

Package: glex (via r-universe)

October 9, 2026

Type Package

Title Global Explanations for Tree-Based Models

Version 0.7.0.9000

Description Global explanations for tree-based models by decomposing regression or classification functions into the sum of main components and interaction components of arbitrary order. Calculates SHAP values and q-interaction SHAP for all values of q for tree-based models such as xgboost.

License GPL-3

URL <https://github.com/PlantedML/glex>, <https://plantedml.com/glex/>

BugReports <https://github.com/PlantedML/glex/issues>

Depends R (>= 4.1.0)

Imports checkmate, data.table, foreach, ggplot2, jsonlite, progress, Rcpp (>= 1.0.8), scico, stats, utils

Suggests covr, doParallel, ISLR2, knitr, patchwork, randomPlantedForest (>= 0.3.0), ranger, rmarkdown, testthat (>= 3.0.0), tibble, xgboost (>= 3.0.0)

LinkingTo Rcpp

VignetteBuilder knitr

Remotes PlantedML/randomPlantedForest

Additional_repositories <https://plantedml.r-universe.dev>

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Repository <https://plantedml.r-universe.dev>

Date/Publication 2026-10-09 00:15:38 UTC

RemoteUrl <https://github.com/PlantedML/glex>

RemoteRef HEAD
RemoteSha ccffa061baed8a71e3b3e75a5d94fc50681e656f

Contents

autoplot.glex	2
autoplot.glex_vi	4
bike	5
dummy_groups	5
glex	6
glex_explain	10
glex_options	11
glex_vi	12
group_components	13
plot_pdp	14
print.glex	15
subset_components	16
theme_glex	17
Index	18

autoplot.glex	<i>Plot Prediction Components</i>
---------------	-----------------------------------

Description

Plotting the main effects among the prediction components is effectively identical to a partial dependence plot, centered to 0.

Usage

```
## S3 method for class 'glex'  
autoplot(object, predictors, ...)  
  
plot_main_effect(object, predictor, rug_sides = "b", ...)  
  
plot_threeway_effects(object, predictors, rug_sides = "b", ...)  
  
plot_twoway_effects(object, predictors, rug_sides = "b", ...)
```

Arguments

object	Object of class glex .
...	Used for future expansion.
predictor, predictors	(character) vector of predictor names, e.g. "x1" to plot main effect of x1, and c("x1", "x2") to plot the interaction term x1:x2.

`rug_sides` (character(1): "b") Sides to plot rug (see [ggplot2::geom_rug\(\)](#)) plot on for continuous predictors.. Default is "b" for both sides. Set to "none" to disable rug plot.

Value

A ggplot2 object.

See Also

[plot_pdp\(\)](#)

Other Visualization functions: [autoplot.glex_vi\(\)](#), [glex_explain\(\)](#), [plot_pdp\(\)](#)

Examples

```
library(randomPlantedForest)

# introduce factor variables to show categorical feature handling
mtcars$cyl <- factor(mtcars$cyl)
mtcars$vs <- factor(mtcars$vs)

# Fit forest, get components
set.seed(12)
rpfit <- rpf(mpg ~ cyl + wt + hp + drat + vs, data = mtcars, ntrees = 25, max_interaction = 3)
components <- glex(rpfit, mtcars)

# Main effects ----
plot_main_effect(components, "wt")
plot_main_effect(components, "cyl")

# plot_threeway_effects(components, c("hr", "temp", "workingday"))

library(randomPlantedForest)

# 2-degree interaction effects ----
# 2d continuous, scatterplot of arbitrary orientation
plot_tway_effects(components, c("wt", "drat"))
# flipped: plot_tway_effects(components, c("drat", "wt"))

# continuous + categorical (forces continuous on x axis, colors by categorical)
plot_tway_effects(components, c("wt", "cyl"))
# identical: plot_tway_effects(components, c("cyl", "wt"))

# 2d categorical, heatmap of arbitrary orientation
plot_tway_effects(components, c("vs", "cyl"))
plot_tway_effects(components, c("cyl", "vs"))
```

autoplot.glex_vi *Plot glex Variable Importances*

Description

Plot glex Variable Importances

Usage

```
## S3 method for class 'glex_vi'
autoplot(
  object,
  by_degree = FALSE,
  threshold = 0,
  max_interaction = NULL,
  scale = "absolute",
  ...
)
```

Arguments

<code>object</code>	Object of class <code>glex_vi</code> , see glex_vi() .
<code>by_degree</code>	(logical(1): FALSE) Optionally sum values by degree of interaction, resulting in one contribution score for all main effects, all second-order interactions, etc.
<code>threshold</code>	(numeric(1): 0) Optional threshold to filter output to include only importance scores greater than this value. Refers to the chosen scale.
<code>max_interaction</code>	(integer(1): NULL) Optionally filter plot to show terms up to the specified degree of interaction. Similar to <code>threshold</code> , all other terms will be aggregated under a "Remaining terms" label.
<code>scale</code>	("absolute") Plot average absolute contributions (default) or the same value but scaled by the average prediction ("relative").
<code>...</code>	(Unused)

Value

A [ggplot](#) object.

See Also

[glex_vi](#)

Other Visualization functions: [autoplot.glex\(\)](#), [glex_explain\(\)](#), [plot_pdp\(\)](#)

bike	<i>Bikesharing data</i>
------	-------------------------

Description

A reduced version of the Bikeshare data as included with ISLR2. The dataset has been converted to a `data.table::data.table()`, with the following changes:

Usage

```
bike
```

Format

An object of class `data.table` (inherits from `data.frame`) with 8645 rows and 11 columns.

Details

- `hr` has been converted to a numeric
- `workingday` was recoded to a binary factor with labels `c("No Workingday", "Workingday")`
- `season` was recoded to a factor with labels `c("Winter", "Spring", "Summer", "Fall")`
- Variables `atemp`, `day`, `registered` and `casual` were removed

Source

Bikeshare in package ISLR2

dummy_groups	<i>Derive feature groups from the factors behind a dummy encoding</i>
--------------	---

Description

Builds the groups list for `group_components()` from the original, un-encoded data: every factor column of data whose encoded level columns are found in `object$x` becomes one group. By default, level columns are expected under the `model.matrix()` naming convention (`paste0(feature, levels)`), e.g. `season` with level `"Winter"` becomes `"seasonWinter"`), which covers both one-hot (`~ f - 1`) and treatment (`~ f`) coding – levels without a matching column (such as a dropped reference level) are simply skipped.

Usage

```
dummy_groups(
  object,
  data,
  naming = function(feature, levels) paste0(feature, levels)
)
```

Arguments

object	(glex) Object of class glex.
data	(data.frame) The data before dummy encoding; its factor and character columns define the candidate groups. Other columns are ignored.
naming	(function(feature, levels)) Maps a factor name and its levels to the encoded column names. Defaults to the <code>model.matrix()</code> convention <code>paste0(feature, levels)</code> .

Details

If the data was encoded with a different scheme, pass `naming`: a function of (feature, levels) returning the encoded column names, e.g. `function(feature, levels) paste(feature, levels, sep = "_")`. It must reproduce the column names the model was trained with, i.e. those found in `object$x`.

Value

Named list of encoded column names, one element per matched factor, suitable as the `groups` argument of `group_components()`.

Examples

```
library(xgboost)
set.seed(1)
n <- 200
d <- data.frame(
  f = factor(sample(c("a", "b", "c"), n, replace = TRUE)),
  x1 = rnorm(n)
)
y <- c(a = 0, b = 2, c = -1)[d$f] + d$x1 + rnorm(n)
x <- model.matrix(~ f + x1 - 1, d)

xg <- xgboost(x, y, nrounds = 10, max_depth = 3, nthreads = 1)
gl <- glex(xg, x)

dummy_groups(gl, d)
grouped <- group_components(gl, dummy_groups(gl, d))
names(grouped$m)
```

glex

Global explanations for tree-based models.

Description

Global explanations for tree-based models by decomposing regression or classification functions into the sum of main components and interaction components of arbitrary order. Calculates SHAP values and q-interaction SHAP for all values of q for tree-based models such as xgboost.

Usage

```
glex(object, x, max_interaction = NULL, features = NULL, ...)
```

```
## Default S3 method:
```

```
glex(object, ...)
```

```
## S3 method for class 'rpf'
```

```
glex(object, x, max_interaction = NULL, features = NULL, ...)
```

```
## S3 method for class 'xgb.Booster'
```

```
glex(
  object,
  x,
  max_interaction = NULL,
  features = NULL,
  max_background_sample_size = NULL,
  weighting_method = "fastpd",
  ...
)
```

```
## S3 method for class 'ranger'
```

```
glex(
  object,
  x,
  max_interaction = NULL,
  features = NULL,
  max_background_sample_size = NULL,
  weighting_method = "fastpd",
  ...
)
```

Arguments

object	Model to be explained, either of class <code>xgb.Booster</code> or <code>rpf</code> .
x	Data to be explained. For <code>xgboost</code> models with categorical features (fit on a <code>data.frame</code> with factor columns), a <code>data.frame</code> whose factor columns have the same levels in the same order as the training data: the model stores category codes, not levels, so a mismatch cannot be detected beyond a missing level (this is the same contract as <code>predict()</code>). For <code>ranger</code> models, a <code>data.frame</code> with the same factor columns as the training data, encoded as <code>predict()</code> does under any respect <code>.unordered.factors</code> option. Otherwise a numeric matrix or <code>data.frame</code> .
max_interaction	(integer(1): NULL) Maximum interaction size to consider. Defaults to using all possible interactions available in the model. For <code>xgboost</code> , this defaults to the <code>max_depth</code> parameter of the model fit. If not set in <code>xgboost</code> , the default value of 6 is assumed.

features	Vector of column names in <code>x</code> to calculate components for. Default is <code>NULL</code> , i.e. all features are used.
...	Further arguments passed to methods. Arguments a method does not use are disregarded with a warning.
max_background_sample_size	The maximum number of background samples used for the FastPD algorithm, only used when <code>weighting_method = "fastpd"</code> . Defaults to <code>nrow(x)</code> .
weighting_method	Use either "path-dependent", "fastpd" (default), or "empirical". See References for details.

Details

For parallel execution using `xgboost` models, register a backend, e.g. with `doParallel::registerDoParallel()`.

The different weighting methods are described in detail in Liu et al. (2025). The default method is "fastpd" as it consistently estimates the correct partial dependence function.

Value

Decomposition of the regression or classification function. A list with elements:

- `shap`: SHAP values, derived from the functional decomposition as $\phi_j = \sum_{S \ni j} m_S / |S|$. This reconstruction is only valid if the decomposition is complete: if it is constrained (see `constrained`), the components no longer sum to the full model prediction and the SHAP efficiency property cannot hold, so `shap` is a scalar NA (with a warning) while `m` remains valid. For multiclass models, columns are class-specific like those of `m`. Note that `randomPlantedForest` models report a single intercept for all classes, so for multiclass models `intercept + rowSums(shap)` reconstructs the predicted class scores only approximately.
- `m`: Functional decomposition into all main and interaction components in the model, up to the degree specified by `max_interaction`. The variable names correspond to the original variable names, with `:` separating interaction terms as one would specify in a [formula](#) interface.
- `intercept`: Intercept term, the expected value of the prediction.
- `constrained`: Character vector naming the arguments that constrained the decomposition ("max_interaction", "features"), or `character(0)` if it is complete. Use `length(x$constrained) > 0` to check whether `shap` is valid. A constraint that only drops terms whose value is zero leaves the decomposition unchanged; `glex()` confirms this against the model's predictions, reports it with a message, and treats the result as complete.
- `remainder`: What the dropped terms are collectively worth, per observation: `prediction - (intercept + rowSums(m))`. Present exactly when the decomposition is constrained, and absent otherwise, so `intercept + rowSums(m) + remainder` reconstructs the prediction in either case. For multiclass `randomPlantedForest` models it is class-wise, mirroring `m`.

Like `m` and `shap`, it is on the scale that the model is decomposed on, which for `xgboost` is the **link** scale and not the response: for a `binary:logistic` model the reconstruction gives the margin, and `plogis(intercept + rowSums(m) + remainder)` gives the predicted probability. `ranger` probability forests and `randomPlantedForest` are decomposed on the response scale, where no such back-transformation is needed. Adding `remainder` to a probability is therefore never correct for `xgboost`.

References

Liu, J., Steensgaard, T., Wright, M. N., Pfister, N., & Hiabu, M. (2025). *Fast Estimation of Partial Dependence Functions using Trees*. Proceedings of the 42nd International Conference on Machine Learning, PMLR 267:39496-39534. [PMLR](#) | [arXiv:2410.13448](#)

Examples

```
# Random Planted Forest -----
library(randomPlantedForest)

rp <- rpf(mpg ~ ., data = mtcars[1:26, ], max_interaction = 2)

glex_rpf <- glex(rp, mtcars[27:32, ])
str(glex_rpf, list.len = 5)

# xgboost -----
library(xgboost)
x <- as.matrix(mtcars[, -1])
y <- mtcars$mpg
xg <- xgboost(x[1:26, ], y[1:26],
  max_depth = 4, learning_rate = .1,
  nrounds = 10, verbosity = 0, nthreads = 1
)
glex(xg, x[27:32, ])
glex(xg, mtcars[27:32, ])

## Not run:
# Parallel execution
doParallel::registerDoParallel()
glex(xg, x[27:32, ])

## End(Not run)

# ranger -----
library(ranger)
x <- as.matrix(mtcars[, -1])
y <- mtcars$mpg
rf <- ranger(
  x = x[1:26, ], y = y[1:26],
  num.trees = 5, max.depth = 3,
  node.stats = TRUE
)
glex(rf, x[27:32, ])
glex(rf, mtcars[27:32, ])

## Not run:
# Parallel execution
doParallel::registerDoParallel()
glex(rf, x[27:32, ])
```

```
## End(Not run)
```

glex_explain	<i>Explain a single prediction</i>
--------------	------------------------------------

Description

Plots the prediction components for a single observation, identified by the row number in the dataset used with `glex()`. Since the resulting plot can be quite busy due to potentially large amounts of elements, it is highly recommended to use `predictors`, `max_interaction`, or `threshold` to restrict the number of elements in the plot.

Usage

```
glex_explain(
  object,
  id,
  threshold = 0,
  max_interaction = NULL,
  predictors = NULL,
  class = NULL,
  barheight = 0.5
)
```

Arguments

<code>object</code>	Object of class <code>glex</code> containing prediction components and data to be explained.
<code>id</code>	(integer(1)) Row ID of the observation to be explained in <code>object\$x</code> .
<code>threshold</code>	(numeric(1): 0) Threshold to filter output by in case of many negligible effects.
<code>max_interaction</code>	(integer(1): NULL) Optionally filter plot to show terms up to the specified degree of interaction. Similar to <code>threshold</code> , all other terms will be aggregated under a "Remaining terms" label.
<code>predictors</code>	(character: NULL) Vector of column names in <code>\$x</code> to restrict plot to.
<code>class</code>	(character: NULL) For multiclass targets, specifies the target class to limit output.
<code>barheight</code>	(numeric(1): 0.5) Relative height of horizontal bars. Preferred value may depend on the number of vertical elements, hence it may be necessary to adjust this value as needed.

Value

A `ggplot` object.

See Also

Other Visualization functions: [autoplot.glex\(\)](#), [autoplot.glex_vi\(\)](#), [plot_pdp\(\)](#)

Examples

```
set.seed(1)
# Random Planted Forest -----
library(randomPlantedForest)

rp <- rpf(mpg ~ ., data = mtcars[1:26, ], max_interaction = 2)

glex_rpf <- glex(rp, mtcars[27:32, ])

glex_explain(glex_rpf, id = 3, predictors = "hp", threshold = 0.01)
```

glex_options

Package options for glex plots

Description

The color choices used across all glex visualization functions can be adjusted globally via [options\(\)](#):

Details

- `glex.palette(NULL)`: Diverging palette used to color continuous interaction effects in [plot_twoway_effects\(\)](#) and [plot_threeway_effects\(\)](#). The default NULL uses a blue/red gradient built from `glex.colors_sign`, matching the look of the Python shap and shapiq packages. Set to the name of a diverging [scico](#) palette (e.g. "vik", "roma") to use that instead.
- `glex.palette_discrete("Dark2")`: Discrete palette used to color categorical predictors in interaction plots. Accepts a vector of colors (used via [ggplot2::scale_color_manual\(\)](#)), the string "okabe-ito" (the colorblind-safe Okabe-Ito palette via [grDevices::palette.colors\(\)](#)), the name of a [scico](#) palette, or the name of an [RColorBrewer](#) palette.
- `glex.colors_sign(c("#008BFB", "#FF0051"))`: Two colors for negative and positive contributions in [glex_explain\(\)](#), also used as the endpoints of the default continuous gradient. The defaults follow the blue/red convention familiar from the Python shap and shapiq packages.
- `glex.color_line("#194155")`: Color for main effect lines and columns drawn by [autoplot\(\)](#) and [plot_pdp\(\)](#).

Examples

```
# Use a scico palette for continuous effects instead of the default gradient
options(glex.palette = "roma")

# Restore the default shap-style gradient
options(glex.palette = NULL)
```

```
# Categorical predictors: Okabe-Ito, a scico/brewer palette, or custom colors
options(glex.palette_discrete = "okabe-ito")
options(glex.palette_discrete = "batlow")
options(glex.palette_discrete = c("#E69F00", "#56B4E9", "#009E73"))
```

glex_vi

*Variable Importance for Main and Interaction Terms***Description**

Variable Importance for Main and Interaction Terms

Usage

```
glex_vi(object, ...)
```

Arguments

object	Object of class glex.
...	(Unused)

Details

The m reported here is the average absolute value of m as reported by `glex()`, aggregated by term:

$$m = \frac{1}{n} \sum_{i=1}^n |m_i|$$

In turn, `m_rel` rescales m by the average prediction of the model (m_0 , intercept as reported by `glex()`):

$$m_rel = \frac{m}{m_0}$$

Value

A `data.table::data.table()` with columns:

- `degree` (integer): Degree of interaction of the term, with 1 being main effects, 2 being 2-degree interactions etc.
- `term` (character): Model term, e.g. main effect `x1` or interaction term `x1:x2`, `x1:x3:x5` etc.
- `class` (factor): For multiclass targets only: The associated target class. Lists all classes in the target, not limited to the majority vote.
- `m` (numeric): Average absolute contribution of term, see Details.
- `m_rel` (numeric): m but relative to the average prediction (intercept in `glex()` output).

See Also[autoplot.glex_vi](#)**Examples**

```

set.seed(1)

# xgboost ----
library(xgboost)
x <- as.matrix(mtcars[, -1])
y <- mtcars$mpg
xg <- xgboost(x[1:26, ], y[1:26],
  max_depth = 4, learning_rate = .1,
  nrounds = 10, verbosity = 0, nthreads = 1
)
glex_xgb <- glex(xg, x[27:32, ])
vi_xgb <- glex_vi(glex_xgb)

library(ggplot2)
autoplot(vi_xgb)
autoplot(vi_xgb, by_degree = TRUE)

```

group_components	<i>Group features of a decomposition</i>
------------------	--

Description

Aggregates the terms of a decomposition such that a set of features is treated as a single feature, named after the group. All terms involving only members of a group are summed into the group's main effect, and terms mixing group members with other features become interactions of the group. Since the decomposition is additive, this regrouping is exact: components still sum to the model prediction, and SHAP values of a group are the sums of its members' SHAP values, preserving the efficiency property.

Usage

```
group_components(object, groups)
```

Arguments

object	(glex) Object of class glex.
groups	(list) Named list of character vectors of feature names in object\$x. Each vector becomes a single feature named after the list element, e.g. <code>list(f = c("fa", "fb", "fc"))</code> . Features may appear in at most one group; features not listed remain unchanged.

Details

The motivating use case is dummy-encoded categorical features in `xgboost`: the decomposition of a one-hot encoded factor spreads its effect across the dummies and their interactions (e.g. `fa:fb`), which are artifacts of the encoding rather than meaningful interactions. Grouping the dummies restores the factor as one feature, e.g. `groups = list(f = c("fa", "fb", "fc"))` turns `fa`, `fb`, `fc`, `fa:fb`, ... into `f`, and `fa:x1`, `fa:fb:x1`, ... into `f:x1`.

For the grouped columns of `x`, a factor is reconstructed when the group's columns form a dummy encoding (0/1 values, at most one active per row), with the active column's name as the level and "(base)" for rows with no active column (treatment coding). Otherwise the group's column in `x` is NA and plot functions cannot be used for that group's main effect.

Value

Object of class `glex` with grouped `m`, `shap` and `x`.

Examples

```
library(xgboost)
set.seed(1)
n <- 200
f <- factor(sample(c("a", "b", "c"), n, replace = TRUE))
x1 <- rnorm(n)
y <- c(a = 0, b = 2, c = -1)[f] + x1 + c(a = 1, b = 0, c = -1)[f] * x1 + rnorm(n)
x <- cbind(model.matrix(~ f - 1), x1)

xg <- xgboost(x, y, nrounds = 10, max_depth = 3, nthreads = 1)
gl <- glex(xg, x)
names(gl$m)

grouped <- group_components(gl, groups = list(f = c("fa", "fb", "fc")))
names(grouped$m)
all.equal(rowSums(grouped$m), rowSums(gl$m))
```

plot_pdp

Partial Dependence Plot

Description

A version of `plot_main_effect` with the intercept term (horizontal line) added, resulting in a partial dependence plot.

Usage

```
plot_pdp(object, predictor, rug_sides = "b", ...)
```

Arguments

object	Object of class glex .
predictor	(character(1)) predictor names, e.g. "x1" to plot main effect of x1.
rug_sides	(character(1): "b") Sides to plot rug (see ggplot2::geom_rug()) plot on for continuous predictors.. Default is "b" for both sides. Set to "none" to disable rug plot.
...	Used for future expansion.

Value

A ggplot2 object.

See Also

[plot_main_effect\(\)](#)

Other Visualization functions: [autoplot.glex\(\)](#), [autoplot.glex_vi\(\)](#), [glex_explain\(\)](#)

Examples

```
library(randomPlantedForest)

# introduce factor variables to show categorical feature handling
mtcars$cyl <- factor(mtcars$cyl)
mtcars$vs <- factor(mtcars$vs)

# Fit forest, get components
set.seed(12)
rpf_fit <- rpf(mpg ~ cyl + wt + hp + drat + vs, data = mtcars, ntrees = 25, max_interaction = 3)
components <- glex(rpf_fit, mtcars)

plot_pdp(components, "wt")
plot_pdp(components, "cyl")
```

print.glex

Print glex objects

Description

This is implemented mainly to avoid flooding the console in cases where the glex object uses many terms, which leads to a large amount of column names of \$m being printed to the console. This function wraps [str\(\)](#) with a truncated output for a more compact representation.

Usage

```
## S3 method for class 'glex'
print(x, ...)
```

Arguments

x	Object to print.
...	(Unused)

Examples

```
# Random Planted Forest ----
library(randomPlantedForest)
rp <- rpf(mpg ~ hp + wt + drat, data = mtcars[1:26, ], max_interaction = 2)

glex(rp, mtcars[27:32, ])
```

subset_components	<i>Subset components</i>
-------------------	--------------------------

Description

Subset components

Usage

```
subset_components(components, term)

subset_component_names(components, term)
```

Arguments

components	An object of class glex.
term	(character(1)) A main term name to subset by, e.g. "x1".

Value

- subset_components(): An object of class glex.
- subset_component_names(): A character vector.

Examples

```
library(randomPlantedForest)

# introduce factor variables to show categorical feature handling
mtcars$cyl <- factor(mtcars$cyl)
mtcars$vs <- factor(mtcars$vs)

# Fit forest, get components
set.seed(12)
rpfit <- rpf(mpg ~ cyl + wt + hp + drat + vs, data = mtcars, ntrees = 25, max_interaction = 3)
components <- glex(rpfit, mtcars)
```

```
# Get component object with only "hp" and its interactions
subset_components(components, "hp")

subset_component_names(components, "hp")
```

theme_glex

A ggplot2 theme for glex plots

Description

This is a slight variation of `ggplot2::theme_minimal()` with increased font size.

Usage

```
theme_glex(
  base_size = 13,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  grid_x = TRUE,
  grid_y = FALSE
)
```

Arguments

<code>base_size</code>	(13) Base font size, given in pts.
<code>base_family</code>	("") Base font family
<code>base_line_size</code> , <code>base_rect_size</code>	(base_size / 22) Base size for line and rect elements
<code>grid_x</code>	(TRUE) Display horizontal grid lines?
<code>grid_y</code>	(FALSE) Display vertical grid lines?

Value

A ggplot2 theme object

Examples

```
library(ggplot2)

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_glex()
```

Index

* Visualization functions

autoplot.glex, 2
autoplot.glex_vi, 4
glex_explain, 10
plot_pdp, 14

* datasets

bike, 5

autoplot.glex, 2
autoplot.glex(), 4, 11, 15
autoplot.glex_vi, 4, 13
autoplot.glex_vi(), 3, 11, 15

bike, 5

data.table::data.table(), 5, 12
dummy_groups, 5

formula, 8

ggplot, 4, 10
ggplot2::geom_rug(), 3, 15
ggplot2::scale_color_manual(), 11
ggplot2::theme_minimal(), 17
glex, 2, 6, 10, 15
glex_explain, 10
glex_explain(), 3, 4, 11, 15
glex_options, 11
glex_vi, 4, 12
glex_vi(), 4
grDevices::palette.colors(), 11
group_components, 13
group_components(), 5, 6

model.matrix(), 5, 6

options(), 11

plot_main_effect, 14
plot_main_effect (autoplot.glex), 2
plot_main_effect(), 15

plot_pdp, 14
plot_pdp(), 3, 4, 11
plot_threeway_effects (autoplot.glex), 2
plot_threeway_effects(), 11
plot_twoway_effects (autoplot.glex), 2
plot_twoway_effects(), 11
print.glex, 15

ranger, 7
RColorBrewer, 11

scico, 11
str(), 15
subset_component_names
 (subset_components), 16
subset_components, 16

theme_glex, 17

xgboost, 7