Package ‘binomialMix’

March 23, 2020

Title Mixture Models for Binomial and Longitudinal Data
Version 1.0.1
Date 2020-03-23
Author Faustine Bousquet <faustine.bousquet@umontpellier.fr>, Sophie Lèbre <sophie.lebre@umontpellier.fr>, Christian Lavergne <christian.lavergne@umontpellier.fr>
Maintainer Faustine Bousquet <faustine.bousquet@umontpellier.fr>

Description
Provides a clustering method of non-gaussian longitudinal data with a mixture of generalized linear models. The longitudinal data should be defined as repeated observations for each individual. The number of observations for each individual can be different. In runEM(), an expectation-maximization algorithm is developed for both binomial and longitudinal data mixture model.

License GPL-3
Depends R (>= 3.6)
Imports lubridate (>= 1.7.0), Rmpfr, MASS, gmp, dplyr, stringr, rlang, stats
Suggests testthat, devtools, roxygen2, knitr, rmarkdown, ggplot2, qpdf

VignetteBuilder knitr
Encoding UTF-8
RoxygenNote 6.1.1
LazyData true
NeedsCompilation no
Repository CRAN
Date/Publication 2020-03-23 11:40:06 UTC

R topics documented:

adcampaign ................................................................. 2
extract_id ................................................................. 3
extract_target ............................................................ 3
**adcampaign**

_adcampaign_

extract_variables ........................................ 4
Incomplete_Loglikelihood_binomiale .......................... 4
init_design_matrices ......................................... 5
init_lambda ....................................................... 6
init_subset ....................................................... 6
init_tau ............................................................ 7
log_density_binom .............................................. 7
my_BIC ............................................................. 8
my_ICL .............................................................. 8
runEM ............................................................... 9
update_beta ....................................................... 10
update_tau ......................................................... 11
update_w ........................................................... 11
update_z ........................................................... 12

Index 13

**adcampaign**  

_Advertising campaign dataset_

**Description**

Advertising campaign dataset

**Usage**

adcampaign

**Format**

A data frame with 29848 observations on the following 9 variables.

_id_ a factor variable with 80 levels representing the 80 campaigns we want to cluster

timestamp_ymd a POSIXct variable corresponding to the datetime each data is collected

yearDay a factor with day levels of the year

day a factor variable with 7 levels representing the 7 days of the week

timeSlot a factor with levels 6 levels representing 6 different timeSlot: 00h-4h, 4h-8h, 8h-12h, 12h-16h, 16h-20h, 20h-00h

app_or_site a factor with 2 levels app site, representing the 2 types of support where an advertising is displayed

impressions a numeric vector counting the number of times an advertising is displayed on a defined timestamp

click a numeric vector counting the number of times an advertising is clicked on a defined timestamp

ctr a numeric vector corresponding to the number of clicks divided by the number of impressions
**extract_id**

*Source*

These data are extracted from TabMo database.

*Examples*

```r
library(binomialMix)
summary(adcampaign)
```

---

**extract_id**

*Extract levels as numeric from id column of the dataset*

*Description*

Extract levels as numeric from id column of the dataset

*Usage*

```r
extract_id(df, col_id)
```

*Arguments*

- `df` A dataframe
- `col_id` A character value corresponding to id column name

*Value*

`dist_id` The numeric levels of id column from df

*Examples*

```r
extract_id(adcampaign,"id")
```

---

**extract_target**

*Extract target value of GLM*

*Description*

Extract target value of GLM

*Usage*

```r
extract_target(formula)
```

*Arguments*

- `formula` A character formula with target variable and predictor variables
Incomplete_Loglikelihood_binomiale

Value

y The target variable from formula in character type

Examples

extract_target("ctr-timeSlot")

extract_variables Extract variables from GLM model

Description

Extract variables from GLM model

Usage

extract_variables(formula)

Arguments

formula A character formula with target variable and predictor variables

Value

formula_var The predictor variables from formula in formula type

Examples

extract_variables("ctr-timeSlot")

Incomplete_Loglikelihood_binomiale

Calculate the incomplete loglikelihood from mixture of binomial

Description

Calculate the incomplete loglikelihood from mixture of binomial

Usage

Incomplete_Loglikelihood_binomiale(df, col_id = "id", target, var_weights, df_id, matrix_id, lamb, b_hk, K)
Arguments

- df: A dataframe
- col_id: A character value corresponding to id column name
- target: A character value corresponding to the target variable
- var_weights: A character value corresponding to the weights variable
- df_id: A list of dataframe filter by id levels
- matrix_id: A list of design matrices filter by id levels
- lamb: A numeric vector of proportion into the different clusters
- b_hk: A matrix of estimated beta
- K: A numeric value of number of clusters chosen for the mixture

Value

- result: A numeric value of incomplete loglikelihood

Description

Initialize design matrices from dataframe to cluster

Usage

init_design_matrices(formula, df, col_id = "id")

Arguments

- formula: A character formula with target variable and predictor variables
- df: A dataframe to cluster
- col_id: A character value corresponding to name of id column in the dataframe df

Value

- result_list: A list containing the df filter by id levels, the design matrices filter by id levels, the number of rows for df filter by id levels

Examples

init_design_matrices("ctr~timeSlot",adcampaign,"id")
**init_lambda**  
*Initialize the vector lambda of mixture proportion*

**Description**  
Initialize the vector lambda of mixture proportion

**Usage**

```r
init_lambda(K)
```

**Arguments**

- **K**  
  A numeric value corresponding to the number of cluster

**Value**

- `result`  
  A numeric vector of length K

**Examples**

```r
init_lambda(K=3)
```

---

**init_subset**  
*Initialize the estimation of beta*

**Description**  
Initialize the estimation of beta

**Usage**

```r
init_subset(df, K, col_id = "id")
```

**Arguments**

- **df**  
  A dataframe
- **K**  
  The number of dataframe to obtain depending on the number of cluster chosen for the mixture
- **col_id**  
  A character value corresponding to id column name

**Value**

- `subset_df`  
  A list of K subset of dataframe

**Examples**

```r
init_subset(adcampaign,3,"id")
```
init_tau

**Description**

Initialize the matrix probability of each levels id to be in the clusters

**Usage**

```
init_tau(df, K, col_id = "id")
```

**Arguments**

- `df`: A dataframe
- `K`: The number of dataframe to obtain depending on the number of cluster chosen for the mixture
- `col_id`: A character value corresponding to id column name

**Value**

- `result_matrix`: A matrix of dimension: rows number is the number of cluster K, columns number is the number of distinct levels from id column

**Examples**

```
init_tau(adcampaign,3,"id")
```

log_density_binom

**Description**

Calculate de log density of a binomial

**Usage**

```
log_density_binom(y, matrix_id, b_hk, k, var_weights)
```

**Arguments**

- `y`: A dataframe corresponding to a specific id levels from col_id
- `matrix_id`: A design matrix corresponding to a specific id levels from col_id
- `b_hk`: A matrix of estimated beta
- `k`: A numeric value to select the beta from a specific cluster
- `var_weights`: A character value corresponding to the weights variable
Value

res A numeric value

my_BIC  Calculate the Bayesian Information Criterion (BIC)

Description

Calculate the Bayesian Information Criterion (BIC)

Usage

my_BIC(nb_param, logl, nb_obs)

Arguments

nb_param The number of parameters estimated by the EM
logl A numeric value which is the maximum value from Incomplete Loglikelihood
nb_obs A numeric value corresponding to the rows number of the whole dataframe

Value

BIC The numeric value of the BIC

my_ICL  Calculate the Integrated Complete Likelihood (ICL)

Description

Calculate the Integrated Complete Likelihood (ICL)

Usage

my_ICL(data, col_id, nb_cluster, nb_param, logl, val_tau, nb_obs)

Arguments

data A dataframe
col_id A character value corresponding to id column name
nb_cluster A numeric value of number of clusters chosen for the mixture
nb_param The number of parameters estimated by the EM
logl A numeric value which is the maximum value from Incomplete Loglikelihood
val_tau A matrix of probability which rows number is the K clusters, columns number is the number of distinct id levels
nb_obs A numeric value corresponding to the rows number of the whole dataframe
runEM

Value

ICL The numeric value of the ICL

Description

This function is the main function of this package. The objective is to provide a clustering of the 80 campaigns that we have on our dataset. The specification of this algorithm is that we can have longitudinal data, i.e. n observations for a single campaign.

Usage

runEM(formula, var_weights, K, df, col_id = "id")

Arguments

formula A formula or Character which links target variable and predictor variables

var_weights A character value corresponding to the weights variable

K A numeric value representing the number of clusters chosen for the mixture

df A dataframe to cluster

col_id A character value (colname) corresponding to the id column name

Value

a summary list of EM algorithm results: loglikelihood, beta/lambda/tau estimation at each iteration, bic/icl value, number of fisher iteration at each EM iteration

Examples

## Load data:
data(adcampaign)
## Run mixture:
## Not run:
result_mixture<-runEM(formula="ctr~timeSlot",
                      var_weights="impressions",
                      K=2,
                      df=adcampaign,
                      col_id="id")

## Analysis of results:
plot(result_mixture[[1]],type="l") #gives you the loglikelihood evolution
# list of the estimated parameter for each cluster for each iteration:
result_mixture[[2]]
# list of the estimated parameter for each cluster for each iteration
result_mixture[[3]] #list of ids proportion in each cluster for each iteration
#list of matrices containing probability to be in cluster k for each id:
```r
result_mixture[[4]]
# BIC value:
result_mixture[[5]]
# ICL value:
result_mixture[[6]]
# list of number fisher scoring iterations for each iteration
result_mixture[[7]]

## End(Not run)
```

---

**update_beta**

*M-step: update of beta parameters*

**Description**

M-step: update of beta parameters

**Usage**

```r
update_beta(formula, df, k, col_id, tau, m, w_inv, z, matrix_id)
```

**Arguments**

- `formula` A character formula with target variable and predictor variables
- `df` A dataframe
- `k` The numeric value of the specific cluster to be updated
- `col_id` A character value corresponding to id column name
- `tau` A matrix of dimension: rows number is the number of cluster K, columns number is the number of distinct levels id
- `m` A numeric iterative value
- `w_inv` An inverse matrix representing the W matrix in the beta equation for the M step
- `z` Working data in the EM algorithm
- `matrix_id` A list of design matrices filter by id levels

**Value**

- `result_beta` Estimated beta for cluster k
**update_tau**

_E-step : update of tau_

**Description**

E-step : update of tau

**Usage**

update_tau(df, K, col_id = "id", beta_hk, lambda, m, df_id, n_c, matrix_id, var_weights, target)

**Arguments**

- **df**: A dataframe
- **K**: The numeric value of the total number of clusters
- **col_id**: A character value corresponding to id column name
- **beta_hk**: A matrix of estimated beta
- **lambda**: A numeric vector of proportion into the different clusters
- **m**: A numeric iterative value
- **df_id**: A list of dataframe filter by id levels
- **n_c**: A numeric vector containing the number of rows for each distinct id levels
- **matrix_id**: A list of design matrices filter by id levels
- **var_weights**: A character value corresponding to the weights variable
- **target**: A character value corresponding to the target variable

**Value**

result_pi Estimated probabilities of tau matrix

**update_w**

_M-step : Update the diagonal matrix W from beta iterative equation_

**Description**

M-step : Update the diagonal matrix W from beta iterative equation

**Usage**

update_w(df, col_id = "id", var_weights, beta_up, df_id, matrix_id)
Arguments

- **df**: A dataframe
- **col_id**: A character value corresponding to id column name
- **var_weights**: A character value corresponding to the weights variable
- **beta_up**: A matrix of estimated beta in a specific cluster k
- **df_id**: A list of dataframe filter by id levels
- **matrix_id**: A list of design matrices filter by id levels

Value

- **omega_inv**: An up-to-date diagonal matrix W

---

**update_z**

*M-step : Update the matrix of working variables Z from beta iterative equation*

---

**Description**

M-step: Update the matrix of working variables Z from beta iterative equation

**Usage**

`update_z(df, col_id = "id", target, beta_up, df_id, matrix_id)`

**Arguments**

- **df**: A dataframe
- **col_id**: A character value corresponding to id column name
- **target**: A character value corresponding to the target variable
- **beta_up**: A matrix of estimated beta in a specific cluster k
- **df_id**: A list of dataframe filter by id levels
- **matrix_id**: A list of design matrices filter by id levels

**Value**

- **work_z**: An up-to-date matrix of working variables Z
Index

*Topic datasets
    adcampaign, 2

adcampaign, 2

extract_id, 3
extract_target, 3
extract_variables, 4

Incomplete_Loglikelihood_binomiale, 4
init_design_matrices, 5
init_lambda, 6
init_subset, 6
init_tau, 7

log_density_binom, 7

my_BIC, 8
my_ICL, 8

runEM, 9

update_beta, 10
update_taus, 11
update_w, 11
update_z, 12