Package 'MetaboVariation'

February 24, 2022

Title Exploring individual variation in metabolomic profiles

Version 0.0.0.9000

Description Generalised linear model fitted via Bayesian framework using STAN for identifying individuals with biomarker values outside the 95% central credible interval at an individual level.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports brms, circlize, ComplexHeatmap, doParallel, dplyr, foreach,
future, grid, magrittr, parallel, plotly, reshape2, rstan,
scales, stringr, readxl, tidyr, ggplot2

Depends R (>= 2.10)

LazyData true

Suggests rmarkdown, knitr

VignetteBuilder knitr

NeedsCompilation no

Author Shubbham Gupta [aut, cre],
Lorraine Brennan [aut],
Claire Gormley [aut]

Maintainer Shubbham Gupta <`shubbham.gupta@ucdconnect.ie`>

R topics documented:

`flaggedIndividuals` *Gives flagged individuals in a tablular format*

Description

From the class "MetaboVariation", the results contain the individuals that have been flagged as they have values outside the 95% central credible interval based on the results. This function provides the details of those individuals in a tabular format.

Usage

```
flaggedIndividuals(
  model,
  data,
  individual_id,
  covariates = NULL,
  threshold = 2
)
```

Arguments

`model`	An object of class "MetaboVariation"
`data`	An object of class data.frame (or one that can be coerced to that class) containing data of all variables used for modelling.
`individual_id`	A character string that denotes the column name of individual ids present in the dataset passed.
`covariates`	A list of column names present in the data that denotes the covariates data for the individuals in the data.
`threshold`	A number to filter out individuals that have been flagged in less than "threshold" metabolites. By default, it is set at 2.

Value

A data.frame of the flagged individuals that have values outside of 95% credible interval.

Examples

```
## Not run:
data(metabol.data)
metabolite_list = colnames(metabol.data)[5:length(colnames(metabol.data))]
metabolites = get.metabolites(list = metabolite_list)
covariates = c("SexM.1F.2","Age","BMI")
individual_id = "Sample.Id"
model = MetaboVariation(data = metabol.data,individual_ids = individual_id
,metabolite = metabolites[1:3],covariates = covariates,full_posterior = TRUE)
flaggedIndividuals(model,data = metabol.data,
individual_id = individual_id,covariates=covariates)
```

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

get.metabolites — *Extract metabolite names present in data*

Description

get_metabolites() is a function that will extract the unique metabolites names from the columns present in data, when different timepoints of metabolites are listed as different columns in data.

Usage

```
get.metabolites(list, divider = "_", start = TRUE)
```

Arguments

`list`	A list containing names of columns that store metabolites values across timepoints.
`divider`	A character value that separates the metabolite name and timepoint value. The default is "_"
`start`	A binary value that tells the function if the metabolite name is followed by timepoint or the timepoint is followed by metabolite. For eg: With divider "_", column names in format "MetaboliteA_1" should have start=TRUE, while column names in format "1_MetaboliteA" should have start=FALSE.

Value

A list of unique metabolite names

Examples

```
data(metabol.data)
metabolite_list = colnames(metabol.data)[5:length(colnames(metabol.data))]
metabolites = get.metabolites(list = metabolite_list)
metabolites
```

`metabol.data` *Metabolites data of individuals*

Description

A random dataset that contains metabolites levels of 3 metabolites for different individuals taken on different timepoints.

Usage

```
data(metabol.data)
```

Format

A data frame with 164 rows and 16 columns. NA in the data means that the metabolite value is missing for the individual for that timepoint.

Age This contains the age of the individual

BMI This contains the BMI value of the individual

SexM.1F.2 Gender of the individual is recorded here. It's a factor column where 1 denotes "Male" while 2 denotes "Female"

metabolA_1 Contains the value of "A" metabolite for the first timepoint.

metabolA_2 Contains the value of "A" metabolite for the second timepoint.

metabolA_3 Contains the value of "A" metabolite for the third timepoint.

metabolA_4 Contains the value of "A" metabolite for the fourth timepoint.

metabolB_1 Contains the value of "B" metabolite for the first timepoint.

metabolB_2 Contains the value of "B" metabolite for the second timepoint.

metabolB_3 Contains the value of "B" metabolite for the third timepoint.

metabolB_4 Contains the value of "B" metabolite for the fourth timepoint.

metabolC_1 Contains the value of "C" metabolite for the first timepoint.

metabolC_2 Contains the value of "C" metabolite for the second timepoint.

metabolC_3 Contains the value of "C" metabolite for the third timepoint.

metabolC_4 Contains the value of "C" metabolite for the fourth timepoint.

Individual_id Contains the individual id for the individual that provided the measurements for various metabolites.

`MetaboVariation` *Models the metabolomic profiles to get individual traits*

Description

Fits a generalised linear model on the passed metabolites and provide results for single or multiple metabolites. It works on bayesian framework using the function brm from BRMS package.

Usage

```
MetaboVariation(
  data,
  individual_ids,
  metabolite,
  covariates = NULL,
  save_brms_model = FALSE,
  full_posterior = FALSE,
  iter = 2000,
  warmup = floor(iter/2),
  ...
)
```

Arguments

`data`	An object of class data.frame (or one that can be coerced to that class) containing data of all variables used for modelling.
`individual_ids`	A character string that denotes the column name of individual ids present in the dataset passed.
`metabolite`	A string or vector containing metabolites name that needs to be modeled.
`covariates`	A list of column names present in the data that denotes the covariates data for the individuals in the data.
`save_brms_model`	A logical value indicating whether the model should be saved or not. By default, it is set as FALSE
`full_posterior`	A logical value indicating whether the full posterior distribution should be included or not. By default, it is set as FALSE and the summarized result is passed by the function.
`iter`	Number of total iterations per chain (including warmup; defaults to 2000).
`warmup`	A positive integer specifying number of warmup (aka burnin) iterations. This also specifies the number of iterations used for stepsize adaptation, so warmup draws should not be used for inference. The number of warmup should not be larger than `iter` and the default is `iter/2`.
`...`	Further arguments are passed to brm for backend processes.

Value

modelling returns an object of "MetaboVariation" class with "meta.single_model" and "meta.multi_model" subclass. Sublass "meta.multi_model" is a collection of subclass "meta.single_model" objects. And subclass "meta.single_model" contains following values.

- significant_covariates - a list of covariates that influences the metabolite value significantly for the individuals. Based on the posterior distribution of the covariates, if zero lies in the 90% credible interval, it is assumed that particular covariate has no significant realtion with the metabolite value.
- result - It stores a summarized result for the posterior predictive distribution of the cohort. It contains the mean, 95% credible interval range, tails of the credible interval and the original value of the individual for that timepoint. It also has a flag that shows whether the original values lies within the credible interval or not.
- iterations - Number of total iterations per chain used for modelling (including warmup; defaults to 2000).
- warmup - A positive integer specifying number of warmup (aka burnin) iterations. This also specifies the number of iterations used for stepsize adaptation, so warmup draws should not be used for inference. The number of warmup should not be larger than iter and the default is iter/2.
- Rhat - Rhat refers to the potential scale reduction statistic, also known as the Gelman-Rubin statistic. Thus, it measures the extent to which chains are converging. The further the value of the statistic from 1, the worse. Model is considered good if the value lies between 0.9 and 1.05.
- metabolite - Contains the name of metabolite on which modelling is done.
- full_posterior - It contains the full posterior predictive distiribution of the data. It can be used for further analysis as per the user requirements.

Examples

```
## Not run:
data(metabol.data)
metabolite_list = colnames(metabol.data)[5:length(colnames(metabol.data))]
metabolites = get.metabolites(list = metabolite_list)
covariates = c("SexM.1F.2","Age","BMI")
individual_id = "Sample.Id"
model = MetaboVariation(data = metabol.data,individual_ids = individual_id,
metabolite = metabolites[1], covariates = covariates)
model = MetaboVariation(data = metabol.data,individual_ids = individual_id,
metabolite = metabolites[1:3], full_posterior = TRUE)

## End(Not run)
```

`plot.MetaboVariation` *Plot results from "MetaboVariation" object.*

Description

Plot the results for the class "MetaboVariation". It plots using plotly and circlize package.

Usage

```
## S3 method for class 'MetaboVariation'
plot(x, type = "circos", timepoints = NULL, threshold = 2, ...)
```

Arguments

`x`	An object of class "MetaboVariation"
`type`	A character specifying the plot type (eg: "circos", "individual_count", "metabolites_count"). By default, "circos" is shown.
`timepoints`	A numeric vector containing timepoints to be shown in the plots. By default, it is NULL and will show all the timepoints
`threshold`	A number to filter out individuals that have been flagged in less than "threshold" metabolites. By default, it is set at 2.
`...`	Further arguments are ignored.

Value

plot will show the results of modelling in the following three ways

- Circos plot - A circular plot that will show the 95% credible interval for all the individuals across timepoints. If the actual value is outside the interval, that bar will be flagged as black and the individual's id will be flagged and shown in **bold**.
- Metabolites count plot - A bar plot that shows the number of individuals are flagged in each timepoints out of all individuals. For "meta.single_model" subclass, number of bars will be same as the number of timepoints for that single metabolite. For meta.multi_model" subclass, each timepoints is shown with different color for each metabolite present in the subclass.
- Individual count plot - A heatmap that shows how many time a particular individual is flagged for a specific metabolite. The minimum count is 0 while the maximum count is the number of total timepoints passed. This plot is only valid for "meta.multi_model" subclass.

Examples

```
## Not run:
data(metabol.data)
metabolite_list = colnames(metabol.data)[5:length(colnames(metabol.data))]
metabolites = get.metabolites(list = metabolite_list)
covariates = c("SexM.1F.2","Age","BMI")
individual_id = "Sample.Id"
```

```
model = MetaboVariation(data = metabol.data,individual_ids = individual_id,
metabolite = metabolites[1:3], covariates = covariates,full_posterior = TRUE)
plot(x = model, type = "circos")
plot(x = model, type = "metabolites_count",timepoints = c(1,2,4))
plot(x = model, type = "individual_count",threshold = 3)

## End(Not run)
```

plotMetabolite *Initiate a data distribution visualization for the given metabolite.*

Description

It shows a violin plot for different timepoints for the passed metabolites. Violin plot is a plot that shows a distribution of the numeric list.

Usage

```
plotMetabolite(data, metabolite, points = FALSE)
```

Arguments

`data`	An object of class data.frame (or one that can be coerced to that class) containing data of all variables used in the plot
`metabolite`	A character or a list of characters containing metabolites name for which the plot will be drawn. If a single metabolite is passed then a plot for that metabolite is drawn. If a list of metabolite is passed then each metabolite will be shown in a different plot.
`points`	As violin plot shows the distribution, observations along the end of tail are also shown by default. If you want to remove these points, then set points = FALSE.

Value

It returns plot for the metabolites passed.

Examples

```
data(metabol.data)
metabolite_list = colnames(metabol.data)[5:length(colnames(metabol.data))]
metabolites = get.metabolites(list = metabolite_list)
plotMetabolite(data = metabol.data,metabolite = metabolites[1])
plotMetabolite(data = metabol.data,metabolite = metabolites[1:3],points = TRUE)
```

Index