MetaboVariation

Shubbham Gupta, Isobel Claire Gormley, and Lorraine Brennan

Introduction

Variations in individuals' metabolic profiles are the result of their genetic makeup, environmental, and lifestyle factors. Hence, we have developed the **MetaboVariation** package to facilitate the flagging of individuals with intra-individual variation in their metabolite levels using repeated measures data.

Expanding on our previous work with the univariate **MetaboVariation** approach [1], we now employ a multivariate Bayesian generalised linear model (BGLM) that considers dependencies among metabolites. This multivariate approach allows us to flag individuals by considering all metabolites and their inherent dependencies, unlike previous research that focused on a single metabolites independently. Consequently, individuals whose observed metabolite levels deviate from their individual posterior predictive interval at a specified time point are flagged. The **MetaboVariation** package is built on the MCMCglmm package [2], which fits the multivariate BGLM. The approach is flexible, allowing for the consideration of multiple posterior predictive interval widths, and a detailed examination of metabolite variations at the individual level.

This document gives an overview of **MetaboVariation** functionalities. See `help(package = MetaboVariation)` for further details and references available via `citation("MetaboVariation")`.

Walk through

Prerequisite packages

To use the **MetaboVariation** package, the user should have the R software environment installed on their machine. The **MetaboVariation** package has the following dependencies which will be installed along with the package if not already installed on the machine:

- circlize, coda, ComplexHeatmap, doParallel, dplyr, foreach, ggplot2, grid, MCMCglmm, parallel, plotly, reshape2, stringr, tidyr, scales.

To install and load the **MetaboVariation** package:

```r
install.packages("MetaboVariation")
library(MetaboVariation)
```

A simulated data example

The object `metabol.data` is a simulated dataset that contains metabolite levels of 5 metabolites for 150 individuals measured at four time points. The covariates sex, age and BMI are also available for these individuals.

To load the data and examine its column names type:

```r
data(metabol.data)
colnames(metabol.data)
#>  [1] "Individual_id" "SexM.1F.2"     "Age"           "BMI"
#>  [5] "metA_1"        "metB_1"        "metC_1"        "metD_1"
#>  [9] "metE_1"        "metA_2"        "metB_2"        "metC_2"
#> [13] "metD_2"        "metE_2"        "metA_3"        "metB_3"
#> [17] "metC_3"        "metD_3"        "metE_3"        "metA_4"
#> [21] "metB_4"        "metC_4"        "metD_4"        "metE_4"
```

The subject IDs of individuals are stored in the column named `Individual_id`, while information on sex, age, and BMI are stored in `SexM.1F.2`, `Age`, and `BMI`, respectively. The column name `metX_Y` represents the level of metabolite **X** at time point **Y**.

Extracting metabolite names

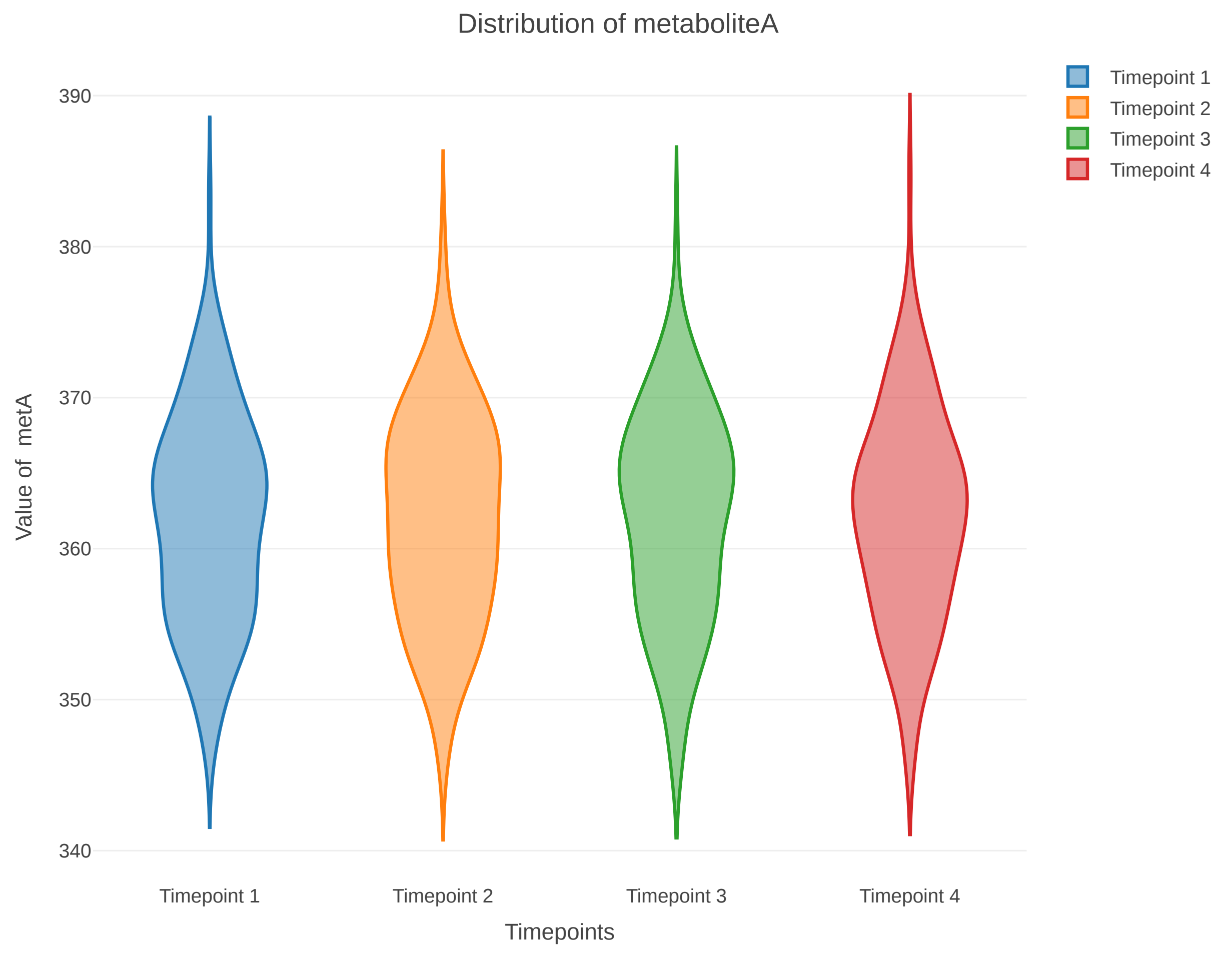
To extract the names of metabolites that are present in the data, the function `get.metabolites` reads the names of columns that contain metabolite values and provides a list of unique metabolites. The user should not pass any unnecessary column names (e.g., any column names that relate to covariate or individual ID information) to the function. The function will return a vector containing the names of unique metabolites. To do this type:

```
metabolite_list = colnames(metabol.data)[5:length(colnames(metabol.data))]
metabolites = get.metabolites(list = metabolite_list)
metabolites
#> [1] "metA" "metB" "metC" "metD" "metE"
```

Visualising the distributions of metabolites

The user can plot the distributions of metabolites for each time point using the function `metabolite.plot`. The user can pass either a single metabolite or multiple metabolites to this function. For example, to view the distribution of the metaboliteA metabolite type:

```
metabolite.plot(data = metabol.data,metabolite = metabolites[1],
                main="Distribution of metaboliteA")
```



Modelling the data to flag individuals with intra-individual variation in metabolite levels

Specifications of the priors used in the BGLM requires consideration. The **MetaboVariation** package allows you to specify your own prior parameters based on specific settings, or alternatively, default settings are available.

Prior specification

You can specify your own prior parameters using the following structure, or let the package calculate them automatically based on fitting an independent BGLM to the metabolite data.

```r
# Set the prior mean vector with zeros, one for each metabolite
prior_mean <- rep(0, length(metabolites))
# Set the prior variance vector with ones, one for each metabolite
prior_V <- rep(1, length(metabolites))
# Set the prior residual covariance matrix as an identity matrix
prior_R <- diag(1,length(metabolites))
# Set the prior random effects covariance matrix as an identity matrix
prior_G <- diag(1,length(metabolites))

# Combine all the priors into a list
prior <- list(
  B = list(mu = prior_mean, V = diag(prior_V)),
  R = list(V = prior_R, nu = length(metabolites) * 1.5),
  G = list(G1 = list(V = prior_G, nu = length(metabolites) * 1.5))
)
```

- B: This is the prior for the regression coefficients, assumed to follow a multivariate normal distribution with prior mean of zero and prior covariance matrix set to a diagonal matrix where the diagonal elements are the variances of the regression coefficients.
- R and G: These are assumed to follow inverse Wishart prior distributions. The scale matrix V for both R and G is set to an identity matrix. The degrees of freedom (nu) are set to 150% of the number of metabolites (M) to balance between overfitting and capturing metabolite dependencies effectively.

If nu is too close to the number of metabolites, the resulting prior distribution would be very wide, indicating high uncertainty and potentially underfitting the data. Conversely, if nu is too large, the distribution would be tightly concentrated around the scale matrix, potentially leading to overfitting by not allowing enough flexibility to capture the true variability in the data.

Under default settings, the package calculates the priors for B based on fitting an independent BGLM to the metabolite data. As for R and G, the scale matrix is a dense matrix with diagonal terms as the variance of metabolites calculated by fitting an independent BGLM to the metabolite data and non-zero off-diagonal terms. The off-diagonal values are set to 0.1. The signs of these off-diagonal terms match the signs from the correlation matrix of the metabolite data.

Modelling the data

The main function, `MetaboVariation`, performs the modeling. The function requires the data, the names of covariate columns (if any), the individual ID column and list of metabolites as arguments. To find the individuals with intra-individual variation, the BGLM provides the posterior predictive distribution for every individual at each time point, for each metabolite. These are used to identify individuals with observed metabolite levels outside the `interval.width`% highest posterior density (HPD) interval at one time point; such individuals with notable variation are flagged. The width of the HPD interval is set by the `interval.width` argument where the default is a vector containing three values `c(0.95,0.975,0.99)`. The function returns summarised results that contains the upper and lower limit of the HPD intervals and whether the individual is flagged or not under each HPD interval width.

Multivariate dependent model

By setting `type="dependent"`, `MetaboVariation` models the data using a multivariate model that takes dependencies between metabolites into account.

To fit the model, type:

```r
model = MetaboVariation(data = metabol.data, individual_ids = "Individual_id",
          metabolite = metabolites,covariates = c("SexM.1F.2","Age","BMI"),
          interval.width =c(0.95,0.975,0.99),type="dependent")
```

```r
names(model)
#> [1] "result"                 "significant_covariates" "chain_convergence"
#> [4] "type"                   "model"
```

The function returns a list that contains the following values:

- result - a data frame containing the summary of the HPDs of the posterior predictive distributions of the individuals in the cohort and a binary flag for each HPDs showing whether the individual is flagged under

eash HPD interval width or not. The data frame contains the mean, the lower and upper bounds of the HPD intervals and the observed value of the metabolite for the individual for that timepoint. The binary flag shows whether the observed value lies within the HPD interval or not where 1 denotes the observed value lies outside the interval and 0 denotes the observed value lies in the interval. The row names specify the individual, time point, and metabolite for each observation.
  - significant_covariates - a data frame containing the estimates and 95% credible intervals for each regression coefficient for each covariate across all metabolites along with a binary flag indicating whether each covariate is significant in the model for the corresponding metabolite.
  - chain_convergence - the potential scale reduction statistic, also known as the Gelman-Rubin statistic which measures the extent to which chains are converging [3]. The further the value of the statistic from 1, the poorer the convergence of the chains. The MCMC chains are considered converged if the value lies between 0.9 and 1.05.
  - model - contains BGLM model outputs.
  - type - shows which model is fitted. It can be either “dependent” or “independent”.

See `?MetaboVariation` for further details about the function.

The following code displays the first five observations flagged within the 95% HPD intervals. The row names indicate the individual, the time point, and the metabolite for which they have been flagged. The output includes three HPD interval widths (0.95, 0.975, 0.99), providing the lower and upper bounds for each of these HPD intervals. Additionally, there is a flag column that indicates whether the individual is flagged within each HPD interval.

```
head(model$result[model$result[,"flag0.95"]==1,])
#>                  mean   lwr0.95   upr0.95  lwr0.975  upr0.975   lwr0.99
#> 11 1 metB   46.71275  43.71696  49.31910  43.69325  49.92331  42.82390
#> 109 1 metC  95.61841  93.16444  98.53489  92.39658  98.53489  92.10585
#> 110 1 metE 300.23288 297.58099 303.30789 297.16379 303.77024 296.80246
#> 122 1 metA 369.93380 367.38293 372.97372 366.77254 373.13867 366.23741
#> 126 1 metA 368.50945 365.82566 371.40447 365.33812 371.54775 365.10780
#> 127 1 metA 357.92564 355.19356 361.22895 354.79492 361.58909 353.99554
#>               upr0.99  original flag0.95 flag0.975 flag0.99
#> 11 1 metB   49.97699  50.63484        1         1        1
#> 109 1 metC  99.36083  93.13228        1         0        0
#> 110 1 metE 304.23929 296.13965        1         1        1
#> 122 1 metA 373.34641 373.08306        1         0        0
#> 126 1 metA 371.96070 371.55486        1         1        0
#> 127 1 metA 361.79758 354.86673        1         0        0
```

Independent model

Another way of fitting the model is independently. By setting `type="independent"`, `MetaboVariation` treats each metabolite independently.

To fit the model, type:

```
independent_model = MetaboVariation(data = metabol.data, individual_ids = "Individual_id",
          metabolite = metabolites,covariates = c("SexM.1F.2","Age","BMI"),
          interval.width=c(0.95,0.975,0.99),type="independent")
```

Posterior predictive checking

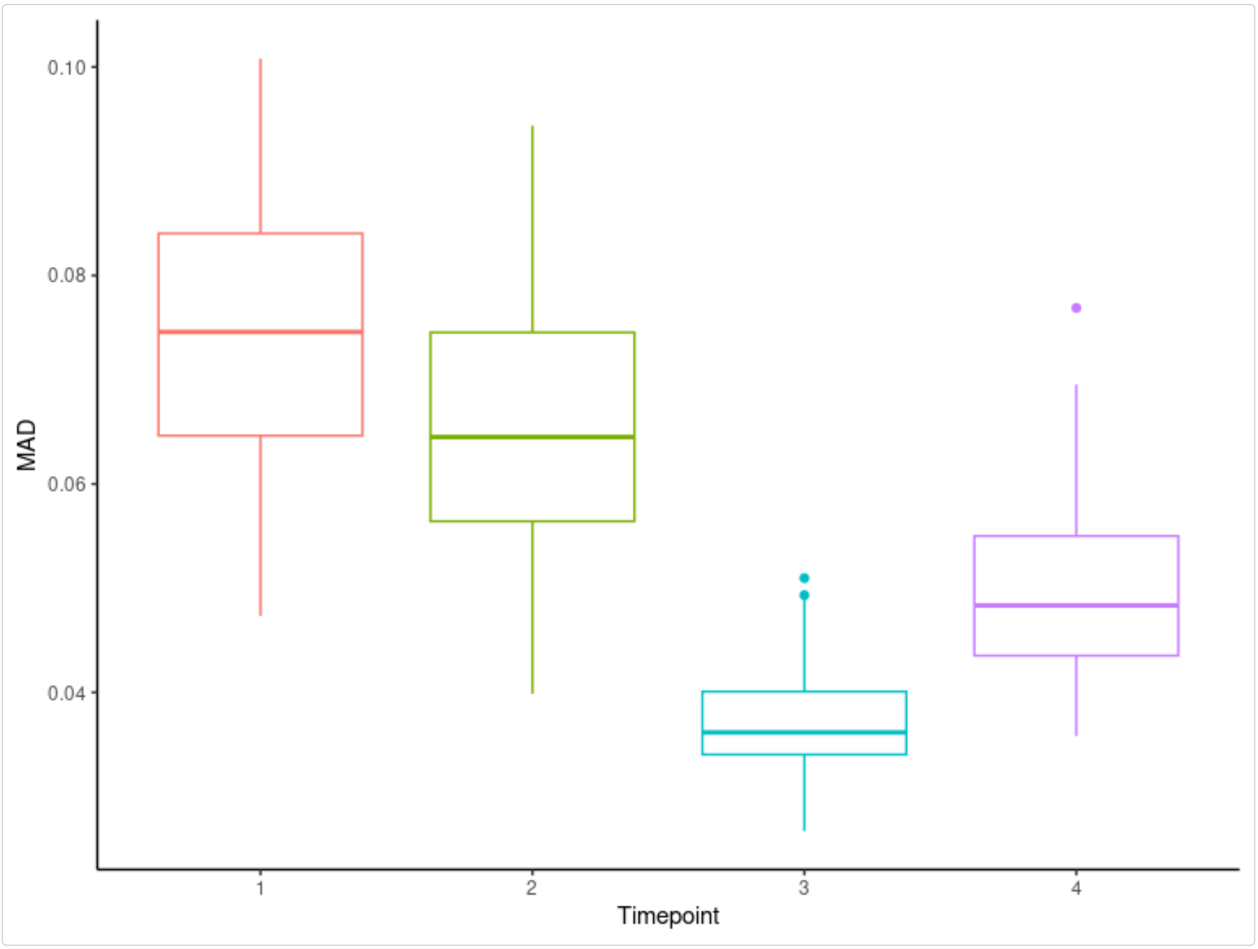
Now that we have two different models—one that considers dependencies between metabolites and one that treats metabolites independently—it is advisable to perform a posterior predictive check to determine which model better fits the data. The function `predictive_check` performs a posterior predictive check assess the fit of the model. The function generates 100 replicate datasets by default from the posterior predictive distributions. The function then provides a box plot that shows the mean absolute difference (MAD) between correlation matrices of the original dataset and those obtained from the replicated datasets over each time point. The function additionally returns the actual values of MAD for each replicate data set at each timepoint. The following code performs the posterior predictive check for both multivariate model and independent model.

```
multivariate_check = predictive_check(data = metabol.data, model = model,
          timepoints = c(1,2,3,4), metabolites = metabolites, replication = 100)
```

```
independent_check = predictive_check(data = metabol.data, model = independent_model,
          timepoints = c(1,2,3,4), metabolites = metabolites, replication = 100)
```

The following code shows the boxplot of posterior predictive check of multivariate model.

```
multivariate_check$plot
```



The following shows the first six rows of MAD values.

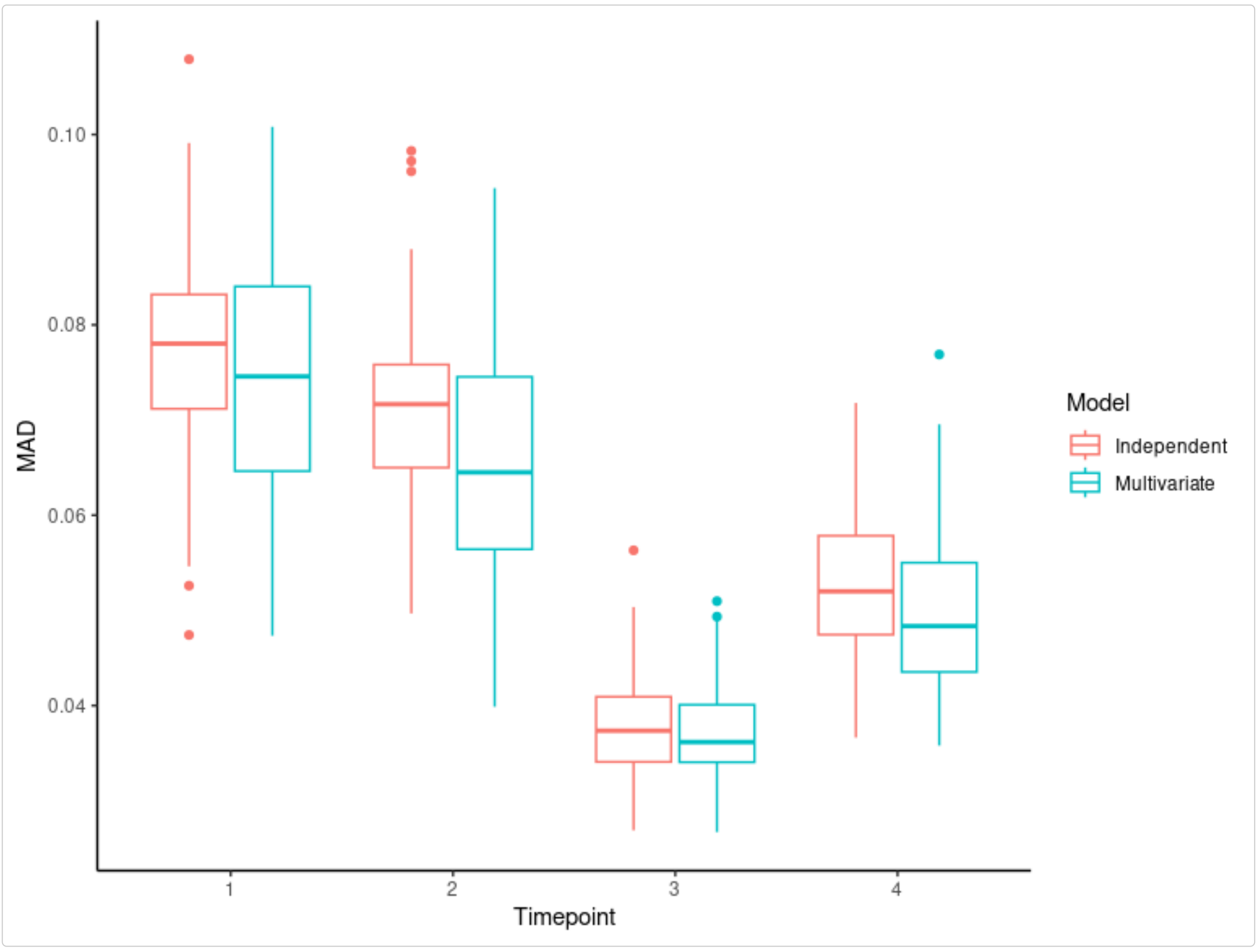
```
head(multivariate_check$mad)
```

```
#>   Timepoint        MAD
#> 1         1 0.09471758
#> 2         1 0.07928552
#> 3         1 0.09797898
#> 4         1 0.08355929
#> 5         1 0.07297066
#> 6         1 0.09531774
```

The following code compares the model fit between multivariate and independent models by using the MAD values from each model's posterior predictive check.

```
# Setting model type for each MAD dataframe
multivariate_check$mad$Model = "Multivariate"
independent_check$mad$Model = "Independent"

plotdf = rbind(multivariate_check$mad,independent_check$mad)
ggplot2::ggplot(plotdf,ggplot2::aes(x=Timepoint,y=MAD,color = Model))+
  ggplot2::geom_boxplot() + ggplot2::labs(colour ="Model",x="Timepoint",y="MAD")
+ ggplot2::theme_classic()
```



The multivariate model shows lower MAD at each time point, suggesting a better fit compared to the independent model. Therefore, we will proceed to examine the results of the multivariate model.

Plotting the results

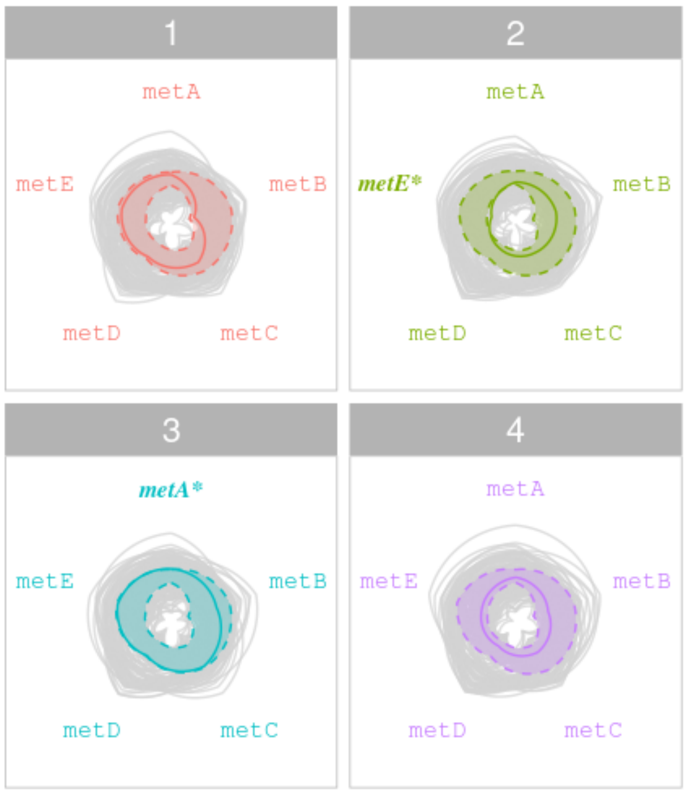
After fitting the BGLM, the user can use the following plotting functions to visualise the results.

Radar plot

The function `radar.plot` visualises the metabolic profile of an individual across all time points and all metabolites. This function requires a **MetaboVariation** object, a specific `interval` width and the `individual` ID to generate the radar plot.

Here's how you can use the `radar.plot` function:

```r
radar.plot(model,interval = 0.95,individual = 150)
```

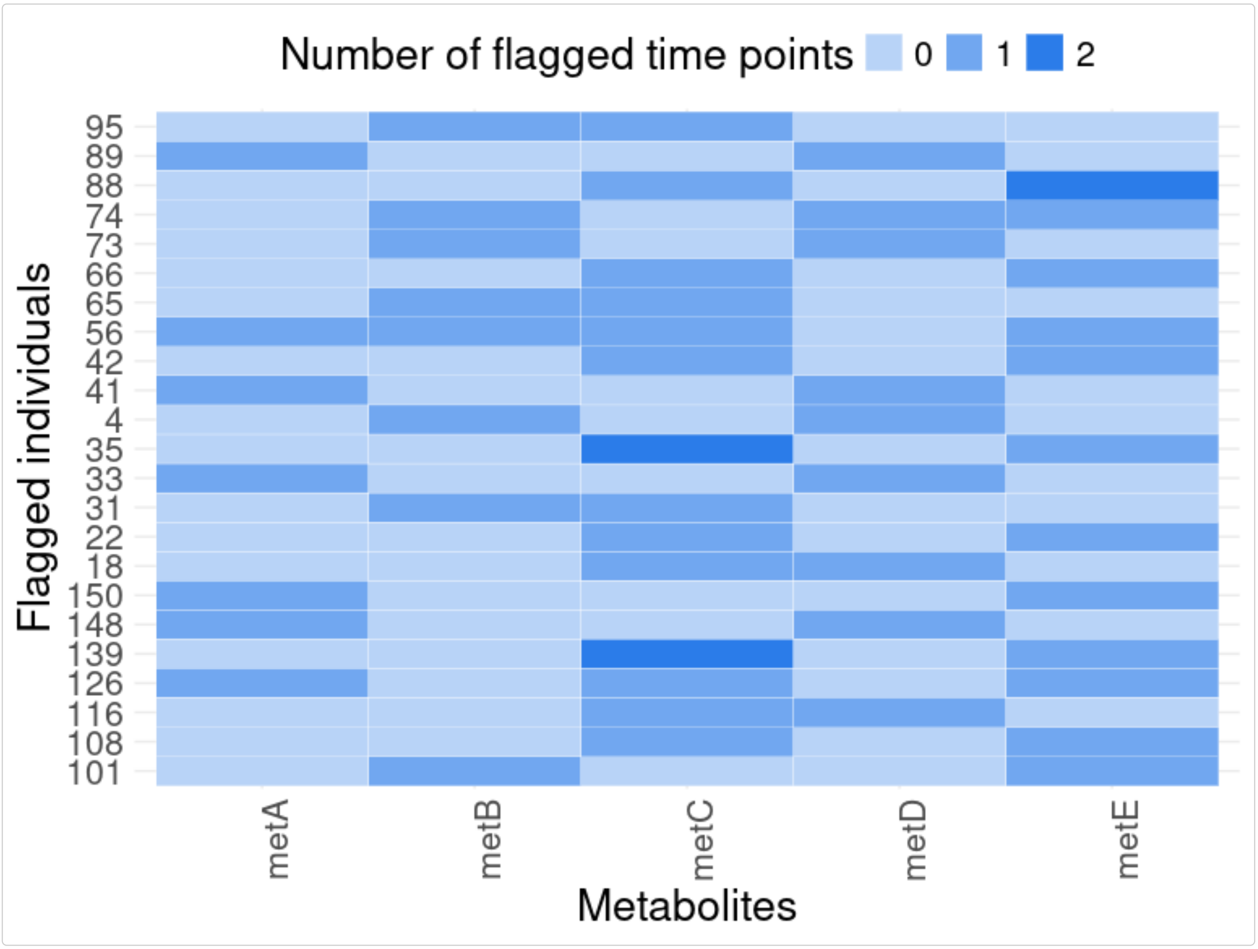


The radar plots display intra-individual variations in metabolite levels across four time points, highlighting significant variations. Metabolites marked with an asterisk (*) are flagged for intra-individudal variation at each respective time point.

Metabolite heatmap

The `metabolite.heatmap` function visualises the flagged individuals across all metabolites. This helps in identifying patterns or variations in certain metabolites. This function needs a **MetaboVariation** object, along with `interval` width and a `threshold` argument to generate the plot. The argument `threshold` indicates the minimum number of metabolites an individual must be flagged in to be shown in the heatmap.

```r
metabolite.heatmap(model = model,interval = 0.975,threshold = 2)
```



The plot above shows the individuals that have been flagged in at least two metabolites. The darker blue shows the metabolites where the individual is flagged in two time points out of four.

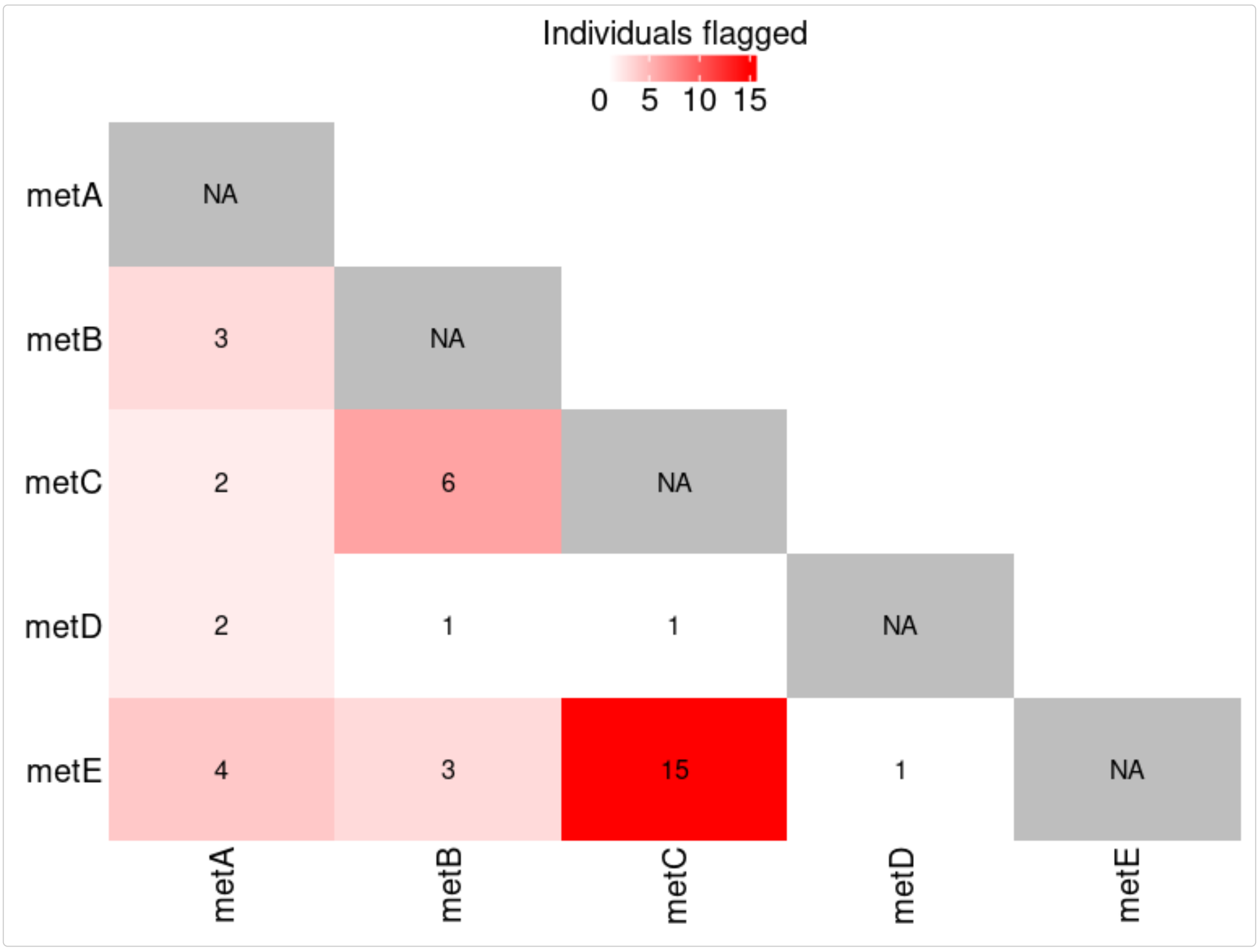
Metabolite-pair heatmap

The `metpair.heatmap` function visualises the number of flagged individuals in pairs of metabolites (in the corresponding row and column) that contributeto the flagging of the individuals within a single time point.

This function needs a **MetaboVariation** object, along with a specific `interval` width to generate the plot. The interval refers to the width of the highest posterior density (HPD) that you want to visualise.

To plot the heatmap, type:

```
metpair.heatmap(model = model,interval = 0.95)
```

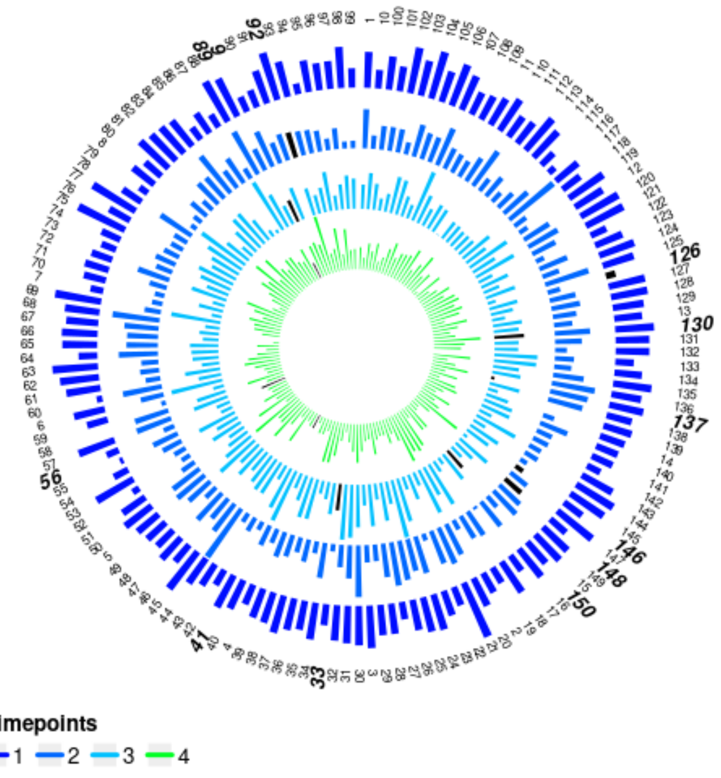


The heat map illustrates the number of individuals flagged for intra-individual variation across pairs of metabolites within a time point. Darker shades of red indicate higher numbers of flagged individuals. Notably, 15 individuals are flagged for the pair metE and metC while 5 individuals are flagged for the pairs metA and metC, and metA and metE. This visualisation underscores the need for a comprehensive multivariate analysis, as multiple metabolite pairs contribute to identifying variations.

Circos plot

The `circos.plot` plot shows the posterior predictive HPD interval for all time points for all individuals for a single metabolite. The outer circle shows the individuals' labels while each inner circle represents a time point. The length of a bar represents the width of the posterior predictive interval and a black bar indicates that a particular individual has been flagged.

```
circos.plot(model = model,metabolite = metabolites[1],interval = 0.975)
```



The circos plot visualises the 97.5% posterior predictive HPD intervals for all individuals across four time points for a single metabolite “metA”. From the plot, 14 individuals are flagged: 2 at time point 1 (blue), 4 at time point 2 (cyan), 6 at time point 3 (green), and 2 at time point 4 (black).

References

[1] Gupta, S., Gormley, I. C., & Brennan, L. (2023). MetaboVariation: Exploring Individual Variation in Metabolite Levels. *Metabolites*, 13(2), 164.

[2] Hadfield, J. D. (2010). MCMC Methods for Multi-Response Generalized Linear Mixed Models: The MCMCglmm R Package. *Journal of Statistical Software*, 33(2), 1–22.

[3] Gelman, A., & Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4), 457–472.