Package 'ingredients'

April 9, 2019

Title Effects and Importances of Model Ingredients

Version 0.3.1

Description Collection of tools for assessment of feature importance and feature effects.
Key functions are:
feature_importance() for assessment of global level feature importance,
ceteris_paribus() for calculation of the what-if plots,
partial_dependency() for partial dependency plots,
conditional_dependency() for conditional dependency plots,
accumulated_dependency() for accumulated local effects plots,
aggregate_profiles() and cluster_profiles() for aggregation of ceteris paribus profiles,
theme_drwhy() with a 'ggplot2' skin for all plots,
generic print() and plot() for better usability of selected explainers.
The package 'ingredients' is a part of the 'DrWhy.AI' universe (Biecek 2018) <arXiv:1806.08915>.

Depends R (>= 3.0)

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports DALEX, ggplot2

Suggests gbm, gower, randomForest, titanic, xgboost, testthat, dplyr,
r2d3, ggpubr, jsonlite

URL https://ModelOriented.github.io/ingredients/

BugReports https://github.com/ModelOriented/ingredients/issues

NeedsCompilation no

Author Przemyslaw Biecek [aut, cre] (<https://orcid.org/0000-0001-8423-1823>),
Hubert Baniecki [ctb]

Maintainer Przemyslaw Biecek <przemyslaw.biecek@gmail.com>

Repository CRAN

Date/Publication 2019-04-09 12:10:08 UTC

R topics documented:

`accumulated_dependency` *Accumulated Local Effects Profiles aka ALEPlots*

Description

Accumulated Local Effects Profiles accumulate local changes in Ceteris Paribus Profiles. Function 'accumulated_dependency' calls 'ceteris_paribus' and then 'aggregate_profiles'.

Usage

```
accumulated_dependency(x, ...)

## S3 method for class 'explainer'
accumulated_dependency(x, variables = NULL,
  N = 500, variable_splits = NULL, grid_points = 101, ...)

## Default S3 method:
```

```
accumulated_dependency(x, data,
  predict_function = predict, label = class(x)[1], variables = NULL,
  grid_points = grid_points, variable_splits = variable_splits,
  N = 500, ...)

## S3 method for class 'ceteris_paribus_explainer'
accumulated_dependency(x, ...,
  variables = NULL)
```

Arguments

x	a model to be explained, or an explainer created with function 'DALEX::explain()' or object of the class 'ceteris_paribus_explainer'.
...	other parameters
variables	names of variables for which profiles shall be calculated. Will be passed to 'calculate_variable_splits()'. If NULL then all variables from the validation data will be used.
N	number of observations used for calculation of partial dependency profiles. By default, 500 observations will be chosen randomly.
variable_splits	named list of splits for variables, in most cases created with 'calculate_variable_splits()'. If NULL then it will be calculated based on validation data avaliable in the 'explainer'.
grid_points	number of points for profile. Will be passed to 'calculate_variable_splits()'.
data	validation dataset Will be extracted from 'x' if it's an explainer
predict_function	predict function Will be extracted from 'x' if it's an explainer
label	name of the model. By default it's extracted from the 'class' attribute of the model

Details

Find more detailes in the Accumulated Local Dependency Chapter.

Value

an 'aggregated_profiles_explainer' geom

References

ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence (PD) Plots https://cran.r-project.org/package=ALEPlot, Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes")
pdp_glm <- accumulated_dependency(explain_titanic_glm, N = 50, variables = c("age", "fare"))
head(pdp_glm)
plot(pdp_glm)


library("randomForest")
 model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                     fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                               data = titanic[,-9],
                               y = titanic$survived)

pdp_rf <- accumulated_dependency(explain_titanic_rf)
plot(pdp_rf)
```

`aggregate_profiles` *Aggregate Ceteris Paribus Profiles*

Description

The function 'aggregate_profiles' calculates an aggregate of ceteris paribus profiles. It can be: Partial Dependency Profile (average across Ceteris Paribus Profiles), Conditional Dependency Profile (local weighted average across Ceteris Paribus Profiles) or Accumulated Local Dependency Profile (cummulated average local changes in Ceteris Paribus Profiles).

Usage

```
aggregate_profiles(x, ..., only_numerical = TRUE, groups = NULL,
  type = "partial", variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function 'ceteris_paribus()'
`...`	other explainers that shall be plotted together
`only_numerical`	a logical. If TRUE then only numerical variables will be plotted. If FALSE then only categorical variables will be plotted.

groups	a variable name that will be usef for grouping. By default 'NULL' which means that no groups shall be calculated
type	either 'partial'/'conditional'/'accumulated' for parital dependence, conditional profiles of accumulated local effects
variables	if not NULL then only 'variables' will be presented

Value

an 'aggregated_profiles_explainer' layer

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")

library("randomForest")
 model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived)

selected_passangers <- select_sample(titanic, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
head(cp_rf)

pdp_rf_p <- aggregate_profiles(cp_rf, variables = "age", type = "partial")
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_c <- aggregate_profiles(cp_rf, variables = "age", type = "conditional")
pdp_rf_c$`_label_` <- "RF_conditional"
pdp_rf_a <- aggregate_profiles(cp_rf, variables = "age", type = "accumulated")
pdp_rf_a$`_label_` <- "RF_accumulated"
plot(pdp_rf_p, pdp_rf_c, pdp_rf_a, color = "_label_")

pdp_rf <- aggregate_profiles(cp_rf, variables = "age",
                             groups = "gender")
head(pdp_rf)
plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_aggreagated_profiles(pdp_rf, size = 3, color = "_label_")
```

`calculate_oscillations` *Calculate Oscillations for Ceteris Paribus Explainer*

Description

Oscillations are proxies for local feature importance at the instance level.

Usage

```
calculate_oscillations(x, sort = TRUE, ...)
```

Arguments

`x`	a ceteris_paribus explainer produced with the 'ceteris_paribus()' function
`sort`	a logical value. If TRUE then rows are sorted along the oscillations
`...`	other arguments

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic_small[,-9],
                           y = titanic_small$survived == "yes")
cp_rf <- ceteris_paribus(explain_titanic_glm, titanic_small[1, ])
calculate_oscillations(cp_rf)


library("randomForest")
set.seed(59)

apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
      no.rooms + district, data = apartments)

explainer_rf <- explain(apartments_rf_model,
      data = apartmentsTest, y = apartmentsTest$m2.price)

apartment <- apartmentsTest[1,]
```

```
cp_rf <- ceteris_paribus(explainer_rf, apartment)
calculate_oscillations(cp_rf)
```

`calculate_variable_profile` *Internal Function for Individual Variable Profiles*

Description

This function calculates individual variable profiles (ceteris paribus profiles), i.e. series of predictions from a model calculated for observations with altered single coordinate.

Usage

```
calculate_variable_profile(data, variable_splits, model,
  predict_function = predict, ...)
```

Arguments

`data`	set of observations. Profile will be calculated for every observation (every row)
`variable_splits`	named list of vectors. Elements of the list are vectors with points in which profiles should be calculated. See an example for more details.
`model`	a model that will be passed to the `predict_function`
`predict_function`	function that takes data and model and returns numeric predictions. Note that the ... arguments will be passed to this function.
`...`	other parameters that will be passed to the `predict_function`

Details

Note that `calculate_variable_profile` function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

a data frame with profiles for selected variables and selected observations

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")

library("randomForest")
set.seed(59)
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
                                      no.rooms + district, data = apartments)
vars <- c("construction.year", "surface", "floor", "no.rooms", "district")
variable_splits <- calculate_variable_split(apartments, vars)
new_apartment <- apartmentsTest[1:10, ]
profiles <- calculate_variable_profile(new_apartment, variable_splits,
                                apartments_rf_model)
head(profiles)

# only subset of observations
small_apartments <- select_sample(apartmentsTest, n = 10)
small_apartments
small_profiles <- calculate_variable_profile(small_apartments, variable_splits,
                                apartments_rf_model)
head(small_profiles)

# neighbors for a selected observation
new_apartment <- apartments[1, 2:6]
small_apartments <- select_neighbours(apartmentsTest, new_apartment, n = 10)
small_apartments
small_profiles <- calculate_variable_profile(small_apartments, variable_splits,
                                apartments_rf_model)
head(new_apartment)
head(small_profiles)
```

---

calculate_variable_split

Internal Function for Split Points for Selected Variables

Description

This function calculate candidate splits for each selected variable. For numerical variables splits are calculated as percentiles (in general uniform quantiles of the length grid_points). For all other variables splits are calculated as unique values.

Usage

```
calculate_variable_split(data, variables = colnames(data),
  grid_points = 101)
```

Arguments

data	validation dataset. Is used to determine distribution of observations.
variables	names of variables for which splits shall be calculated
grid_points	number of points used for response path

Details

Note that calculate_variable_split function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

A named list with splits for selected variables

Examples

```
library("DALEX")
 ## Not run:
library("randomForest")
set.seed(59)
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
                                      no.rooms + district, data = apartments)
vars <- c("construction.year", "surface", "floor", "no.rooms", "district")
calculate_variable_split(apartments, vars)

## End(Not run)
```

---

ceteris_paribus *Ceteris Paribus Profiles aka Individual Variable Profiles*

Description

This explainer works for individual observations. For each observation it calculates Ceteris Paribus Profiles for selected variables. Such profiles can be used to hypothesize about model results if selected variable is changed. For this reason it is also called 'What-If Profiles'.

Usage

```
ceteris_paribus(x, ...)

## S3 method for class 'explainer'
ceteris_paribus(x, new_observation, y = NULL,
  variables = NULL, variable_splits = NULL, grid_points = 101, ...)

## Default S3 method:
ceteris_paribus(x, data, predict_function = predict,
  new_observation, y = NULL, variables = NULL,
  variable_splits = NULL, grid_points = 101, label = class(x)[1],
  ...)
```

Arguments

x	a model to be explained, or an explainer created with the 'DALEX::explain()' function.
...	other parameters
new_observation	a new observation with columns that corresponds to variables used in the model
y	true labels for 'new_observation'. If specified then will be added to ceteris paribus plots.
variables	names of variables for which profiles shall be calculated. Will be passed to 'calculate_variable_splits()'. If NULL then all variables from the validation data will be used.
variable_splits	named list of splits for variables, in most cases created with 'calculate_variable_splits()'. If NULL then it will be calculated based on validation data avaliable in the 'explainer'.
grid_points	number of points for profile. Will be passed to 'calculate_variable_splits()'.
data	validation dataset. It will be extracted from 'x' if it's an explainer
predict_function	predict function. It will be extracted from 'x' if it's an explainer
label	name of the model. By default it's extracted from the 'class' attribute of the model

Details

Find more detailes in Ceteris Paribus Chapter.

Value

An object of the class 'ceteris_paribus_explainer'. It's a data frame with calculated average responses.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                      data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")
```

```
cp_rf <- ceteris_paribus(explain_titanic_glm, titanic[1,])
cp_rf
plot(cp_rf, variables = "age")


 library("randomForest")
 model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived,
                           label = "Random Forest v7")

# select few passangers
selected_passangers <- select_sample(titanic, n = 20)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red")
```

`ceteris_paribus_2d` *Ceteris Paribus 2D Plot*

Description

This function calculates ceteris paribus profiles for grid of values spanned by two variables. It may be useful to identify or present interactions between two variables.

Usage

```
ceteris_paribus_2d(explainer, observation, grid_points = 101,
  variables = NULL)
```

Arguments

`explainer`	a model to be explained, preprocessed by the 'DALEX::explain' function
`observation`	a new observarvation for which predictions need to be explained
`grid_points`	number of points used for response path. Will be used for both variables
`variables`	if specified, then only these variables will be explained

Value

An object of the class 'ceteris_paribus_2d_explainer'. It's a data frame with calculated average responses.

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes")
cp_rf <- ceteris_paribus_2d(explain_titanic_glm, titanic[1,])
head(cp_rf)
plot(cp_rf)

library("randomForest")
set.seed(59)

apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
      no.rooms + district, data = apartments)

explainer_rf <- explain(apartments_rf_model,
      data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)

new_apartment <- apartmentsTest[1, ]
new_apartment

wi_rf_2d <- ceteris_paribus_2d(explainer_rf, observation = new_apartment,
         variables = c("surface", "floor", "no.rooms"))
head(wi_rf_2d)
plot(wi_rf_2d)
```

`cluster_profiles` *Cluster Ceteris Paribus Profiles*

Description

Function 'cluster_profiles' calculates aggregates of ceteris paribus profiles based on hierarchical clustering.

Usage

```
cluster_profiles(x, ..., aggregate_function = mean,
  only_numerical = TRUE, center = FALSE, k = 3, variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function 'ceteris_paribus()'
`...`	other explainers that shall be plotted together

aggregate_function	a function for profile aggregation. By default it's 'mean'
only_numerical	a logical. If TRUE then only numerical variables will be plotted. If FALSE then only categorical variables will be plotted.
center	shall profiles be centered before clustering
k	number of clusters for the hclust function
variables	if not NULL then only 'variables' will be presented

Details

Find more detailes in the Clustering Profiles Chapter.

Value

a 'aggregated_profiles_explainer' layer

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
titanic <- na.omit(titanic)
selected_passangers <- select_sample(titanic, n = 100)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")
cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)
clust_rf <- cluster_profiles(cp_rf, k = 3, variables = "age")
plot(clust_rf)


 library("randomForest")
 model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived == "yes",
                           label = "Random Forest v7")

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
```

```
head(pdp_rf)
clust_rf <- cluster_profiles(cp_rf, k = 3, variables = "age")
head(clust_rf)

plot(clust_rf, color = "_label_") +
  show_aggreagated_profiles(pdp_rf, color = "black", size = 3)

plot(cp_rf, color = "grey", variables = "age") +
  show_aggreagated_profiles(clust_rf, color = "_label_", size = 2)

clust_rf <- cluster_profiles(cp_rf, k = 3, center = TRUE, variables = "age")
head(clust_rf)
```

`conditional_dependency` *Conditional Dependency Profiles*

Description

Conditional Dependency Profiles (aka Local Profiles) average localy Ceteris Paribus Profiles. Function 'conditional_dependency' calls 'ceteris_paribus' and then 'aggregate_profiles'.

Usage

```
conditional_dependency(x, ...)

## S3 method for class 'explainer'
conditional_dependency(x, variables = NULL,
  N = 500, variable_splits = NULL, grid_points = 101, ...)

## Default S3 method:
conditional_dependency(x, data,
  predict_function = predict, label = class(x)[1], variables = NULL,
  grid_points = grid_points, variable_splits = variable_splits,
  N = 500, ...)

## S3 method for class 'ceteris_paribus_explainer'
conditional_dependency(x, ...,
  variables = NULL)

local_dependency(x, ...)
```

Arguments

`x`	a model to be explained, or an explainer created with function 'DALEX::explain()' or object of the class 'ceteris_paribus_explainer'.
`...`	other parameters

variables	names of variables for which profiles shall be calculated. Will be passed to 'calculate_variable_splits()'. If NULL then all variables from the validation data will be used.
N	number of observations used for calculation of partial dependency profiles. By default 500.
variable_splits	named list of splits for variables, in most cases created with 'calculate_variable_splits()'. If NULL then it will be calculated based on validation data avaliable in the 'explainer'.
grid_points	number of points for profile. Will be passed to 'calculate_variable_splits()'.
data	validation dataset, will be extracted from 'x' if it's an explainer
predict_function	predict function, will be extracted from 'x' if it's an explainer
label	name of the model. By default it's extracted from the 'class' attribute of the model

Details

Find more detailes in Local Dependency Profiles Chapter.

Value

an 'aggregated_profile_explainer' layer

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes")

pdp_rf <- conditional_dependency(explain_titanic_glm, N = 50)
plot(pdp_rf)


library("titanic")
library("randomForest")

titanic_small <- titanic_train[,c("Survived", "Pclass", "Sex", "Age",
```

```
                                    "SibSp", "Parch", "Fare", "Embarked")]
titanic_small$Survived <- factor(titanic_small$Survived)
titanic_small$Sex <- factor(titanic_small$Sex)
titanic_small$Embarked <- factor(titanic_small$Embarked)
titanic_small <- na.omit(titanic_small)
rf_model <- randomForest(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,
                         data = titanic_small)
explainer_rf <- explain(rf_model, data = titanic_small,
                        y = titanic_small$Survived == "1", label = "RF")

pdp_rf <- conditional_dependency(explainer_rf)
plot(pdp_rf)
```

`feature_importance` *Feature Importance Plots*

Description

This function calculates variable importance based on the drop in the Loss function after single-variable-perturbations. For this reason it is also called the Variable Dropout Plot.

Usage

```
feature_importance(x, ...)

## S3 method for class 'explainer'
feature_importance(x,
  loss_function = loss_root_mean_square, ..., type = "raw",
  n_sample = NULL)

## Default S3 method:
feature_importance(x, data, y, predict_function,
  loss_function = loss_root_mean_square, ..., label = class(x)[1],
  type = "raw", n_sample = NULL)
```

Arguments

`x`	a model to be explained, or an explainer created with function 'DALEX::explain()'.
`...`	other parameters
`loss_function`	a function thet will be used to assess variable importance
`type`	character, type of transformation that should be applied for dropout loss. 'raw' results raw drop lossess, 'ratio' returns `drop_loss/drop_loss_full_model` while 'difference' returns `drop_loss - drop_loss_full_model`
`n_sample`	number of observations that should be sampled for calculation of variable importance. If NULL then variable importance will be calculated on whole dataset (no sampling).

`data`	validation dataset, will be extracted from 'x' if it's an explainer
`y`	true labels for 'data', will be extracted from 'x' if it's an explainer
`predict_function`	predict function, will be extracted from 'x' if it's an explainer
`label`	name of the model. By default it's extracted from the 'class' attribute of the model

Details

Find more detailes in the Feature Importance Chapter.

Value

An object of the class 'feature_importance'. It's a data frame with calculated average response.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")

vd_rf <- feature_importance(explain_titanic_glm)
plot(vd_rf)


library("randomForest")

 titanic <- na.omit(titanic)
 model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")

vd_rf <- feature_importance(explain_titanic_rf)
plot(vd_rf)

HR_rf_model <- randomForest(status~., data = HR, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR, y = HR$status)
vd_rf <- feature_importance(explainer_rf, type = "raw",
                            loss_function = loss_cross_entropy)
```

```
head(vd_rf)
plot(vd_rf)

HR_glm_model <- glm(status == "fired"~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired")
vd_glm <- feature_importance(explainer_glm, type = "raw",
                        loss_function = loss_root_mean_square)
head(vd_glm)
plot(vd_glm)

library("xgboost")
model_martix_train <- model.matrix(status == "fired" ~ . -1, HR)
data_train <- xgb.DMatrix(model_martix_train, label = HR$status == "fired")
param <- list(max_depth = 2, eta = 1, silent = 1, nthread = 2,
              objective = "binary:logistic", eval_metric = "auc")
HR_xgb_model <- xgb.train(param, data_train, nrounds = 50)
explainer_xgb <- explain(HR_xgb_model, data = model_martix_train,
                     y = HR$status == "fired", label = "xgboost")
vd_xgb <- feature_importance(explainer_xgb, type = "raw")
head(vd_xgb)
plot(vd_xgb, vd_glm)
```

`partial_dependency` *Partial Dependency Profiles*

Description

Partial Dependency Profiles are averages from Ceteris Paribus Profiles. Function 'partial_dependency' calls 'ceteris_paribus' and then 'aggregate_profiles'.

Usage

```
partial_dependency(x, ...)

## S3 method for class 'explainer'
partial_dependency(x, variables = NULL, N = 500,
  variable_splits = NULL, grid_points = 101, ...)

## Default S3 method:
partial_dependency(x, data, predict_function = predict,
  label = class(x)[1], variables = NULL, grid_points = grid_points,
  variable_splits = variable_splits, N = 500, ...)

## S3 method for class 'ceteris_paribus_explainer'
partial_dependency(x, ...,
  variables = NULL)
```

Arguments

x	a model to be explained, or an explainer created with function 'DALEX::explain()' or object of the class 'ceteris_paribus_explainer'.
...	other parameters
variables	names of variables for which profiles shall be calculated. Will be passed to 'calculate_variable_splits()'. If NULL then all variables from the validation data will be used.
N	number of observations used for calculation of partial dependency profiles. By default 500.
variable_splits	named list of splits for variables, in most cases created with 'calculate_variable_splits()'. If NULL then it will be calculated based on validation data avaliable in the 'explainer'.
grid_points	number of points for profile. Will be passed to 'calculate_variable_splits()'.
data	validation dataset, will be extracted from 'x' if it's an explainer
predict_function	predict function, will be extracted from 'x' if it's an explainer
label	name of the model. By default it's extracted from the 'class' attribute of the model

Details

Find more detailes in the Partial Dependence Profiles Chapter.

Value

an 'aggregated_profiles_explainer' layer

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")

pdp_rf <- partial_dependency(explain_titanic_glm, N = 50)
plot(pdp_rf)
```

```
 library("randomForest")
 model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived,
                           label = "Random Forest v7")

pdp_rf <- partial_dependency(explain_titanic_rf, variables = "age")
plot(pdp_rf)

pdp_rf <- partial_dependency(explain_titanic_rf)
plot(pdp_rf)
```

`plot.aggregated_profiles_explainer` *Adds a Layer with Aggregated Profiles*

Description

Function 'show_aggreagated_profiles' adds a layer to a plot created with 'plot.ceteris_paribus_explainer'.

Usage

```
## S3 method for class 'aggregated_profiles_explainer'
plot(x, ..., size = 1,
  alpha = 1, color = "#371ea3", facet_ncol = NULL,
  variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function 'ceteris_paribus()'
`...`	other explainers that shall be plotted together
`size`	a numeric. Size of lines to be plotted
`alpha`	a numeric between 0 and 1. Opacity of lines
`color`	a character. Either name of a color or name of a variable that should be used for coloring
`facet_ncol`	number of columns for the 'facet_wrap()'
`variables`	if not NULL then only 'variables' will be presented

Value

a ggplot2 layer

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")

pdp_rf_p <- partial_dependency(explain_titanic_glm, N = 50)
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_l <- conditional_dependency(explain_titanic_glm, N = 50)
pdp_rf_l$`_label_` <- "RF_local"
pdp_rf_a<- accumulated_dependency(explain_titanic_glm, N = 50)
pdp_rf_a$`_label_` <- "RF_accumulated"
head(pdp_rf_p)
plot(pdp_rf_p, pdp_rf_l, pdp_rf_a, color = "_label_")


 library("randomForest")
 titanic <- na.omit(titanic)
 model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived == "yes",
                           label = "Random Forest v7")

selected_passangers <- select_sample(titanic, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf_p <- aggregate_profiles(cp_rf, variables = "age", type = "partial")
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_c <- aggregate_profiles(cp_rf, variables = "age", type = "conditional")
pdp_rf_c$`_label_` <- "RF_conditional"
pdp_rf_a <- aggregate_profiles(cp_rf, variables = "age", type = "accumulated")
pdp_rf_a$`_label_` <- "RF_accumulated"
head(pdp_rf_p)
plot(pdp_rf_p, pdp_rf_c, pdp_rf_a, color = "_label_")

plot(cp_rf, variables = "age") +
show_observations(cp_rf, variables = "age") +
```

```
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_aggreagated_profiles(pdp_rf, size = 2)

plot(pdp_rf, variables = "age")
```

`plot.ceteris_paribus_2d_explainer` *Plot Ceteris Paribus 2D Explanations*

Description

Function 'ceteris_paribus_2d_explainer' plots What-If Plots for a single prediction / observation.

Usage

```
## S3 method for class 'ceteris_paribus_2d_explainer'
plot(x, ..., split_ncol = NULL,
  add_raster = TRUE, add_contour = TRUE, bins = 3,
  add_observation = TRUE, pch = "+", size = 6)
```

Arguments

`x`	a ceteris paribus explainer produced with the 'ceteris_paribus_2d' function
`...`	currently will be ignored
`split_ncol`	number of columns for the 'facet_wrap'
`add_raster`	if TRUE then 'geom_raster' will be added to present levels with diverging colors
`add_contour`	if TRUE then 'geom_contour' will be added to present contours
`bins`	number of contours to be added
`add_observation`	if TRUE then 'geom_point' will be added to present observation that is explained
`pch`	character, symbol used to plot observations
`size`	numeric, size of individual datapoints

Value

a ggplot2 object

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")

library("randomForest")
set.seed(59)

apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
      no.rooms + district, data = apartments)

explainer_rf <- explain(apartments_rf_model,
      data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)

new_apartment <- apartmentsTest[1, ]
new_apartment

wi_rf_2d <- ceteris_paribus_2d(explainer_rf, observation = new_apartment)
head(wi_rf_2d)

plot(wi_rf_2d)
plot(wi_rf_2d, add_contour = FALSE)
plot(wi_rf_2d, add_observation = FALSE)
plot(wi_rf_2d, add_raster = FALSE)

# HR data
model <- randomForest(status ~ gender + age + hours + evaluation + salary, data = HR)
pred1 <- function(m, x)   predict(m, x, type = "prob")[,1]
explainer_rf_fired <- explain(model, data = HR[,1:5],
   y = HR$status == "fired",
   predict_function = pred1, label = "fired")

new_emp <- HR[1, ]
new_emp

wi_rf_2d <- ceteris_paribus_2d(explainer_rf_fired, observation = new_emp)
head(wi_rf_2d)

plot(wi_rf_2d)
```

---

plot.ceteris_paribus_explainer

Plots Ceteris Paribus Profiles

Description

Function 'plot.ceteris_paribus_explainer' plots Individual Variable Profiles for selected observations. Various parameters help to decide what should be plotted, profiles, aggregated profiles, points or rugs.

Usage

```
## S3 method for class 'ceteris_paribus_explainer'
plot(x, ..., size = 1, alpha = 1,
  color = "#46bac2", only_numerical = TRUE, facet_ncol = NULL,
  variables = NULL)
```

Arguments

x	a ceteris paribus explainer produced with function 'ceteris_paribus()'
...	other explainers that shall be plotted together
size	a numeric. Size of lines to be plotted
alpha	a numeric between 0 and 1. Opacity of lines
color	a character. Either name of a color or name of a variable that should be used for coloring
only_numerical	a logical. If TRUE then only numerical variables will be plotted. If FALSE then only categorical variables will be plotted.
facet_ncol	number of columns for the 'facet_wrap()'
variables	if not NULL then only 'variables' will be presented

Details

Find more detailes in Ceteris Paribus Chapter.

Value

a ggplot2 object

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes")
cp_rf <- ceteris_paribus(explain_titanic_glm, titanic[1,])
cp_rf
plot(cp_rf, variables = "age")
```

```
library("randomForest")
model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + class + embarked +
                                   fare + sibsp + parch,  data = titanic)
model_titanic_rf

explain_titanic_rf <- explain(model_titanic_rf,
                          data = titanic[,-9],
                          y = titanic$survived == "yes",
                          label = "Random Forest v7")

selected_passangers <- select_sample(titanic, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

plot(cp_rf, variables = "age") +
show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red")
```

`plot.ceteris_paribus_oscillations` *Plot Ceteris Paribus Oscillations*

Description

Function 'plot.ceteris_paribus_oscillations' plots local variable importance plots calculated as oscillations in the Ceteris Paribus Profiles.

Usage

```
## S3 method for class 'ceteris_paribus_oscillations'
plot(x, ..., bar_width = 10)
```

Arguments

`x`	a ceteris paribus oscillation explainer produced with function 'calculate_oscillations()'
`...`	other explainers that shall be plotted together
`bar_width`	width of bars. By default 10

Value

a ggplot2 object

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")

library("randomForest")
set.seed(59)

apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
      no.rooms + district, data = apartments)

explainer_rf <- explain(apartments_rf_model,
      data = apartmentsTest, y = apartmentsTest$m2.price)

apartment <- apartmentsTest[1:2,]

cp_rf <- ceteris_paribus(explainer_rf, apartment)
plot(cp_rf, color = "_ids_")

vips <- calculate_oscillations(cp_rf)
vips
plot(vips)
```

---

plot.feature_importance_explainer

Plots Variable Importance

Description

Function `plot.feature_importance_explainer` plots variable importance calculated as changes in the loss function after variable drops. It uses output from `feature_importance` function that corresponds to permutation based measure of variable importance. Variables are sorted in the same order in all panels. The order depends on the average drop out loss. In different panels variable contributions may not look like sorted if variable importance is different in different in different models.

Usage

```
## S3 method for class 'feature_importance_explainer'
plot(x, ..., max_vars = NULL,
  bar_width = 10)
```

Arguments

`x`	a variable dropout explainer produced with the 'feature_importance' function
`...`	other explainers that shall be plotted together
`max_vars`	maximum number of variables that shall be presented for for each model. By default NULL what means all variables
`bar_width`	width of bars. By default 10

Details

Find more details in the Feature Importance Chapter.

Value

a ggplot2 object

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")
vd_rf <- feature_importance(explain_titanic_glm)
plot(vd_rf)


library("randomForest")

 model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")

vd_rf <- feature_importance(explain_titanic_rf)
plot(vd_rf)

HR_rf_model <- randomForest(status~., data = HR, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR, y = HR$status)
vd_rf <- feature_importance(explainer_rf, type = "raw",
                            loss_function = loss_cross_entropy)
head(vd_rf)
plot(vd_rf)

HR_glm_model <- glm(status == "fired"~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired")
vd_glm <- feature_importance(explainer_glm, type = "raw",
                         loss_function = loss_root_mean_square)
head(vd_glm)
plot(vd_glm)
```

```
library("xgboost")
model_martix_train <- model.matrix(status == "fired" ~ . -1, HR)
data_train <- xgb.DMatrix(model_martix_train, label = HR$status == "fired")
param <- list(max_depth = 2, eta = 1, silent = 1, nthread = 2,
              objective = "binary:logistic", eval_metric = "auc")
HR_xgb_model <- xgb.train(param, data_train, nrounds = 50)
explainer_xgb <- explain(HR_xgb_model, data = model_martix_train,
                     y = HR$status == "fired", label = "xgboost")
vd_xgb <- feature_importance(explainer_xgb, type = "raw")
head(vd_xgb)

plot(vd_glm, vd_xgb, bar_width = 5)
```

---

plotD3 *Plot Feature Importance Objects in D3 with r2d3 Package.*

Description

Function `plotD3.feature_importance_explainer` plots dropouts for variables used in the model. It uses output from `feature_importance` function that corresponds to permutation based measure of feature importance. Variables are sorted in the same order in all panels. The order depends on the average drop out loss. In different panels variable contributions may not look like sorted if variable importance is different in different models.

Usage

```
plotD3(x, ...)

## S3 method for class 'feature_importance_explainer'
plotD3(x, ..., max_vars = NULL,
  bar_width = 12, split = "model", scale_height = FALSE,
  margin = 0.15)
```

Arguments

x	a feature importance explainer produced with the 'feature_importance' function
...	other explainers that shall be plotted together
max_vars	maximum number of variables that shall be presented for for each model. By default NULL what means all variables
bar_width	width of bars in px. By default 12px
split	either "model" or "feature" determines the plot layout
scale_height	should the height of plot scale with window size? By default it's FALSE
margin	extend x axis domain range to adjust the plot. Usually value between 0.1 and 0.3, by default it's 0.15

Value

an ‘r2d3‘ object.

Examples

```
## Not run:
library("DALEX")
library("ingredients")
library("caret")

rf_model <- train(m2.price~., data = apartments, method="rf", ntree = 100)
explainer_rf <- explain(rf_model, data = apartments_test[,2:6],
                        y = apartments_test$m2.price, label="rf")
fi_rf <- feature_importance(explainer_rf, loss_function = loss_root_mean_square)

head(fi_rf)
plotD3(fi_rf)

svm_model <- train(m2.price~., data = apartments, method="svmLinear")
explainer_svm <- explain(svm_model, data = apartments_test[,2:6],
                        y = apartments_test$m2.price, label="svm")
fi_svm <- feature_importance(explainer_svm, loss_function = loss_root_mean_square)

head(fi_svm)
plotD3(fi_rf, fi_svm)

plotD3(fi_rf, fi_svm, split = "feature")

plotD3(fi_rf, fi_svm, max_vars = 3, bar_width = 16, scale_height = TRUE)
plotD3(fi_rf, fi_svm, max_vars = 3, bar_width = 16, split = "feature", scale_height = TRUE)
plotD3(fi_rf, margin = 0.2)

## End(Not run)
```

---

print.aggregated_ceteris_paribus_explainer

Prints Aggregated Profiles

Description

Prints Aggregated Profiles

Usage

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

Arguments

x	an individual variable profile explainer produced with the 'aggregate_profiles()' function
...	other arguments that will be passed to 'head()'

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")
selected_passangers <- select_sample(titanic, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)
head(cp_rf)
pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)


library("randomForest")
 model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived)

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)
```

---

print.ceteris_paribus_explainer

Prints Individual Variable Explainer Summary

Description

Prints Individual Variable Explainer Summary

Usage

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

Arguments

x	an individual variable profile explainer produced with the ‘ceteris_paribus()‘ function
...	other arguments that will be passed to ‘head()‘

Examples

```
library("DALEX")

library("randomForest")
set.seed(59)

apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
      no.rooms + district, data = apartments)

explainer_rf <- explain(apartments_rf_model,
      data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)

apartments_small <- select_sample(apartmentsTest, 10)

cp_rf <- ceteris_paribus(explainer_rf, apartments_small)
cp_rf
```

---

select_neighbours *Select Subset of Rows Closest to a Specified Observation*

Description

This function selects subset of rows from data set. This is useful if data is large and we need just a sample to calculate profiles.

Usage

```
select_neighbours(data, observation, variables = NULL,
  distance = gower::gower_dist, n = 20, frac = NULL)
```

Arguments

data	set of observations
observation	single observation
variables	names of variables that shall be used for calculation of distance. By default these are all variables present in ‘data‘ and ‘observation‘

distance	the distance function, by default the ‘gower_dist‘ function.
n	number of neighbours to select
frac	if ‘n‘ is not specified (NULL), then will be calculated as ‘frac‘ * number of rows in ‘data‘. Either ‘n‘ or ‘frac‘ need to be specified.

Details

Note that `select_neighbours` function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

a data frame with selected rows

Examples

```
library("DALEX")

new_apartment <- apartments[1, 2:6]
small_apartments <- select_neighbours(apartmentsTest, new_apartment, n = 10)
new_apartment
small_apartments
```

---

select_sample *Select Subset of Rows*

Description

This function selects subset of rows from data set. This is useful if data is large and we need just a sample to calculate profiles.

Usage

```
select_sample(data, n = 100, seed = 1313)
```

Arguments

data	set of observations. Profile will be calculated for every observation (every row)
n	named list of vectors. Elements of the list are vectors with points in which profiles should be calculated. See an example for more details.
seed	seed for random number generator.

Details

Note that `select_subsample` function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

a data frame with selected rows

Examples

```
library("DALEX")
small_apartments <- select_sample(apartmentsTest)
head(small_apartments)
```

---

show_aggreagated_profiles

Adds a Layer with Aggregated Profiles

Description

Function 'show_aggreagated_profiles' adds a layer to a plot created with 'plot.ceteris_paribus_explainer'.

Usage

```
show_aggreagated_profiles(x, ..., size = 0.5, alpha = 1,
  color = "#371ea3", variables = NULL)
```

Arguments

x	a ceteris paribus explainer produced with function 'ceteris_paribus()'
...	other explainers that shall be plotted together
size	a numeric. Size of lines to be plotted
alpha	a numeric between 0 and 1. Opacity of lines
color	a character. Either name of a color or name of a variable that should be used for coloring
variables	if not NULL then only 'variables' will be presented

Value

a ggplot2 layer

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
selected_passangers <- select_sample(titanic, n = 100)

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
```

```
                          data = titanic[,-9],
                          y = titanic$survived == "yes")

cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)
pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_aggreagated_profiles(pdp_rf, size = 3)


library("randomForest")
 model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                          data = titanic[,-9],
                          y = titanic$survived)

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_aggreagated_profiles(pdp_rf, size = 3)

plot(pdp_rf, variables = "age", color = "grey")
```

`show_observations` *Adds a Layer with Observations to a Profile Plot*

Description

Function 'show_observations' adds a layer to a plot created with 'plot.ceteris_paribus_explainer' for selected observations. Various parameters help to decide what should be plotted, profiles, aggregated profiles, points or rugs.

Usage

```
show_observations(x, ..., size = 2, alpha = 1, color = "#371ea3",
  only_numerical = TRUE, variables = NULL)
```

Arguments

x	a ceteris paribus explainer produced with function ‘ceteris_paribus()‘
...	other explainers that shall be plotted together
size	a numeric. Size of lines to be plotted
alpha	a numeric between 0 and 1. Opacity of lines
color	a character. Either name of a color or name of a variable that should be used for coloring
only_numerical	a logical. If TRUE then only numerical variables will be plotted. If FALSE then only categorical variables will be plotted.
variables	if not NULL then only ‘variables‘ will be presented

Value

a ggplot2 layer

Examples

```
library("DALEX")
 ## Not run:
library("titanic")
library("randomForest")

titanic_small <- titanic_train[,c("Survived", "Pclass", "Sex", "Age",
                                  "SibSp", "Parch", "Fare", "Embarked")]
titanic_small$Survived <- factor(titanic_small$Survived)
titanic_small$Sex <- factor(titanic_small$Sex)
titanic_small$Embarked <- factor(titanic_small$Embarked)
titanic_small <- na.omit(titanic_small)
rf_model <- randomForest(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,
                         data = titanic_small)
explainer_rf <- explain(rf_model, data = titanic_small,
                        y = titanic_small$Survived == "1", label = "RF")

selected_passangers <- select_sample(titanic_small, n = 100)
cp_rf <- ceteris_paribus(explainer_rf, selected_passangers)
cp_rf

plot(cp_rf, variables = "Age", color = "grey") +
show_observations(cp_rf, variables = "Age", color = "grey") +
  show_rugs(cp_rf, variables = "Age", color = "red")


## End(Not run)
```

show_rugs *Adds a Layer with Rugs to a Profile Plot*

Description

Function 'show_rugs' adds a layer to a plot created with 'plot.ceteris_paribus_explainer' for selected observations. Various parameters help to decide what should be plotted, profiles, aggregated profiles, points or rugs.

Usage

```
show_rugs(x, ..., size = 0.5, alpha = 1, color = "#371ea3",
  only_numerical = TRUE, sides = "b", variables = NULL)
```

Arguments

x	a ceteris paribus explainer produced with function 'ceteris_paribus()'
...	other explainers that shall be plotted together
size	a numeric. Size of lines to be plotted
alpha	a numeric between 0 and 1. Opacity of lines
color	a character. Either name of a color or name of a variable that should be used for coloring
only_numerical	a logical. If TRUE then only numerical variables will be plotted. If FALSE then only categorical variables will be plotted.
sides	a string containing any of "trbl", for top, right, bottom, and left. Passed to geom rug.
variables	if not NULL then only 'variables' will be presented

Value

a ggplot2 layer

Examples

```
library("DALEX")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
selected_passangers <- select_sample(titanic, n = 100)

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                      data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")
```

```
cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)
pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_aggreagated_profiles(pdp_rf, size = 3)


library("randomForest")
 model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)
 model_titanic_rf

 explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic[,-9],
                           y = titanic$survived)

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_aggreagated_profiles(pdp_rf, size = 3)

plot(pdp_rf, variables = "age", color = "grey")
```

---

theme_drwhy *DrWhy Theme for ggplot Objects*

Description

DrWhy Theme for ggplot Objects

Usage

```
theme_drwhy()

theme_drwhy_vertical()

theme_drwhy_blank()

theme_drwhy_colors(n = 2)

theme_drwhy_colors_gradient()
```

Arguments

n	number of colors for color palette

Value

theme for ggplot2 objects

Index