Package 'offsetreg'

April 11, 2024

Title An Extension of 'Tidymodels' Supporting Offset Terms

Version 1.1.0

Maintainer Matt Heaphy <mattrmattrs@gmail.com>

Description Extend the 'tidymodels' ecosystem <https://www.tidymodels.org/> to enable the creation of predictive models with offset terms. Models with offsets are most useful when working with count data or when fitting an adjustment model on top of an existing model with a prior expectation. The former situation is common in insurance where data is often weighted by exposures. The latter is common in life insurance where industry mortality tables are often used as a starting point for setting assumptions.

License MIT + file LICENSE

URL https://github.com/mattheaphy/offsetreg/, https://mattheaphy.github.io/offsetreg/

BugReports https://github.com/mattheaphy/offsetreg/issues

Encoding UTF-8

RoxygenNote 7.3.0

Imports generics, glue, parsnip (>= 1.2.0), poissonreg, rlang, stats

Suggests broom, glmnet, knitr, recipes, rmarkdown, rpart, testthat (>= 3.0.0), tune, workflows, rsample, xgboost

Config/testthat/edition 3

Depends R (>= 4.1)

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Matt Heaphy [aut, cre, cph]

Repository CRAN

Date/Publication 2024-04-11 14:40:03 UTC

R topics documented:

boost_tree_offset 2
decision_tree_exposure 3
glmnet_offset 5
glm_offset 6
poisson_reg_offset 7
rpart_exposure 8
us_deaths 9
xgb_train_offset 10

Index **13**

`boost_tree_offset` *Boosted Poisson Trees with Offsets*

Description

`boost_tree_offset()` defines a model that creates a series of Poisson decision trees with predefined offsets forming an ensemble. Each tree depends on the results of previous trees. All trees in the ensemble are combined to produce a final prediction. This function can be used for count regression models only.

Usage

```
boost_tree_offset(
  mode = "regression",
  engine = "xgboost_offset",
  mtry = NULL,
  trees = NULL,
  min_n = NULL,
  tree_depth = NULL,
  learn_rate = NULL,
  loss_reduction = NULL,
  sample_size = NULL,
  stop_iter = NULL
)
```

Arguments

`mode`	A single character string for the type of model. The only possible value for this model is "regression"
`engine`	A single character string specifying what computational engine to use for fitting.
`mtry`	A number for the number (or proportion) of predictors that will be randomly sampled at each split when creating the tree models (specific engines only).
`trees`	An integer for the number of trees contained in the ensemble.

`min_n`	An integer for the minimum number of data points in a node that is required for the node to be split further.
`tree_depth`	An integer for the maximum depth of the tree (i.e. number of splits) (specific engines only).
`learn_rate`	A number for the rate at which the boosting algorithm adapts from iteration-to-iteration (specific engines only). This is sometimes referred to as the shrinkage parameter.
`loss_reduction`	A number for the reduction in the loss function required to split further (specific engines only).
`sample_size`	A number for the number (or proportion) of data that is exposed to the fitting routine. For `xgboost`, the sampling is done at each iteration while `C5.0` samples once during training.
`stop_iter`	The number of iterations without improvement before stopping (specific engines only).

Details

This function is similar to `parsnip::boost_tree()` except that specification of an offset column is required.

Value

A model specification object with the classes `boost_tree_offset` and `model_spec`.

See Also

`parsnip::boost_tree()`

Examples

```
parsnip::show_model_info("boost_tree_offset")

boost_tree_offset()
```

`decision_tree_exposure` *Poisson Decision Trees with Exposures*

Description

`decision_tree_exposure()` defines a Poisson decision tree model with weighted exposures (observation times).

Usage

```
decision_tree_exposure(
  mode = "regression",
  engine = "rpart_exposure",
  cost_complexity = NULL,
  tree_depth = NULL,
  min_n = NULL
)
```

Arguments

mode	A single character string for the type of model. The only possible value for this model is "regression"
engine	A single character string specifying what computational engine to use for fitting.
cost_complexity	A positive number for the the cost/complexity parameter (a.k.a. Cp) used by CART models (specific engines only).
tree_depth	An integer for maximum depth of the tree.
min_n	An integer for the minimum number of data points in a node that are required for the node to be split further.

Details

This function is similar to parsnip::decision_tree() except that specification of an exposure column is required.

Value

A model specification object with the classes decision_tree_exposure and model_spec.

See Also

parsnip::decision_tree()

Examples

```
parsnip::show_model_info("decision_tree_exposure")

decision_tree_exposure()
```

glmnet_offset *Fit Penalized Generalized Linear Models with an Offset*

Description

This function is a wrapper around `glmnet::glmnet()` that uses a column from `x` as an offset.

Usage

```
glmnet_offset(
  x,
  y,
  family,
  offset_col = "offset",
  weights = NULL,
  lambda = NULL,
  alpha = 1
)
```

Arguments

`x`	Input matrix
`y`	Response variable
`family`	A function or character string describing the link function and error distribution.
`offset_col`	Character string. The name of a column in `data` containing offsets.
`weights`	Optional weights to use in the fitting process.
`lambda`	A numeric vector of regularization penalty values
`alpha`	A number between zero and one denoting the proportion of L1 (lasso) versus L2 (ridge) regularization. • `alpha = 1`: Pure lasso model • `alpha = 0`: Pure ridge model

Details

Outside of the `tidymodels` ecosystem, `glmnet_offset()` has no advantages over `glmnet::glmnet()` since that function allows for offsets to be specified in its `offset` argument.

Within `tidymodels`, `glmnet_offset()` provides an advantage because it will ensure that offsets are included in the data whenever resamples are created.

The `x`, `y`, `family`, `lambda`, `alpha` and `weights` arguments have the same meanings as `glmnet::glmnet()`. See that function's documentation for full details.

Value

A `glmnet` object. See `glmnet::glmnet()` for full details.

See Also

glmnet::glmnet()

Examples

```
us_deaths$off <- log(us_deaths$population)
x <- model.matrix(~ age_group + gender + off, us_deaths)[, -1]
glmnet_offset(x, us_deaths$deaths, family = "poisson", offset_col = "off")
```

---

glm_offset *Fit Generalized Linear Models with an Offset*

Description

This function is a wrapper around stats::glm() that uses a column from data as an offset.

Usage

```
glm_offset(
  formula,
  family = "gaussian",
  data,
  offset_col = "offset",
  weights = NULL
)
```

Arguments

formula	A model formula
family	A function or character string describing the link function and error distribution.
data	Optional. A data frame containing variables used in the model.
offset_col	Character string. The name of a column in data containing offsets.
weights	Optional weights to use in the fitting process.

Details

Outside of the tidymodels ecosystem, glm_offset() has no advantages over stats::glm() since that function allows for offsets to be specified in the formula interface or its offset argument.

Within tidymodels, glm_offset() provides an advantage because it will ensure that offsets are included in the data whenever resamples are created.

The formula, family, data, and weights arguments have the same meanings as stats::glm(). See that function's documentation for full details.

Value

A glm object. See stats::glm() for full details.

See Also

stats::glm()

Examples

```
us_deaths$off <- log(us_deaths$population)
glm_offset(deaths ~ age_group + gender, family = "poisson",
           us_deaths, offset_col = "off")
```

---

poisson_reg_offset *Poisson regression models with offsets*

Description

poisson_reg_offset() defines a generalized linear model of count data with an offset that follows a Poisson distribution.

Usage

```
poisson_reg_offset(
  mode = "regression",
  penalty = NULL,
  mixture = NULL,
  engine = "glm_offset"
)
```

Arguments

mode	A single character string for the type of model. The only possible value for this model is "regression".
penalty	A non-negative number representing the total amount of regularization (glmnet only).
mixture	A number between zero and one (inclusive) giving the proportion of L1 regularization (i.e. lasso) in the model. • mixture = 1 specifies a pure lasso model, • mixture = 0 specifies a ridge regression model, and • 0 < mixture < 1 specifies an elastic net model, interpolating lasso and ridge. Available for glmnet and spark only.
engine	A single character string specifying what computational engine to use for fitting.

Details

This function is similar to `parsnip::poisson_reg()` except that specification of an offset column is required.

Value

A model specification object with the classes `poisson_reg_offset` and `model_spec`.

See Also

`parsnip::poisson_reg()`

Examples

```
parsnip::show_model_info("poisson_reg_offset")

poisson_reg_offset()
```

`rpart_exposure` *Poisson Recursive Partitioning and Regression Trees with Exposures*

Description

This function is a wrapper around `rpart::rpart()` for Poisson regression trees using weighted exposures (observation times).

Usage

```
rpart_exposure(
  formula,
  data,
  exposure_col = "exposure",
  weights = NULL,
  control,
  cost,
  shrink = 1,
  ...
)
```

Arguments

`formula`	A model formula that contains a single response variable on the left-hand side.
`data`	Optional. A data frame containing variables used in the model.
`exposure_col`	Character string. The name of a column in `data` containing exposures.
`weights`	Optional weights to use in the fitting process.

control	A list of hyperparameters. See rpart::rpart.control().
cost	A vector of non-negative costs for each variable in the model.
shrink	Optional parameter for the splitting function. Coefficient of variation of the prior distribution.
...	Alternative input for arguments passed to rpart::rpart.control().

Details

Outside of the tidymodels ecosystem, rpart_exposure() has no advantages over rpart::rpart() since that function allows for exposures to be specified in the formula interface by passing cbind(exposure, y) as a response variable.

Within tidymodels, rpart_exposure() provides an advantage because it will ensure that exposures are included in the data whenever resamples are created.

The formula, data, weights, control, and cost arguments have the same meanings as rpart::rpart(). shrink is passed to rpart::rpart()'s parms argument via a named list. See that function's documentation for full details.

Value

An rpart model

See Also

rpart::rpart()

Examples

```
rpart_exposure(deaths ~ age_group + gender, us_deaths,
               exposure_col = "population")
```

---

us_deaths *United States Deaths 2011-2020*

Description

United States deaths, population estimates, and crude mortality rates for ages 25+ from the CDC Multiple Causes of Death Files.

Usage

```
us_deaths
```

Format

A data frame with 140 rows and 6 columns.

gender Gender

age_group Attained age groups

year Calendar year

deaths Number of deaths

population Population estimate

qx Crude mortality rate equal to `deaths / population`

Source

Centers for Disease Control and Prevention, National Center for Health Statistics. National Vital Statistics System, Mortality 1999-2020 on CDC WONDER Online Database, released in 2021. Data are from the Multiple Cause of Death Files, 1999-2020, as compiled from data provided by the 57 vital statistics jurisdictions through the Vital Statistics Cooperative Program. Accessed at http://wonder.cdc.gov/mcd-icd10.html on Jan 15, 2024."

`xgb_train_offset` *Boosted Poisson Trees with Offsets via* `xgboost`

Description

`xgb_train_offset()` and `xgb_predict_offset()` are wrappers for `xgboost` tree-based models where all of the model arguments are in the main function. These functions are nearly identical to the parsnip functions `parsnip::xgb_train()` and `parsnip::xg_predict_offset()` except that the objective "count:poisson" is passed to `xgboost::xgb.train()` and an offset term is added to the data set.

Usage

```
xgb_train_offset(
  x,
  y,
  offset_col = "offset",
  weights = NULL,
  max_depth = 6,
  nrounds = 15,
  eta = 0.3,
  colsample_bynode = NULL,
  colsample_bytree = NULL,
  min_child_weight = 1,
  gamma = 0,
  subsample = 1,
  validation = 0,
```

```
  early_stop = NULL,
  counts = TRUE,
  ...
)

xgb_predict_offset(object, new_data, offset_col = "offset", ...)
```

Arguments

x	A data frame or matrix of predictors
y	A vector (numeric) or matrix (numeric) of outcome data.
offset_col	Character string. The name of a column in data containing offsets.
weights	A numeric vector of weights.
max_depth	An integer for the maximum depth of the tree.
nrounds	An integer for the number of boosting iterations.
eta	A numeric value between zero and one to control the learning rate.
colsample_bynode	Subsampling proportion of columns for each node within each tree. See the counts argument below. The default uses all columns.
colsample_bytree	Subsampling proportion of columns for each tree. See the counts argument below. The default uses all columns.
min_child_weight	A numeric value for the minimum sum of instance weights needed in a child to continue to split.
gamma	A number for the minimum loss reduction required to make a further partition on a leaf node of the tree
subsample	Subsampling proportion of rows. By default, all of the training data are used.
validation	The *proportion* of the data that are used for performance assessment and potential early stopping.
early_stop	An integer or NULL. If not NULL, it is the number of training iterations without improvement before stopping. If validation is used, performance is base on the validation set; otherwise, the training set is used.
counts	A logical. If FALSE, colsample_bynode and colsample_bytree are both assumed to be *proportions* of the proportion of columns affects (instead of counts).
...	Other options to pass to xgb.train() or xgboost's method for predict().
object	An xgboost object.
new_data	New data for predictions. Can be a data frame, matrix, xgb.DMatrix

Value

A fitted xgboost object.

Examples

```
us_deaths$off <- log(us_deaths$population)
x <- model.matrix(~ age_group + gender + off, us_deaths)[, -1]

mod <- xgb_train_offset(x, us_deaths$deaths, "off",
                        eta = 1, colsample_bynode = 1,
                        max_depth = 2, nrounds = 25,
                        counts = FALSE)

xgb_predict_offset(mod, x, "off")
```

Index

∗ **datasets**
us_deaths, 9

boost_tree_offset, 2

decision_tree_exposure, 3

glm_offset, 6
glmnet::glmnet(), *5, 6*
glmnet_offset, 5

parsnip::boost_tree(), *3*
parsnip::decision_tree(), *4*
parsnip::poisson_reg(), *8*
poisson_reg_offset, 7

rpart::rpart(), *8, 9*
rpart::rpart.control(), *9*
rpart_exposure, 8

stats::glm(), *6, 7*

us_deaths, 9

xgb_predict_offset *(*xgb_train_offset*)*, 10
xgb_train_offset, 10