Package ‘workflowsets’

April 16, 2021

Title Create a Collection of 'tidymodels' Workflows

Version 0.0.2

Description A workflow is a combination of a model and preprocessors (e.g. a formula, recipe, etc.) (Kuhn and Silge (2021) <https://www.tmwr.org/>). In order to try different combinations of these, an object can be created that contains many workflows. There are functions to create workflows en masse as well as training them and visualizing the results.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1.9001

Imports rlang, tune (>= 0.1.3), withr, dplyr (>= 1.0.0), purrr, tibble (>= 3.1.0), vctrs, tidyr, workflows (>= 0.2.2), cli, ggplot2, stats, prettyunits, rsample (>= 0.0.9), generics

Language en-US

Suggests spelling, recipes (>= 0.1.15), modeldata, parsnip, testthat (>= 3.0.0), covr, yardstick, dials, kknn, knitr, rmarkdown

Depends R (>= 2.10)

    https://workflowsets.tidymodels.org

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Max Kuhn [aut, cre] (<https://orcid.org/0000-0003-2402-136X>), RStudio [cph]

Maintainer Max Kuhn <max@rstudio.com>

Repository CRAN

Date/Publication 2021-04-16 16:50:02 UTC
**R topics documented:**

- `as_workflow_set` ............................................. 2
- `autoplot.workflow_set` ........................................ 3
- `collect_metrics.workflow_set` ................................ 4
- `comment_add` .................................................. 5
- `leave_var_out_formulas` ....................................... 6
- `option_add` .................................................... 8
- `option_list` .................................................... 9
- `pull_workflow_set_result` ..................................... 9
- `rank_results` .................................................. 10
- `two_class_set` .................................................. 11
- `workflow_map` .................................................. 11
- `workflow_set` .................................................. 13

**Index**

<table>
<thead>
<tr>
<th>as_workflow_set</th>
<th>Save results from tuning or resampling functions as a workflow set</th>
</tr>
</thead>
</table>

**Description**

If results have been generated directly from functions like `tune::tune_grid()`, they can be combined into a workflow set using this function.

**Usage**

```r
as_workflow_set(...)```

**Arguments**

- `...` One or more named objects. Names should be unique and the objects should have at least one of the following classes: `iteration_results`, `tune_results`, `resample_results`, or `tune_race`. Each element should also contain the original workflow (accomplished using the `save_workflow` option in the control function).

**Value**

A workflow set. Note that the option column will not reflect the options that were used to create each object.
Description

This autoplot() method can performance metrics that have been ranked using a metric. It can also run autoplot() on the individual results (per wflow_id).

Usage

```r
## S3 method for class 'workflow_set'
autoplot(
  object,
  rank_metric = NULL,
  metric = NULL,
  id = "workflow_set",
  select_best = FALSE,
  std_errs = qnorm(0.95),
  ...
)
```

Arguments

- `object`: A workflow_set whose elements have results.
- `rank_metric`: A character string for which metric should be used to rank the results.
- `metric`: A character vector for which metrics (apart from `rank_metric`) to be included in the visualization.
- `id`: A character string for what to plot. If a value of "workflow_set" is used, the results of each model (and sub-model) are ordered and plotted. Alternatively, a value of the workflow set's wflow_id can be given and the autoplot() method is executed on that workflow's results.
- `select_best`: A logical; should the results only contain the numerically best submodel per workflow?
- `std_errs`: The number of standard errors to plot (if the standard error exists).
- `...`: Other options to pass to autoplot().

Details

The x-axis is the workflow rank in the set (a value of one being the best) versus the performance metric(s) on the y-axis. With multiple metrics, there will be facets for each metric.

If multiple resamples are used, confidence bounds are shown for each result.

Value

A ggplot object.
collect_metrics.workflow_set

Obtain and format results produced by tuning functions for workflow sets

Examples

```r
autoplot(two_class_res)
autoplot(two_class_res, select_best = TRUE)
autoplot(two_class_res, id = "yj_trans_cart", metric = "roc_auc")
```

Description

Return a tibble of performance metrics for all models or submodels.

Usage

```r
## S3 method for class 'workflow_set'
collect_metrics(x, summarize = TRUE, ...)

## S3 method for class 'workflow_set'
collect_predictions(
  x,
  summarize = TRUE,
  parameters = NULL,
  select_best = FALSE,
  metric = NULL,
  ...
)
```

Arguments

- **x** A workflow_set object where all workflows have been evaluated.
- **summarize** A logical for whether the performance estimates should be summarized via the mean (over resamples) or the raw performance values (per resample) should be returned along with the resampling identifiers. When collecting predictions, these are averaged if multiple assessment sets contain the same row.
- **...** Not currently used.
- **parameters** An optional tibble of tuning parameter values that can be used to filter the predicted values before processing. This tibble should only have columns for each tuning parameter identifier (e.g. "my_param" if tune("my_param") was used).
- **select_best** A single logical for whether the numerically best results are retained. If TRUE, the parameters argument is ignored.
- **metric** A character string for the metric that is used for select_best.
Details

When applied to a workflow set, the metrics and predictions that are returned do not contain the actual tuning parameter columns and values (unlike when these collect functions are run on other objects). The reason is that workflow sets can contain different types of models or models with different tuning parameters.

If the columns are needed, there are two options. First, the .config column can be used to merge the tuning parameter columns into an appropriate object. Alternatively, the map() function can be used to get the metrics from the original objects (see the example below).

Value

A tibble.

See Also

tune::collect_metrics(), rank_results()

Examples

library(dplyr)
library(purrr)
library(tidyr)

# ------------------------------------------------------------------------------

collect_metrics(two_class_res)

# Alternatively, if the tuning parameter values are needed:
two_class_res %>%
dplyr::filter(grepl("cart", wflow_id)) %>%
mutate(metrics = map(result, collect_metrics)) %>%
dplyr::select(wflow_id, metrics) %>%
tidyr::unnest(cols = metrics)

collect_metrics(two_class_res, summarize = FALSE)
Usage

comment_add(x, id, ..., append = TRUE, collapse = "\n")

comment_get(x, id)

comment_reset(x, id)

comment_print(x, id = NULL, ...)

Arguments

x
A workflow set

id
A single character string for a value in the wflow_id column. For comment_print(), id can be a vector or NULL (and this indicates that all comments are printed).

...
One or more character strings.

append
A logical value to determine if the new comment should be added to the existing values.

collapse
A character string that separates the comments.

Value

comment_add() and comment_reset() return an updated workflow set. comment_get() returns a character string. comment_print() returns NULL invisibly.

Examples

two_class_set %>% comment_get("none_cart")

new_set <-
two_class_set %>%
  comment_add("none_cart", "What does ‘cart’ stand for?")
  comment_add("none_cart", "Classification And Regression Trees.")

comment_print(new_set)

new_set %>% comment_get("none_cart")

new_set %>% comment_reset("none_cart") %>% comment_get("none_cart")

Description

Create formulas without each predictor.
Usage

leave_var_out_formulas(formula, data, full_model = TRUE, ...)

Arguments

formula A model formula that contains at least two predictors.
data A data frame.
full_model A logical; should the list include the original formula?
... Options to pass to stats::model.frame()

Details

The new formulas obey the hierarchy rule so that interactions without main effects are not included (unless the original formula contains such terms).

Factor predictors are left as-is (i.e., no indicator variables are created).

Value

A named list of formulas

See Also

workflow_set()

Examples

data(penguins, package = "modeldata")

leave_var_out_formulas(
  bill_length_mm ~ .,
  data = penguins
)

leave_var_out_formulas(
  bill_length_mm ~ (island + sex)^2 + flipper_length_mm,
  data = penguins
)

leave_var_out_formulas(
  bill_length_mm ~ (island + sex)^2 + flipper_length_mm +
  I(flipper_length_mm^2),
  data = penguins
)
Description

These functions are helpful for manipulating the information in the option column.

Usage

option_add(x, ..., id = NULL, strict = FALSE)

option_remove(x, ...)

option_add_parameters(x, id = NULL, strict = FALSE)

Arguments

x
A workflow set

... A list of named options. For option_remove() this can be a series of unquoted option names.

id
A character string of one or more values from the wflow_id column that indicates which options to update. By default, all workflows are updated.

strict
A logical; show execution stop if existing options are being replaced?

Details

option_add() is used to update all of the options in a workflow set.

option_remove() will eliminate specific options across rows.

option_add_parameters() adds a parameter object to the option column (if parameters are being tuned).

Note that executing a function on the workflow set, such as tune_grid(), will add any options given to that function to the option column.

Value

An updated workflow set.

Examples

two_class_set %>%
  option_add(a = 1)

two_class_set %>%
  option_add(a = 1) %>%
  option_add(b = 2, id = "none_cart")
library(tune)
two_class_set %>%
  option_add_parameters()

option_list

Make a classed list of options

Description
This function returns a named list with an extra class of "workflow_set_options" that has corresponding formatting methods for printing inside of tibbles.

Usage
option_list(...)  

Arguments
...
A set of named options (or nothing)

Value
A classed list.

Examples

option_list(a = 1, b = 2)
option_list()

pull_workflow_set_result
Extract elements from a workflow set

Description
pull_workflow_set_result() retrieves the results of workflow_map() for a particular workflow while pull_workflow() extracts the unfitted workflow from the info column.

Usage

pull_workflow_set_result(x, id)
pull_workflow(x, id)
Arguments

x A workflow set.

id A single character string for a workflow ID.

Value

pull_workflow_set_result() produces a tune_result or resample_results object. pull_workflow() returns an unfit workflow object.

Examples

library(tune)
pull_workflow_set_result(two_class_res, "none_cart")
pull_workflow(two_class_res, "none_cart")

rank_results

Rank the results by a metric

Description

This function sorts the results by a specific performance metric.

Usage

rank_results(x, rank_metric = NULL, select_best = FALSE)

Arguments

x A workflow set that has all results.

rank_metric A character string for a metric.

select_best A logical; should the results only contain the numerically best submodel per workflow.

Details

If some models have the exact same performance, rank(value, ties.method = "random") is used (with a reproducible seed) so that all ranks are integers.

No columns are returned for the tuning parameters since they are likely to be different (or not exist) for some models. The wflow_id and .config columns can be used to determine the corresponding parameter values.

Value

A tibble with columns: wflow_id, .config, .metric, mean, std_err, n, preprocessor, model, and rank.
**Examples**

```r
rank_results(chi_features_res)
rank_results(chi_features_res, select_best = TRUE)
rank_results(chi_features_res, rank_metric = "rsq")
```

---

**two_class_set**

---

**Example Data Sets**

**Description**

Example Data Sets

**Details**

Example workflow sets and associated model fits.

two_class_set and two_class_res were generated using the data in the package file `example-data/two-class-set.R`

chi_features_set and chi_features_res were generated using the data in the package file `example-data/chi-features-res.R`. It is meant to approximate the sequence of models built in Section 1.3 of Kuhn and Johnson (2019).

**Value**

Workflow sets.

**References**

Max Kuhn and Kjell Johnson (2019) *Feature Engineering and Selection*, [https://bookdown.org/max/FES/a-more-complex-example.html](https://bookdown.org/max/FES/a-more-complex-example.html)

**Examples**

```r
data(two_class_set)
two_class_set
```

---

**workflow_map**

**Process a series of workflows**

**Description**

`workflow_map()` will execute the same function across the workflows in the set. The various `tune_*()` functions can be used as well as `fit_resamples()`.
Usage

workflow_map(
  object,
  fn = "tune_grid",
  verbose = FALSE,
  seed = sample.int(10^4, 1),
  ...
)

Arguments

object A workflow set.
fn The function to run. Acceptable values are: `tune::tune_grid()`, `tune::tune_bayes()`, `tune::fit_resamples()`, `finetune::tune_race_anova()`, `finetune::tune_race_win_loss()`, or `finetune::tune_sim_anneal()`.
verbose A logical for logging progress.
seed A single integer that is set prior to each function execution.
... Options to pass to the modeling function. See details below.

Details

When passing options, anything passed in the ... will be combined with any values in the option column. The values in ... will override that column’s values and the new options are added to the options column.

Any failures in execution result in the corresponding row of results to contain a try-error object.

In cases where a model has no tuning parameters is mapped to one of the tuning functions, `tune::fit_resamples()` will be used instead and a warning is issued if verbose = TRUE.

If a workflow required packages that are not installed, a message is printed and workflow_map() continues with the next workflow (if any).

Value

An updated workflow set. The option column will be updated with any options for the tune package functions given to workflow_map(). Also, the results will be added to the result column. If the computations for a workflow fail, an try-catch object will be saved in place of the results (without stopping execution).

See Also

`workflow_set()`, `as_workflow_set()`, `pull_workflow_set_result()`

Examples

# An example of processed results
chi_features_res

# Code examples at
if (interactive()) {
  system.file("example-data", package = "workflowsets")
}

---

**workflow_set**  
*Generate a set of workflow objects from preprocessing and model objects*

**Description**  
Generate a set of workflow objects from preprocessing and model objects

**Usage**  

```r
workflow_set(preproc, models, cross = TRUE)
```

**Arguments**  

- `preproc`  
  A list (preferably named) with preprocessing objects: formulas, recipes, or `workflows::workflow_variables()`.

- `models`  
  A list (preferably named) of `parsnip` model specifications.

- `cross`  
  A logical: should all combinations of the preprocessors and models be used to create the workflows? If `FALSE`, the length of `preproc` and `models` should be equal.

**Details**  
The preprocessors that can be combined with the model objects can be one or more of:

- A traditional R formula.
- A recipe definition (un-prepared) via `recipes::recipe()`.
- A selectors object created by `workflows::workflow_variables()`.

Since `preproc` is a named list column, any combination of these can be used in that argument (i.e., `preproc` can be mixed types).

**Value**  
A tibble with extra class 'workflow_set'. A new set includes four columns (but others can be added):

- `wflow_id` contains character strings for the preprocessor/workflow combination. These can be changed but must be unique.
- `info` is a list column with tibbles containing more specific information, including any comments added using `comment_add()`. This tibble also contains the workflow object (which can be easily retrieved using `pull_workflow()`).
- `option` is a list column that will include a list of optional arguments passed to the functions from the `tune` package. They can be added manually via `option_add()` or automatically when options are passed to `workflow_map()`.
- `result` is a list column that will contain any objects produced when `workflow_map()` is used.
See Also

workflow_map(), comment_add(), option_add(), as_workflow_set()

Examples

library(workflowsets)
library(workflows)
library(modeldata)
library(recipes)
library(parsnip)
library(dplyr)
library(rsample)
library(tune)
library(yardstick)

# ---------------------------------------------------------------------------
data(cells)
cells <- cells %>% dplyr::select(-case)
set.seed(1)
val_set <- validation_split(cells)

# ---------------------------------------------------------------------------
basic_recipe <-
  recipe(class ~ ., data = cells) %>%
  step_YeoJohnson(all_predictors()) %>%
  step_normalize(all_predictors())
pca_recipe <-
  basic_recipe %>%
  step_pca(all_predictors(), num_comp = tune())
ss_recipe <-
  basic_recipe %>%
  step_spatialsign(all_predictors())

# ---------------------------------------------------------------------------
knn_mod <-
  nearest_neighbor(neighbors = tune(), weight_func = tune()) %>%
  set_engine("kknn") %>%
  set_mode("classification")
lr_mod <-
  logistic_reg() %>%
  set_engine("glm")

# ---------------------------------------------------------------------------
preproc <- list(none = basic_recipe, pca = pca_recipe, sp_sign = ss_recipe)
models <- list(knn = knn_mod, logistic = lr_mod)

cell_set <- workflow_set(preproc, models, cross = TRUE)
cell_set

# Using variables and formulas

# Select predictors by their names
channels <- paste0("ch_", 1:4)
preproc <- purrr::map(channels, ~ workflow_variables(class, c(contains(!!.x))))
names(preproc) <- channels
preproc$everything <- class ~ .
preproc

cell_set_by_group <- workflow_set(preproc, models["logistic"])
cell_set_by_group
Index

* datasets
  two_class_set, 11
  as_workflow_set, 2
  as_workflow_set(), 12, 14
  autoplot.workflow_set, 3
  chi_features_res(two_class_set), 11
  chi_features_set(two_class_set), 11
  collect_metrics.workflow_set, 4
  collect_predictions.workflow_set
    (collect_metrics.workflow_set), 4
  comment_add, 5
  comment_add(), 13, 14
  comment_get(comment_add), 5
  comment_print(comment_add), 5
  comment_reset(comment_add), 5
  leave_var_out_formulas, 6
  option_add, 8
  option_add(), 13, 14
  option_add_parameters(option_add), 8
  option_list, 9
  option_remove(option_add), 8
  pull_workflow
    (pull_workflow_set_result), 9
  pull_workflow(), 13
  pull_workflow_set_result, 9
  pull_workflow_set_result(), 12
  rank_results, 10
  rank_results(), 5
  recipes::recipe(), 13
  stats::model.frame(), 7
  tune::collect_metrics(), 5
  tune::fit_resamples(), 12
  tune::tune_bayes(), 12
  tune::tune_grid(), 2, 12
  two_class_res(two_class_set), 11
  two_class_set, 11
  workflow_map, 11
  workflow_map(), 9, 13, 14
  workflow_set, 13
  workflow_set(), 7, 12
  workflows::workflow_variables(), 13