "" otherwise
hhs	"*" if this signal is available at the Health and Human Services region level, "" otherwise
nation	"*" if this signal is available at the national level, "" otherwise

summary.covidcast_signal *Summarize* covidcast_signal *object*

Description

Prints a variety of summary statistics about the underlying data, such as median values, the date range included, sample sizes, and so on, for an object returned by `covidcast_signal()`.

Usage

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

Arguments

`object`	The `covidcast_signal` object.
`...`	Additional arguments, for compatibility with `summary()`. Currently unused.

Value

No return value; called only to print summary statistics.

Index