

# Package ‘vivo’

June 17, 2019

**Title** Local Variable Importance via Oscillations of Ceteris Paribus Profiles

**Version** 0.1.0

**Description** Provides an easy to calculate variable importance measure based on Ceteris Paribus plot and is calculated in eight variants. We obtain eight variants measure through the possible combinations of three parameters such as absolute\_deviation, point and density.

**Depends** R (>= 3.0)

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Imports** ggplot2, dplyr, ingredients

**Suggests** knitr, rmarkdown, DALEX, mlbench, randomForest, gridExtra, grid, lattice, testthat

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**URL** <https://github.com/MI2DataLab/vivo>

**BugReports** <https://github.com/MI2DataLab/vivo/issues>

**NeedsCompilation** no

**Author** Anna Kozak [aut, cre],  
Przemyslaw Biecek [aut, ths]

**Maintainer** Anna Kozak <[anna1993kozak@gmail.com](mailto:anna1993kozak@gmail.com)>

**Repository** CRAN

**Date/Publication** 2019-06-17 14:30:03 UTC

## R topics documented:

calculate_weight . . . . .	2
local_variable_importance . . . . .	3
plot.local_importance . . . . .	4

---

calculate_weight	<i>Calculated empirical density and weight based on variable split.</i>
------------------	---

---

### Description

This function calculate an empirical density of raw data based on variable split from Ceteris Paribus profiles. Then calculated weight for values generated by `ingredients::ceteris_paribus()`.

### Usage

```
calculate_weight(profiles, data, variable_split)
```

### Arguments

profiles	data.frame generated by <code>ingredients::ceteris_paribus()</code>
data	data.frame with raw data to model
variable_split	list generated by <code>ingredients::calculate_variable_split()</code>

### Value

Return an weight based on empirical density.

### Examples

```
library("DALEX", warn.conflicts = FALSE, quietly = TRUE)
data(apartments)

library("ingredients", warn.conflicts = FALSE, quietly = TRUE)
split <- calculate_variable_split(apartments, variables = colnames(apartments))

library("randomForest", warn.conflicts = FALSE, quietly = TRUE)
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface +
                                   floor + no.rooms, data = apartments)

explainer_rf <- explain(apartments_rf_model, data = apartmentsTest[,2:5],
                       y = apartmentsTest$m2.price)

new_apartment <- data.frame(construction.year = 1998, surface = 88, floor = 2L, no.rooms = 3)

profiles <- ceteris_paribus(explainer_rf, new_apartment)

library("vivo")
calculate_weight(profiles, data = apartments[, 2:5], variable_split = split)
```

---

`local_variable_importance`*Local Variable Importance measure based on Ceteris Paribus profiles.*

---

## Description

This function calculate local importance measure in eight variants. We obtain eight variants measure through the possible options of three parameters such as ‘absolute\_deviation’, ‘point’ and ‘density’.

## Usage

```
local_variable_importance(profiles, data, absolute_deviation = TRUE,  
  point = TRUE, density = TRUE)
```

## Arguments

<code>profiles</code>	data.frame generated by <code>ingredients::ceteris_paribus()</code>
<code>data</code>	data.frame with raw data to model
<code>absolute_deviation</code>	logical parameter, if ‘absolute_deviation = TRUE’ then measure is calculated as absolute deviation, else is calculated as a root from average squares
<code>point</code>	logical parameter, if ‘point = TRUE’ then measure is calculated as a distance from $f(x)$ , else measure is calculated as a distance from average profiles
<code>density</code>	logical parameter, if ‘density = TRUE’ then measure is weighted based on the density of variable, else is not weighted

## Value

A data.frame of the class ‘local\_variable\_importance’. It’s a data.frame with calculated local variable importance measure.

## Examples

```
library("DALEX")  
data(apartments)  
  
library("randomForest")  
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface +  
  floor + no.rooms, data = apartments)  
  
explainer_rf <- explain(apartments_rf_model, data = apartmentsTest[,2:5],  
  y = apartmentsTest$m2.price)  
  
new_apartment <- data.frame(construction.year = 1998, surface = 88, floor = 2L, no.rooms = 3)
```

```

library("ingredients")
profiles <- ceteris_paribus(explainer_rf, new_apartment)

library("vivo")
local_variable_importance(profiles, apartments[,2:5],
                          absolute_deviation = TRUE, point = TRUE, density = TRUE)

local_variable_importance(profiles, apartments[,2:5],
                          absolute_deviation = TRUE, point = TRUE, density = FALSE)

local_variable_importance(profiles, apartments[,2:5],
                          absolute_deviation = TRUE, point = FALSE, density = TRUE)

```

---

`plot.local_importance` *Plot Local Variable Importance measure*

---

### **Description**

Function `plot.local_importance` plots local importance measure based on Ceteris Paribus profiles.

### **Usage**

```

## S3 method for class 'local_importance'
plot(x, ...)

```

### **Arguments**

```

x          object returned from 'local_variable_importance()' function
...       other parameters

```

### **Value**

a `ggplot2` object

### **Examples**

```

library("DALEX")
data(apartments)

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

explainer_rf <- explain(apartments_rf_model, data = apartmentsTest[,2:5],
                      y = apartmentsTest$m2.price)

```

```
new_apartment <- data.frame(construction.year = 1998, surface = 88, floor = 2L, no.rooms = 3)

library("ingredients")
profiles <- ceteris_paribus(explainer_rf, new_apartment)

library("vivo")
measure <- local_variable_importance(profiles, apartments[,2:5],
                                     absolute_deviation = TRUE, point = TRUE, density = FALSE)

plot(measure)
```

# Index

`calculate_weight`, [2](#)

`local_variable_importance`, [3](#)

`plot.local_importance`, [4](#)