Package 'delphiBackfillCorrection'

September 26, 2022

Type Package

Title Correct signal outliers

Version 1.0

Date 2022-08-24

Author Jingjing Tang

Maintainer Jingjing Tang <jtang2@andrew.cmu.edu>

Description Takes auxiliary output from COVIDcast API data pipelines and adjusts unusual values using a lasso-penalized quantile regression. Output is used for research and model development.

License file LICENSE

Depends R (>= 3.5.0),

Imports dplyr, plyr, readr, tibble, stringr, covidcast, quantgen, arrow, evalcast, jsonlite, lubridate, tidyr, zoo, utils, rlang, parallel

Suggests knitr (>= 1.15), rmarkdown (>= 1.4), testthat (>= 1.0.1), covr (>= 2.2.2)

RoxygenNote 7.2.0

Encoding UTF-8

NeedsCompilation no

R topics documented:

add_7davs_and_target . . . 2
add_dayofweek . . . 3
add_params_for_dates . . . 3
add_shift . . . 4
add_sqrtscale . . . 4
add_weekofmonth . . . 5
create_dir_not_exist . . . 5
create_name_pattern . . . 6
data_filteration . . . 6

delta . . . 7
est_priors . . . 7
evaluate . . . 9
export_test_result . . . 9
fill_missing_updates . . . 10
fill_rows . . . 11
frac_adj . . . 11
frac_adj_with_pseudo . . . 13
generate_filename . . . 13
get_7dav . . . 14
get_files_list . . . 15
get_model . . . 15
get_populous_counties . . . 16
get_weekofmonth . . . 16
main . . . 17
main_local . . . 17
model_training_and_testing . . . 18
objective . . . 20
read_data . . . 20
read_params . . . 21
run_backfill . . . 22
run_backfill_local . . . 23
subset_valid_files . . . 24
training_days_check . . . 24
validity_checks . . . 25

Index **26**

`add_7davs_and_target` *Add 7dav and target to the data Target is the updates made ref_lag days after the first release*

Description

Add 7dav and target to the data Target is the updates made ref_lag days after the first release

Usage

```
add_7davs_and_target(df, value_col, refd_col, lag_col, ref_lag = REF_LAG)
```

Arguments

`df`	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
`value_col`	string specifying name of value (counts) field within the input dataframe.
`refd_col`	string specifying name of reference date field within the input dataframe.
`lag_col`	string specifying name of lag field within the input dataframe.
`ref_lag`	max lag to use for training

add_dayofweek *Add one hot encoding for day of a week info in terms of reference and issue date*

Description

Add one hot encoding for day of a week info in terms of reference and issue date

Usage

```
add_dayofweek(df, time_col, suffix, wd = WEEKDAYS_ABBR)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
time_col	string specifying name of column used for the date, can be either reference date or issue date
suffix	suffix added to indicate which kind of date is used
wd	vector of days of a week

add_params_for_dates *Add params related to date*

Description

Target is the updates made ref_lag days after the first release

Usage

```
add_params_for_dates(df, refd_col, lag_col)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
refd_col	string specifying name of reference date field within the input dataframe.
lag_col	string specifying name of lag field within the input dataframe.

add_shift *Used for data shifting in terms of reference date*

Description

Used for data shifting in terms of reference date

Usage

```
add_shift(df, n_day, refd_col)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
n_day	number of days to be shifted
refd_col	string specifying name of reference date field within the input dataframe.

add_sqrtscale *Add columns to indicate the scale of value at square root level*

Description

Add columns to indicate the scale of value at square root level

Usage

```
add_sqrtscale(train_data, test_data, max_raw, value_col)
```

Arguments

train_data	Data Frame containing training data
test_data	Data Frame for testing
max_raw	the maximum value in the training data at square root level
value_col	string specifying name of value (counts) field within the input dataframe.

add_weekofmonth *Add one hot encoding for week of a month info in terms of issue date*

Description

Add one hot encoding for week of a month info in terms of issue date

Usage

```
add_weekofmonth(df, time_col, wm = WEEK_ISSUES)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
time_col	string specifying name of column used for the date, can be either reference date or issue date
wm	vector of weeks of a month

create_dir_not_exist *Create directory if not already existing*

Description

Create directory if not already existing

Usage

```
create_dir_not_exist(path)
```

Arguments

path	string specifying a directory to create

create_name_pattern *Create pattern to match input files of a given type and signal*

Description

Create pattern to match input files of a given type and signal

Usage

```
create_name_pattern(indicator, signal, file_type = c("daily", "rollup"))
```

Arguments

indicator	string specifying the name of the indicator as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
file_type	string specifying time period coverage of input files. Either "daily" or "rollup"

data_filteration *Filtration for training and testing data with different lags*

Description

Filtration for training and testing data with different lags

Usage

```
data_filteration(test_lag, geo_train_data, geo_test_data, lag_pad)
```

Arguments

test_lag	integer number of days ago to predict for
geo_train_data	training data for a certain location
geo_test_data	testing data for a certain location
lag_pad	lag padding for training

delta *Sum of squared error*

Description

Sum of squared error

Usage

```
delta(fit, actual)
```

Arguments

fit	estimated values
actual	actual values

est_priors *Main function for the beta prior approach Estimate the priors for the beta distribution based on data for a certain day of a week*

Description

Main function for the beta prior approach Estimate the priors for the beta distribution based on data for a certain day of a week

Usage

```
est_priors(
  train_data,
  prior_test_data,
  geo,
  value_type,
  dw,
  taus,
  covariates,
  response,
  lp_solver,
  lambda,
  indicator,
  signal,
  geo_level,
  signal_suffix,
  training_end_date,
  model_save_dir,
  start = c(0, log(10)),
```

```
  base_pseudo_denom = 1000,
  base_pseudo_num = 10,
  train_models = TRUE,
  make_predictions = TRUE
)
```

Arguments

train_data	Data Frame containing training data
prior_test_data	Data Frame for testing
geo	string specifying the name of the geo region (e.g. FIPS code for counties)
value_type	string describing signal type. Either "count" or "fraction".
dw	column name to indicate which day of a week it is
taus	numeric vector of quantiles to be predicted. Values must be between 0 and 1.
covariates	character vector of column names serving as the covariates for the model
response	the column name of the response variable
lp_solver	string specifying the lp solver to use in Quantgen fitting. Either "glpk" or "gurobi". For faster optimization, use Gurobi (requires separate installation of the ‘gurobi‘ package).
lambda	the level of lasso penalty
indicator	string specifying the name of the indicator as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
geo_level	string describing geo coverage of input data. Either "state" or "county".
signal_suffix	string specifying value column name ending to be appended to standard value column names from ‘params$num_col‘ and ‘params$denom_col‘. Used for non-standard value column names and when processing multiple signals from a single input dataframe, as with ‘quidel‘’s age buckets.
training_end_date	the most recent training date
model_save_dir	directory containing trained models
start	the initialization of the the points in nlm
base_pseudo_denom	the pseudo counts added to denominator if little data for training
base_pseudo_num	the pseudo counts added to numerator if little data for training
train_models	boolean indicating whether to train models (TRUE). If FALSE previously trained models (stored locally) will be used instead. Default is TRUE.
make_predictions	boolean indicating whether to generate and save corrections (TRUE) or not. Default is TRUE.

`evaluate` *Evaluation of the test results based on WIS score The WIS score calculation is based on the weighted_interval_score function from the 'evalcast' package from Delphi*

Description

Evaluation of the test results based on WIS score The WIS score calculation is based on the weighted_interval_score function from the 'evalcast' package from Delphi

Usage

```
evaluate(test_data, taus)
```

Arguments

`test_data`	dataframe with a column containing the prediction results of each requested quantile. Each row represents an update with certain (reference_date, issue_date, location) combination.
`taus`	numeric vector of quantiles to be predicted. Values must be between 0 and 1.

`export_test_result` *Export the result to customized directory*

Description

Export the result to customized directory

Usage

```
export_test_result(
  test_data,
  coef_data,
  indicator,
  signal,
  geo_level,
  signal_suffix,
  lambda,
  training_end_date,
  value_type,
  export_dir
)
```

Arguments

test_data	test data containing prediction results
coef_data	data frame containing the estimated coefficients
indicator	string specifying the name of the indicator as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
geo_level	string describing geo coverage of input data. Either "state" or "county".
signal_suffix	string specifying value column name ending to be appended to standard value column names from ‘params$num_col‘ and ‘params$denom_col‘. Used for non-standard value column names and when processing multiple signals from a single input dataframe, as with ‘quidel‘’s age buckets.
lambda	the level of lasso penalty
training_end_date	the most recent training date
value_type	string describing signal type. Either "count" or "fraction".
export_dir	path to directory to save output to

---

fill_missing_updates *Get pivot table, filling NANs. If there is no update on issue date D but previous reports exist for issue date D_p < D, all the dates between [D_p, D] are filled with with the reported value on date D_p. If there is no update for any previous issue date, fill in with 0.*

Description

Get pivot table, filling NANs. If there is no update on issue date D but previous reports exist for issue date D_p < D, all the dates between [D_p, D] are filled with with the reported value on date D_p. If there is no update for any previous issue date, fill in with 0.

Usage

```
fill_missing_updates(df, value_col, refd_col, lag_col)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
value_col	string specifying name of value (counts) field within the input dataframe.
refd_col	string specifying name of reference date field within the input dataframe.
lag_col	string specifying name of lag field within the input dataframe.

`fill_rows` *Re-index, fill na, make sure all reference date have enough rows for updates*

Description

Re-index, fill na, make sure all reference date have enough rows for updates

Usage

```
fill_rows(df, refd_col, lag_col, min_refd, max_refd, ref_lag = REF_LAG)
```

Arguments

`df`	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
`refd_col`	string specifying name of reference date field within the input dataframe.
`lag_col`	string specifying name of lag field within the input dataframe.
`min_refd`	the earliest reference date considered in the data
`max_refd`	the latest reference date considered in the data
`ref_lag`	max lag to use for training

Value

df_new Data Frame with filled rows for missing lags

`frac_adj` *Update fraction using beta prior approach*

Description

Update fraction using beta prior approach

Usage

```
frac_adj(
  train_data,
  test_data,
  prior_test_data,
  indicator,
  signal,
  geo_level,
  signal_suffix,
  lambda,
```

```
  value_type,
  geo,
  training_end_date,
  model_save_dir,
  taus = TAUS,
  lp_solver = LP_SOLVER,
  train_models = TRUE,
  make_predictions = TRUE
)
```

Arguments

train_data	Data Frame containing training data
test_data	testing data
prior_test_data	testing data for the lag -1 model
indicator	string specifying the name of the indicator as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
geo_level	string describing geo coverage of input data. Either "state" or "county".
signal_suffix	string specifying value column name ending to be appended to standard value column names from ‘params$num_col‘ and ‘params$denom_col‘. Used for non-standard value column names and when processing multiple signals from a single input dataframe, as with ‘quidel‘’s age buckets.
lambda	the level of lasso penalty
value_type	string describing signal type. Either "count" or "fraction".
geo	string specifying the name of the geo region (e.g. FIPS code for counties)
training_end_date	the most recent training date
model_save_dir	directory containing trained models
taus	numeric vector of quantiles to be predicted. Values must be between 0 and 1.
lp_solver	string specifying the lp solver to use in Quantgen fitting. Either "glpk" or "gurobi". For faster optimization, use Gurobi (requires separate installation of the ‘gurobi‘ package).
train_models	boolean indicating whether to train models (TRUE). If FALSE previously trained models (stored locally) will be used instead. Default is TRUE.
make_predictions	boolean indicating whether to generate and save corrections (TRUE) or not. Default is TRUE.

frac_adj_with_pseudo *Update fraction based on the pseudo counts for numerators and denominators*

Description

Update fraction based on the pseudo counts for numerators and denominators

Usage

```
frac_adj_with_pseudo(data, dw, pseudo_num, pseudo_denom, num_col, denom_col)
```

Arguments

data	Data Frame
dw	character to indicate the day of a week. Can be NULL for all the days
pseudo_num	the estimated counts to be added to numerators
pseudo_denom	the estimated counts to be added to denominators
num_col	name of numerator column in the input dataframe
denom_col	name of denominator column in the input dataframe

generate_filename *Construct filename for model with given parameters*

Description

Construct filename for model with given parameters

Usage

```
generate_filename(
  indicator,
  signal,
  geo_level,
  signal_suffix,
  lambda,
  training_end_date = "",
  geo = "",
  value_type = "",
  test_lag = "",
  tau = "",
  dw = "",
  beta_prior_mode = FALSE,
  model_mode = TRUE
)
```

Arguments

indicator	string specifying the name of the indicator as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
geo_level	string describing geo coverage of input data. Either "state" or "county".
signal_suffix	string specifying value column name ending to be appended to standard value column names from ‘params$num_col‘ and ‘params$denom_col‘. Used for non-standard value column names and when processing multiple signals from a single input dataframe, as with ‘quidel‘’s age buckets.
lambda	the level of lasso penalty
training_end_date	the most recent training date
geo	string specifying the name of the geo region (e.g. FIPS code for counties)
value_type	string describing signal type. Either "count" or "fraction".
test_lag	integer number of days ago to predict for
tau	decimal quantile to be predicted. Values must be between 0 and 1.
dw	string, indicate the day of a week
beta_prior_mode	bool, indicate whether it is for a beta prior model
model_mode	bool, indicate whether the file name is for a model

Value

path to file containing model object

---

get_7dav *Calculate 7 day moving average for each issue date The 7dav for date D reported on issue date D_i is the average from D-7 to D-1*

Description

Calculate 7 day moving average for each issue date The 7dav for date D reported on issue date D_i is the average from D-7 to D-1

Usage

```
get_7dav(pivot_df, refd_col)
```

Arguments

pivot_df	Data Frame where the columns are issue dates and the rows are reference dates
refd_col	string specifying name of reference date field within the input dataframe.

get_files_list *List valid input files.*

Description

List valid input files.

Usage

```
get_files_list(indicator, signal, params, sub_dir)
```

Arguments

indicator	string specifying the name of the indicator as used in 'parquet' input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in 'parquet' input data filenames. One indicator can be associated with multiple signals.
params	named list containing modeling and data settings. Must include the following elements: 'ref_lag', 'testing_window', 'test_dates', 'training_days', 'num_col', 'denom_col', 'taus', 'lambda', 'export_dir', 'lp_solver', 'input_dir', 'cache_dir', 'geo_levels', and 'value_types'.
sub_dir	string specifying the indicator-specific directory within the general input directory 'params$input_dir'

get_model *Train model using quantile regression with Lasso penalty, or load from disk*

Description

Train model using quantile regression with Lasso penalty, or load from disk

Usage

```
get_model(
  model_path,
  train_data,
  covariates,
  tau,
  lambda,
  lp_solver,
  train_models
)
```

Arguments

model_path	path to read model from or to save model to
train_data	Data Frame containing training data
covariates	character vector of column names serving as the covariates for the model
tau	decimal quantile to be predicted. Values must be between 0 and 1.
lambda	the level of lasso penalty
lp_solver	string specifying the lp solver to use in Quantgen fitting. Either "glpk" or "gurobi". For faster optimization, use Gurobi (requires separate installation of the ‘gurobi‘ package).
train_models	boolean indicating whether to train models (TRUE). If FALSE previously trained models (stored locally) will be used instead. Default is TRUE.

---

get_populous_counties *Subset list of counties to those included in the 200 most populous in the US*

Description

Subset list of counties to those included in the 200 most populous in the US

Usage

```
get_populous_counties()
```

---

get_weekofmonth *Get week of a month info according to a date*

Description

All the dates on or before the ith Sunday but after the (i-1)th Sunday is considered to be the ith week. Notice that If there are 4 or 5 weeks in total, the ith weeks is labeled as i and the dates in the 5th week this month are actually in the same week with the dates in the 1st week next month and those dates are sparse. Thus, we assign the dates in the 5th week to the 1st week. If there are 6 weeks in total, the 1st, 2nd, 3rd, 4th, 5th, 6th weeks are labeled as c(1, 1, 2, 3, 4, 1) which means we will merge the first, second and the last weeks together.

Usage

```
get_weekofmonth(date)
```

Arguments

date	Date object

Value

a integer indicating which week it is in a month

main *Perform backfill correction on all desired signals and geo levels*

Description

Perform backfill correction on all desired signals and geo levels

Usage

```
main(params)
```

Arguments

params	named list containing modeling and data settings. Must include the following elements: ‘ref_lag‘, ‘testing_window‘, ‘test_dates‘, ‘training_days‘, ‘num_col‘, ‘denom_col‘, ‘taus‘, ‘lambda‘, ‘export_dir‘, ‘lp_solver‘, ‘input_dir‘, ‘cache_dir‘, ‘geo_levels‘, and ‘value_types‘.

main_local *Main function to correct a single local signal*

Description

Main function to correct a single local signal

Usage

```
main_local(
  input_dir,
  export_dir,
  test_start_date,
  test_end_date,
  num_col,
  denom_col,
  value_type = c("count", "fraction"),
  training_days = TRAINING_DAYS,
  testing_window = TESTING_WINDOW,
  lambda = LAMBDA,
  ref_lag = REF_LAG,
  lp_solver = LP_SOLVER
)
```

Arguments

input_dir	path to the directory containing input data
export_dir	path to directory to save output to
test_start_date	Date or string in the format "YYYY-MM-DD" to start making predictions on
test_end_date	Date or string in the format "YYYY-MM-DD" to stop making predictions on
num_col	name of numerator column in the input dataframe
denom_col	name of denominator column in the input dataframe
value_type	string describing signal type. Either "count" or "fraction".
training_days	integer number of days to use for training
testing_window	the testing window used for saving the runtime. Could set it to be 1 if time allows
lambda	the level of lasso penalty
ref_lag	max lag to use for training
lp_solver	string specifying the lp solver to use in Quantgen fitting. Either "glpk" or "gurobi". For faster optimization, use Gurobi (requires separate installation of the ‘gurobi‘ package).

---

model_training_and_testing

Fetch model and use to generate predictions/perform corrections

---

Description

Fetch model and use to generate predictions/perform corrections

Usage

```
model_training_and_testing(
  train_data,
  test_data,
  taus,
  covariates,
  lp_solver,
  lambda,
  test_lag,
  geo,
  value_type,
  model_save_dir,
  indicator,
  signal,
  geo_level,
  signal_suffix,
```

```
  training_end_date,
  train_models = TRUE,
  make_predictions = TRUE
)
```

Arguments

train_data	Data Frame containing training data
test_data	Data frame for testing
taus	numeric vector of quantiles to be predicted. Values must be between 0 and 1.
covariates	character vector of column names serving as the covariates for the model
lp_solver	string specifying the lp solver to use in Quantgen fitting. Either "glpk" or "gurobi". For faster optimization, use Gurobi (requires separate installation of the ‘gurobi‘ package).
lambda	the level of lasso penalty
test_lag	integer number of days ago to predict for
geo	string specifying the name of the geo region (e.g. FIPS code for counties)
value_type	string describing signal type. Either "count" or "fraction".
model_save_dir	directory containing trained models
indicator	string specifying the name of the indicator as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in ‘parquet‘ input data filenames. One indicator can be associated with multiple signals.
geo_level	string describing geo coverage of input data. Either "state" or "county".
signal_suffix	string specifying value column name ending to be appended to standard value column names from ‘params$num_col‘ and ‘params$denom_col‘. Used for non-standard value column names and when processing multiple signals from a single input dataframe, as with ‘quidel‘’s age buckets.
training_end_date	Most recent training date
train_models	boolean indicating whether to train models (TRUE). If FALSE previously trained models (stored locally) will be used instead. Default is TRUE.
make_predictions	boolean indicating whether to generate and save corrections (TRUE) or not. Default is TRUE.

objective *Generate objection function*

Description

Generate objection function

Usage

```
objective(theta, x, prob, ...)
```

Arguments

theta	parameters for the distribution in log scale
x	vector of quantiles
prob	the expected probabilities
...	additional arguments

read_data *Read a parquet file into a dataframe*

Description

Read a parquet file into a dataframe

Usage

```
read_data(input_dir)
```

Arguments

input_dir	path to the directory containing input data

`read_params` *Return params file as an R list*

Description

Reads a parameters file. If the file does not exist, the function will create a copy of '"params.json.template" and read from that.

Usage

```
read_params(
  path = "params.json",
  template_path = "params.json.template",
  train_models = TRUE,
  make_predictions = TRUE
)
```

Arguments

`path`	path to the parameters file; if not present, will try to copy the file "params.json.template"
`template_path`	path to the template parameters file
`train_models`	boolean indicating whether to train models (TRUE). If FALSE previously trained models (stored locally) will be used instead. Default is TRUE.
`make_predictions`	boolean indicating whether to generate and save corrections (TRUE) or not. Default is TRUE.

Details

A params list should contain the following fields. If not included, they will be filled with default values when possible.

params$ref_lag: reference lag, after x days, the update is considered to be the response. 60 is a reasonable choice for CHNG outpatient data params$input_dir: link to the input data file params$testing_window: the testing window used for saving the runtime. Could set it to be 1 if time allows params$test_dates: list of two elements, the first one is the start date and the second one is the end date params$training_days: set it to be 270 or larger if you have enough data params$num_col: the column name for the counts of the numerator, e.g. the number of COVID claims params$denom_col: the column name for the counts of the denominator, e.g. the number of total claims params$geo_level: character vector of "state" and "county", by default params$taus: vector of considered quantiles params$lambda: the level of lasso penalty params$export_dir: directory to save corrected data to params$lp_solver: LP solver to use in quantile_lasso(); "gurobi" or "glpk"

Value

a named list of parameters values

run_backfill *Get backfill-corrected estimates for a single signal + geo combination*

Description

Get backfill-corrected estimates for a single signal + geo combination

Usage

```
run_backfill(
  df,
  params,
  training_end_date,
  refd_col = "time_value",
  lag_col = "lag",
  signal_suffixes = c(""),
  indicator = "",
  signal = ""
)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
params	named list containing modeling and data settings. Must include the following elements: 'ref_lag', 'testing_window', 'test_dates', 'training_days', 'num_col', 'denom_col', 'taus', 'lambda', 'export_dir', 'lp_solver', 'input_dir', 'cache_dir', 'geo_levels', and 'value_types'.
training_end_date	the most recent training date
refd_col	string specifying name of reference date field within the input dataframe.
lag_col	string specifying name of lag field within the input dataframe.
signal_suffixes	character vector specifying value column name endings to be appended to standard value column names from 'params$num_col' and 'params$denom_col'. Used for non-standard value column names and when processing multiple signals from a single input dataframe, as with 'quidel''s age buckets.
indicator	string specifying the name of the indicator as used in 'parquet' input data filenames. One indicator can be associated with multiple signals.
signal	string specifying the name of the signal as used in 'parquet' input data filenames. One indicator can be associated with multiple signals.

run_backfill_local *Corrected estimates from a single local signal*

Description

Corrected estimates from a single local signal

Usage

```
run_backfill_local(
  df,
  export_dir,
  test_date_list,
  value_cols,
  value_type,
  taus = TAUS,
  test_lags = TEST_LAGS,
  training_days = TRAINING_DAYS,
  testing_window = TESTING_WINDOW,
  ref_lag = REF_LAG,
  lambda = LAMBDA,
  lp_solver = LP_SOLVER
)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
export_dir	path to directory to save output to
test_date_list	Date vector of dates to make predictions for
value_cols	character vector of numerator and/or denominator field names
value_type	string describing signal type. Either "count" or "fraction".
taus	numeric vector of quantiles to be predicted. Values must be between 0 and 1.
test_lags	integer vector of number of days ago to predict for
training_days	integer number of days to use for training
testing_window	the testing window used for saving the runtime. Could set it to be 1 if time allows
ref_lag	max lag to use for training
lambda	the level of lasso penalty
lp_solver	string specifying the lp solver to use in Quantgen fitting. Either "glpk" or "gurobi". For faster optimization, use Gurobi (requires separate installation of the ‘gurobi‘ package).

subset_valid_files *Return file names only if they contain data to be used in training*

Description

Parse filenames to find included dates. Use different patterns if file includes daily or rollup (multiple days) data.

Usage

```
subset_valid_files(files_list, file_type = c("daily", "rollup"), params)
```

Arguments

files_list	character vector of input files of a given ‘file_type‘
file_type	string specifying time period coverage of input files. Either "daily" or "rollup"
params	named list containing modeling and data settings. Must include the following elements: ‘ref_lag‘, ‘testing_window‘, ‘test_dates‘, ‘training_days‘, ‘num_col‘, ‘denom_col‘, ‘taus‘, ‘lambda‘, ‘export_dir‘, ‘lp_solver‘, ‘input_dir‘, ‘cache_dir‘, ‘geo_levels‘, and ‘value_types‘.

training_days_check *Check available training days*

Description

Check available training days

Usage

```
training_days_check(issue_date, training_days = TRAINING_DAYS)
```

Arguments

issue_date	contents of input data’s ‘issue_date‘ column
training_days	integer number of days to use for training

validity_checks *Check input data for validity*

Description

Check input data for validity

Usage

```
validity_checks(df, value_type, num_col, denom_col, signal_suffixes)
```

Arguments

df	Data Frame of aggregated counts within a single location reported for each reference date and issue date.
value_type	string describing signal type. Either "count" or "fraction".
num_col	name of numerator column in the input dataframe
denom_col	name of denominator column in the input dataframe
signal_suffixes	character vector specifying value column name endings to be appended to standard value column names from ‘params$num_col‘ and ‘params$denom_col‘. Used for non-standard value column names and when processing multiple signals from a single input dataframe, as with ‘quidel‘’s age buckets.

Value

list of input dataframe augmented with lag column, if it didn’t already exist, and character vector of one or two value column names, depending on requested ‘value_type‘

Index

add_7davs_and_target, 2
add_dayofweek, 3
add_params_for_dates, 3
add_shift, 4
add_sqrtscale, 4
add_weekofmonth, 5

create_dir_not_exist, 5
create_name_pattern, 6

data_filteration, 6
delta, 7

est_priors, 7
evaluate, 9
export_test_result, 9

fill_missing_updates, 10
fill_rows, 11
frac_adj, 11
frac_adj_with_pseudo, 13

generate_filename, 13
get_7dav, 14
get_files_list, 15
get_model, 15
get_populous_counties, 16
get_weekofmonth, 16

main, 17
main_local, 17
model_training_and_testing, 18

objective, 20

read_data, 20
read_params, 21
run_backfill, 22
run_backfill_local, 23

subset_valid_files, 24

training_days_check, 24

validity_checks, 25