Package 'workflows'

December 30, 2019

Title Modeling Workflows

Version 0.1.0

Description Managing both a 'parsnip' model and a preprocessor, such as a model formula or recipe from 'recipes', can often be challenging. The goal of 'workflows' is to streamline this process by bundling the model alongside the preprocessor, all within the same object.

License MIT + file LICENSE

URL https://github.com/tidymodels/workflows

BugReports https://github.com/tidymodels/workflows/issues

Depends R (>= 3.2)

Imports cli (>= 2.0.0), ellipsis (>= 0.2.0), generics, glue, hardhat, parsnip (>= 0.0.4), rlang (>= 0.4.1)

Suggests covr, knitr, recipes, rmarkdown, testthat (>= 2.3.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Author Davis Vaughan [aut, cre], RStudio [cph]

Maintainer Davis Vaughan <davis@rstudio.com>

Repository CRAN

Date/Publication 2019-12-30 14:00:02 UTC

R topics documented:

`add_formula` *Add formula terms to a workflow*

Description

- `add_formula()` specifies the terms of the model through the usage of a formula.
- `remove_formula()` removes the formula as well as any downstream objects that might get created after the formula is used for preprocessing, such as terms. Additionally, if the model has already been fit, then the fit is removed.
- `update_formula()` first removes the formula, then replaces the previous formula with the new one. Any model that has already been fit based on this formula will need to be refit.

Usage

```
add_formula(x, formula, ..., blueprint = NULL)

remove_formula(x)

update_formula(x, formula, ..., blueprint = NULL)
```

Arguments

`x`	A workflow
`formula`	A formula specifying the terms of the model. It is advised to not do preprocessing in the formula, and instead use a recipe if that is required.
`...`	Not used.
`blueprint`	A hardhat blueprint used for fine tuning the preprocessing. If `NULL`, `hardhat::default_formula_bluepr` is used.

Details

To fit a workflow, one of `add_formula()` or `add_recipe()` *must* be specified, but not both.

Value

`x`, updated with either a new or removed formula preprocessor.

Examples

```
workflow <- workflow()
workflow <- add_formula(workflow, mpg ~ cyl)
workflow

remove_formula(workflow)

update_formula(workflow, mpg ~ disp)
```

`add_model` *Add a model to a workflow*

Description

- `add_model()` adds a parsnip model to the workflow.
- `remove_model()` removes the model specification as well as any fitted model object. Any extra formulas are also removed.
- `update_model()` first removes the model then adds the new specification to the workflow.

Usage

```
add_model(x, spec, formula = NULL)

remove_model(x)

update_model(x, spec, formula = NULL)
```

Arguments

`x`	A workflow.
`spec`	A parsnip model specification.
`formula`	An optional formula override to specify the terms of the model. Typically, the terms are extracted from the formula or recipe preprocessing methods. However, some models (like survival and bayesian models) use the formula not to preprocess, but to specify the structure of the model. In those cases, a formula specifying the model structure must be passed unchanged into the model call itself. This argument is used for those purposes.

Details

`add_model()` is a required step to construct a minimal workflow.

Value

`x`, updated with either a new or removed model.

Examples

```
library(parsnip)

lm_model <- linear_reg()
lm_model <- set_engine(lm_model, "lm")

regularized_model <- set_engine(lm_model, "glmnet")

workflow <- workflow()
workflow <- add_model(workflow, lm_model)
workflow

workflow <- add_formula(workflow, mpg ~ .)
workflow

remove_model(workflow)

fitted <- fit(workflow, data = mtcars)
fitted

remove_model(fitted)

remove_model(workflow)

update_model(workflow, regularized_model)
update_model(fitted, regularized_model)
```

---

add_recipe *Add a recipe to a workflow*

Description

- `add_recipe()` specifies the terms of the model and any preprocessing that is required through the usage of a recipe.
- `remove_recipe()` removes the recipe as well as any downstream objects that might get created after the recipe is used for preprocessing, such as the prepped recipe. Additionally, if the model has already been fit, then the fit is removed.
- `update_recipe()` first removes the recipe, then replaces the previous recipe with the new one. Any model that has already been fit based on this recipe will need to be refit.

Usage

```
add_recipe(x, recipe, ..., blueprint = NULL)

remove_recipe(x)

update_recipe(x, recipe, ..., blueprint = NULL)
```

Arguments

x	A workflow
recipe	A recipe created using recipes::recipe()
...	Not used.
blueprint	A hardhat blueprint used for fine tuning the preprocessing. If NULL, hardhat::default_recipe_bluepri is used.

Details

To fit a workflow, one of add_formula() or add_recipe() *must* be specified, but not both.

Value

x, updated with either a new or removed recipe preprocessor.

Examples

```
library(recipes)

recipe <- recipe(mpg ~ cyl, mtcars)
recipe <- step_log(recipe, cyl)

workflow <- workflow()
workflow <- add_recipe(workflow, recipe)
workflow

remove_recipe(workflow)

update_recipe(workflow, recipe(mpg ~ cyl, mtcars))
```

---

bivariate *Example Bivariate Classification Data*

Description

Example Bivariate Classification Data

Details

These data are a simplified version of the segmentation data contained in caret. There are three columns: A and B are predictors and the column Class is a factor with levels "One" and "Two". There are three data sets: one for training (n = 1009), validation (n = 300), and testing (n = 710).

Value

bivariate_train, bivariate_test, bivariate_val
tibbles

Examples

```
data(bivariate)
```

`control_workflow` *Control object for a workflow*

Description

`control_workflow()` holds the control parameters for a workflow.

Usage

```
control_workflow(control_parsnip = NULL)
```

Arguments

`control_parsnip`
: A parsnip control object. If `NULL`, a default control argument is constructed from `parsnip::control_parsnip()`.

Value

A `control_workflow` object for tweaking the workflow fitting process.

Examples

```
control_workflow()
```

`fit-workflow` *Fit a workflow object*

Description

Fitting a workflow currently involves two main steps:

- Preprocessing the data using a formula preprocessor, or by calling `recipes::prep()` on a recipe.
- Fitting the underlying parsnip model using `parsnip::fit.model_spec()`.

Usage

```
## S3 method for class 'workflow'
fit(object, data, ..., control = control_workflow())
```

Arguments

object	A workflow
data	A data frame of predictors and outcomes to use when fitting the workflow
...	Not used
control	A control_workflow() object

Details

In the future, there will also be *postprocessing* steps that can be added after the model has been fit.

Value

The workflow object, updated with a fit parsnip model in the objectfitfit slot.

Examples

```
library(parsnip)
library(recipes)

model <- linear_reg()
model <- set_engine(model, "lm")

base_workflow <- workflow()
base_workflow <- add_model(base_workflow, model)

formula_workflow <- add_formula(base_workflow, mpg ~ cyl + log(disp))

fit(formula_workflow, mtcars)

recipe <- recipe(mpg ~ cyl + disp, mtcars)
recipe <- step_log(recipe, disp)

recipe_workflow <- add_recipe(base_workflow, recipe)

fit(recipe_workflow, mtcars)
```

---

predict-workflow *Predict from a workflow*

Description

This is the predict() method for a fit workflow object. The nice thing about predicting from a workflow is that it will:

- Preprocess new_data using the preprocessing method specified when the workflow was created and fit. This is accomplished using hardhat::forge(), which will apply any formula preprocessing or call recipes::bake() if a recipe was supplied.
- Call parsnip::predict.model_fit() for you using the underlying fit parsnip model.

Usage

```
## S3 method for class 'workflow'
predict(object, new_data, type = NULL, opts = list(), ...)
```

Arguments

object	A workflow that has been fit by `fit.workflow()`
new_data	A data frame containing the new predictors to preprocess and predict on
type	A single character value or NULL. Possible values are "numeric", "class", "prob", "conf_int", "pred_int", "quantile", or "raw". When NULL, `predict()` will choose an appropriate value based on the model's mode.
opts	A list of optional arguments to the underlying predict function that will be used when `type = "raw"`. The list should not include options for the model object or the new data being predicted.
...	Arguments to the underlying model's prediction function cannot be passed here (see opts). There are some `parsnip` related options that can be passed, depending on the value of type. Possible arguments are: • `level`: for types of "conf_int" and "pred_int" this is the parameter for the tail area of the intervals (e.g. confidence level for confidence intervals). Default value is 0.95. • `std_error`: add the standard error of fit or prediction for types of "conf_int" and "pred_int". Default value is FALSE. • `quantile`: the quantile(s) for quantile regression (not implemented yet) • `time`: the time(s) for hazard probability estimates (not implemented yet)

Value

A data frame of model predictions, with as many rows as `new_data` has.

Examples

```
library(parsnip)
library(recipes)

training <- mtcars[1:20,]
testing <- mtcars[21:32,]

model <- linear_reg()
model <- set_engine(model, "lm")

workflow <- workflow()
workflow <- add_model(workflow, model)

recipe <- recipe(mpg ~ cyl + disp, training)
recipe <- step_log(recipe, disp)

workflow <- add_recipe(workflow, recipe)
```

```
fit_workflow <- fit(workflow, training)

# This will automatically `bake()` the recipe on `testing`,
# applying the log step to `disp`, and then fit the regression.
predict(fit_workflow, testing)
```

`workflow` *Create a workflow*

Description

A `workflow` is a container object that aggregates information required to fit and predict from a model. This information might be a recipe used in preprocessing, specified through `add_recipe()`, or the model specification to fit, specified through `add_model()`.

Usage

```
workflow()
```

Value

A new `workflow` object.

Examples

```
library(recipes)

rec <- recipe(mpg ~ cyl, mtcars)
rec <- step_log(rec, cyl)

wrk <- workflow()
wrk <- add_recipe(wrk, rec)
```

`workflow-extractors` *Extract elements of a workflow*

Description

These functions extract various elements from a workflow object. If they do not exist yet, an error is thrown.

- `pull_workflow_preprocessor()` returns either the formula or recipe used for preprocessing.
- `pull_workflow_spec()` returns the parsnip model specification.
- `pull_workflow_fit()` returns the parsnip model fit.

- `pull_workflow_mold()` returns the preprocessed "mold" object returned from `hardhat::mold()`. It contains information about the preprocessing, including either the prepped recipe or the formula terms object.
- `pull_workflow_prepped_recipe()` returns the prepped recipe. It is extracted from the mold object returned from `pull_workflow_mold()`.

Usage

```
pull_workflow_preprocessor(x)

pull_workflow_spec(x)

pull_workflow_fit(x)

pull_workflow_mold(x)

pull_workflow_prepped_recipe(x)
```

Arguments

x	A workflow

Value

The extracted value from the workflow, `x`, as described in the description section.

Examples

```
library(parsnip)
library(recipes)

model <- linear_reg()
model <- set_engine(model, "lm")

recipe <- recipe(mpg ~ cyl + disp, mtcars)
recipe <- step_log(recipe, disp)

base_workflow <- workflow()
base_workflow <- add_model(base_workflow, model)

recipe_workflow <- add_recipe(base_workflow, recipe)
formula_workflow <- add_formula(base_workflow, mpg ~ cyl + log(disp))

fit_recipe_workflow <- fit(recipe_workflow, mtcars)
fit_formula_workflow <- fit(formula_workflow, mtcars)

# The preprocessor is either a recipe or a formula
pull_workflow_preprocessor(recipe_workflow)
pull_workflow_preprocessor(formula_workflow)

# The `spec` is the parsnip spec before it has been fit.
```

```
# The `fit` is the fit parsnip model.
pull_workflow_spec(fit_formula_workflow)
pull_workflow_fit(fit_formula_workflow)

# The mold is returned from `hardhat::mold()`, and contains the
# predictors, outcomes, and information about the preprocessing
# for use on new data at `predict()` time.
pull_workflow_mold(fit_recipe_workflow)

# A useful shortcut is to extract the prepped recipe from the workflow
pull_workflow_prepped_recipe(fit_recipe_workflow)

# That is identical to
identical(
  pull_workflow_mold(fit_recipe_workflow)$blueprint$recipe,
  pull_workflow_prepped_recipe(fit_recipe_workflow)
)
```

Index