Package 'orbital'

March 13, 2026

Title Predict with 'tidymodels' Workflows in Databases

Version 0.5.1

Description Turn 'tidymodels' workflows into objects containing the sufficient sequential equations to perform predictions. These smaller objects allow for low dependency prediction locally or directly in databases.

License MIT + file LICENSE

URL https://github.com/tidymodels/orbital, https://orbital.tidymodels.org

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

Depends R (>= 4.1)

Imports cli, dplyr, generics (>= 0.1.2), rlang

Suggests arrow, bonsai, DBI, dbplyr, dtplyr, duckdb, earth, embed, glmnet, glue, gt, hardhat, jsonlite, kknn, knitr, lightgbm, modeldata, parsnip, partykit, probably, R6, randomForest, ranger, recipes, rmarkdown, rpart, RSQLite, rstanarm, sparklyr, splines2, tailor, testthat (>= 3.0.0), themis, tibble, tidypredict (>= 1.1.0), workflows, xgboost

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate, rmarkdown, gt

Config/testthat/edition 3

Config/usethis/last-upkeep 2025-06-05

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre], Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

Maintainer Emil Hvitfeldt <emil.hvitfeldt@posit.co>

Repository CRAN

Date/Publication 2026-03-13 07:00:02 UTC

Contents

`augment.orbital_class` *Augment using orbital objects*

Description

`augment()` will add column(s) for predictions to the given data.

Usage

```
## S3 method for class 'orbital_class'
augment(x, new_data, ...)
```

Arguments

`x`	An orbital object.
`new_data`	A data frame or remote database table.
`...`	Not currently used.

Details

This function is a shorthand for the following code

```
dplyr::bind_cols(
  predict(orbital_obj, new_data),
  new_data
)
```

Note that `augment()` works better and safer than above as it also works on data set in data bases.

This function is confirmed to not work work in spark data bases or arrow tables.

Value

A modified data frame or remote database table.

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)

wf_fit <- fit(wf_spec, mtcars)

orbital_obj <- orbital(wf_fit)

augment(orbital_obj, mtcars)
```

---

estimate_orbital_size *Estimate orbital expression character count*

Description

Estimates the character count of the orbital expression that would be generated for a model, without actually generating it. This is useful during hyperparameter tuning when you want to track SQL size as a metric but don't want to pay the cost of generating the full orbital object for every candidate model.

Usage

```
estimate_orbital_size(x, ...)

## S3 method for class 'xgb.Booster'
estimate_orbital_size(x, ...)

## S3 method for class 'lgb.Booster'
estimate_orbital_size(x, ...)

## S3 method for class 'ranger'
estimate_orbital_size(x, ...)

## S3 method for class 'randomForest'
estimate_orbital_size(x, ...)

## S3 method for class 'rpart'
estimate_orbital_size(x, ...)
```

```
## S3 method for class 'constparty'
estimate_orbital_size(x, ...)

## S3 method for class 'catboost.Model'
estimate_orbital_size(x, ...)

## S3 method for class 'glm'
estimate_orbital_size(x, ...)

## S3 method for class 'lm'
estimate_orbital_size(x, ...)

## S3 method for class 'glmnet'
estimate_orbital_size(x, ..., penalty = NULL)

## S3 method for class 'earth'
estimate_orbital_size(x, ...)

## S3 method for class 'recipe'
estimate_orbital_size(x, ...)

## S3 method for class 'workflow'
estimate_orbital_size(x, ...)

## S3 method for class 'tailor'
estimate_orbital_size(x, ...)
```

Arguments

`x`	A fitted model object, workflow, prepped recipe, or fitted tailor.
`...`	Additional arguments passed to methods.
`penalty`	For glmnet models, the penalty value (lambda) to use. If the model was fit with a single lambda, this is used by default. Otherwise, you must specify a value.

Details

The estimation uses model metadata (tree structure, number of parameters, feature names) to approximate the size of the resulting orbital expression. The estimates are typically within 5-10% of the actual size.

For tree-based models, this function is much faster than generating the full orbital object because it only needs to inspect the tree structure, not convert each tree to an R expression.

This function aims to support all the same models and preprocessing operations as `orbital()`. If you find a case where `orbital()` works but `estimate_orbital_size()` does not, please file an issue.

Value

An integer estimate of the total character count of the orbital expression.

See Also

orbital() for generating orbital objects.

Examples

```
library(xgboost)

# Estimate size for an xgboost model
x <- as.matrix(mtcars[, -1])
y <- mtcars[, 1]
model <- xgboost(x = x, y = y, nrounds = 50, max_depth = 4, verbosity = 0)

estimate_orbital_size(model)


library(recipes)
library(workflows)
library(parsnip)

# Estimate size for a workflow
rec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

wf <- workflow() |>
  add_recipe(rec) |>
  add_model(linear_reg()) |>
  fit(mtcars)

estimate_orbital_size(wf)
```

---

orbital *Turn tidymodels objects into orbital objects*

Description

Fitted workflows, parsnip objects, and recipes objects can be turned into an orbital object that contain all the information needed to perform predictions.

Usage

```
orbital(x, ..., prefix = ".pred", type = NULL, separate_trees = FALSE)
```

Arguments

x	A fitted workflow, parsnip, or recipes object.
...	Not currently used.

prefix	A single string, specifies the prediction naming scheme. If x produces a prediction, tidymodels standards dictate that the predictions will start with .pred. This is not a valid name for some data bases.
type	A vector of strings, specifies the prediction type. Regression models allow for "numeric" and classification models allow for "class" and "prob". Multiple values are allowed to produce hard and soft predictions for classification models. Defaults to NULL which defaults to "numeric" for regression models and "class" for classification models.
separate_trees	A single logical. For tree ensemble models (xgboost, lightgbm, catboost, ranger, randomForest), should each tree be output as a separate expression? This can improve performance when predicting in databases by allowing parallel evaluation of trees. Defaults to FALSE. See vignette("separate-trees") for details.

Details

An orbital object contains all the information that is needed to perform predictions. This makes the objects substantially smaller than the original objects. The main downside with this object is that all the input checking has been removed, and it is thus up to the user to make sure the data is correct.

The printing of orbital objects reduce the number of significant digits for easy viewing, the can be changes by using the digits argument of print() like so print(orbital_object, digits = 10). The printing likewise truncates each equation to fit on one line. This can be turned off using the truncate argument like so print(orbital_object, truncate = FALSE).

Full list of supported models and recipes steps can be found here: vignette("supported-models").

These objects will not be useful by themselves. They can be used to predict() with, or to generate code using functions such as orbital_sql() or orbital_dt().

Value

An orbital object.

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)

wf_fit <- fit(wf_spec, mtcars)

orbital(wf_fit)

# Also works on parsnip object by itself
fit(lm_spec, mpg ~ disp, data = mtcars) |>
```

```
  orbital()

# And prepped recipes
prep(rec_spec) |>
  orbital()
```

---

orbital_dt *Convert to data.table code*

Description

Returns data.table code that is equivilant to prediction code.

Usage

```
orbital_dt(x)
```

Arguments

x	An orbital object. This function requires dtplyr to be installed to run. The resulting code will likely need to be adopted to your use-case. Most likely by removing the initial `copy(data-name)` at the start.

Value

data.table code.

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)

wf_fit <- fit(wf_spec, mtcars)

orbital_obj <- orbital(wf_fit)

orbital_dt(orbital_obj)
```

`orbital_inline` *Convert orbital objects to quosures*

Description

Use orbital object splicing function to apply orbital prediction in a quosure aware function such as `dplyr::mutate()`.

Usage

```
orbital_inline(x)
```

Arguments

`x`	An orbital object.

Details

This function is mostly going to be used for Dots Injection. This function is used internally in predict(), but is also exported for user flexibility. Should be used with `!!!` as seen in the examples.

Note should be taken that using this function modifies existing variables and creates new variables, unlike predict() which only returns predictions.

Value

a list of quosures.

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)

wf_fit <- fit(wf_spec, mtcars)

orbital_obj <- orbital(wf_fit)

orbital_inline(orbital_obj)

library(dplyr)

mtcars |>
```

```
  mutate(!!!orbital_inline(orbital_obj))
```

`orbital_json_read` *Read orbital json file*

Description

Reading an orbital object from disk

Usage

```
orbital_json_read(path)
```

Arguments

`path`	file on disk.

Details

This function is aware of the `version` field of the orbital object, and will read it in correctly, according to its specification.

Value

An orbital object.

See Also

`orbital_json_write()`

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)

wf_fit <- fit(wf_spec, mtcars)

orbital_obj <- orbital(wf_fit)

tmp_file <- tempfile()
```

```
orbital_json_write(orbital_obj, tmp_file)

orbital_json_read(tmp_file)
```

`orbital_json_write` *Save orbital object as json file*

Description

Saving an orbital object to disk in a human and machine readable way.

Usage

```
orbital_json_write(x, path)
```

Arguments

`x`	An orbital object.
`path`	file on disk.

Details

The structure of the resulting JSON file allows for easy reading, both by orbital itself with `orbital_json_read()`, but potentially by other packages and langauges. The file is versioned by the `version` field to allow for changes why being backwards combatible with older file versions.

Value

Nothing.

See Also

`orbital_json_read()`

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)

wf_fit <- fit(wf_spec, mtcars)
```

```
orbital_obj <- orbital(wf_fit)

tmp_file <- tempfile()

orbital_json_write(orbital_obj, tmp_file)

readLines(tmp_file)
```

`orbital_r_fun` *Turn orbital object into a R function*

Description

Returns a R file that contains a function that output predictions when applied to data frames.

Usage

```
orbital_r_fun(x, name = "orbital_predict", file)
```

Arguments

`x`	An orbital object.
`name`	Name of created function. Defaults to '"orbital_predict"'.
`file`	A file name.

Details

The generated function is only expected to work on data frame objects. The generated function doesn't require the orbital package to be loaded. Depending on what models and steps are used, other packages such as dplyr will need to be loaded as well.

Value

Nothing.

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)
```

```
wf_fit <- fit(wf_spec, mtcars)

orbital_obj <- orbital(wf_fit)

file_name <- tempfile()

orbital_r_fun(orbital_obj, file = file_name)

readLines(file_name)
```

`orbital_sql` *Convert to SQL code*

Description

Returns SQL code that is equivilant to prediction code.

Usage

```
orbital_sql(x, con)
```

Arguments

x	An orbital object.
con	A connection object.

Details

This function requires a database connection object, as the resulting code SQL code can differ depending on the type of database.

Value

SQL code.

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)
```

```
wf_fit <- fit(wf_spec, mtcars)

orbital_obj <- orbital(wf_fit)

library(dbplyr)
con <- simulate_dbi()

orbital_sql(orbital_obj, con)
```

`predict.orbital_class` *Prediction using orbital objects*

Description

Running prediction on data frame of remote database table, without needing to load original packages used to fit model.

Usage

```
## S3 method for class 'orbital_class'
predict(object, new_data, ...)
```

Arguments

`object`	An orbital object.
`new_data`	A data frame or remote database table.
`...`	Not currently used.

Details

Using this function should give identical results to running `predict()` or `bake()` on the orginal object.

The prediction done will only return prediction colunms, a opposed to returning all modified functions as done with `orbital_inline()`.

Value

A modified data frame or remote database table.

Examples

```
library(workflows)
library(recipes)
library(parsnip)

rec_spec <- recipe(mpg ~ ., data = mtcars) |>
  step_normalize(all_numeric_predictors())

lm_spec <- linear_reg()

wf_spec <- workflow(rec_spec, lm_spec)

wf_fit <- fit(wf_spec, mtcars)

orbital_obj <- orbital(wf_fit)

predict(orbital_obj, mtcars)
```

Index