Package 'evalcast'

October 12, 2020

Type Package

Title Evaluating COVID-19 Forecasters Built on Covidcast

Version 0.1.0

Description This package is a tool for those developing probabilistic COVID-19 forecasters. It provides functionality for accurately evaluating forecaster performance. Crucially, evalcast leverages the covidcast R package's ``as of'' capability, which allows one to get the data that would have been known as of a particular date in the past. This is important for honest evaluation of COVID-19 forecasters because data sources often perform ``backfill'' in which previous estimates about the past are updated. Without properly accounting for backfill, traditional backtesting can lead to overly optimistic evaluations of one's forecaster. Furthermore, naively training on historical data that has already been backfilled may lead a trained model to rely too heavily on the most recent data that has yet to settle. Such forecasters may end up performing far worse in prospective evaluation than in backtesting.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Remotes github::cmu-delphi/covidcast/R-packages/covidcast@main

Imports covidcast, dplyr, magrittr, lubridate, purrr, tibble, tidyr, stringr, readr, MMWRweek, zoo, rvest, xml2, rlang

Suggests testthat

NeedsCompilation no

Author Jacob Bien [aut, cre],
Alden Green [aut],
Balasubramanian Narasimhan [aut],
Samyak Rajanala [ctb],
Aaron Rumack [ctb],
Ryan Tibshirani [ctb]

Maintainer Jacob Bien <jbien@usc.edu>

R topics documented:

`evalcast-package` *evalcast*

Description

A package to evaluate forecasters using the `covidcast` R package.

Author(s)

Jacob Bien Maintainer: Jacob Bien <jbien@usc.edu>

`absolute_error` *Compute absolute error*

Description

Computes absolute error between the actual value and the median of the forecast distribution.

Usage

```
absolute_error(quantile_forecasts, actual_value)
```

all_attr *Return list of attributes*

Description

Given a list of cards, returns a list of the same length giving the values of that attribute across all cards.

Usage

```
all_attr(cards, attribute)
```

Arguments

cards	a list of predictions cards or a list of scorecards
attribute	name of attribute

baseline_forecaster *Baseline forecaster*

Description

This serves as a template for a forecaster. It's not intended to be a great forecaster.

Usage

```
baseline_forecaster(
  df,
  forecast_date,
  signals,
  incidence_period = c("epiweek", "day"),
  ahead,
  geo_type
)
```

Arguments

df	a data frame of the format that is outputted by covidcast_signal.
forecast_date	date on which forecasts will be made about some period (e.g., epiweek). For example, if forecast_date is ymd("2020-05-11"), incidence_period is "day", and ahead = 3, then, we'd be making forecasts for "2020-05-14".
signals	a tibble with columns "data_source" and "signal" that specifies which variables from the covidcast API will be used by my_forecaster. The first row of signals is taken to be the response. If using incidence_period "epiweek", thre response should be something for which summing daily values over an epiweek makes sense (e.g., counts or proportions but not log(counts) or log(proportions)). Available data sources and signals are documented in the [COVIDcast signal documentation](https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html) documentation. Can optionally include a column "first_day" giving the earliest day of data needed from that data source.

`incidence_period`
: either "epiweek" or "day"

`ahead`
: vector of (one or more) integers. How many epiweeks/days ahead are you forecasting? If incidence_period is "epiweek" and forecast_date is Sunday or Monday, then ahead = 1 means the epiweek that includes the forecast date; if forecast_date is Tuesday-Saturday, then it is the following epiweek. If incidence_period is "day" then ahead = 1 means the day after forecast date.

`geo_type`
: one of the geographical types, matching the terms used on the api ("county", "dma", "hrr", "msa", "state")

Value

A data frame with columns "ahead", "location", "probs", "quantiles". The quantiles column gives the probs-quantile of the forecast distribution for that location and ahead.

`check_valid_forecaster_output` *Check that forecaster's output is valid*

Description

Check that forecaster's output is valid

Usage

```
check_valid_forecaster_output(pred_card)
```

Arguments

`pred_card`
: tibble of form created by `get_predictions`

`download_signal` *Download signal from covidcast*

Description

This is a simple wrapper to `covidcast_signal` that is less verbose.

Usage

```
download_signal(...)
```

`evaluate_predictions` *Create a list of scorecards*

Description

This function performs backtesting. It does the following:

1. Takes a list of predictions cards (as created by `get_predictions`). These should be from a single forecaster, each card corresponding to a different forecast date.
2. Downloads from the covidcast API the latest available data to compute what actually occurred (summing the response over the incidence period).
3. Computes various user-specified error measures.

The result is a list of score cards, where each list element corresponds to a distinct value of ahead. A scorecard is a data frame in which each row corresponds to a location-day pair, and the columns give the values of the error measures (along with other information including the forecast distributions and the actual response values).

Usage

```
evaluate_predictions(
  predictions_cards,
 err_measures = list(wis = weighted_interval_score, ae = absolute_error, coverage_80 =
    interval_coverage(alpha = 0.2)),
  backfill_buffer = 10
)
```

Arguments

`predictions_cards`
: a list of prediction cards from the same forecaster that are all for the same prediction task, meaning they are for the same response, incidence_period, ahead, and geo_type. Each should be from a different forecast date. A predictions card is created by the function `get_predictions`.

`err_measures`
: a named list of one or more functions, where each function takes a data frame with two columns "probs" and "quantiles" and an actual (i.e. observed) scalar value and returns some measure of error. If empty, returns the scorecard without any error measure columns.

`backfill_buffer`
: How many days until response is deemed trustworthy enough to be taken as correct? See details for more.

Details

Backfill refers to the process by which some data sources go back in time updating previously reported values. Suppose it is Sept. 14 and we are evaluating our predictions for what happened in the previous epiweek (Sept. 6 - Sept. 12). Although we may be able to calculate a value for "actual", we might not trust this value since on Sept. 16, backfill may occur changing what is known about the period Sept. 6 - Sept. 12. There are two consequences of this phenomenon. First, running this function on different dates may result in different estimates of the error. Second, we may not trust the evaluations we get that are too recent. The parameter `backfill_buffer` specifies how long of

a buffer period we should enforce. This will be dependent on the data source and signal and is left to the user to determine. If backfill is not relevant for the particular signal you are predicting, then you can set backfill_buffer to 0.

Value

a list of score cards (one for each ahead)

`filter_predictions` *Filter a list of predictions cards based on attributes*

Description

Filter a list of predictions cards based on attributes

Usage

```
filter_predictions(
  predictions_cards,
  name_of_forecaster = NULL,
  response_data_source = NULL,
  response_signal = NULL,
  forecast_date = NULL,
  incidence_period = NULL,
  ahead = NULL,
  geo_type = NULL
)
```

Arguments

`predictions_cards` a list of predictions cards

`get_covidhub_predictions` *Get predictions from a forecaster on COVIDHub*

Description

This function simply converts the predictions of forecasters submitting to the COVID Hub https://github.com/reichlab/cov forecast-hub/ to the format of a predictions card, so it can be easily evaluated and compared.

Usage

```
get_covidhub_predictions(covid_hub_forecaster_name, forecast_dates = NULL, ...)
```

Arguments

`covid_hub_forecaster_name` the name of the forecaster matching what it is called on covid hub

`forecast_dates` a vector of class Date on which forecasts will be made. e.g. `c(lubridate::ymd("2020-09-07"),lub`

`...` additional parameters to be passed to `filter_predictions`.

Details

For now, this function only supports (i) incident not cumulative predictions and (ii) epiweek not daily incidence_period predictions.

Value

a list of predictions cards

---

get_forecast_dates *Get Available Forecast Dates for Forecaster on COVID Hub*

Description

Retrieves the forecast dates that a forecaster submitted to the COVID Hub https://github.com/reichlab/covid19-forecast-hub/

Usage

```
get_forecast_dates(covid_hub_forecaster_name)
```

Arguments

`covid_hub_forecaster_name`
: the name of a forecaster on the COVID Hub.

---

get_predictions *Get predictions*

Description

For each of the provided forecast dates, runs a forecaster using the data that would have been available as of that given forecast date. Returns a list of "predictions cards." A prediction card is a data frame giving the forecast distributions of a given forecaster for a given forecast task. A forecast task is specified by the forecast date, ahead, response, incidence period, and geo_type (e.g., 1-epiweek ahead death forecasting at the state level with predictions made using the information as of Sept. 14).

Usage

```
get_predictions(
  forecaster,
  name_of_forecaster,
  signals,
  forecast_dates,
  incidence_period,
  ahead,
  geo_type,
  geo_values = "*"
)
```

Arguments

`forecaster`	a function that outputs a tibble with columns...
`name_of_forecaster`	the name of forecaster
`signals`	a tibble with columns "data_source" and "signal" that specifies which variables from the covidcast API will be used by my_forecaster. The first row of signals is taken to be the response. If using incidence_period "epiweek", thre response should be something for which summing daily values over an epiweek makes sense (e.g., counts or proportions but not log(counts) or log(proportions)). Available data sources and signals are documented in the [COVIDcast signal documentation](https://cmu-delphi.github.io/delphi-epidata/api/covidcast_signals.html) documentation. Can optionally include a column "first_day" giving the earliest day of data needed from that data source.
`forecast_dates`	a vector of class Date on which forecasts will be made. e.g. `c(lubridate::ymd("2020-09-07"),lub`
`incidence_period`	either "epiweek" or "day"
`ahead`	vector of (one or more) integers. How many epiweeks/days ahead are you forecasting? If incidence_period is "epiweek" and forecast_date is Sunday or Monday, then ahead = 1 means the epiweek that includes the forecast date; if forecast_date is Tuesday-Saturday, then it is the following epiweek. If incidence_period is "day" then ahead = 1 means the day after forecast date.
`geo_type`	one of the geographical types, matching the terms used on the api ("county", "dma", "hrr", "msa", "state")
`geo_values`	see covidcast_signal for a description of this parameter.

Details

A predictions card has two columns:

1. location - the FIPS code of the location. For counties, this is the same as the geo_value used in the API. However, for states, the location and geo_value are different because the API uses state abbreviations instead of FIPS for geo_value.
2. forecast_distribution - this is a list column... each element itself contains a tibble with the covidhub quantiles.

A predictions card has attributes that specify the exact forecasting task that was being carried out and the name of the forecaster.

Value

a list of predictions cards

get_predictions_single_date *Get predictions cards for a single date*

Description

Get predictions cards for a single date

Usage

```
get_predictions_single_date(
  forecaster,
  name_of_forecaster,
  signals,
  forecast_date,
  incidence_period,
  ahead,
  geo_type,
  geo_values
)
```

get_target_response *returns data frame with column names "forecast_date", "location", "target_start", "target_end", "actual"*

Description

returns data frame with column names "forecast_date", "location", "target_start", "target_end", "actual"

Usage

```
get_target_response(signals, forecast_dates, incidence_period, ahead, geo_type)
```

intersect_locations *Remove locations that are not in all cards*

Description

Remove locations that are not in all cards

Usage

```
intersect_locations(cards)
```

Arguments

cards	a list of predictions cards or a list of scorecards

interval_coverage *Generate interval coverage error measure function*

Description

This function returns an error measure function indicating whether a central interval covers the actual value. The interval is defined as the (alpha / 2)-quantile to the (1 - alpha / 2)-quantile.

Usage

```
interval_coverage(alpha)
```

Arguments

alpha	used to specify the nominal coverage of the interval

plot_coverage *Plot the interval coverage*

Description

Plot the interval coverage

Usage

```
plot_coverage(scorecards, alpha = 0.2, type = c("all", "one"))
```

Arguments

scorecards	a list of different forecasters scorecards, all on the same forecasting task (i.e., same ahead, etc.)
alpha	location of vertical line if type = "all" is 1-alpha; if type="one" then 1-alpha is the nominal coverage probability shown.
type	whether to show coverage across all nominal levels (in which case averaging is performed across forecast dates and locations) or whether to show it for one specific alpha value.

plot_measure *Plot a measure*

Description

Plot a measure

Usage

```
plot_measure(scorecards, err_name, type = "boxplot")
```

Arguments

scorecards	a list of scorecards
err_name	the name of a column appearing in all scorecards

plot_width *Plot the interval width*

Description

Interval width does not depend on the actual outcome, so this function can be called on predictions cards in addition to scorecards.

Usage

```
plot_width(cards, alpha = 0.2)
```

Arguments

cards	a list of different forecasters scorecards (or predictions cards), all on the same forecasting task (i.e., same ahead, etc.)
location	of vertical line

unique_attr *Return unique value of attribute or throw error*

Description

If TRUE, returns the unique value; if FALSE, throws an error.

Usage

```
unique_attr(cards, attribute)
```

Arguments

cards	a list of predictions cards or a list of scorecards
attribute	name of attribute

`weighted_interval_score` *Compute weighted interval score*

Description

For more details, see https://arxiv.org/abs/2005.12881

Usage

```
weighted_interval_score(quantile_forecasts, actual_value)
```

Details

Bracher, J., Ray, E. L., Gneiting, T., & Reich, N. G. (2020). Evaluating epidemic forecasts in an interval format. arXiv preprint arXiv:2005.12881.

Index