Package 'FLAME'

December 18, 2018

Type Package

Title A Fast Large-Scale Almost Matching Exactly Approach to Causal Inference

Version 1.0.0

Description The 'FLAME' (Fast Large-scale Almost Matching Exactly) package implements the matching algorithm in Roy, Rudin, Volfovsky, and Wang (2017) <arXiv:1707.06315>. 'FLAME' performs matching of treatment and control units in the potential outcomes framework for large categorical datasets. 'FLAME' creates matches that include as many covariates as possible, and sequentially drops covariates that are less useful based on a match quality measure. Match quality combines two important elements – it considers predictive power from machine learning on a hold out training set, and a balancing factor to ensure that it does not remove a covariate that would ruin overlap between treatment and control groups. Currently the 'FLAME' package applies to categorical data, and provides two approaches for implementation - bit vectors and database management systems (e.g., 'PostgreSQL', 'SQLite'). For data that has been adequately processed and fits in memory, bit vectors should be applied. For extremely large data that does not fit in memory, database systems should be applied.

Maintainer Chia-Rui (Jerry) Chang <chiarui424@gmail.com>

URL https://chiarui424.github.io/FLAME/

BugReports https://github.com/chiarui424/FLAME/issues

Depends R (>= 3.3)

Imports reticulate (>= 1.9), graphics, stats, latticeExtra, dplyr, RPostgreSQL, RSQLite, gmp, lattice, rlang

Suggests knitr, rmarkdown, ggplot2, ggpubr, e1071

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Chia-Rui (Jerry) Chang [aut, cre],
Cynthia Rudin [aut],
Alexander Volfovsky [aut],
Sudeepa Roy [aut],
Tianyu Wang [ctb]

Repository CRAN

Date/Publication 2018-12-17 23:40:08 UTC

R topics documented:

`ATE` *Compute Average Treatment Effect*

Description

`ATE` computes average treatment effect of the matched subsamples by a weighted average of the estimated treatment effects in each matched group. The weight is the number of matched units.

Usage

```
ATE(FLAME_object)
```

Arguments

`FLAME_object`	object returned by applying the FLAME algorithm (`FLAME_bit`, `FLAME_PostgreSQL`, or `FLAME_SQLite`)

Value

average treatment effect (ATE) of the matched subsamples

Examples

```
data(toy_data)
result <- FLAME::FLAME_bit(data = toy_data, holdout = toy_data)
FLAME::ATE(result)
```

AVG_EFFECT *Compute Estimated Treatment Effects*

Description

AVG_EFFECT computes estimated treatment effects. Estimated treatment effect is the weighted average of CATEs, with weight being the number of units in each matched group.

Usage

```
AVG_EFFECT(CATE_object)
```

Arguments

CATE_object	object returned by applying CATE function

Value

estimated treatment effects

Examples

```
data(toy_data)
result <- FLAME::FLAME_bit(data = toy_data, holdout = toy_data)
CATE_object <- FLAME::CATE(FLAME_object = result, cov_name = c("X1", "X2"), cov_val = c("2", "2"))
FLAME::AVG_EFFECT(CATE_object)
```

CATE *Get the Size and CATE of Matched Group(s)*

Description

CATE provides detailed information of the conditional average treatment effect (CATE) and size of each matched group. First, given number of covariates used for matching, `CATE(FLAME_object, num_covs = x)` returns the covariate values, size and CATE of each matched group. Second, given a covariate combination, `CATE(FLAME_object, num_covs = x, cov_name = c("x1", "x2", ...), cov_val = c(0,1,...))` returns the CATE and size of the matched group. Third, if user would like to see all matched groups given a specific covariate combination even when the FLAME algorithm performs matching with more than the number of covariates specified, `CATE(FLAME_object, cov_name = c("x1", "x2", ...), cov_val = c(0,1,...))` returns all matched groups containing the covariate combination.

Usage

```
CATE(FLAME_object, num_covs = NULL, cov_name = NULL, cov_val = NULL)
```

Arguments

FLAME_object	object returned by applying the FLAME algorithm (FLAME_bit or FLAME_PostgreSQL or FLAME_SQLite)
num_covs	number of covariates used for matching
cov_name	a vector of covariate names
cov_val	a vector of covariate values, where the value position should match cov_name position. In addition, it has to be in character R data type.

Value

data frame with covariate values, CATE, and size of each matched group

Examples

```
data(toy_data)
result <- FLAME::FLAME_bit(data = toy_data, holdout = toy_data)
FLAME::CATE(FLAME_object = result, num_covs  = 2)
FLAME::CATE(FLAME_object = result, num_covs  = 2, cov_name = c("X1", "X2"), cov_val = c("2", "2"))
FLAME::CATE(FLAME_object = result, cov_name = c("X1", "X2"), cov_val = c("2", "2"))
```

CATE_plot *Summarize CATEs of All Matched Groups by Boxplot*

Description

Given CATE_object, CATE_plot visualizes CATEs of all matched groups by boxplot

Usage

```
CATE_plot(CATE_object)
```

Arguments

CATE_object	object returned by applying CATE function

Value

boxplot

Examples

```
data(toy_data)
result <- FLAME::FLAME_bit(data = toy_data, holdout = toy_data)
CATE_object <- FLAME::CATE(FLAME_object = result, cov_name = c("X1", "X2"), cov_val = c("2", "2"))
FLAME::CATE_plot(CATE_object)
```

Data_Generation *Generate Synthetic Data*

Description

`Data_Generation` generates synthetic data, where each covariate is a binary variable. The function used to create synthetic data can be found

Usage

```
Data_Generation(num_control, num_treated, num_cov_dense,
  num_cov_unimportant, U)
```

Arguments

`num_control`	number of samples in the control group
`num_treated`	number of samples in the treated group
`num_cov_dense`	number of important covariates
`num_cov_unimportant`	number of unimportant covariates
`U`	non-linear term coefficient

Value

synthetic data

Examples

```
Data_Generation(10, 10, 10, 5, 5)
Data_Generation(10, 10, 5, 10, 5)
```

FLAME_bit *Bit Vectors Implementation*

Description

FLAME_bit applies FLAME matching algorithm based on bit vectors. The required arguments include (1) data and (2) holdout. The rest of the arguments are optional.

Usage

```
FLAME_bit(data, holdout, tradeoff = 0.1, compute_var = FALSE,
  PE_function = NULL, model = NULL, ridge_reg = NULL,
  lasso_reg = NULL, tree_depth = NULL)
```

Arguments

data	input data
holdout	holdout training data
tradeoff	tradeoff parameter to compute Match Quality (optional, default = 0.1)
compute_var	indicator variable of computing variance (optional, default = FALSE)
PE_function	user defined function to compute predictive error (optional)
model	user defined model - Linear, Ridge, Lasso, or DecisionTree (optional)
ridge_reg	L2 regularization parameter if model = Ridge (optional)
lasso_reg	L1 regularization parameter if model = Lasso (optional)
tree_depth	maximum depth of decision tree if model = DecisionTree (optional)

Value

(1) list of covariates FLAME performs matching at each iteration, (2) Sizes, conditional average treatment effects (CATEs), and variance (if compute_var = TRUE) of matches at each iteration, (3) match quality at each iteration, and (4) the original data with additional column *matched*, indicating the number of covariates each unit is matched on. If a unit is never matched, then *matched* will be 0.

Examples

```
data(toy_data)
FLAME_bit(data = toy_data, holdout = toy_data)
```

FLAME_PostgreSQL *PostgreSQL Database Implementation*

Description

FLAME_PostgreSQL applies the FLAME algorithm based on PostgreSQL. If your computer system does not have PostgreSQL installed, install from here. For setup of PostgreSQL server, please refer to this tutorial. User must connect to PostgreSQL server in R using the command dbConnect(dbDriver('PostgreSQL'), db 'your_password') and name the connection as **db**

Usage

```
FLAME_PostgreSQL(db, data, holdout, compute_var = FALSE,
  tradeoff = 0.1, PE_function = NULL, model = NULL,
  ridge_reg = NULL, lasso_reg = NULL, tree_depth = NULL)
```

Arguments

db	name of the database connection (**must name the connection as db**)
data	input data
holdout	holdout training data
compute_var	indicator variable of computing variance (optional, default = FALSE)
tradeoff	tradeoff parameter to compute Match Quality (optional, default = 0.1)
PE_function	user defined function to compute predictive error (optional)
model	user defined model - Linear, Ridge, Lasso, or DecisionTree (optional)
ridge_reg	L2 regularization parameter if model = Ridge (optional)
lasso_reg	L1 regularization parameter if model = Lasso (optional)
tree_depth	maximum depth of decision tree if model = DecisionTree (optional)

Value

(1) list of covariates FLAME performs matching at each iteration, (2) Sizes, conditional average treatment effects (CATEs), and variance (if compute_var = TRUE) of matches at each iteration, (3) match quality at each iteration, and (4) the original data with additional column *matched*, indicating the number of covariates each unit is matched on. If a unit is never matched, then *matched* will be 0.

Examples

```
data <- data(toy_data)

drv <- dbDriver('PostgreSQL')

db <- dbConnect(drv, dbname="FLAME", host='localhost',
```

```
port=5432, user="postgres", password = 'new_password')

FLAME_PostgreSQL(db = db, data = data, holdout = data)

dbDisconnect(db)
```

FLAME_SQLite *SQLite Database Implementation*

Description

FLAME_SQLite applies the FLAME algorithm based on SQLite. FLAME_SQLite does not require external database installment. However, user should connect to a temporary database with command dbConnect(SQLite(),"tempdb_name") and name the connection as **db**. The required arguments include (1) db, (2) data, and (3) holdout. The rest of the arguments are optional.

Usage

```
FLAME_SQLite(db, data, holdout, compute_var = FALSE, tradeoff = 0.1,
  PE_function = NULL, model = NULL, ridge_reg = NULL,
  lasso_reg = NULL, tree_depth = NULL)
```

Arguments

db	name of the connection to temporary database (**must name the connection as db**)
data	input data
holdout	holdout training data
compute_var	indicator variable of computing variance (optional, default = FALSE)
tradeoff	tradeoff parameter to compute Match Quality (optional, default = 0.1)
PE_function	user defined function to compute predictive error (optional)
model	user defined model - Linear, Ridge, Lasso, or DecisionTree (optional)
ridge_reg	L2 regularization parameter if model = Ridge (optional)
lasso_reg	L1 regularization parameter if model = Lasso (optional)
tree_depth	maximum depth of decision tree if model = DecisionTree (optional)

Value

(1) list of covariates FLAME performs matching at each iteration, (2) Sizes, conditional average treatment effects (CATEs), and variance (if compute_var = TRUE) of matches at each iteration, (3) match quality at each iteration, and (4) the original data with additional column *matched*, indicating the number of covariates each unit is matched on. If a unit is never matched, then *matched* will be 0.

Examples

```
data <- data(toy_data)

db <- dbConnect(SQLite(),"tempdb")

FLAME_SQLite(db = db, data = data, holdout = data)

dbDisconnect(db)
```

MATCH *Get Matched Units Given Certain Covariate Combination*

Description

Get Matched Units Given Certain Covariate Combination

Usage

```
MATCH(FLAME_object, cov_name, cov_val)
```

Arguments

FLAME_object	object returned by applying the FLAME algorithm (FLAME_bit or FLAME_PostgreSQL or FLAME_SQLite)
cov_name	a vector of covariate names
cov_val	a vector of covariate values, where the value position should match cov_name position

Value

data frame with all matched units

Examples

```
data(toy_data)
result <- FLAME::FLAME_bit(data = toy_data, holdout = toy_data)
FLAME::MATCH(FLAME_object = result, cov_name = c("X1", "X2"), cov_val = c("2", "2"))
```

summary *Summary*

Description

FLAME_summary provides brief summary of FLAME implementation.

Usage

```
summary(FLAME_object)
```

Arguments

FLAME_object	object returned by applying the FLAME algorithm (FLAME_bit, FLAME_PostgreSQL, or FLAME_SQLite)

Value

(1) Number of units matched (2) Average treatment effect (ATE)

Examples

```
data(toy_data)
result <- FLAME::FLAME_bit(data = toy_data, holdout = toy_data)
FLAME::summary(result)
```

summary_plot *visualize matching process*

Description

summary_plot visualizes matching process, summarizing each covariate dropped, number of matched units, and average CATE at each iteration.

Usage

```
summary_plot(FLAME_object)
```

Arguments

FLAME_object	object returned by applying the FLAME algorithm (FLAME_bit, FLAME_PostgreSQL, or FLAME_SQLite)

Value

summary plot of the FLAME algorithm

Examples

```
data(toy_data)
result <- FLAME::FLAME_bit(data = toy_data, holdout = toy_data)
FLAME::summary_plot(result)
```

`toy_data` *Toy Data*

Description

This toy data has 100 observations/units with 20 categorical covariates.

Usage

```
toy_data
```

Format

A data frame with 2190410 rows and 86 covariates:

X1 9 levels

X2 9 levels

X3 9 levels

X4 9 levels

X5 8 levels

X6 8 levels

X7 8 levels

X8 7 levels

X9 5 levels

X10 5 levels

X11 4 levels

X12 3 levels

X13 3 levels

X14 2 levels

X15 2 levels

X16 2 levels

X17 2 levels

X18 2 levels

X19 2 levels

X20 2 levels

outcome Outcome Variable

treated Treated = 1, Control = 0

Index