Package 'sqlscore'

October 14, 2022

Version 0.1.4

Title Utilities for Generating SQL Queries from Model Objects

Description Provides utilities for generating SQL queries (particularly CREATE TABLE statements) from R model objects. The most important use case is generating SQL to score a generalized linear model or related model represented as an R object, in which case the package handles parsing formula operators and including the model's response function.

License MIT + file LICENSE

URL `https://github.com/wwbrannon/sqlscore/`

BugReports `https://github.com/wwbrannon/sqlscore/issues`

Depends R (>= 3.3.0)

Imports dbplyr (>= 1.0.0)

Suggests testthat, arm, glmnet, mboost, covr

RoxygenNote 6.1.1

NeedsCompilation no

Author William Brannon [aut, cre]

Maintainer William Brannon <`wwbrannon@email.wm.edu`>

Repository CRAN

Date/Publication 2019-03-17 06:40:03 UTC

R topics documented:

create_statement 2
linpred 3
score_expression 5
select_statement 6
sqlscore 7

Index **9**

create_statement *Generate a CREATE TABLE statement from a model*

Description

Generate a CREATE TABLE statement to score the passed model on a preexisting database table. The statement will generate predictions entirely in the database, with no need to fetch data into R. Models need not be GLMs, but their prediction steps must consist of applying a response function to a linear predictor.

Usage

```
create_statement(mod, dest_table, src_table, dest_schema = NULL,
  dest_catalog = NULL, src_schema = NULL, src_catalog = NULL,
  drop = FALSE, temporary = FALSE, pk = c("id"), response = NULL,
  con = dbplyr::simulate_dbi())
```

Arguments

`mod`	A supported model object.
`dest_table`	The unqualified DB name of the destination table.
`src_table`	The unqualified DB name of the source table.
`dest_schema`	The DB schema of the destination table.
`dest_catalog`	The DB catalog of the destination table.
`src_schema`	The DB schema of the source table.
`src_catalog`	The DB catalog of the source table.
`drop`	Whether to generate a DROP TABLE IF EXISTS before the CREATE TABLE.
`temporary`	Whether the destination table should be a temporary table.
`pk`	A vector of primary key column names.
`response`	The name of a custom response function to apply to the linear predictor.
`con`	A DBI connection to control the details of SQL generation; defaults to dbplyr::simulate_dbi() for the best guess at portable SQL.

Details

An open database connection can be passed as the ‘con‘ argument, or the ‘dbplyr::simulate_*‘ functions can be used in applications which don’t have a DB connection when they need to generate SQL.

Value

A dbplyr SQL object representing the SELECT statement.

Supported packages

Specific packages and models that are known to work include: glm and lm from package:stats, cv.glmnet from package:glmnet, glmboost from package:mboost, and bayesglm from package:arm.

Default S3 methods are for objects structured like those of class "glm", so models not listed here may work if they resemble those objects, but are not guaranteed to.

Warning

Note that if the model object transformed its training data before fitting (e.g., centering and scaling predictors), the generated SQL statement will not include those transformations. A future release may include that functionality, but centering and scaling in particular are difficult to do efficiently and portably in SQL.

Examples

```
# Basic create statements
mod <- glm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris)
create_statement(mod, src_table="tbl_name", dest_table="target_tbl")
create_statement(mod, src_table="tbl_name", src_schema="schema_name",
                 src_catalog="catalog_name", dest_table="target_tbl")
create_statement(mod, src_table="tbl_name", src_schema="schema_name",
                 src_catalog="catalog_name", dest_table="target_tbl",
                 dest_schema="target_schema", dest_catalog="target_catalog",
                 pk=c("lab", "specimen_id"))

#With a custom response function
create_statement(mod, src_table="tbl_name", src_schema="schema_name",
                 dest_table="target_tbl", response="probit")

# With a model-derived non-identity response function
mod <- glm(Sepal.Length > 5.0 ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris, family=binomial("logit"))
create_statement(mod, src_table="tbl_name", dest_table="target_tbl")

#With formula operators
x <- matrix(rnorm(100*20),100,20)
colnames(x) <- sapply(1:20, function(x) paste0("X", as.character(x)))
x <- as.data.frame(x)
mod <- glm(X2 ~ X3 + X5 + X15*X8, data=x)
create_statement(mod, src_table="tbl_name", dest_table="target_tbl")
create_statement(mod, src_table="tbl_name", dest_table="target_tbl",
                 response="cauchit")
```

`linpred` *Unevaluated prediction expressions for models*

Description

Generate an unevaluated call corresponding to the predict step of the passed model. The call represents the linear predictor in terms of elementary functions on the underlying column names. Before translation into SQL, it should have a response function applied by score_expression (which may be a no-op in the case of the identity response).

Usage

```
linpred(mod)
```

Arguments

`mod`	A supported model object.

Value

An unevaluated R call object representing the linear predictor.

Warning

The Binomial models in glmboost return coefficients which are 1/2 the coefficients fit by a call to glm(..., family=binomial(...)), because the response variable is internally recoded to -1 and +1. sqlscore multiplies the returned coefficients by 2 to put them back on the same scale as glm, and adds the glmboost offset to the intercept before multiplying.

Examples

```
# A Gaussian GLM including factors
mod <- glm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris)
linpred(mod)

# A binomial GLM - linear predictor is unaffected
mod <- glm(Sepal.Length > 5.0 ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris, family=binomial("logit"))
linpred(mod)

#With formula operators
x <- matrix(rnorm(100*20),100,20)
colnames(x) <- sapply(1:20, function(x) paste0("X", as.character(x)))
x <- as.data.frame(x)
mod <- glm(X2 ~ X3 + X5 + X15*X8, data=x)
linpred(mod)
```

`score_expression` *Unevaluated prediction expressions for models*

Description

Generate an unevaluated call corresponding to the predict step of the passed model. The call represents the response function of the linear predictor in terms of elementary functions on the underlying column names, and is suitable for direct translation into SQL.

Usage

```
score_expression(mod, response = NULL)
```

Arguments

`mod`	A supported model object.
`response`	The name of a custom response function to apply to the linear predictor.

Value

An unevaluated R call object representing the response function of the linear predictor.

Warning

The Binomial models in glmboost return coefficients which are 1/2 the coefficients fit by a call to glm(..., family=binomial(...)), because the response variable is internally recoded to -1 and +1. sqlscore multiplies the returned coefficients by 2 to put them back on the same scale as glm, and adds the glmboost offset to the intercept before multiplying.

Examples

```
# A Gaussian GLM including factors
mod <- glm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris)
score_expression(mod)

# A binomial GLM - linear predictor is unaffected
mod <- glm(Sepal.Length > 5.0 ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris, family=binomial("logit"))
score_expression(mod)

#With a hand-specified response function
score_expression(mod, response="probit")

#With formula operators
x <- matrix(rnorm(100*20),100,20)
colnames(x) <- sapply(1:20, function(x) paste0("X", as.character(x)))
x <- as.data.frame(x)
mod <- glm(X2 ~ X3 + X5 + X15*X8, data=x)
```

```
score_expression(mod)
```

select_statement *Generate a SELECT statement from a model*

Description

Generate a SELECT statement to score the passed model on a preexisting database table. The statement will generate predictions entirely in the database, with no need to fetch data into R. Models need not be GLMs, but their prediction steps must consist of applying a response function to a linear predictor.

Usage

```
select_statement(mod, src_table, src_schema = NULL, src_catalog = NULL,
  pk = c("id"), response = NULL, con = dbplyr::simulate_dbi())
```

Arguments

`mod`	A supported model object.
`src_table`	The unqualified DB name of the source table.
`src_schema`	The DB schema of the source table.
`src_catalog`	The DB catalog of the source table.
`pk`	A vector of primary key column names.
`response`	The name of a custom response function to apply to the linear predictor.
`con`	A DBI connection to control the details of SQL generation; defaults to dbplyr::simulate_dbi() for the best guess at portable SQL.

Details

An open database connection can be passed as the ‘con‘ argument, or the ‘dbplyr::simulate_*‘ functions can be used in applications which don’t have a DB connection when they need to generate SQL.

Value

A dbplyr SQL object representing the SELECT statement.

Supported packages

Specific packages and models that are known to work include: glm and lm from package:stats, cv.glmnet from package:glmnet, glmboost from package:mboost, and bayesglm from package:arm.

Default S3 methods are for objects structured like those of class "glm", so models not listed here may work if they resemble those objects, but are not guaranteed to.

Warning

Note that if the model object transformed its training data before fitting (e.g., centering and scaling predictors), the generated SQL statement will not include those transformations. A future release may include that functionality, but centering and scaling in particular are difficult to do efficiently and portably in SQL.

Examples

```
# Basic select statements
mod <- glm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris)
select_statement(mod, src_table="tbl_name")
select_statement(mod, src_table="tbl_name", src_schema="schema_name",
                 src_catalog="catalog_name")
select_statement(mod, src_table="tbl_name", src_schema="schema_name",
                 src_catalog="catalog_name", pk=c("lab", "specimen_id"))

#With a custom response function
select_statement(mod, src_table="tbl_name", src_schema="schema_name",
                 response="probit")

# With a model-derived non-identity response function
mod <- glm(Sepal.Length > 5.0 ~ Sepal.Width + Petal.Length + Petal.Width + Species,
           data=datasets::iris, family=binomial("logit"))
select_statement(mod, src_table="tbl_name")

#With formula operators
x <- matrix(rnorm(100*20),100,20)
colnames(x) <- sapply(1:20, function(x) paste0("X", as.character(x)))
x <- as.data.frame(x)
mod <- glm(X2 ~ X3 + X5 + X15*X8, data=x)
select_statement(mod, src_table="tbl_name")
select_statement(mod, src_table="tbl_name", response="cauchit")
```

`sqlscore` *sqlscore: Utilities to score GLMs and related models in SQL.*

Description

The sqlscore package provides utilities for generating sql queries (particularly CREATE TABLE statements) from R model objects. The most important use case is generating SQL to score a GLM or related model represented as an R object, in which case the package handles parsing formula operators and including the model's response function. The models scored need not be generalized linear models, strictly speaking, but their prediction steps must consist of applying a response function to a linear predictor. The package handles escaping and dealing with formula operators, and provides a way to use a custom response function if desired.

Function overview

The SQL-generating functions create_statement and select_statement do what their names suggest and generate CREATE TABLE and SELECT statements for model scoring. Helper functions include linpred(), which generates an R call object representing the linear predictor, and score_expression, an S3 generic that handles wrapping the linear predictor in the response function.

Supported models

Specific packages and models that are known to work include: glm and lm from package:stats, cv.glmnet from package:glmnet, glmboost from package:mboost, and bayesglm from package:arm.

Default S3 methods are for objects structured like those of class "glm", so models not listed here may work if they resemble those objects, but are not guaranteed to.

Index

create_statement, 2

linpred, 3

score_expression, 5
select_statement, 6
sqlscore, 7
sqlscore-package *(sqlscore)*, 7