Package: fiphde (via r-universe)

September 13, 2024

Title Forecasting Influenza in Support of Public Health Decision Making

Version 2.0.2

Description Miscellaneous functions for retrieving data, creating and evaluating time series forecasting models for influenza-like illness (ILI) and influenza hospitalizations in the United States.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports dplyr, fable, fabletools (>= 0.3.2), lubridate, magrittr, MMWRweek, purrr, tibble, tidyr, tsibble, trending (>= 0.1.0), trendeval (>= 0.1.0), ggplot2, stringr, jsonlite, stats, scales, readr, shiny, httr, distfromq

Depends R (>= 2.10)

Suggests testthat (>= 3.0.0), plotly, shinyWidgets, here, DT, knitr, feasts, urca, rmarkdown, waiter, rplanes

Remotes signaturescience/rplanes, reichlab/distfromq

Config/testthat/edition 3

VignetteBuilder knitr

URL https://signaturescience.github.io/fiphde/, https://github.com/signaturescience/fiphde

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

RemoteUrl https://github.com/signaturescience/fiphde

RemoteRef HEAD

RemoteSha 6718966133498f8bc0f55b6ffc41f7fc9e1e1201

Contents

.mcga *Make clean column names*

Description

This helper is used in ilinet and who_nrevss functions to clean column names of values returned from the APIs.

Usage

```
.mcga(tbl)
```

Arguments

tbl	Input tibble with columns to rename

Value

A tibble with clean column names

clin_nowcast *Nowcast clinical laboratory percent positive flu data*

Description

This function provides a naive nowcasting method for clinical laboratory percent positive flu data. The methodology simply averages the last 4 weeks of available data and uses this average as the value for the number of weeks specified to replace. The function will always add 1 additional week to the observed data and (optionally) replace the number of weeks specified in the "weeks_to_replace" argument. This is useful given that there is reporting lag in the NREVSS clinical laboratory percent positive flu data.

Usage

```
clin_nowcast(clin, weeks_to_replace = 1)
```

Arguments

clin	Data prepared with get_cdc_clin
weeks_to_replace	Number of retrospective weeks to replace with nowcast; default is 1

Value

A `tibble` with the following columns:

- **abbreviation**: Abbreviation for the location
- **location**: FIPS code for the location
- **epiyear**: Year of reporting (in epidemiological week calendar)
- **epiweek**: Week of reporting (in epidemiological week calendar)
- **week_start**: Date of beginning (Sunday) of the given epidemiological week
- **p_positive**: Percentage of positive specimens
- **n_positive**: Total number of positive specimens
- **total**: Total number of specimens tested

Examples

```
## Not run:

# Get data for Texas
tx_clin <-
  get_cdc_clin(region = "state") %>%
  dplyr::filter(location == "48")

# Look at most recent observations
tx_clin %>%
  dplyr::arrange(week_start) %>%
  tail()

# Now augment with default 1 week nowcast
tx_clin %>%
  clin_nowcast(., weeks_to_replace = 1) %>%
  dplyr::arrange(week_start) %>%
  tail()

# And again augmented with 2 week nowcast instead
tx_clin %>%
  clin_nowcast(., weeks_to_replace = 2) %>%
  dplyr::arrange(week_start) %>%
  tail()


## End(Not run)
```

`density_probs` *Calculate categorical probability density*

Description

This unexported helper function is used to build a distribution from quantile forecasts and then calculate the probability density for the thresholds associated with each category forecasted: large increase, increase, stable, decrease, large decrease.

Usage

```
density_probs(df, n_horizons = 5, ...)
```

Arguments

`df`	Data frame with forecasts and categorical thresholds joined
`n_horizons`	Number of horizons ahead
`...`	Additional arguments passed to `distfromq::distfromq()`

Value

Data frame with probabilities for each rate change category

`fiphde_launcher` *FIPHDE explorer app launcher*

Description

The explorer app allows a user to view plots of forecasts, inspect tabular output of submission files, and download subsets of forecast submission data. The app includes an interface to interactively select locations to include in the plots, table, and download. This function wraps `shiny::runApp` and accepts arguments for the observed data against which the forecasts should be plotted, as well as the directory containing submission files, both of which are temporarily attached to the global environment for use during the app session. Additional arguments passed to `...` will be inherited by shiny::runApp.

The explorer is meant to review *candidate* submission files. As such, the app is written to expect that submission files in the "submission_dir" argument are named with ".candidate.csv" suffix. For examples of submission files and the naming convention see `system.file("extdata", "submission-example", package = "fiphde")`. Note that submission files can be included for multiple models in model-specific sub-directories.

Usage

```
fiphde_launcher(.data, submission_dir, app_dir = NULL, ...)
```

Arguments

.data	A tibble with historical data for trend leading up to forecast
submission_dir	Full path to directory of submission files containing forecast submissions to explore; submission files in the directory must be named with ".candidate.csv" suffix
app_dir	Full path to directory of explorer app; default is NULL and app directory will be resolved from system.file("app", package="fiphde")
...	Additional arguments to be passed to shiny::runApp

Value

This function starts a Shiny app. On exit it removes objects (see ".data" and "submission_dir") that are temporarily attached and used by the app session.

Examples

```
## Not run:
# Path to the submission example
submission_dir <- system.file("extdata", "submission-example", package = "fiphde")
# Prepare data for explorer app
prepped_hosp <-
  get_hdgov_hosp(limitcols = TRUE) %>%
  prep_hdgov_hosp(statesonly=TRUE, min_per_week = 0, remove_incomplete = TRUE) %>%
  dplyr::filter(abbreviation != "DC")
# Launch the explorer app
fiphde_launcher(.data = prepped_hosp,
                submission_dir = submission_dir,
                host = "0.0.0.0",
                launch.browser = TRUE,
                port = 80)

## End(Not run)
```

---

forecast_categorical *Forecast categorical targets*

Description

This function takes probabilistic flu hospitalization forecast input and converts the forecasted values for each location to a categorical "change" indicator. The criteria for each level ("large decrease", "decrease", "stable", "increase", "large increase") was defined by the CDC (see link in references). The algorithm evaluates absolute changes in counts and rates (per 100k individuals) for the most recently observed week and a 2 week ahead forecasted horizon. This procedure runs independently for each location, and results in a formatted tabular output that includes each possible level and its corresponding probability of being observed (calculated from probabilistic quantiles) for every location.

Usage

```
forecast_categorical(
  .forecast,
  .observed,
  method = "density",
  format = "hubverse",
  horizon = 4
)
```

Arguments

`.forecast`	A tibble with "submission-ready" probabilistic flu hospitalization forecast data (i.e., tibble contained in list element returned from format_for_submission)
`.observed`	A tibble with observed flu admission data (i.e., tibble output from prep_hdgov_hosp)
`method`	The categorical forecasting method to use; must be one of `"density"` or `"interpolation"`; default is `"density"`
`format`	The submission format to be used; must be one of `"hubverse"` or `"legacy"`; default is `"hubverse"`
`horizon`	The number of horizons ahead to forecast; must be one of `4` or `5`; default is `4`

Value

A `tibble` with formatted categorical forecasts.

If format is `"hubverse"` the tibble will have the following columns:

- **reference_date**: Date of reference for forecast submission
- **horizon**: Horizon for the given forecast
- **target**: Name of forecasted target
- **target_end_date**: Last date of the forecasted target (e.g., Saturday of the given epidemiological week)
- **location**: Name or geographic identifier (e.g., FIPS code) for location for the given forecast
- **output_type**: Type of forecasted value (e.g., "quantile")
- **output_type_id**: The quantile for the forecasted value if output_type is "quantile"
- **value**: The forecasted value

If format is `"legacy"` the tibble will have the following columns:

- **forecast_date**: Date of forecast
- **target**: Horizon and name of forecasted target
- **target_end_date**: Last date of the forecasted target (e.g., Saturday of the given epidemiological week)
- **location**: Name or geographic identifier (e.g., FIPS code) for location for the given forecast
- **type**: One of either "point" or "quantile" for the forecasted value
- **quantile**: The quantile for the forecasted value; `NA` if "type" is `"point"`
- **value**: The forecasted value

References

https://github.com/cdcepi/Flusight-forecast-data/blob/master/data-experimental/README.md

Examples

```
## Not run:
# Retrieve hospitalization data
h_raw <- get_hdgov_hosp(limitcols=TRUE)
# Prepare and summarize hospitalization data to weekly resolution
prepped_hosp <- prep_hdgov_hosp(h_raw)
# Create a keyed time series tibble with only locations of interest
prepped_tsibble <- make_tsibble(prepped_hosp,
                                    epiyear = epiyear,
                                    epiweek=epiweek,
                                    key=location)

# Run with default constrained ARIMA, nonseasonal ETS, no NNETAR
hosp_fitfor <- ts_fit_forecast(prepped_tsibble,
                               horizon=4L,
                               outcome="flu.admits",
                               covariates=TRUE)
# Prepare forecast for quantile submission format
forc <- format_for_submission(hosp_fitfor$tsfor, method = "ts", format = "legacy")
# Run categorical summary of quantiles for the time series ensemble
forecast_categorical(forc$ensemble, prepped_hosp, method = "interpolation", format = "legacy")

## End(Not run)
```

`forecast_ili` *Forecast ILI*

Description

This function forecasts ILI up to a specified future horizon. The models used can be parameterized with a "models" argument (for more details see ts_fit_forecast). By default, the function will use an ARIMA approach to model all locations in the input historical ILI data and then use the fitted models to forecast out to each of the horizons.

Usage

```
forecast_ili(
  ilidat,
  horizon = 4L,
  trim_date = NULL,
  models = list(arima = "PDQ(0,0,0)+pdq(1:2,0:2,0)")
)
```

Arguments

ilidat	Data returned from get_cdc_ili
horizon	Optional horizon periods through which the forecasts should be generated; default is 4
trim_date	Earliest start date you want to use for ILI data; default NULL doesn't trim
models	The list of model parameters passed to ts_fit_forecast; defaults to list(arima="PDQ(0,0,0)+pdq(1:2,0

Value

A named list containing:

- **ilidat**: The data sent into the function filtered to the location and the trim_date. Select columns returned.
- **ilidat_tsibble**: The tsibble class object returned by running make_tsibble on the data above.
- **ili_fit**: The fit from fabletools::model.
- **ili_forecast**: The forecast from fabletools::forecast at the specified horizon.
- **ili_future**: The horizon-number of weeks of ILI data forecasted into the future.
- **ili_bound**: The data in 1 bound to the data in 5.
- **arima_params**: A tibble with ARIMA model parameters for each location (if type="arima").
- **locstats**: A tibble with missing data information on all locations.
- **removed**: A tibble with locations removed because of high missing ILI data.

Examples

```
## Not run:
# Retrieve ILI data
ilidat <- get_cdc_ili(region = c("national", "state", "hhs"),
                      years = 2010:lubridate::year(lubridate::today()))

# Using data only from march 2020 forward, for US only
ilidat_us <- ilidat %>% dplyr::filter(location=="US")
# Replace most recent week with nowcast data, and nowcast last week
ilidat_us <- ilidat_us %>% replace_ili_nowcast(weeks_to_replace=1)
ilifor_us <- forecast_ili(ilidat_us, horizon=4L, trim_date="2020-03-01")
# Take a look at objects that come out ILI forecasting procedure
ilifor_us$ili_fit
ilifor_us$arima_params
ilifor_us$ili_forecast
head(ilifor_us$ili_bound)
tail(ilifor_us$ili_bound, 10)

## End(Not run)
```

`format_for_submission` *Format forecasts for submission*

Description

This function prepares forecasts to adhere to probabilistic forecast submission guidelines for consortia such as FluSight.

Usage

```
format_for_submission(
  .forecasts,
  method = "ts",
  .target = "wk ahead inc flu hosp",
  format = "hubverse",
  horizon_shift = 1
)
```

Arguments

`.forecasts`	Forecasts to be formatted for submission; if method is `"ts"` this should be forecasts from ts_fit_forecast; otherwise this must be a `tibble` with forecast output (e.g., output from glm_forecast) with a column designating "location"
`method`	Method for forecasting; default is `"ts"` which will trigger the use of ts_format_for_submission internally
`.target`	Name of the target in the forecast; default is `"wk ahead inc flu hosp"`
`format`	The submission format to be used; must be one of `"hubverse"` or `"legacy"`; default is `"hubverse"`
`horizon_shift`	Number of horizons to shift backwards to align with reference date; only used if format is `"hubverse"`; default is `1`

Value

A named list of tibbles with probabilistic forecasts (one for each model), formatted for submission.

If format is `"hubverse"` each tibble will have the following columns:

- **reference_date**: Date of reference for forecast submission
- **horizon**: Horizon for the given forecast
- **target**: Name of forecasted target
- **target_end_date**: Last date of the forecasted target (e.g., Saturday of the given epidemiological week)
- **location**: Name or geographic identifier (e.g., FIPS code) for location for the given forecast
- **output_type**: Type of forecasted value (e.g., "quantile")
- **output_type_id**: The quantile for the forecasted value if output_type is "quantile"

- **value**: The forecasted value

If format is "legacy" each tibble will have the following columns:

- **forecast_date**: Date of forecast
- **target**: Horizon and name of forecasted target
- **target_end_date**: Last date of the forecasted target (e.g., Saturday of the given epidemiological week)
- **location**: Name or geographic identifier (e.g., FIPS code) for location for the given forecast
- **type**: One of either "point" or "quantile" for the forecasted value
- **quantile**: The quantile for the forecasted value; NA if "type" is "point"
- **value**: The forecasted value

References

https://github.com/signaturescience/FluSight-forecast-hub/tree/main/model-output#forecast-file-format

https://github.com/cdcepi/Flusight-forecast-data/blob/master/data-forecasts/README.md

Examples

```
## Not run:
# Get raw data from healthdata.gov
h_raw <- get_hdgov_hosp(limitcols=TRUE)

# Prep, and make a tsibble
prepped_hosp <- prep_hdgov_hosp(h_raw, statesonly=TRUE)
prepped_hosp_tsibble <- make_tsibble(prepped_hosp,
                                     epiyear = epiyear,
                                     epiweek=epiweek,
                                     key=location)
# Limit to only Virginia and US
prepped_hosp_tsibble <-
  prepped_hosp_tsibble %>%
  dplyr::filter(location %in% c("US", "51"))

# Fit a model
hosp_fitfor <- ts_fit_forecast(prepped_hosp_tsibble,
                               horizon=4L,
                               outcome="flu.admits",
                               covariates=TRUE)

# Format for submission
formatted_list <- format_for_submission(hosp_fitfor$tsfor, method = "ts", format = "legacy")
formatted_list

## End(Not run)
```

`get_cdc_clin` *Retrieve clinical laboratory percent positive flu data*

Description

This function returns weekly state and/or national clinical laboratory percent positivity data from the NREVSS reporting instrument via the CDC FluView API.

Usage

```
get_cdc_clin(region = "both", years = NULL)
```

Arguments

`region`	Either "state", "national", or "both". Defaults to `"both"` to return state and national data combined.
`years`	A vector of years to retrieve data for. CDC has data going back to 1997. Default value (`NULL`) retrieves **all** years.

Value

A `tibble` with the following columns:

- **abbreviation**: Abbreviation for the location
- **location**: FIPS code for the location
- **epiyear**: Year of reporting (in epidemiological week calendar)
- **epiweek**: Week of reporting (in epidemiological week calendar)
- **week_start**: Date of beginning (Sunday) of the given epidemiological week
- **p_positive**: Percentage of positive specimens
- **n_positive**: Total number of positive specimens
- **total**: Total number of specimens tested

References

https://gis.cdc.gov/grasp/fluview/Phase_6_Cleared_Help.pdf

Examples

```
## Not run:
# Get all clinical lab flu positivity data
all_clin <- get_cdc_clin()
all_clin
# Alternatively look at a specific location and time
# This 2021 will return weekly data
# Starting at beginning of 2021/22 season
# Ending the week before start of 2022/23 season
```

```
va_clin <-
  get_cdc_clin(region = "state", years = 2021) %>%
  dplyr::filter(location == "51")
va_clin

## End(Not run)
```

`get_cdc_hosp` *Retrieve hospitalization data from FluSurv-NET*

Description

This function retrieves historical FluSurv-NET hospitalization data via the CDC FluView API.

Usage

```
get_cdc_hosp(years = NULL)
```

Arguments

`years`	A vector of years to retrieve data for (i.e. 2014 for CDC flu season 2014-2015). CDC has data going back to 2009 and up until the *previous* flu season. Default value (`NULL`) retrieves **all** years.

Value

A `tibble` with the following columns:

- **location**: FIPS code for the location
- **abbreviation**: Abbreviation for the location
- **region**: Name of the region
- **epiyear**: Year of reporting (in epidemiological week calendar)
- **epiweek**: Week of reporting (in epidemiological week calendar)
- **week_start**: Date of beginning (Sunday) of the given epidemiological week
- **week_end**: Date of end (Saturday) of the given epidemiological week
- **rate**: The cumulative rate per 100k
- **weekly_rate**: The weekly rate per 100k
- **season**: The flu season to which the given epidemiological week belongs

References

https://gis.cdc.gov/GRASP/fluview/FluViewPhase3QuickReferenceGuide.pdf

Examples

```
## Not run:
# Retrieve FluSurv-Net hospitalization data for specific year(s)
get_cdc_hosp(years=2019)

## End(Not run)
```

`get_cdc_ili` *Retrieve ILI data from ILINet*

Description

This function pulls ILINet data from the CDC FluView API. Data are available historically and can be pulled at the state, national, or HHS region level.

Usage

```
get_cdc_ili(region = c("national", "state", "hhs"), years = NULL)
```

Arguments

`region`	Either "state", "national", or "hhs"; defaults to `c("national", "state", "hhs")` for all three.
`years`	A vector of years to retrieve data for. CDC has data going back to 1997. Default value (`NULL`) retrieves **all** years.

Value

A `tibble` with the following columns:

- **location**: FIPS code for the location
- **region_type**: The type of location
- **abbreviation**: Abbreviation for the location
- **region**: Name of the region
- **epiyear**: Year of reporting (in epidemiological week calendar)
- **epiweek**: Week of reporting (in epidemiological week calendar)
- **week_start**: Date of beginning (Sunday) of the given epidemiological week
- **weighted_ili**: Population-weighted percentage of ILI outpatient visits
- **unweighted_ili**: Unweighted percentage of ILI outpatient visits
- **ilitotal**: Total number of ILI outpatient visits reported
- **num_providers**: Number of providers reporting
- **total_patients**: Total number of outpatient visits reported
- **population**: Total population for the given location

References

https://gis.cdc.gov/grasp/fluview/FluViewPhase1QuickReferenceGuide.pdf

Examples

```
## Not run:
# Retrieve ILI data for specific years and regions
get_cdc_ili(region="national", years=2021)
get_cdc_ili(region="hhs", years=2021)
get_cdc_ili(region="state", years=2021) %>% dplyr::filter(abbreviation=="VA")
get_cdc_ili(region=c("national", "state"), years=2021)

## End(Not run)
```

---

get_hdgov_hosp *Retrieve hospitalization data from HHS*

Description

This function retrieves hospital utilization time series data distributed through healthdata.gov. Data are aggregated to the state granularity from facility level reports via HHS TeleTracking, HHS Protect, and the National Healthcare Safety Network (historically). Users can optionally filter to include all fields or restrict to a prespecified set of fields relevant to COVID and influenza hospital utilization. The results are returned as a `tibble`.

Usage

```
get_hdgov_hosp(
  endpoint = "https://healthdata.gov/api/views/g62h-syeh/rows.csv",
  app_token = Sys.getenv("HEALTHDATA_APP_TOKEN"),
  limitcols = FALSE,
  shift_back = TRUE
)
```

Arguments

`endpoint`	URL to healthdata.gov endpoint
`app_token`	App token from healthdata.gov; default is to look for environment variable called `"HEALTHDATA_APP_TOKEN"` and if a token is not supplied to proceed with possibility of rate limitation (see "Details" for more information)
`limitcols`	Logical as to whether or not to limit to prespecified set of columns (see "Value" section for more details); default `FALSE`
`shift_back`	Logical as to whether or not the dates for the retrieved data should be shifted back to reflect previous day; default is `TRUE`

Details

The data retrieval will proceed whether or not an API token has been supplied via the `app_token` argument. However, to avoid possible rate limits it is recommended to retrieve a token for the healthdata.gov API (https://healthdata.gov/profile/edit/developer_settings), and add that token as an entry to `.Renviron` with `HEALTHDATA_APP_TOKEN="yourtokenhere"`.

Value

A `tibble` with at least the following columns:

- **state**: Abbreviation of the state
- **date**: Date of report
- **flu.admits**: Count of flu cases among admitted patients on previous day
- **flu.admits.cov**: Coverage (number of hospitals reporting) for incident flu cases
- **flu.deaths**: Count of flu deaths on previous day
- **flu.deaths.cov**: Coverage (number of hospitals reporting) for flu deaths
- **flu.icu**: Count of flu cases among ICU patients on previous day
- **flu.icu.cov**: Coverage (number of hospitals reporting) for flu ICU cases
- **flu.tot**: Count of total flu cases among admitted patients
- **flu.tot.cov**: Coverage (number of hospitals reporting) for total flu cases
- **cov.admits**: Count of COVID cases among admitted patients on previous day
- **cov.admits.cov**: Coverage (number of hospitals reporting) for incident COVID cases
- **cov.deaths**: Count of COVID deaths on previous day
- **cov.deaths.cov**: Coverage (number of hospitals reporting) for COVID deaths

If `limitcols=TRUE` then the only columns returned will be those listed above. However, if `limitcols=FALSE` then the function will additionally return *all* other fields in the state-aggregated hospitalization data.

References

https://healthdata.gov/Hospital/COVID-19-Reported-Patient-Impact-and-Hospital-Capa/g62h-syeh

https://dev.socrata.com/foundry/healthdata.gov/g62h-syeh

Examples

```
## Not run:
# Retrieve hospitalization data (all columns)
get_hdgov_hosp()
# Retrieve hospitalization data (limited columns)
get_hdgov_hosp(limitcols=TRUE)

## End(Not run)
```

`get_nowcast_ili` *Retrieve ILI nowcast*

Description

This function pulls the ILI nowcast from CMU Delphi's ILI Nearby API. Observed ILINet data is typically reported with a lag, and the ILI nowcast can be used to augment the ILI data stream. The functionality here depends on availability of the ILI Nearby API (see 'Details' section).

Usage

```
get_nowcast_ili(
  epiyearweeks = NULL,
  dates = lubridate::today() - c(14, 7),
  state = NULL,
  boundatzero = TRUE
)
```

Arguments

`epiyearweeks`	A vector of epiyear-epiweeks to retrieve data for, e.g., 202150, 202151, etc. Exclusive with dates
`dates`	A vector of dates to retrieve data for, e.g., ""2021-12-12" or "2021-12-19". Exclusive with epiyearweek. Defaults to two weeks prior.
`state`	A vector of states to retrieve (two-letter abbreviation). Default `NULL` retrieves all states, national, and hhs regions. See examples.
`boundatzero`	Logical as to whether or not the values should be truncated at 0 (i.e., non-negative); default is `TRUE`

Details

As of October 2022 ILInearby was no longer being updated. As such, the `get_nowcast_ili()` will likely return 'NA'. See https://github.com/cmu-delphi/delphi-epidata/issues/993.

Value

Either `NA` (if the API can't be reached) or a `tibble` with the following columns:

- **location**: FIPS code for the location
- **abbreviation**: Abbreviation for the location
- **epiyear**: Year of reporting (in epidemiological week calendar)
- **epiweek**: Week of reporting (in epidemiological week calendar)
- **weigthed_ili_now**: Nowcasted ILI value

References

https://delphi.cmu.edu/nowcast/

Examples

```
## Not run:
# Defaults to the previous two weeks for all states
get_nowcast_ili()

# Otherwise specify one or the other, not both
get_nowcast_ili(epiyearweeks=c("202150", "202151"), dates=NULL)
get_nowcast_ili(epiyearweeks=NULL, dates=c("2021-12-12", "2021-12-19"))

# Get just one state for the last years worth of data (back 52 weeks to 1 week)
get_nowcast_ili(epiyearweeks=NULL,
                dates=lubridate::today()-seq(52*7, 7, -7),
                state="FL")

## End(Not run)
```

`glm_fit` *Fit glm models*

Description

This helper function is used in glm_wrap to fit a list of models and select the best one. The model selection procedure will use the root mean square error (RMSE) metric implemented in yardstick::rmse to select the best model.

Usage

```
glm_fit(.data, .models, complete = TRUE)
```

Arguments

`.data`	Data including all explanatory and outcome variables needed for modeling; must include column for "location"
`.models`	List of models defined as trending::trending_model objects
`complete`	Logical as to whether or not all observations for covariates must be available in a given model; default is `TRUE`

Value

A `tibble` containing characteristics from the "best" `glm` model including:

- **model_class**: The "type" of model for the best fit
- **fit**: The fitted model object for the best fit stored as a list column
- **location**: The geographic unit being modeled
- **data**: Original model fit data as a `tibble` in a list column

`glm_forecast` *Forecast glm models*

Description

This function uses fitted model object from glm_fit and future covariate data to create probablistic forecasts at specific quantiles derived from the "alpha" parameter.

Usage

```
glm_forecast(
  .data,
  new_covariates = NULL,
  fit,
  alpha = c(0.01, 0.025, seq(0.05, 0.45, by = 0.05)) * 2
)
```

Arguments

`.data`	Data including all explanatory and outcome variables needed for modeling
`new_covariates`	Tibble with one column per covariate, and n rows for n horizons being forecasted
`fit`	Fitted model object from glm_fit; note must be accessed from first element in "fit" column
`alpha`	Vector specifying the threshold(s) to be used for prediction intervals; alpha of `0.05` would correspond to 95% PI; default is `c(0.01, 0.025, seq(0.05, 0.45, by = 0.05)) * 2` to create a range of intervals

Value

A `tibble` with forecasted data including the following columns:

- **epiweek**: The epidemiological week for the forecasted horizon
- **epiyear**: The epidemiological year for the forecasted horizon
- **quantile**: The quantile for the forecasted value; `NA` for point estimate
- **value**: The forecasted value

`glm_quibble` *Get quantiles from prediction intervals*

Description

This helper function runs the trending::predict.trending_fit method on a fitted model at specified values of "alpha" in order to create a range of prediction intervals. The processing also includes steps to convert the alpha to corresponding quantile values at upper and lower bounds. See "Details" for more information on the translation of "alpha" to quantile values. This function is used internally in glm_forecast.

Usage

```
glm_quibble(
  fit,
  new_data,
  alpha = c(0.01, 0.025, seq(0.05, 0.45, by = 0.05)) * 2
)
```

Arguments

`fit`	Fitted model object from glm_fit; note must be accessed from first element in "fit" column
`new_data`	A `tibble` with new data on which the trending::predict.trending_fit method should run
`alpha`	Vector specifying the threshold(s) to be used for prediction intervals (PI); alpha of `0.05` would correspond to 95% PI; default is `c(0.01, 0.025, seq(0.05, 0.45, by = 0.05)) * 2` to create a range of intervals

Details

The "alpha" parameter defines the width of prediction interval (PI). For example, an alpha = 0.05 would correspond to a 95% PI. This function uses the PI(s) (per the alpha value(s) specified) to construct a range of quantiles that fall at lower and upper bound of each PI. Continuing from the example of alpha = 0.05, the quantile estimates returned would fall at 0.025 (lower bound of PI) and 0.975 (upper bound of PI).

Value

A `tibble` with forecasted data including the following columns:

- **epiweek**: The epidemiological week for the forecasted horizon
- **epiyear**: The epidemiological year for the forecasted horizon
- **quantile**: The quantile for the forecasted value; `NA` for point estimate
- **value**: The forecasted value

`glm_wrap` *Run glm modeling and forecasting*

Description

This is a wrapper function that pipelines influenza hospitalization modeling (glm_fit) and forecasting (glm_forecast).

Usage

```
glm_wrap(
  .data,
  .models,
  new_covariates = NULL,
  horizon = 4,
  alpha = c(0.01, 0.025, seq(0.05, 0.45, by = 0.05)) * 2
)
```

Arguments

`.data`	Data including all explanatory and outcome variables needed for modeling
`.models`	List of models defined as trending::trending_model objects
`new_covariates`	A `tibble` with one column per covariate, and n rows for n horizons being forecasted
`horizon`	Number of weeks ahead for forecasting
`alpha`	Vector specifying the threshold(s) to be used for prediction intervals (PI); alpha of `0.05` would correspond to 95% PI; default is `c(0.01, 0.025, seq(0.05, 0.45, by = 0.05)) * 2` to create a range of intervals

Value

Named list with two elements:

- **model**: Output from glm_fit with selected model fit
- **forecasts**: Output from glm_forecast with forecasts from each horizon combined as a single tibble

Examples

```
## Not run:
# Retrieve data to be used in fitting models
hosp_va <-
 get_hdgov_hosp(limitcols=TRUE) %>%
 prep_hdgov_hosp(statesonly=TRUE, min_per_week = 0, remove_incomplete = TRUE) %>%
 dplyr::filter(abbreviation == "VA")

# Define list of models
```

```
models <-
 list(
   poisson = trending::glm_model(flu.admits ~ hosp_rank + ili_rank, family = "poisson"),
  quasipoisson = trending::glm_model(flu.admits ~ hosp_rank + ili_rank, family = "quasipoisson"),
   negbin = trending::glm_nb_model(flu.admits ~ hosp_rank + ili_rank)
 )

# Create new covariate data to feed into forecast procedure
new_cov <-
  dplyr::tibble(
    date = max(hosp_va$week_start) + c(7,14,21,28),
    epiweek = lubridate::epiweek(date),
    epiyear = lubridate::epiyear(date)
  ) %>%
  dplyr::left_join(
    fiphde:::historical_severity, by="epiweek"
  ) %>%
  dplyr::select(-epiweek,-epiyear)

# Run the glm wrapper to fit and forecast
va_glm_res <- glm_wrap(.data = hosp_va, .models = models, new_covariates = new_cov, horizon = 4)
va_glm_res


## End(Not run)
```

`hospitalizations` *Laboratory-confirmed influenza hospitalizations*

Description

Adapted from `cdcfluview::hospitalizations`.

This unexported helper function leverages the CDC FluView API to pull influenza hospitalizations collected by surveillance instruments (including FluSurv-NET). The data retrieved can be parameterized by geographic granularity and/or flu season, and includes hospitalization rates by age group. The function is used internally by get_cdc_hosp.

Usage

```
hospitalizations(
  surveillance_area = c("flusurv", "eip", "ihsp"),
  region = "all",
  years = NULL
)
```

Arguments

`surveillance_area`
: One of `"flusurv"`, `"eip"`, or `"ihsp"`

region	Individual region within the surveillance area selected; default "all" mimics selecting "Entire Network" from the CDC FluView application drop down; see "Details" for list of valid region values for each surveillance area
years	A vector of years to retrieve data for (i.e. 2014 for CDC flu season 2014-2015). CDC has data for this API going back to 2009 and up until the *previous* flu season. Default value (NULL) means retrieve **all** years. NOTE: if you happen to specify a 2-digit season value (i.e. 56 == 2016-2017) the function is smart enough to retrieve by season ID vs convert that to a year.

Details

NOTE: The list of regions was compiled in February 2023 by querying the CDC FluView API. Individual regions may not be accessible in all cases. As of late 2023, the query was only returning results for the "Entire Network" selection.

Each possible value "surveillance_area" ("flusurv", "eip", or "ihsp") can be further queried by region. The following is a list of valid regions:

- **flusurv**: "Entire Network"
- **eip**: "Entire Network", "California", "Colorado", "Connecticut", "Georgia", "Maryland", "Minnesota", "New Mexico", "New York - Albany", "New York - Rochester", "Oregon, "Tennessee"
- **ihsp**: "Entire Network", "Idaho", "Iowa", "Michigan", "Ohio", "Oklahoma", "Rhode Island", "South Dakota", "Utah"

References

- Hospital Portal
- cdcfluview package

---

hubverse_format *Hubverse formatting*

Description

This unexported helper is used internally inside in format_for_submission. It specifically updates formatting for Hubverse guidelines.

Usage

```
hubverse_format(dat, horizon_shift = 1)
```

Arguments

dat	Forecast prepped in "legacy" format
horizon_shift	Number of horizons to shift backwards to align with reference date; default is 1

Value

Formatted `tibble`

References

https://github.com/signaturescience/FluSight-forecast-hub/tree/main/model-output#forecast-file-format

https://github.com/cdcepi/Flusight-forecast-data/blob/master/data-forecasts/README.md

---

`ilinet` *Retrieve ILINet surveillance data*

Description

Adapted from `cdcfluview::ilinet`.

This unexported helper function retrieves current and historical ILINet surveillance data for the identified region via the CDC FluView API. The function is used internally in get_cdc_ili. Data returned include weighted and unweighted ILI percentage, as well as age-specific ILI outpatient visit counts for each location / epidemiological week.

Usage

```
ilinet(region = c("national", "hhs", "census", "state"), years = NULL)
```

Arguments

`region`	One of `"national"`, `"hhs"`, `"census"`, or `"state"`
`years`	A vector of years to retrieve data for (i.e. `2014` for CDC flu season 2014-2015). CDC has data for this API going back to 1997. Default value (`NULL`) means retrieve **all** years. NOTE: if you happen to specify a 2-digit season value (i.e. `57` == 2017-2018) the function is smart enough to retrieve by season ID vs convert that to a year.

References

- cdcfluview package
- CDC FluView Portal

is_monday *Check Monday*

Description

This is a helper function to see if today is Monday.

Usage

```
is_monday()
```

Value

Logical indicating whether or not today is Monday

Examples

```
is_monday()
```

make_tsibble *Make* tsibble

Description

This function converts an input `tibble` with columns for lubridate::epiyear and lubridate::epiweek into a tsibble::tsibble object. The `tsibble` has columns specifying indices for the time series as well as a date for the Monday of the epiyear/epiweek combination at each row.

Usage

```
make_tsibble(df, epiyear, epiweek, key = location)
```

Arguments

`df`	A `tibble` containing columns `epiyear` and `epiweek`.
`epiyear`	Unquoted variable name containing the MMWR epiyear.
`epiweek`	Unquoted variable name containing the MMWR epiweek.
`key`	Unquoted variable name containing the name of the column to be the tsibble key. See tsibble::as_tsibble.

Value

A `tsibble` containing additional columns `monday` indicating the date for the Monday of that epiweek, and `yweek` (a yearweek vctr class object) that indexes the `tsibble` in 1 week increments.

Examples

```
# Create an example tibble
d <- tibble::tibble(epiyear=c(2020, 2020, 2021, 2021),
                    epiweek=c(52, 53, 1, 2),
                    location="US",
                    somedata=101:104)
# Convert to tsibble (keyed time series tibble)
make_tsibble(d, epiyear = epiyear, epiweek=epiweek, key=location)
```

---

mmwr_week_to_date *Convert MMWR format to date*

Description

Adapted from `cdcfluview::mmwr_week_to_date`.

This function transforms MMWR epidemiological year+week (or year+week+day) to a date object. This was implemented based on the `cdcfluview::mmwr_week_to_date` function, which adapted similar functionality from the `MMWRweek` package.

Usage

```
mmwr_week_to_date(year, week, day = NULL)
```

Arguments

`year`	Vector of epidemiological year(s); must be same length as "week" and "day" (unless "day" is `NULL`)
`week`	Vector of epidemiological week(s); must be same length as "year" and "day" (unless "day" is `NULL`)
`day`	Vector of day(s); must be same length as "week" and "year" (unless set to is `NULL`); default is `NULL` and the day returned will be the first day of the epidemiological week (i.e., Sunday)

Value

Vector of date objects as with as many elements as input year(s), week(s), day(s)

References

- cdcfluview package

Examples

```
mmwr_week_to_date(2020,1)
mmwr_week_to_date(2020,1,5)
mmwr_week_to_date(c(2020,2021,2022),c(1,2,8), c(1,1,7))
```

mnz *Minimum non-zero*

Description

Helper function to get the minimum non-zero positive value from a vector. Used internally in mnz_replace.

Usage

```
mnz(x)
```

Arguments

x	A numeric vector

Value

The minimum non-zero positive value from x

Examples

```
x <- c(.1, 0, -.2, NA, .3, .4, .0001, -.3, NA, 999)
x
mnz(x)
```

mnz_replace *Minimum non-zero replacement*

Description

Replace zeros and negative values with the minimum non-zero positive value from a vector.

Usage

```
mnz_replace(x)
```

Arguments

x	A numeric vector

Value

A vector of the same length with negatives and zeros replaced with the minimum nonzero value of that vector.

Examples

```
x <- c(.1, 0, -.2, NA, .3, .4, .0001, -.3, NA, 999)
x
mnz(x)
mnz_replace(x)
tibble::tibble(x) %>% dplyr::mutate(x2=mnz_replace(x))
```

`plot_forecast` *Plot forecasts*

Description

This function serves as a plotting mechanism for prepped forecast submission data. The plots show the historical trajectory of the truth data supplied along with the forecasted point estimates and (optionally) the prediction interval. All plots are faceted by location.

Note that the ".data" and "submission" arguments to this function expect incoming data prepared in a certain format. See the argument documentation and "Details" for more information.

Usage

```
plot_forecast(
  .data,
  submission,
  location = "US",
  pi = 0.95,
  .model = NULL,
  .outcome = "flu.admits",
  format = "legacy"
)
```

Arguments

`.data`	A data frame with historical truth data for all locations and outcomes in submission targets
`submission`	Formatted submission (e.g., a `tibble` containing forecasts prepped with format_for_submission)
`location`	Vector specifying locations to filter to; `'US'` by default.
`pi`	Width of prediction interval to plot; default is `0.95` for 95% PI; if set to `NULL` the PI will not be plotted
`.model`	Name of the model used to generate forecasts; default is `NULL` and the name of the model will be assumed to be stored in a column called "model" in formatted submission file
`.outcome`	The name of the outcome variable you're plotting in the historical data; defaults to `"flu.admits"`
`format`	The submission format to be used; must be one of `"hubverse"` or `"legacy"`; default is `"legacy"`

Details

To plot the forecasted output alongside the observed historical data, both the ".data" and "submission" data must be prepared at the same geographic and temporal resolutions. The data frame passed to ".data" must include the column specified in the ".outcome" argument as well as the following columns:

- **location**: FIPS location code
- **week_end**: Date of the last day (Saturday) in the given epidemiological week

If format is "legacy" the "submission" data should be a probabilistic forecast prepared as a `tibble` with at minimum the following columns:

- **forecast_date**: Date of forecast
- **target**: Horizon and name of forecasted target
- **target_end_date**: Last date of the forecasted target (e.g., Saturday of the given epidemiological week)
- **location**: FIPS code for location
- **type**: One of either "point" or "quantile" for the forecasted value
- **quantile**: The quantile for the forecasted value; `NA` if "type" is `"point"`
- **value**: The forecasted value

If format is "hubverse" the "submission" data should be a probabilistic forecast prepared as a `tibble` with at minimum the following columns:

- **reference_date**: Date of reference for forecast submission
- **horizon**: Horizon for the given forecast
- **target**: Name of forecasted target
- **target_end_date**: Last date of the forecasted target (e.g., Saturday of the given epidemiological week)
- **location**: Name or geographic identifier (e.g., FIPS code) for location for the given forecast
- **output_type**: Type of forecasted value (e.g., "quantile")
- **output_type_id**: The quantile for the forecasted value if output_type is "quantile"
- **value**: The forecasted value

The "submission" data may optionally include a column with the name of the model used, such that multiple models can be visualized in the same plot.

Value

A `ggplot2` plot object with line plots for outcome trajectories faceted by location

Examples

```
## Not run:
# Get some data
h_raw <- get_hdgov_hosp(limitcols=TRUE)

# Prep all the data
prepped_hosp_all <- prep_hdgov_hosp(h_raw)

# What are the last four weeks of recorded data?
last4 <-
  prepped_hosp_all %>%
  dplyr::distinct(week_start) %>%
  dplyr::arrange(week_start) %>%
  tail(4)

# Remove those
prepped_hosp <-
  prepped_hosp_all %>%
  dplyr::anti_join(last4, by="week_start")

# Make a tsibble
prepped_hosp_tsibble <- make_tsibble(prepped_hosp,
                                     epiyear = epiyear,
                                     epiweek=epiweek,
                                     key=location)
# Limit to just one state and US
prepped_hosp_tsibble <-
  prepped_hosp_tsibble %>%
  dplyr::filter(location %in% c("US", "51"))

# Fit models and forecasts
hosp_fitfor <- ts_fit_forecast(prepped_hosp_tsibble,
                               horizon=4L,
                               outcome="flu.admits",
                               trim_date=NULL,
                               covariates=TRUE)

# Format for submission
hosp_formatted <- ts_format_for_submission(hosp_fitfor$tsfor)

# Plot with current and all data
plot_forecast(prepped_hosp, hosp_formatted$ensemble)
plot_forecast(prepped_hosp_all, hosp_formatted$ensemble)
plot_forecast(prepped_hosp, hosp_formatted$ensemble, location=c("US", "51"))
plot_forecast(prepped_hosp_all, hosp_formatted$ensemble, location=c("US", "51"))
plot_forecast(prepped_hosp, hosp_formatted$ets)
plot_forecast(prepped_hosp_all, hosp_formatted$ets)
plot_forecast(prepped_hosp, hosp_formatted$arima)
plot_forecast(prepped_hosp_all, hosp_formatted$arima)

# Demonstrating multiple models
prepped_hosp <-
```

```
  h_raw %>%
  prep_hdgov_hosp(statesonly=TRUE, min_per_week = 0, remove_incomplete = TRUE) %>%
  dplyr::filter(abbreviation != "DC") %>%
  dplyr::filter(week_start < as.Date("2022-01-08", format = "%Y-%m-%d"))

tsens_20220110 <-
  system.file("extdata/2022-01-10-SigSci-TSENS.csv", package="fiphde") %>%
  readr::read_csv(show_col_types = FALSE)
creg_20220110 <-
  system.file("extdata/2022-01-10-SigSci-CREG.csv", package="fiphde") %>%
  readr::read_csv(show_col_types = FALSE)
combo_20220110 <- dplyr::bind_rows(
  dplyr::mutate(tsens_20220110, model = "SigSci-TSENS"),
  dplyr::mutate(creg_20220110, model = "SigSci-CREG")
)
plot_forecast(prepped_hosp, combo_20220110, location = "24")
plot_forecast(prepped_hosp, tsens_20220110, location = "24")
plot_forecast(prepped_hosp, combo_20220110, location = c("34","36"))
plot_forecast(prepped_hosp, creg_20220110, location = "US", .model = "SigSci-CREG")
plot_forecast(prepped_hosp, creg_20220110, location = "US", .model = "SigSci-CREG")

## demonstrating different prediction interval widths
plot_forecast(prepped_hosp, combo_20220110, location = "24", pi = 0.5)
plot_forecast(prepped_hosp, combo_20220110, location = "24", pi = 0.9)
plot_forecast(prepped_hosp, combo_20220110, location = "24", pi = 0.95)
plot_forecast(prepped_hosp, combo_20220110, location = "24", pi = NULL)

## End(Not run)
```

`plot_forecast_categorical` *Plot categorical forecasts*

Description

This function creates a bar plot for categorical forecasts. See examples for demonstration of usage.

Usage

```
plot_forecast_categorical(categorical_forecast, format = "hubverse")
```

Arguments

`categorical_forecast`	Either a `tibble` with categorical forecasts created with forecast_categorical or prepared forecast submission in "hubverse" format (see Details)
`format`	Either "hubverse" or "legacy"; the "hubverse" format will require an input forecast that includes output for "pmf" (see Details); default is "hubverse"

Details

The categorical plotting function works both with "legacy" formatting (i.e., format used in the 2022-23 FluSight season) and the "hubverse" formatting (i.e., format used in the 2023-24 FluSight season). Unlike the "legacy" format, the "hubverse" format allows for quantile and categorical forecasts to be co-mingled in the same submission object. If the format is specified as "hubverse", then the `plot_forecast_categorical()` function will interally look for the "pmf" forecasts.

Value

A `ggplot2` object with categorical forecasts shown as a stacked bar plot.

Examples

```
## Not run:
# Retrieve hospitalization data
h_raw <- get_hdgov_hosp(limitcols=TRUE)
# Prepare and summarize hospitalization data to weekly resolution
prepped_hosp <- prep_hdgov_hosp(h_raw)
# Create a keyed time series tibble with only locations of interest
prepped_tsibble <- make_tsibble(prepped_hosp,
                                    epiyear = epiyear,
                                    epiweek=epiweek,
                                    key=location)
# Run with default constrained ARIMA, nonseasonal ETS, no NNETAR
hosp_fitfor <- ts_fit_forecast(prepped_tsibble,
                               horizon=4L,
                               outcome="flu.admits")
# Prepare forecast for quantile submission format
prepped_forecast <- format_for_submission(hosp_fitfor$tsfor, method = "ts")
# Run categorical summary of quantiles for the time series ensemble
categorical_forecast <- forecast_categorical(prepped_forecast$ensemble, prepped_hosp)
# Plot the categorical forecast
plot_forecast_categorical(categorical_forecast, format = "legacy")

## End(Not run)
```

---

pois_forc *Simple Poisson count forecaster*

Description

This function is a helper that forecasts Poisson counts for near-term horizons based on characteristics of recently observed count data. The function effectively takes a rolling average of most recent observations (augmenting with each forecasted horizon as the horizons progress), then uses this average as the parameter for Lambda in a random draw from a Poisson distribution.

Usage

```
pois_forc(.data, .location, .var, horizon = 4)
```

Arguments

.data	Data frame with incoming data that includes a variable with counts (see ".var" argument), and location (must be stored in a column called "location") and a variable for sorting by date (must be stored in a column called "week_start")
.location	The name of the location of interest
.var	Bare, unquoted name of the variable with counts to be forecasted
horizon	The number of horizons ahead to forecast; must be one of 4 or 5; default is 4

Value

Vector with Poisson forecasts for the number of horizons specified.

Examples

```
## Not run:
all_clin <- get_cdc_clin()
va_ahead <-
  dplyr::tibble(
    n_positive = pois_forc(all_clin, .location = "51", n_positive),
    total = pois_forc(all_clin, .location = "51", total),
    p_positive = n_positive / total)
va_ahead

## End(Not run)
```

---

prep_hdgov_hosp *Prep hospitalization data*

Description

This function prepares hospitalization data retrieved using get_hdgov_hosp for downstream forecasting. The function optionally limits to states only, trims to a given date, removes incomplete weeks, and removes locations with little reporting over the last month.

Usage

```
prep_hdgov_hosp(
  hdgov_hosp,
  statesonly = TRUE,
  trim = list(epiyear = 2020, epiweek = 43),
  remove_incomplete = TRUE,
  min_per_week = 1
)
```

Arguments

hdgov_hosp	Daily hospital utilization data from get_hdgov_hosp
statesonly	Logical as to whether or not to limit to US+DC+States only (i.e., drop territories); default is TRUE
trim	Named list with elements for epiyear and epiweek corresponding to the minimum epidemiological week to retain; defaults to list(epiyear=2020, epiweek=43), which is the first date of report in the healthdata.gov hospitalization data; if set to NULL the data will not be trimmed
remove_incomplete	Logical as to whether or not to remove the last week if incomplete; defaults is TRUE.
min_per_week	The minimum number of flu.admits per week needed to retain that state. Default removes states with less than 1 flu admission per week over the last 30 days.

Value

A tibble with hospitalization data summarized to epiyear/epiweek with the following columns:

- **abbreviation**: Abbreviation for the location
- **location**: FIPS code for the location
- **week_start**: Date of beginning (Sunday) of the given epidemiological week
- **monday**: Date of Monday of the given epidemiological week
- **week_end**: Date of end (Saturday) of the given epidemiological week
- **epiyear**: Year of reporting (in epidemiological week calendar)
- **epiweek**: Week of reporting (in epidemiological week calendar)
- **flu.admits**: Count of flu cases among admitted patients on previous day
- **flu.admits.cov**: Coverage (number of hospitals reporting) for incident flu cases
- **ili_mean**: Estimate of historical ILI activity for the given epidemiological week
- **ili_rank**: Rank of the given epidemiological week in terms of ILI activity across season (1 being highest average activity)
- **hosp_mean**: Estimate of historical flu hospitalization rate for the given epidemiological week
- **hosp_rank**: Rank of the given epidemiological week in terms of flu hospitalizations across season (1 being highest average activity)

Examples

```
## Not run:
# Retrieve hospitalization data
hdgov_hosp <- get_hdgov_hosp(limitcols=TRUE)
# Prepare and summarize to weekly resolution
h <- prep_hdgov_hosp(hdgov_hosp)
h

## End(Not run)
```

`replace_ili_nowcast` *Replace ILINet data with nowcast*

Description

This function replaces the weighted ILI retrieved from get_cdc_ili with nowcast data for each of the locations in the original data. The function will first attempt to use ILI Nearby nowcasts pulled using get_nowcast_ili. If the ILI Nearby nowcasts are unavailable, the function will optionally fallback to a pseudo nowcast method that averages the observed ILI for the 4 most recent weeks. The nowcast data will be used to add 1 additional week to the observed ILI data and (optionally) replace the number of weeks specified in the "weeks_to_replace" argument.

Usage

```
replace_ili_nowcast(
  ilidat,
  start_date = NULL,
  weeks_to_replace = 1,
  fallback = TRUE,
  try_api = TRUE
)
```

Arguments

`ilidat`	ILI data retrieved via get_cdc_ili
`start_date`	Date from which to start nowcasting; default is lubridate::today
`weeks_to_replace`	Number of weeks of `ilidat` to replace; default is `1`
`fallback`	Logical as to whether or not to fall back to pseudo nowcast (average of last 4 ILI weeks in the given location) if nowcast data is unavailable; default is `TRUE`
`try_api`	Logical as to whether or not the function should try the ILI Nearby nowcast API; default is `TRUE`; if `FALSE` then the function will not attempt to query the API at all

Value

A `tibble` with the following columns:

- **location**: FIPS code for the location
- **region_type**: The type of location
- **abbreviation**: Abbreviation for the location
- **region**: Name of the region
- **epiyear**: Year of reporting (in epidemiological week calendar)
- **epiweek**: Week of reporting (in epidemiological week calendar)
- **week_start**: Date of beginning (Sunday) of the given epidemiological week
- **weighted_ili**: Population-weighted percentage of ILI outpatient visits

Examples

```
## Not run:
ilidat <- get_cdc_ili(years=2021)
ilidat <-
  ilidat %>%
  dplyr::filter(location=="US" | abbreviation=="VA") %>%
  dplyr::group_by(location) %>%
  dplyr::slice_max(week_start, n=4) %>%
  dplyr::select(location:weighted_ili)
ilidat
iliaug <- replace_ili_nowcast(ilidat, weeks_to_replace=1)
iliaug

## End(Not run)
```

---

round_preserve *Round and preserve vector*

Description

This unexported helper is used to ensure that categorical forecasts are rounded to sum to 1.

Usage

```
round_preserve(x, digits = 0)
```

Arguments

x	Numeric vector with values to round
digits	The number of digits to use in precision; defalut is 0

Value

Vector of same length as "x" with values rounded

---

smoothie *Calculate smoothed and weighted averages of previous observations*

Description

This helper function calculates a weighted average of the last n observations.

Usage

```
smoothie(x, n = 4, weights = c(1, 2, 3, 4))
```

Arguments

x	Incoming vector of observations
n	Number of recent observations to smooth; default is 4
weights	Vector of weights to be applied to last n observations during averaging

Value

Vector of length 1 with the weighted average of last n observations.

Examples

```
## Not run:
## pull and prep weekly US flu hospitalization data
hosp_us <-
  get_hdgov_hosp() %>%
  prep_hdgov_hosp() %>%
  dplyr::filter(location == "US")

## what do the last 4 observations look like?
tail(hosp_us$flu.admits, 4)

## smooth over last 4 with default weights
smoothie(hosp_us$flu.admits, n=4, weights=c(1,2,3,4))

## try smoothing over last 4 with different weights (exponential this time)
smoothie(hosp_us$flu.admits, n=4, weights=exp(1:4))

## End(Not run)
```

---

this_monday *Get Monday*

Description

This function is a helper to get the date for the Monday of the current week. The function determines the current week based on epidemiological week orientation (i.e., week begins with Sunday).

Usage

```
this_monday()
```

Value

Date for the Monday of the current week.

Examples

```
this_monday()
```

this_saturday *Get Saturday*

Description

This function is a helper to get the date for the Saturday of the current week. The function determines the current week based on epidemiological week orientation (i.e., week begins with Sunday).

Usage

```
this_saturday()
```

Value

Date for the Saturday of the current week.

Examples

```
this_saturday()
```

to_num *Clean numeric values*

Description

This unexported helper is used in the ilinet function to strip special characters and empty space and convert a character vector to numeric.

Usage

```
to_num(x)
```

Arguments

x	Input character vector for which special characters should be stripped and converted

Value

Numeric vector

`ts_fit_forecast` *Fit and forecast with time-series approaches*

Description

This function allows the user to fit time series models and forecast values out to a specified horizon. Starting from a `tsibble` object (see make_tsibble), the function fits the models specified as a list in the "models" argument. The "Details" section provides more information on how to parameterize the models used. Note that if the input `tsibble` is "keyed" (e.g., grouped by location) then the procedure will fit and forecast independently for each grouping.

Usage

```
ts_fit_forecast(
  prepped_tsibble,
  outcome = "flu.admits",
  horizon = 4L,
  trim_date = "2021-01-01",
  models = list(arima = "PDQ(0, 0, 0) + pdq(1:2, 0:2, 0)", ets =
    "season(method=\"N\")", nnetar = NULL),
  covariates = TRUE,
  ensemble = TRUE
)
```

Arguments

`prepped_tsibble`	A `tsibble` with data formatted via make_tsibble
`outcome`	The outcome variable to model; default is `"flu.admits"`
`horizon`	Number of weeks ahead to forecast
`trim_date`	The date (YYYY-MM-DD) at which time series models should start fitting; default `"2021-01-01"`; if set to `NULL` the input data will not be trimmed (i.e., all data will be used to fit time series models)
`models`	A list of right hand side formula contents for models you want to run; default is `list(arima='PDQ(0, 0, 0) + pdq(1:2, 0:2, 0)', ets='season(method="N")', nnetar=NULL)` which runs a constrained ARIMA, non-seasonal ETS, and ignores the NNETAR model; see "Details" for more information
`covariates`	Logical. Should flu hospitalization-specific covariates that should be modeled with the time series? If so, historical hospitalization and ILI rank for each epidemiological week, brought in with prep_hdgov_hosp, is added to the ARIMA model.
`ensemble`	Logical as to whether or not the models should be ensembled (using mean); default `TRUE`

Details

When fitting time series models, the set of models used (and their parameters) can be defined via a named list passed to the "models" argument. The list should contain elements that define the right-hand side of model formulas. The function internally uses the fable::fable package, and any models provided must be part of the `fable` ecosystem of time series models. The models passed must be named as "arima", "ets", and "nnetar". To skip any one of these models set the named argument for the given model to `NULL`. The "models" argument defaults to `list(arima = "PDQ(0, 0, 0) + pdq(1:2, 0:2, 0)", ets = "season(method='N')", nnetar = NULL)`. To run an unconstrained ARIMA: `list(arima='PDQ() + pdq()')` (see fable::ARIMA). To run a seasonal exponential smoothing: `list(ets='season(method=c("A", "M", "N"), period="3 months")')` (see fable::ETS). To run an autoregressive neural net with P=1: `list(nnetar="AR(P=1)")` (see fable::NNETAR).

Value

A list of the time series fit, time series forecast, and model formulas.

- **tsfit**: A `mdl_df` class "mable" with one row for each location, columns for arima and ets models.
- **tsfor**: A `fbl_ts` class "fable" with one row per location-model-timepoint up to `horizon` number of time points.
- **formulas**: A list of ARIMA, ETS, and/or NNETAR formulas

References

https://fable.tidyverts.org/

Examples

```
## Not run:
# Retrieve hospitalization data
h_raw <- get_hdgov_hosp(limitcols=TRUE)
# Prepare and summarize hospitalization data to weekly resolution
prepped_hosp <- prep_hdgov_hosp(h_raw)
# Create a keyed time series tibble with only locations of interest
prepped_tsibble <- make_tsibble(prepped_hosp,
                                epiyear = epiyear,
                                epiweek=epiweek,
                                key=location) %>%
  dplyr::filter(location %in% c("US", "51"))

# Run with default constrained ARIMA, nonseasonal ETS, no NNETAR
hospfor1 <- ts_fit_forecast(prepped_tsibble,
                            horizon=4L,
                            outcome="flu.admits",
                            covariates=TRUE)
# Run an unconstrained ARIMA, seasonal ETS, no NNETAR
hospfor2 <- ts_fit_forecast(prepped_tsibble,
                            horizon=4L,
                            outcome="flu.admits",
```

```
                              covariates=TRUE,
                              models=list(arima='PDQ() + pdq()',
                                  ets='season(method=c("A", "M", "N"), period="3 months")',
                                          nnetar=NULL))
# Run an unconstrained ARIMA, seasonal ETS, NNETAR
hospfor3 <- ts_fit_forecast(prepped_tsibble,
                              horizon=4L,
                              outcome="flu.admits",
                              covariates=TRUE,
                              models=list(arima='PDQ() + pdq()',
                                  ets='season(method=c("A", "M", "N"), period="3 months")',
                                          nnetar="AR(P=1)"))

## End(Not run)
```

`ts_format_for_submission` *Format time series forecast*

Description

This function specifically formats time series forecasts generated with ts_fit_forecast to adhere to probabilistic forecast submission guidelines for consortia such as FluSight. It is used as a helper in format_for_submission.

Usage

```
ts_format_for_submission(
  tsfor,
  .target = "wk ahead inc flu hosp",
  .counts = TRUE
)
```

Arguments

`tsfor`	The forecast from ts_fit_forecast
`.target`	Name of the target in the forecast; default is `"wk ahead inc flu hosp"`
`.counts`	Logical; default `TRUE` indicates that the target outcome is a count, and should be rounded off at an integer

Details

Uses quantiles `c(0.01, 0.025, seq(0.05, 0.95, by = 0.05), 0.975, 0.99)` in the built-in `fiphde:::q`, using an accessory table `fiphde:::quidk`.

Value

A named list of tibbles with probabilistic forecasts (one for each model), formatted for submission with the following columns:

- **forecast_date**: Date of forecast
- **target**: Horizon and name of forecasted target
- **target_end_date**: Last date of the forecasted target (e.g., Saturday of the given epidemiological week)
- **location**: FIPS code for location
- **type**: One of either "point" or "quantile" for the forecasted value
- **quantile**: The quantile for the forecasted value; NA if "type" is "point"
- **value**: The forecasted value

References

https://github.com/cdcepi/Flusight-forecast-data/blob/master/data-forecasts/README.md

Examples

```
## Not run:
# Get raw data from healthdata.gov
h_raw <- get_hdgov_hosp(limitcols=TRUE)

# Prep, and make a tsibble
prepped_hosp <- prep_hdgov_hosp(h_raw, statesonly=TRUE)
prepped_hosp_tsibble <- make_tsibble(prepped_hosp,
                                     epiyear = epiyear,
                                     epiweek=epiweek,
                                     key=location)
# Limit to only Virginia and US
prepped_hosp_tsibble <-
  prepped_hosp_tsibble %>%
  dplyr::filter(location %in% c("US", "51"))

# Fit a model
hosp_fitfor <- ts_fit_forecast(prepped_hosp_tsibble,
                               horizon=4L,
                               outcome="flu.admits",
                               covariates=TRUE)

# Format for submission
formatted_list <- ts_format_for_submission(hosp_fitfor$tsfor)
formatted_list

## End(Not run)
```

validate_forecast *Validate forecast submission*

Description

This function will take the prepped forecast data from format_for_submission and run a series of tests to validate the format.

Usage

```
validate_forecast(subdat)
```

Arguments

subdat	A tibble with submission ready forecasts prepped by and stored in output of format_for_submission

Value

Named list with elements for each test (including logical for whether or not test passed and message if failed) and an overall "valid" logical with TRUE if all tests passed an FALSE if at least one failed

Examples

```
## Not run:
# Get raw data from healthdata.gov
h_raw <- get_hdgov_hosp(limitcols=TRUE)

# Prep, and make a tsibble
prepped_hosp <- prep_hdgov_hosp(h_raw, statesonly=TRUE)
prepped_hosp_tsibble <- make_tsibble(prepped_hosp,
                                     epiyear = epiyear,
                                     epiweek=epiweek,
                                     key=location)
# Limit to only Virginia and US
prepped_hosp_tsibble <-
  prepped_hosp_tsibble %>%
  dplyr::filter(location %in% c("US", "51"))

# Fit a model
hosp_fitfor <- ts_fit_forecast(prepped_hosp_tsibble,
                               horizon=4L,
                               outcome="flu.admits",
                               covariates=TRUE)



# Format for submission
formatted_list <- format_for_submission(hosp_fitfor$tsfor, method = "ts")
```

```
# Validate one of the forecasts
# Note that this expects forecast is prepared with forecast date = Monday of the current week
ens_forc <- formatted_list$ensemble
ens_forc$forecast_date <- this_monday()
validate_forecast(ens_forc)

## End(Not run)
```

`who_nrevss` *WHO/NREVSS clinical lab surveillance data*

Description

Adapted from `cdcfluview::who_nrevss`.

This unexported helper function leverages the CDC FluView API to pull flu surveillance data collected from U.S. World Health Organization (WHO) Collaborating Laboratories and National Respiratory and Enteric Virus Surveillance System (NREVSS) laboratories. The data retrieved can be parameterized by geographic granularity and/or flu season. The function is used internally by get_cdc_clin.

Usage

```
who_nrevss(region = c("national", "hhs", "census", "state"), years = NULL)
```

Arguments

`region`	One of `"national"`, `"hhs"`, `"census"`, or `"state"`
`years`	A vector of years to retrieve data for (i.e. `2014` for CDC flu season 2014-2015). CDC has data for this API going back to 1997. Default value (`NULL`) means retrieve **all** years. NOTE: if you specify a 2-digit season value the function will convert that to the corresponding season identifier (i.e. `57` == 2017-2018).

References

- cdcfluview package

Index