Package 'sknifedatar'

October 14, 2022

Title Swiss Knife of Data

Version 0.1.2

Description Extension of the 'modeltime' ecosystem. In addition. Allows fitting of multiple models over multiple time series. It also provides a bridge for using the 'workflowsets' package with 'modeltime'. It includes some functionalities for spatial data and visualization.

License MIT + file LICENSE

URL https://github.com/rafzamb/sknifedatar

BugReports https://github.com/rafzamb/sknifedatar/issues

Depends R (>= 3.6.0)

Imports cli, dplyr (>= 1.0.0), knitr, magrittr, modeltime, parsnip (>= 0.1.4), purrr, rlang (>= 0.1.2), rsample (>= 0.0.9), tibble (>= 3.1.0), tidyr, tune (>= 0.1.3), utils

Suggests earth, ggplot2, recipes (>= 0.1.15), rmarkdown, spelling, timetk (>= 2.6.0), workflows (>= 0.2.2), yardstick (>= 0.0.8), workflowsets

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Rafael Zambrano [aut, cre], Karina Bartolome [aut], Rodrigo Serrano [ctb]

Maintainer Rafael Zambrano <rafazambragit@gmail.com>

Repository CRAN

Date/Publication 2021-06-01 08:00:02 UTC

R topics documented:

`automagic_tabs` *Automatic Generation of Tabs*

Description

It allows to automatically generate the code necessary to group multiple Rmarkdown chunks into tabs. Concatenating all the chunks into a string that can be later knitted and rendered.

Usage

```
automagic_tabs(
  input_data,
  panel_name,
  .output,
  ...,
  tabset_title = "",
  tabset_props = ".tabset-fade .tabset-pills",
```

```
  is_output_distill = TRUE
)
```

Arguments

input_data	Ungrouped tibble with at least 2 columns, one for the title of the tabs and another with the output to be displayed.
panel_name	string with the name of the ID column.
.output	string with the name of the column of the output.
...	additional parameters that correspond to all those available in rmarkdown chunks (fig.align, fig.width, ...).
tabset_title	string title of the .tabset
tabset_props	string defining .tabset properties. Only works with is_output_distill = F
is_output_distill	boolean. is output a distill article.

Details

given a tibble, which must contain an "ID" column (representing the title of the tabs) and another column that stores the output to be generated (plot, text, code, ...), a string is automatically generated which can be later rendered in a Rmarkdown document.

Value

concatenated string of all automatically generated chunks.

See Also

sknifedatar website

Examples

```
library(dplyr)
library(sknifedatar)
library(ggplot2)

dataset <- iris %>%
  group_by(Species) %>%
  tidyr::nest() %>%
  mutate(
   .plot = purrr::map(data, ~ ggplot(.x, aes(x = Sepal.Length, y = Petal.Length)) + geom_point())
  ) %>%
  ungroup()

automagic_tabs(input_data = dataset, panel_name = "Species", .output = ".plot", fig.align='center')

unlink("figure", recursive = TRUE)
```

automagic_tabs2 *Automatic Generation of Tabs with multiple outputs*

Description

It allows to automatically generate the code necessary to group multiple Rmarkdown chunks into tabs. Concatenating all the chunks into a string that can be later knitted and rendered.

Usage

```
automagic_tabs2(
  input_data,
  panel_name,
  ...,
  tabset_title = "",
  tabset_props = ".tabset-fade .tabset-pills",
  chunk_props = list(echo = FALSE, fig.align = "center"),
  is_output_distill = TRUE
)
```

Arguments

input_data	Ungrouped tibble with at least 2 columns, one for the title of the tabs and another with the output to be displayed.
panel_name	column with the ID variable.
...	nested columns that contain outputs to display.
tabset_title	string title of the .tabset
tabset_props	string defining .tabset properties. Only works with is_output_distill = F
chunk_props	named list with additional parameters that correspond to all those available in rmarkdown chunks (fig.align, fig.width, ...).
is_output_distill	boolean. is output a distill article.

Details

given a tiblle, which must contain an "ID" column (representing the title of the tabs) and other columns that stores output to be generated (plot, text, code, ...), a string is automatically generated which can be later rendered in a Rmarkdown document.

Value

concatenated string of all automatically generated chunks.

See Also

sknifedatar website

Examples

```
library(dplyr)
library(sknifedatar)
library(ggplot2)

dataset <- iris %>%
  group_by(Species) %>%
  tidyr::nest() %>%
  mutate(
   .plot = purrr::map(data, ~ ggplot(.x, aes(x = Sepal.Length, y = Petal.Length)) + geom_point()),
    .table = purrr::map(data, ~ summary(.x) %>% knitr::kable())
  ) %>%
  ungroup()

automagic_tabs2(input_data = dataset, panel_name = Species, .plot, .table)

unlink("figure", recursive = TRUE)
```

---

crimes *Buenos Aires crimes data*

Description

Data set that records the date, time, type of crime and geolocation of crimes that occurred between 2017 and 2019. The data was extracted from the public repository of GCBA

Usage

```
crimes
```

Format

data frame with 100 rows y 9 columns:

id: id

fecha: date

franja_horaria: hour from 0 to 23

tipo_delito: crime type

subtipo_delito: crime subtype

comuna: commune

barrio: neighborhood

lat: latitude

long: longitude ...

Source

http://data.buenosaires.gob.ar/dataset/delitos

data_avellaneda — *Vehicle flow through Avellaneda toll in Ciudad Autonoma de Buenos Aires, Argentina.*

Description

Data corresponds to Vehicle flow through Avellaneda toll in Ciudad Autonoma de Buenos Aires.. From January 2009 to December 2020.

Usage

```
data_avellaneda
```

Format

A dataframe with 4383 rows y 2 columns: date is the daily date, value is the number of vehicles no that day.

Source

https://data.buenosaires.gob.ar/dataset/flujo-vehicular-por-unidades-peaje-ausa

data_crime_clime — *Corners of the city of Buenos Aires*

Description

Data set that contains 2023 corners of the city of Buenos Aires, product of the interception of the main streets and avenues. Each row is a corner, the columns represent climatic factors, elements of the physical environment and counts of crimes that occurred in the vicinity of each corner. The original data were extracted from Openstreetmap and GCABA. They were transformed until obtaining the tabular structure that is presented here.

Usage

```
data_crime_clime
```

Format

A data frame with 2023 rows and 136 columns, the variables corner, long and lat, represent the ID of the corner and its geolocation. To see a data science project applied to this dataset see Crime prediction in CABA

Crime variables

For each corner, the number of crimes that occurred in each month of the December 2017 - December-2019 period is recorded. In total there are 25 columns of crime, which refer to the 25 months of the study period. The attributes are arranged chronologically, they can be identified with the prefix "crimes", followed by the month and year, for example: crimes_dec_2017.

Climate variables

4 climatic factors are studied: average temperature, average wind speed, millimeters of water and rainy days. Storing their values in 25 columns for each variable, referring to the 25 months of the December 2017 - December-2019 period. The attributes are ordered chronologically, they can be identified with the prefix of the climatic factor, followed by the month and year.

close environment variables

For each corner, the elements of the physical environment that are within a radius of 250 meters are counted, for example the number of metro stations, police stations, universities, gastronomic places, among others. In total there are 38 environment attributes.

Source

https://rafzamb.github.io/sknifedatar/

---

data_longer_crime *Corners of the city of Buenos Aires with meteorological and environmental factors*

Description

Data set that contains 2023 corners of the city of Buenos Aires, product of the interception of the main streets and avenues. Each row is a corner, the columns represent climatic factors, elements of physical environment and counts of crimes that occurred in the vicinity of each corner. The original data were extracted from Openstreetmap and GCABA. They were transformed until obtaining the tabular structure that is presented here.

Usage

```
data_longer_crime
```

Format

A data frame with 2023 rows and 136 columns, the variables corner, long and lat, represent the ID of the corner and its geolocation. To see a data science project applied to this dataset see Crime prediction in CABA

Crime variables

For each corner, the number of crimes that occurred in each month of the December 2017 - December-2019 period is recorded. In total there are 25 columns of crime, which refer to the 25 months of the study period. The attributes are arranged chronologically, they can be identified with the prefix "crimes", followed by the month and year.

Climate variables

4 climatic factors are studied: average temperature, average wind speed, millimeters of water and rainy days. Storing their values in 25 columns for each variable, referring to the 25 months of the December 2017 - December-2019 period. The attributes are ordered chronologically, they can be identified with the prefix of the climatic factor, followed by the month and year.

Nearby environment variables

For each corner, the elements of the physical environment that are within a radius of 250 meters are counted, for example the number of metro stations, police stations, universities, gastronomic places, among others. In total there are 38 environment attributes.

Source

https://rafzamb.github.io/sknifedatar/

---

emae_series — *Time series of the monthly estimator of Argentine economic activity, grouped by sector*

Description

The EMAE is an indicator that reflects the monthly evolution of the economic activity of all the productive sectors nationwide for Argentina, for more details: EMAE indicator. Data was obtained from all the sectoral EMAE series, from January 2004 to October 2020, from Time Series (API) from the Government Open Data Portal.

Usage

```
emae_series
```

Format

A data frame with 3104 rows and 3 columns, the variable "date" represents the monthly date, "value" the monthly value of the indicator and the sector column indicates the id or economic sector of the series.

Source

https://datos.gob.ar/dataset/jgm_3/archivo/jgm_3.13

`insert_na` *Add NA values to a dataframe*

Description

allows adding NA values to a data frame, selecting the columns and the proportion of desired NAs.

Usage

```
insert_na(.dataset, columns, .p = 0.01, seed = 123)
```

Arguments

`.dataset`	data frame.
`columns`	vector that indicates the name of the columns where the NA values will be added, in the format: c("X1", "X2") for variables X1, X2.
`.p`	value between 0 and 1, indicating the proportion of NA values that will be added.
`seed`	random number seed.

Value

the original data frame, but with the NA values added in the indicated columns.

Examples

```
insert_na(.dataset = iris, columns = c("Sepal.Length","Petal.Length"), .p = 0.25)
```

`intercepcion_calles` *Dataset of the intersection of the main streets and avenues of the city of Buenos Aires, Argentina.*

Description

Data set that records the date, time slot, type of crime and geolocation of crimes that occurred between 2017 and 2019 The data was obtained from the Openstreetmap using the osmdata package, later they were transformed until obtaining the tabular structure that is presented here.

Usage

```
intercepcion_calles
```

Format

A data frame with 2417 rows y 3 columns:

id corner id

lat latitude

long longitude ...

Source

https://rafzamb.github.io/sknifedatar/

`modeltime_multibestmodel`

Gets the best model from a modeltime table

Description

this feature allows you to select the best model for each series, based on a specific evaluation metric.

Usage

```
modeltime_multibestmodel(
  .table,
  .metric = NULL,
  .minimize = TRUE,
  .forecast = TRUE
)
```

Arguments

`.table`	'table_time**' tibble generated with the `modeltime_multifit()` function.
`.metric`	evaluation metric, from `modeltime_accuracy()` of 'modeltime' package: 'mae', 'mape','mase','smape','rmse','rsq'.
`.minimize`	boolean (default = TRUE), TRUE if the error metric should be minimized, FALSE in order to maximize it.
`.forecast`	boolean (default = TRUE), If it is TRUE, it indicates that the `modeltime_multi forecast()` function has already been applied to the object that enters the "**.table**" parameter. This is evaluated by the existence of the column "nested_forecast".

Details

take the object 'table_time' from the output of the function `modeltime_multifit()`, and selects the best model based on the selected metric.

Value

table_time tibble filtered by the best model.

Examples

```
# Data
data_serie <- sknifedatar::table_time

# best_model_emae
sknifedatar::modeltime_multibestmodel(.table = data_serie$table_time,
                                      .metric = "rmse",
                                      .minimize = TRUE,
                                      .forecast = FALSE)
```

`modeltime_multifit` *Fit Multiple Models to Multiple Time Series*

Description

allows multiple models to be fitted over multiple time series, using models from the 'modeltime' package.

Usage

```
modeltime_multifit(serie, .prop, ...)
```

Arguments

`serie`	nested time series.
`.prop`	series train/test partition ratio.
`...`	models or workflows to train (model_1, model2, ...).

Details

the focus of this function is not related to panel series, it is oriented to multiple individual series. Receiving as the first argument "series" a set of nested series (for example through the `nest()` function), then specifying a desired train/test partition ratio for series. The final input to the function are the models to be trained, simply by typing the name of the models separated by commas. The function admits as many models as required.

Value

A list of 2 items. The first component is a tibble with a first column that contains the name of the series, and a second column called "nested_column" that stores the time series, then a column for each model where the trained models or workflows for each series are stored. The last 2 columns, "nested_model" and "calibration", store the "n" trained models for each series and the adjustment metrics on the test partition. The second element is a tibble saved with the name of 'models_accuracy', it allows to visualize the performance of each model for each series according to a set of metrics.

See Also

sknifedatar website

Examples

```
library(modeltime)
library(earth)
nested_serie <-
tidyr::nest(dplyr::filter(sknifedatar::emae_series, date < '2007-02-01'),
            nested_column = -sector)

## Models
mars <- parsnip::mars(mode = 'regression') %>% parsnip::set_engine('earth')

# modeltime_multifit
sknifedatar::modeltime_multifit(serie = head(nested_serie,2),
                                .prop = 0.9,
                                mars)
```

`modeltime_multiforecast` *Forecasting of multiple models over multiple time series*

Description

allows forecasting on multiple time series from multiple fitted models.

Usage

```
modeltime_multiforecast(models_table, .h = NULL, .prop = NULL)
```

Arguments

`models_table`	'table_time' tibble generated with the `modeltime_multifit()` function.
`.h`	prediction horizon of the `modeltime_forecast()` function.
`.prop`	time series split partition ratio. If "h" is specified, this function predicts on the testing partition.

Details

this function takes the 'table_time' object generated with the `modeltime_multifit()` function, the `modeltime_forecast()` from the package 'modeltime' is applied to each model for each series.

Value

'models_table' tibble with a new column called 'nested_forecast' where the predictions are stored.

Examples

```
# Data
data_serie <- sknifedatar::table_time

# Forecast
sknifedatar::modeltime_multiforecast(data_serie$table_time, .prop=0.8)
```

`modeltime_multirefit` *Refit the model or models for multiple time series*

Description

applies the `modeltime_refit()` function from the 'modeltime' package to multiple series and models.

Usage

```
modeltime_multirefit(models_table)
```

Arguments

`models_table`	'table_time' tibble generated from the `modeltime_multifit()` function.

Details

it takes the 'table_time' tibble generated with the `modeltime_multifit()` function and returns the same object but with the models fitted for the complete period.

Value

retrained 'table_time' object.

Examples

```
# Data
library(modeltime)
data_serie <- head(sknifedatar::table_time$table_time,1)
# modeltime_multirefit
sknifedatar::modeltime_multirefit(models_table = data_serie)
```

`modeltime_wfs_bestmodel` *Modeltime best workflow from a set of models*

Description

get best workflows generated from the `modeltime_wfs_fit()` function output.

Usage

```
modeltime_wfs_bestmodel(
  .wfs_results,
  .model = NULL,
  .metric = "rmse",
  .minimize = TRUE
)
```

Arguments

`.wfs_results`	a tibble generated from the `modeltime_wfs_fit()` function.
`.model`	string or number, It can be supplied as follows: "top n," "Top n" or "tOp n", where n is the number of best models to select; n, where n is the number of best models to select; name of the workflow or workflows to select.
`.metric`	metric to get best model from ('mae', 'mape','mase','smape','rmse','rsq')
`.minimize`	a boolean indicating whether to minimize (TRUE) or maximize (FALSE) the metric.

Details

the best model is selected based on a specific metric ('mae', 'mape','mase','smape','rmse','rsq'). The default is to minimize the metric. However, if the model is being selected based on rsq minimize should be FALSE.

Value

a tibble containing the best model based on the selected metric.

Examples

```
library(dplyr)
library(earth)
data <- sknifedatar::data_avellaneda %>% mutate(date=as.Date(date)) %>% filter(date<'2012-06-01')

recipe_date <- recipes::recipe(value ~ ., data = data) %>%
  recipes::step_date(date, features = c('dow','doy','week','month','year'))

mars <- parsnip::mars(mode = 'regression') %>%
```

```
  parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = recipe_date),
  models  = list(M_mars = mars),
  cross   = TRUE)

wffits <- sknifedatar::modeltime_wfs_fit(.wfsets = wfsets,
                                         .split_prop = 0.8,
                                         .serie=data)

sknifedatar::modeltime_wfs_bestmodel(.wfs_results = wffits,
                                     .metric='rsq',
                                     .minimize = FALSE)
```

`modeltime_wfs_fit` *Modeltime workflowsets fit*

Description

allows working with workflow sets and modeltime. Combination of recipes and models are trained and evaluation metrics are returned.

Usage

```
modeltime_wfs_fit(.wfsets, .split_prop, .serie)
```

Arguments

`.wfsets`	workflow_set object, generated with the `workflow_set()` function from the 'workflowsets' package.
`.split_prop`	time series split proportion.
`.serie`	time series dataframe.

Details

Given a workflow_set containing multiple time series recipes and models, adjusts all the possible combinations on a time series. It uses a split proportion in order to train on a time series partition and evaluate metrics on the testing partition.

Value

tbl_df containing the model id (based on workflow_set), model description and metrics on the time series testing dataframe. Also, a .fit_model column is included, which contains each fitted model.

See Also

sknifedatar website

Examples

```
library(dplyr)
library(earth)

data <- sknifedatar::data_avellaneda %>% mutate(date=as.Date(date)) %>% filter(date<'2012-06-01')

recipe_date <- recipes::recipe(value ~ ., data = data) %>%
  recipes::step_date(date, features = c('dow','doy','week','month','year'))

mars <- parsnip::mars(mode = 'regression') %>% parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = recipe_date),
  models  = list(M_mars = mars),
  cross   = TRUE)

sknifedatar::modeltime_wfs_fit(.wfsets = wfsets,
                               .split_prop = 0.8,
                               .serie = data)
```

---

modeltime_wfs_forecast

Modeltime workflow sets forecast

Description

forecast from a set of recipes and models trained by `modeltime_wfs_fit()` function.

Usage

```
modeltime_wfs_forecast(.wfs_results, .series, .split_prop = NULL, .h = NULL)
```

Arguments

`.wfs_results`	tibble of combination of recipes and models fitted, generated with the `modeltime_wfs_fit()` function.
`.series`	time series dataframe.
`.split_prop`	time series split proportion.
`.h`	time series horizon from the `modeltime_forecast()` function from 'modeltime' package.

Details

since it uses the `modeltime_forecast()` function from 'modeltime' package, either the forecast can be made on new data or on a number of periods.

Value

a tibble containing the forecast for each model.

Examples

```
library(dplyr)
library(modeltime)
library(earth)

data <- sknifedatar::data_avellaneda %>%  mutate(date=as.Date(date)) %>%
filter(date<'2012-06-01')

recipe_date <- recipes::recipe(value ~ ., data = data) %>%
  recipes::step_date(date, features = c('dow','doy','week','month','year'))

mars <- parsnip::mars(mode = 'regression') %>%
  parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = recipe_date),
  models  = list(M_mars = mars),
  cross   = TRUE)

wffits <- sknifedatar::modeltime_wfs_fit(.wfsets = wfsets,
                                         .split_prop = 0.8,
                                         .serie=data)

sknifedatar::modeltime_wfs_forecast(.wfs_results=wffits,
                                    .series = data,
                                    .split_prop = 0.8)
```

`modeltime_wfs_heatmap` *Modeltime workflowsets heatmap plot*

Description

generate a heatmap for each recipe and model on a object generated with the `modeltime_wfs_fit()` function.

Usage

```
modeltime_wfs_heatmap(
  .wfs_results,
  metric = "rsq",
  low_color = "#c7e9b4",
  high_color = "#253494"
)
```

Arguments

.wfs_results	a tibble generated with the modeltime_wfs_fit() function.
metric	a metric the metric used for the heatmap values: 'mae', 'mape','mase','smape','rmse','rsq'.
low_color	color for the worst metric (highest error or lowest rsq).
high_color	color for the better metric (lowest error or highest rsq).

Details

assumes that the workflows included in the 'workflow_set' object are named M_name_of_model, since the .model_id is recipe_name*M*name_of_model and the '*M*' is used to separate the recipe from the model name.

Value

a ggplot heatmap.

Examples

```
library(modeltime)
library(dplyr)
library(parsnip)
library(earth)

data <- sknifedatar::data_avellaneda %>% mutate(date=as.Date(date)) %>% filter(date<'2011-01-01')

recipe_date <- recipes::recipe(value ~ ., data = data) %>%
  recipes::step_date(date, features = c('dow','doy','week','month','year'))

mars_backward <- mars(prune_method ='backward', mode = 'regression') %>% set_engine('earth')

mars_forward <- mars(prune_method = 'forward', mode = 'regression') %>% set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    date = recipe_date),
  models  = list(M_mars_backward = mars_backward,
                 M_mars_forward = mars_forward),
  cross   = TRUE)

wffits <- sknifedatar::modeltime_wfs_fit(.wfsets = wfsets,
                                         .split_prop = 0.6,
```

```
                                        .serie=data)

sknifedatar::modeltime_wfs_heatmap(wffits, 'rsq')
```

`modeltime_wfs_multibestmodel` *Get the best workflow for each time series*

Description

obtains the best workflow for each time series based on a performance metric.

Usage

```
modeltime_wfs_multibestmodel(.table, .metric = NULL, .minimize = TRUE)
```

Arguments

`.table`	a tibble that comes from the output of the `modeltime_wfs_multifit()` or `modeltime_wfs_multiforeca` functions. For the `modeltime_wfs_multifit()` function, the 'table_time' object must be selected from the output.
`.metric`	a string of evaluation metric, the following symmetrical can be supplied: 'mae', 'mape','mase', 'smape','rmse','rsq'.
`.minimize`	boolean (default = TRUE), TRUE if the error metric should be minimized, FALSE in order to maximize it.

Value

a tibble, corresponds to the same tibble supplied in the '.table' parameter but with the selection of the best workflow for each series.

Examples

```
library(dplyr)
library(earth)

df <- sknifedatar::emae_series

datex <- '2020-02-01'
df_emae <- df %>%
  dplyr::filter(date <= datex) %>%
  tidyr::nest(nested_column=-sector) %>%
  head(2)

receta_base <- recipes::recipe(value ~ ., data = df %>% select(-sector))

mars <- parsnip::mars(mode = 'regression') %>% parsnip::set_engine('earth')
```

```
wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = receta_base),
  models  = list(M_mars = mars),
  cross   = TRUE)

wfsets_fit <- modeltime_wfs_multifit(.wfs = wfsets,
                                     .prop = 0.8,
                                     serie = df_emae)

sknifedatar::modeltime_wfs_multibestmodel(.table = wfsets_fit$table_time,
                                          .metric = "rmse",
                                          .minimize = TRUE)
```

`modeltime_wfs_multifit` *Fit a workflow_set object over multiple time series*

Description

allows a workflow_set object to be fitted over multiple time series, using models from the 'modeltime' package.

Usage

```
modeltime_wfs_multifit(serie, .prop, .wfs)
```

Arguments

`serie`	nested time series.
`.prop`	series train/test partition ratio.
`.wfs`	worklows_set object.

Value

A list of 2 items. The first component is a tibble with a first column that contains the name of the series, and a second column called 'nested_column' that stores the time series, then a column for each workflow for each series are stored. The last 2 columns, 'nested_model' and 'calibration', store the 'n' trained workflows for each series and the adjustment metrics on the test partition. The second element is a tibble saved with the name of 'models_accuracy', it allows to visualize the performance of each workflow for each series according to a set of metrics.

Examples

```
library(dplyr)
library(earth)

df <- sknifedatar::emae_series

datex <- '2020-02-01'
df_emae <- df %>%
dplyr::filter(date <= datex) %>%
tidyr::nest(nested_column=-sector) %>%
head(2)

receta_base <- recipes::recipe(value ~ ., data = df %>% select(-sector))

mars <- parsnip::mars(mode = 'regression') %>% parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = receta_base),
  models  = list(M_mars = mars),
  cross   = TRUE)

sknifedatar::modeltime_wfs_multifit(.wfs = wfsets,
                                    .prop = 0.8,
                                    serie = df_emae)
```

`modeltime_wfs_multiforecast` *Forecast of a workflow set on multiple time series*

Description

generates forecasts of a workflow set object over multiple time series.

Usage

```
modeltime_wfs_multiforecast(models_table, .h = NULL, .prop = NULL)
```

Arguments

`models_table`	a tibble that comes from the output of the `modeltime_wfs_multifit()`, `modeltime_wfs_multirefit()`, `modeltime_wfs_multibestmodel()` functions. For the `modeltime_wfs_multifit()` function, the 'table_time' object must be selected from the output.
`.h`	prediction horizon of the `modeltime_forecast()` function of the 'modeltime' package.
`.prop`	decimal number, time series split partition ratio. If ".h" is specified, this function predicts on the testing partition.

Value

a tibble, corresponds to the same tibble supplied in the 'models_table' parameter but with an additional column called 'nested_forecast' where the nested previews of the workflows on all the time series are stored.

Examples

```
library(dplyr)
library(earth)

df <- sknifedatar::emae_series

datex <- '2020-02-01'
df_emae <- df %>%
  dplyr::filter(date <= datex) %>%
  tidyr::nest(nested_column=-sector) %>% head(2)

receta_base <- recipes::recipe(value ~ ., data = df %>% select(-sector))

mars <- parsnip::mars(mode = 'regression') %>% parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = receta_base),
  models  = list(M_mars = mars),
  cross   = TRUE)

wfsets_fit <- sknifedatar::modeltime_wfs_multifit(.wfs = wfsets,
                                                  .prop = 0.8,
                                                  serie = df_emae)

sknifedatar::modeltime_wfs_multiforecast(wfsets_fit$table_time,
                                         .prop=0.8)
```

`modeltime_wfs_multirefit`

Refit one or more trained workflows to new data

Description

It allows retraining a set of workflows trained on new data.

Usage

```
modeltime_wfs_multirefit(models_table)
```

Arguments

models_table	a tibble that comes from the output of the modeltime_wfs_multifit(), modeltime_wfs_multiforecas modeltime_wfs_multibestmodel() functions. For the modeltime_wfs_multifit function, the 'table_time' object must be selected from the output.

Value

a tibble, corresponds to the same tibble supplied in the 'models_table' parameter but with the refit of the workflows saved in the 'nested_model' column.

Examples

```
library(dplyr)
library(earth)

df <- sknifedatar::emae_series

datex <- '2020-02-01'
df_emae <- df %>%
  dplyr::filter(date <= datex) %>%
  tidyr::nest(nested_column=-sector) %>%
  head(2)

receta_base <- recipes::recipe(value ~ ., data = df %>% select(-sector))

mars <- parsnip::mars(mode = 'regression') %>% parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = receta_base),
  models  = list(M_mars = mars),
  cross   = TRUE)

wfsets_fit <- modeltime_wfs_multifit(.wfs = wfsets,
                                     .prop = 0.8,
                                     serie = df_emae)

sknifedatar::modeltime_wfs_multirefit(wfsets_fit$table_time)
```

---

modeltime_wfs_rank *Modeltime workflow sets ranking based on a metric*

Description

generates a ranking of models generated with modeltime_wfs_fit() function.

Usage

```
modeltime_wfs_rank(.wfs_results, rank_metric = NULL, minimize = TRUE)
```

Arguments

.wfs_results	a tibble generated with the modeltime_wfs_fit() function.
rank_metric	the metric used to generate the ranking 'mae', 'mape','mase','smape','rmse','rsq'.
minimize	a boolean indicating whether to minimize (TRUE) or maximize (FALSE) the metric

Details

the ranking depends on the metric selected.

Value

a tibble containing the models ranked by a specific metric.

See Also

sknifedatar website

Examples

```
library(dplyr)
library(modeltime)
library(earth)

data <- sknifedatar::data_avellaneda %>%
  mutate(date=as.Date(date)) %>%
  filter(date<'2012-06-01')

recipe_date <- recipes::recipe(value ~ ., data = data) %>%
  recipes::step_date(date, features = c('dow','doy','week','month','year'))

mars <- parsnip::mars(mode = 'regression') %>% parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = recipe_date),
  models  = list(M_mars = mars),
  cross   = TRUE)

wffits <- sknifedatar::modeltime_wfs_fit(.wfsets = wfsets,
                                         .split_prop = 0.8,
                                         .serie = data)

sknifedatar::modeltime_wfs_rank(.wfs_results = wffits,
                                rank_metric = 'rsq',
                                minimize = FALSE)
```

`modeltime_wfs_refit` *Modeltime workflow sets refit*

Description

applies the `modeltime_refit()` function from 'modeltime' package to the object generated from the `modeltime_wfs_fit()` function (or the filtered version after the `modeltime_wfs_bestmodel()` is applied).

Usage

```
modeltime_wfs_refit(.wfs_results, .serie)
```

Arguments

`.wfs_results`	tibble of combination of recipes and models fitted, generated with the `modeltime_wfs_fit()` function.
`.serie`	a time series dataframe.

Details

each workflow is now re-trained using all the available data.

Value

a tibble containing the re-trained models.

Examples

```
library(modeltime)
library(dplyr)
library(earth)

data <- sknifedatar::data_avellaneda %>%
  mutate(date=as.Date(date)) %>%
  filter(date<'2012-06-01')

recipe_date <- recipes::recipe(value ~ ., data = data) %>%
  recipes::step_date(date, features = c('dow','doy','week','month','year'))

mars <- parsnip::mars(mode = 'regression') %>%
  parsnip::set_engine('earth')

wfsets <- workflowsets::workflow_set(
  preproc = list(
    R_date = recipe_date),
  models  = list(M_mars = mars),
  cross   = TRUE)
```

```
wffits <- sknifedatar::modeltime_wfs_fit(.wfsets = wfsets,
                                         .split_prop = 0.8,
                                         .serie = data)

sknifedatar::modeltime_wfs_refit(.wfs_results = wffits,
                                 .serie = data)
```

`multieval` *Evaluation of multiple metrics and predictions*

Description

for a set of predictions from different models, evaluate multiple metrics and return the results in a tabular format that makes it easy to compare the predictions.

Usage

```
multieval(.dataset, .observed, .predictions, .metrics, value_table = FALSE)
```

Arguments

`.dataset`	data frame with the predictions, it must have at least the column with the observed data and at least one column that refers to the predictions of a model.
`.observed`	string with the name of the column that contains the observed data.
`.predictions`	string or vector of strings the columns where the predictions are stored.
`.metrics`	metric or set of metrics to be evaluated, the metrics refer to those allowed by the package 'yardstick' from 'tidymodels'.
`value_table`	TRUE to display disaggregated metrics.

Value

data frame with 4 columns: the evaluation metrics, the estimator used, the value of the metric and the name of the model.

See Also

Crime prediction /multieval

Examples

```
set.seed(123)
library(yardstick) # métricas

predictions <-
  data.frame(truth = runif(100),
             predict_model_1 = rnorm(100, mean = 1,sd =2),
             predict_model_2 = rnorm(100, mean = 0,sd =2),
```

```
              predict_model_3 = rnorm(100, mean = 0,sd =3))

multieval(.dataset = predictions,
          .observed = "truth",
          .predictions = c("predict_model_1","predict_model_2","predict_model_3"),
          .metrics = list(rmse = rmse, rsq = rsq, mae = mae),
          value_table = TRUE)

# Output ----------------------
# A tibble: 9 x 4
# .metric .estimator .estimate model
# <chr>   <chr>          <dbl> <chr>
#   1 mae     standard     1.45    predict_model_1
# 2 mae     standard     1.67    predict_model_2
# 3 mae     standard     2.43    predict_model_3
# 4 rmse    standard     1.78    predict_model_1
# 5 rmse    standard     2.11    predict_model_2
# 6 rmse    standard     3.01    predict_model_3
# 7 rsq     standard     0.00203 predict_model_1
# 8 rsq     standard     0.0158  predict_model_2
# 9 rsq     standard     0.00254 predict_model_3

#$summary_table
# A tibble: 3 x 4
#  model             mae  rmse     rsq
#  <chr>           <dbl> <dbl>   <dbl>
#  1 predict_model_1  1.45  1.78 0.00203
#  2 predict_model_2  1.67  2.11 0.0158
#  3 predict_model_3  2.43  3.01 0.00254
```

`sliding_window` *Mobile sliding window transformation*

Description

allows to apply a monthly moving sliding window transformation on a data set.

Usage

```
sliding_window(data, inicio, pliegues, variables)
```

Arguments

`data`	dataframe that contains historical counts of different events in monthly time frames. Each row is a unique observation and the columns corresponding to the different months of study. The variables must have keywords to be able to select them together. To see an example of the structure of the data, the dataset such contained in this package can be used.
`inicio`	initial month, integer numeric format.

pliegues	vector that starts at 1 and ends in the number of periods to be traversed.
variables	a word or vector that allows you to select the variables together and implement the function for each group.

Details

the operation is as follows, the intermediate month "t" of the entire study period is selected, then the number of events that occurred for each observation in the previous month is calculated, in the last 3 months, 6 months, 12 months and the same month of the previous year.

The procedure described above is replicated in a mobile manner, that is, rolling the time window from t + 1 to n, where n is the last month of study. To see a real use case, visit Crime analysis with tidymodels

Value

a data frame with the ID of the observations and the different counting time slots calculated by variables.

See Also

sknifedatar website

Examples

```
pliegues = 1:13
names(pliegues) = pliegues

variables = c("delitos", "temperatura", "mm_agua", "lluvia", "viento")
names(variables) = variables

data("data_longer_crime")

sliding_window(data = data_longer_crime %>% dplyr::select(-c(long,lat)),
               inicio = 13,
               pliegues = pliegues,
               variables = variables)
```

---

table_time *Fitted models for the EMAE indicator*

Description

set of models fitted on the "emae_series" dataset, this object comes from the output of the `modeltime_multifit()` function. For example, if object = modeltime_multifit then 'object$table_time' is the fitted models table.

Usage

```
table_time
```

Format

a tibble that contains a first column with the name of the series, then the "nested_column" column that stores the time series, then a column for each supplied model where the models or trained workflows for each series are stored. Finally the columns "nested_model" and "calibration" that store the "n" trained models for each series and the adjustment metrics on the test partition.

Source

https://rafzamb.github.io/sknifedatar/

Index