Package 'tidypredict'

May 8, 2026

Title Run Predictions Inside the Database

Version 1.1.0

Description It parses a fitted 'R' model object, and returns a formula in 'Tidy Eval' code that calculates the predictions. It works with several databases back-ends because it leverages 'dplyr' and 'dbplyr' for the final 'SQL' translation of the algorithm. It currently supports lm(), glm(), randomForest(), ranger(), rpart(), earth(), xgb.Booster.complete(), lgb.Booster(), catboost.Model(), cubist(), and ctree() models.

License MIT + file LICENSE

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

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

Depends R (>= 3.6)

Imports cli, dplyr (>= 0.7), generics, jsonlite, knitr, purrr, rlang (>= 1.1.1), tibble, tidyr

Suggests bonsai, covr, Cubist (>= 0.5.1), DBI, dbplyr, earth (>= 5.1.2), glmnet, lightgbm, methods, mlbench, modeldata, nycflights13, parsnip, partykit, randomForest, ranger (>= 0.14.1), rpart (>= 4.1.0), rmarkdown, RSQLite, survival, testthat (>= 3.2.0), withr, xgboost, yaml

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre], Edgar Ruiz [aut], Max Kuhn [aut]

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

Repository CRAN

Date/Publication 2026-02-27 06:30:14 UTC

Contents

`acceptable_formula` *Checks that the formula can be parsed*

Description

Uses an S3 method to check that a given formula can be parsed based on its class. It currently scans for contrasts that are not supported and in-line functions. (e.g: lm(wt ~ as.factor(am))). Since this function is meant for function interaction, as opposed to human interaction, a successful check is silent.

Usage

```
acceptable_formula(model)
```

Arguments

`model`	An R model object

Examples

```
model <- lm(mpg ~ wt, mtcars)
acceptable_formula(model)
```

as_parsed_model *Prepares parsed model object*

Description

Prepares parsed model object

Usage

```
as_parsed_model(x)
```

Arguments

x	A parsed model object

generate_tree_node *Construct a single node of a tree*

Description

Construct a single node of a tree

Usage

```
generate_tree_node(node, calc_mode = "")
```

Arguments

node	a list with named elements path and prediction. See details for more.
calc_mode	character, takes values "" and "calc_mode".

The node list should contain the two lists path and prediction.

The path element has the following structure:

This list can contain 0 or more elemements. The elements but each be of the following format:

- type character, must be "conditional", "set", or "all".
- op character. if type == "conditional" must be "more", "more-equal", "less", or "less-equal". if type == "set" must be "in" on not-in.
- col character.
- val if type == "conditional" and vals if type == "set". Can be character or numeric.

The prediction list has the following structure:

It can either be a singular value or a list. If it is a list it will have the following 4 named elements col, val, op, and is_intercept.

- col character, name of column
- val val, numeric of character
- op character, known values are "none" and "multiply". "none" is used then is_intercept == 1.
- is_interceptinteger, takes values 0 and 1.‘

@keywords internal

---

parse_model *Converts an R model object into a parsed model*

Description

Parses a fitted R model's structure and extracts the components needed to create a dplyr formula for prediction. The parsed model can be serialized (e.g., saved to YAML) and later used to generate predictions without the original model object.

Usage

```
parse_model(model)
```

Arguments

model	An R model object.

Value

A parsed model object with class parsed_model and a model-specific subclass (e.g., pm_xgb, pm_tree, pm_regression). The object contains:

- $general: List with model metadata including model (model type), type (used for S3 dispatch), version (parsed model format version), and model-specific parameters.
- Model-specific fields containing coefficients, tree structures, etc.

Parsed model versions

The $general$version field indicates the parsed model format:

- **Version 1**: Original format. Linear models store coefficients in a data frame. Tree models use flat case_when() expressions where all leaf conditions are at the same level.
- **Version 2**: Improved coefficient storage for linear models (lm, earth). Tree models still use flat case_when().
- **Version 3**: Current format. Tree models (rpart, ranger, randomForest, xgboost, lightgbm, catboost, partykit, cubist) use nested case_when() expressions that mirror the tree structure. This produces more efficient SQL and R code because conditions are evaluated hierarchically rather than checking all leaf paths.

When loading a parsed model saved with an older version, tidypredict automatically uses the appropriate formula builder for backwards compatibility.

Model types

Each parsed model has a type that determines the S3 class used for dispatch:

- `pm_regression`: Linear models (lm, glm, earth, glmnet)
- `pm_tree`: Single trees and forests (rpart, partykit, ranger, randomForest, cubist)
- `pm_xgb`: XGBoost gradient boosting models
- `pm_lgb`: LightGBM gradient boosting models
- `pm_catboost`: CatBoost gradient boosting models

Examples

```
library(dplyr)
df <- mutate(mtcars, cyl = paste0("cyl", cyl))
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = df)
parse_model(model)
```

`path_formula` *Turn a path object into an expression*

Description

Turn a path object into an expression

Usage

```
path_formula(x)
```

Arguments

`x`	a list. The input of this function is a list with 4 values. • `type` character, must be `"conditional"` or `"set"`. • `op` character. if `type == "conditional"` must be `"more"`, `"more-equal"`, `"less"`, or `"less-equal"`. if `type == "set"` must be `"in"` on `not-in`. • `col` character. • `val` if `type == "conditional"` and `vals` if `type == "set"`. Can be character or numeric. @keywords internal

`path_formulas` *Turn a path object into a combined expression*

Description

Turn a path object into a combined expression

Usage

```
path_formulas(path)
```

Arguments

`path`	a list of lists. This list can contain 0 or more elemements. The elements but each be of the following format: • `type` character, must be `"conditional"`, `"set"`, or `"all"`. • `op` character. if `type == "conditional"` must be `"more"`, `"more-equal"`, `"less"`, or `"less-equal"`. if `type == "set"` must be `"in"` on `not-in`. • `col` character. • `val` if `type == "conditional"` and `vals` if `type == "set"`. Can be character or numeric. @keywords internal

`set_catboost_categories`
Set categorical feature mappings for CatBoost model

Description

CatBoost stores categorical features as hash values internally. This function establishes the mapping between hash values and category names by examining a data frame with the same factor columns used during training.

Usage

```
set_catboost_categories(parsed_model, model, data)
```

Arguments

`parsed_model`	A parsed CatBoost model from `parse_model()`
`model`	The original CatBoost model object
`data`	A data frame containing factor columns matching the categorical features used in the model. The factor levels must match those from training.

Details

This function is only needed when using raw CatBoost models (trained with `catboost.train()`). When using parsnip/bonsai, categorical features are handled automatically and this function is not required.

Value

The parsed model with category mappings added

Examples

```
## Not run:
# For raw CatBoost models with categorical features:
pm <- parse_model(catboost_model)
pm <- set_catboost_categories(pm, catboost_model, training_data)
tidypredict_fit(pm)

# For parsnip/bonsai models, this is not needed:
# tidypredict_fit(parsnip_model_fit)  # works automatically

## End(Not run)
```

`tidy.pm_regression` *Tidy the parsed model results*

Description

Tidy the parsed model results

Usage

```
## S3 method for class 'pm_regression'
tidy(x, ...)
```

Arguments

`x`	A parsed_model object
`...`	Reserved for future use

tidypredict_fit *Returns a Tidy Eval formula to calculate fitted values*

Description

It parses a model or uses an already parsed model to return a Tidy Eval formula that can then be used inside a dplyr command.

Usage

```
tidypredict_fit(model)
```

Arguments

model	An R model or a list with a parsed model.

Examples

```
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = mtcars)
tidypredict_fit(model)
```

tidypredict_interval *Returns a Tidy Eval formula to calculate prediction interval.*

Description

It parses a model or uses an already parsed model to return a Tidy Eval formula that can then be used inside a dplyr command.

Usage

```
tidypredict_interval(model, interval = 0.95)
```

Arguments

model	An R model or a list with a parsed model
interval	The prediction interval, defaults to 0.95

Details

The result still has to be added to and subtracted from the fit to obtain the upper and lower bound respectively.

Examples

```
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = mtcars)
tidypredict_interval(model)
```

`tidypredict_test` *Tests base predict function against tidypredict*

Description

Compares the results of predict() and tidypredict_to_column() functions.

Usage

```
tidypredict_test(
  model,
  df = model$model,
  threshold = 1e-12,
  include_intervals = FALSE,
  max_rows = NULL,
  xg_df = NULL
)
```

Arguments

`model`	An R model or a list with a parsed model. It currently supports lm(), glm() and randomForest() models.
`df`	A data frame that contains all of the needed fields to run the prediction. It defaults to the "model" data frame object inside the model object.
`threshold`	The number that a given result difference, between predict() and tidypredict_to_column() should not exceed. For continuous predictions, the default value is 0.000000000001 (1e-12), and for categorical predictions, the default value is 0.
`include_intervals`	Switch to indicate if the prediction intervals should be included in the test. It defaults to FALSE.
`max_rows`	The number of rows in the object passed in the df argument. Highly recommended for large data sets.
`xg_df`	A xgb.DMatrix object, required only for XGBoost models. It defaults to NULL recommended for large data sets.

Examples

```
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = mtcars)
tidypredict_test(model)
```

`tidypredict_to_column` *Adds the prediction columns to a piped command set.*

Description

Adds a new column with the results from tidypredict_fit() to a piped command set. If add_interval is set to TRUE, it will add two additional columns- one for the lower and another for the upper prediction interval bounds.

Usage

```
tidypredict_to_column(
  df,
  model,
  add_interval = FALSE,
  interval = 0.95,
  vars = c("fit", "upper", "lower")
)
```

Arguments

`df`	A data.frame or tibble
`model`	An R model or a parsed model inside a data frame
`add_interval`	Switch that indicates if the prediction interval columns should be added. Defaults to FALSE
`interval`	The prediction interval, defaults to 0.95. Ignored if add_interval is set to FALSE
`vars`	The name of the variables that this function will produce. Defaults to "fit", "upper", and "lower".

Index