MetaboVariation

Shubbham Gupta, Isobel Claire Gormley, and Lorraine Brennan

Introduction

Most metabolomic biomarker research concentrates on disease biomarkers discovered by comparing patients with the illness to matched controls. However, this approach has limited impact on the early detection of disease or metabolic dysfunction biomarkers.

Variability within individuals arises from various factors, leading to differences in metabolic profiles. 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 metabolite. 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 function MCMCglmm from the MCMCglmm package [2], which fits the BGLM. The approach is flexible, allowing for the consideration of multiple posterior predictive interval widths, allowing 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.

Assuming that the user has the **MetaboVariation** package installed, the user first needs to load the package:

```
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 different time points. The covariates sex, age and BMI are also available for these individuals.

To load the data and examine its column type:

```
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

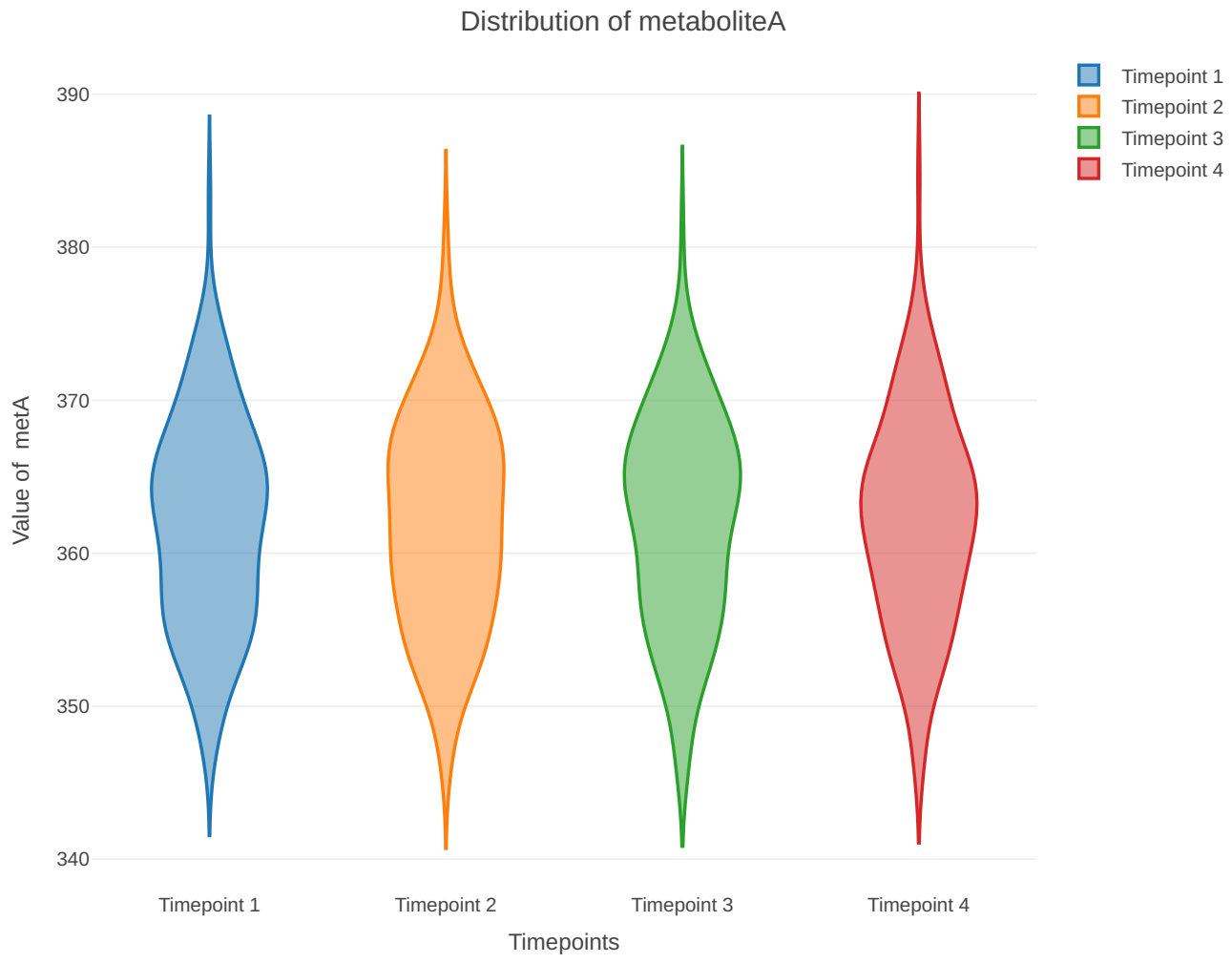
To extract the names of metabolites that are present in the data, the function `get.metabolites` reads the names of columns that contain metabolite values and provides a list of unique metabolites. The user should not pass any unnecessary column names (e.g., any column names that relate to covariate or subject 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 variations 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.

```
# Initialise the prior mean vector with zeros, one for each metabolite
prior_mean <- rep(0, length(metabolites))
# Initialise the prior standard deviation vector with ones, one for each metabolite
prior_sd <- rep(1, length(metabolites))
# Initialise the prior residual covariance matrix as an identity matrix
prior_R <- diag(1,length(metabolites))
# Initialise 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_sd^2)),
  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 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 pacakge calculate the priors for B based on fitting an independent BGLM to the metabolite data. As for R and G, the scale matrixis 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. This function uses BGLM to flag individuals with intra-individual variations in their metabolite levels using repeated measurements. The function requires the data, the names of covariate columns (if any), and the subject 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. The `MetaboVariation` function checks if the observed value of the metabolite for an individual is outside the posterior predictive highest posterior distribution (HPD) interval, and if so, the individual is flagged. The width of the HPD interval is set by the `cutoff` argument where the default is a vector containing three values `c(0.95,0.975,0.99)` representing 95%, 97.5% and 99% respectively. The function returns summarised results that contains the upper and lower limit of HPD interval and whether the individual is flagged or not for each cutoff value.

Multivariate dependent model

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

To fit the model,type:

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

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

The function returns a list that contains the following values:

- model - contains BGLM model outputs.
- type - shows which model is has been fitted.. It can be either “dependent” or “independent”.
- significant_covariates - a data frame containing the estimates and 95% confidence intervals for each covariate across all metabolites along with a binary flag indicating whether each covariate is significant for the corresponding metabolite.
- result - a data frame containing the posterior predictive distributions of the cohort. The result contains the mean, `cutoff` % credible interval width, tails of the credible interval and the observed value of the metabolite for the individual for that time point. The result also has a binary flag that shows whether the observed value lies within the HPD interval or not where 1 denotes the observed value lies in the interval and 0 denotes the observed value lies outside the interval.
- 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.

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 model includes three `cutoffs` (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,])
#>                  fit   lwr0.95   upr0.95  lwr0.975  upr0.975   lwr0.99
#> 11 1 metB   46.78879  43.99568  49.77645  43.44596  49.81772  42.80774
#> 115 1 metD  81.44570  78.83963  84.19326  78.18750  84.40046  78.18750
#> 110 1 metE 300.25089 297.41291 303.31282 297.17539 304.28559 296.15023
#> 122 1 metA 369.88784 367.29350 372.82330 366.84724 373.37293 366.14813
#> 126 1 metA 368.56601 365.67265 371.44141 365.09198 372.08393 364.58272
#> 126 1 metC  96.31870  93.55330  99.12960  92.99060  99.58455  92.96100
#>              upr0.99  original flag0.95 flag0.975 flag0.99
#> 11 1 metB   50.05890  50.63484        1         1        1
#> 115 1 metD  85.23249  78.75315        1         0        0
#> 110 1 metE 304.34851 296.13965        1         1        1
#> 122 1 metA 373.42884 373.08306        1         0        0
#> 126 1 metA 372.73409 371.55486        1         0        0
#> 126 1 metC 100.57650  99.34041        1         0        0
```

Independent model

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

To fit the model, type:

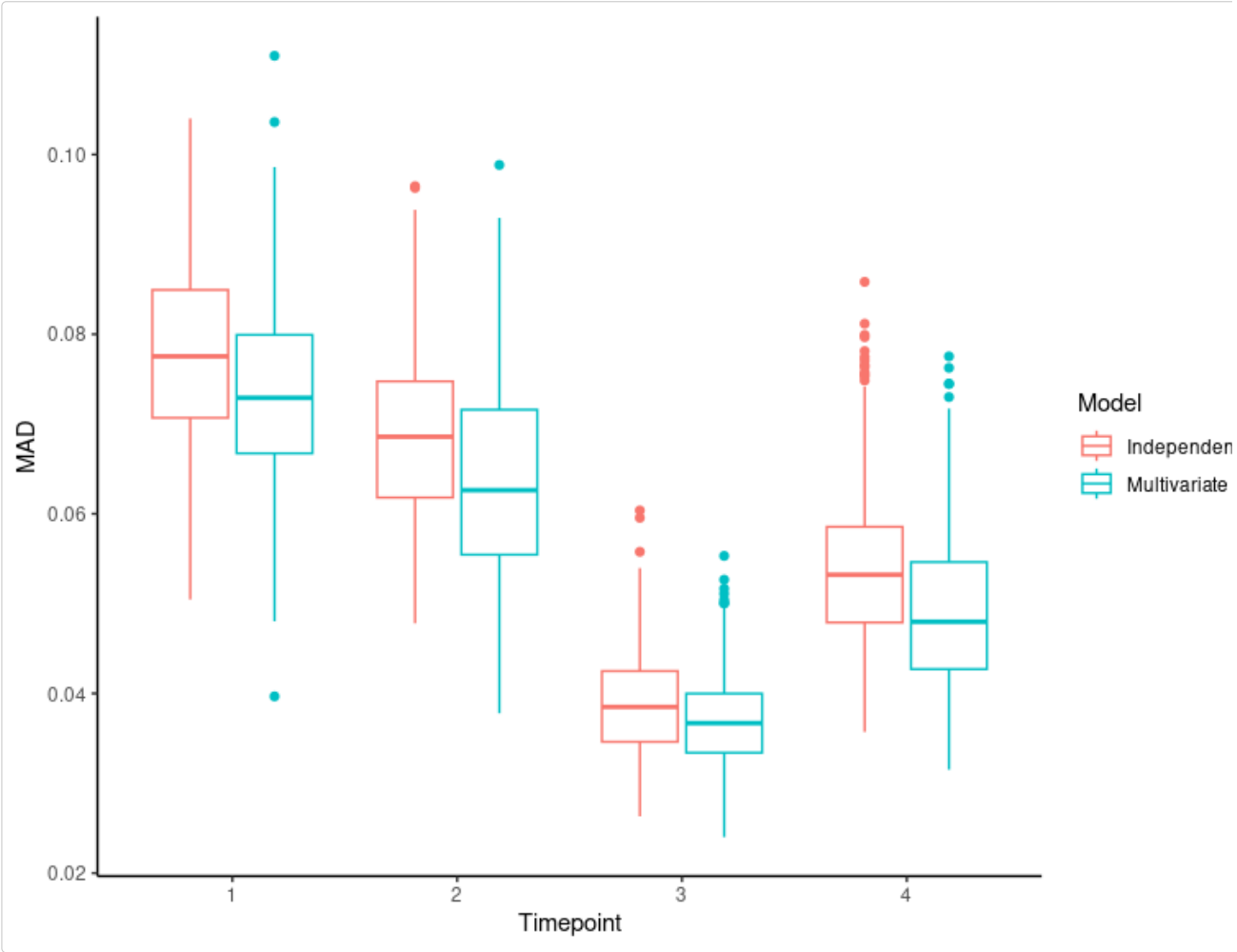
```
independent_model = MetaboVariation(data = metabol.data, individual_ids = "Individual_id",
          metabolite = metabolites,covariates = c("SexM.1F.2","Age","BMI"),
```

```
          cutoff=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 fit the data. The function `predictive_check` performs a posterior predictive check for the two models. It generates 100 replicate datasets from the posterior predictive distributions under both models. 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 replicate datasets using model 1 (red) and model 2 (blue), over each time points.

```
predictive_check(data = metabol.data, model1 = model,model2 = independent_model,
          timepoints = c(1,2,3,4),metabolites = metabolites, replication = 500,
          model1_name = "Multivariate", model2_name = "Independent")
```



The multivariate model shows lower MAD mean across 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

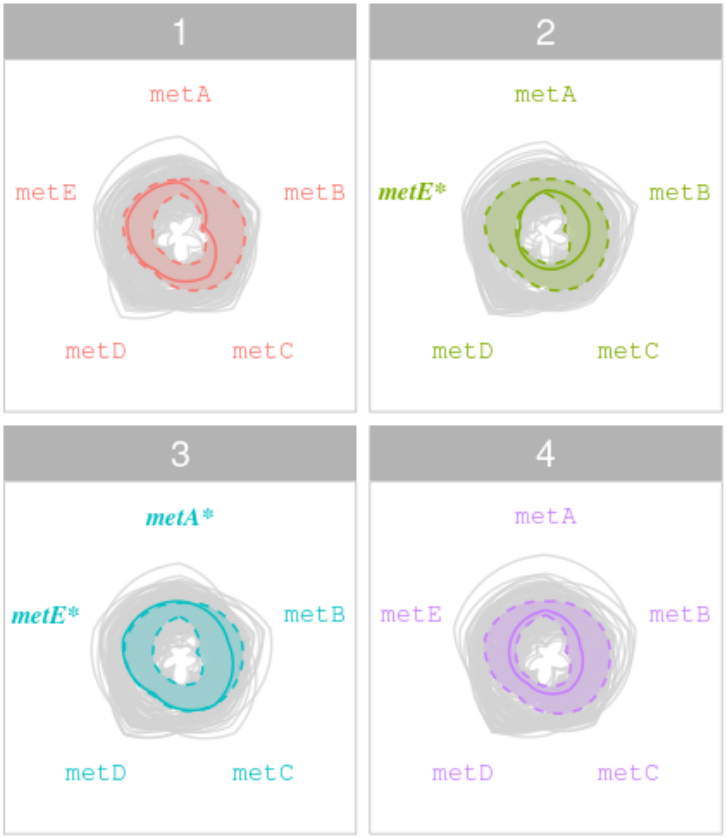
After fitting the BGLM, the user can use the following plotting functions to visualise the results.

Radar plot

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

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

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

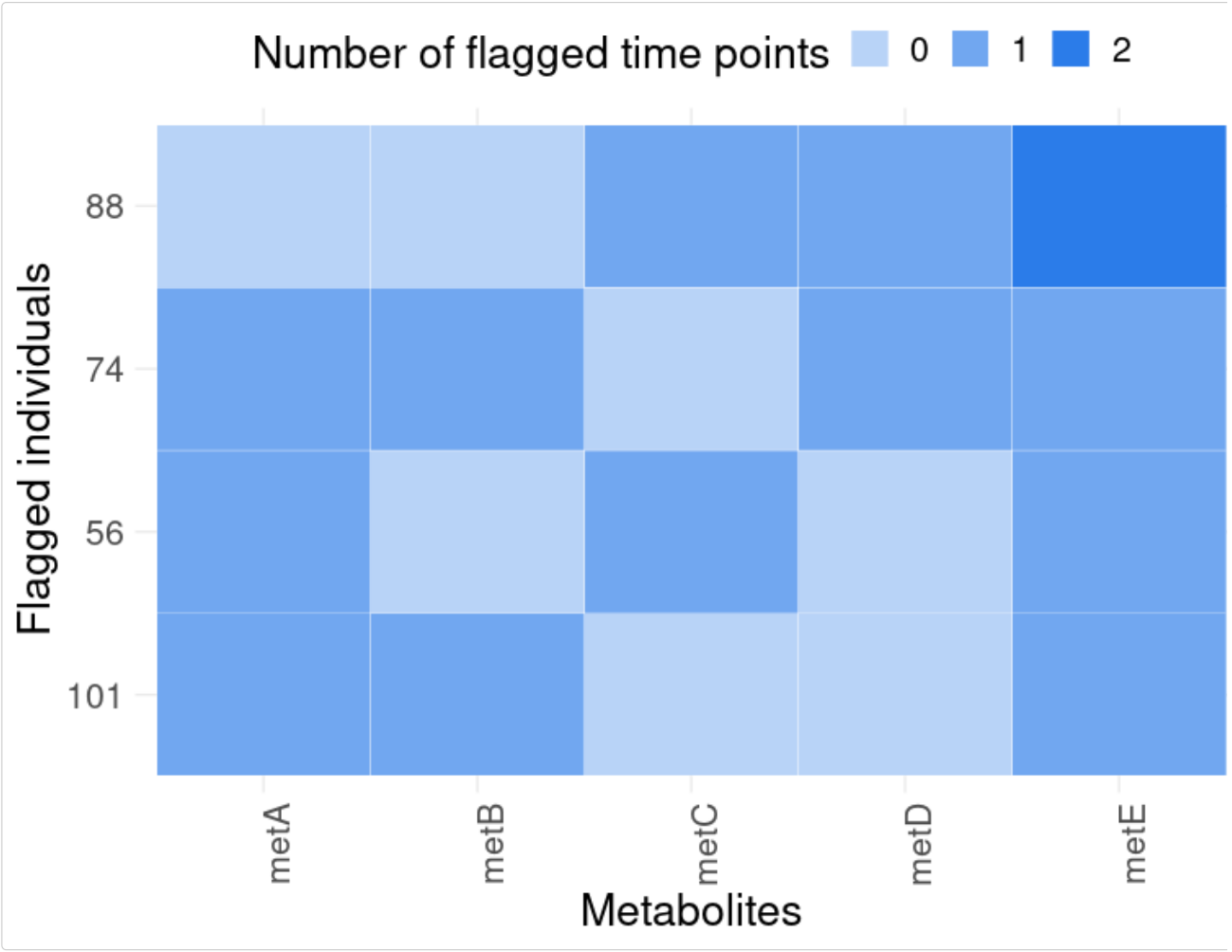


The radar plots display intra-individual variations in metabolite levels across four time points, highlighting significant deviations. Metabolites marked with an asterisk (*) are flagged for intra-individudal variation at each respective time point.

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` and a `threshold` to generate the plot. The argument `threshold` indicates the minimum number of metabolites an individual must be flagged in to be involved in the heatmap.

```
metabolite.heatmap(model = model,interval = 0.975,threshold = 3)
```



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

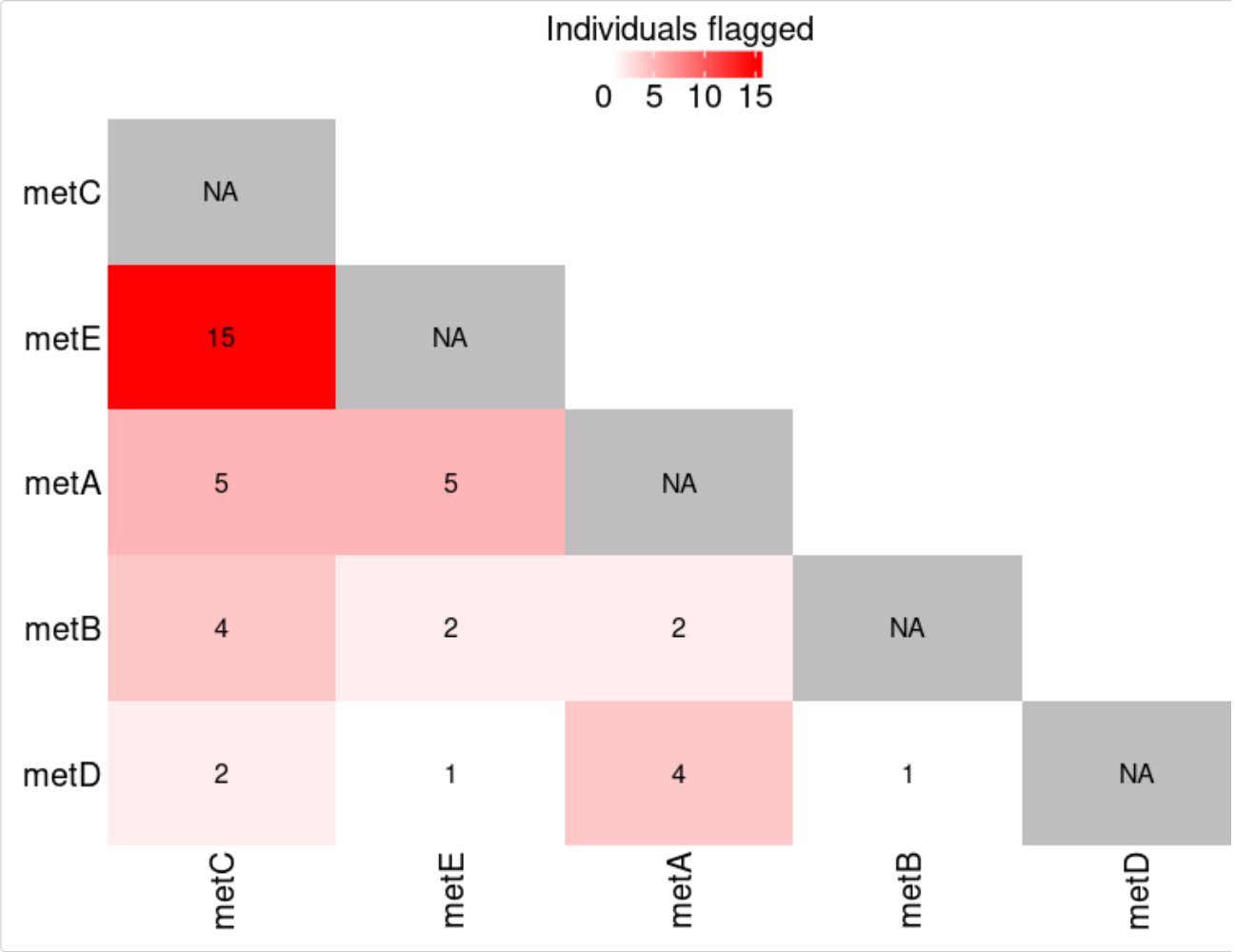
Metabolite-pair heatmap

The `metabolitepair.heatmap` function visualises the number of flagged individuals in the pair of metabolites (in the corresponding row and column) contributes to flagging the individuals within a single time point.

This function needs a **MetaboVariation** object, along with a specific `interval` and `threshold` to generate the plot. The interval refers for the highest posterior distribution (HPD) that you want to visualize and the threshold indicates a positive integer that refers to the minimum number of metabolites an individual must be flagged in a time point to be included in the heatmap.

To plot the heatmap, type:

```
metabolitepair.heatmap(model = model,interval = 0.95,threshold = 1)
```

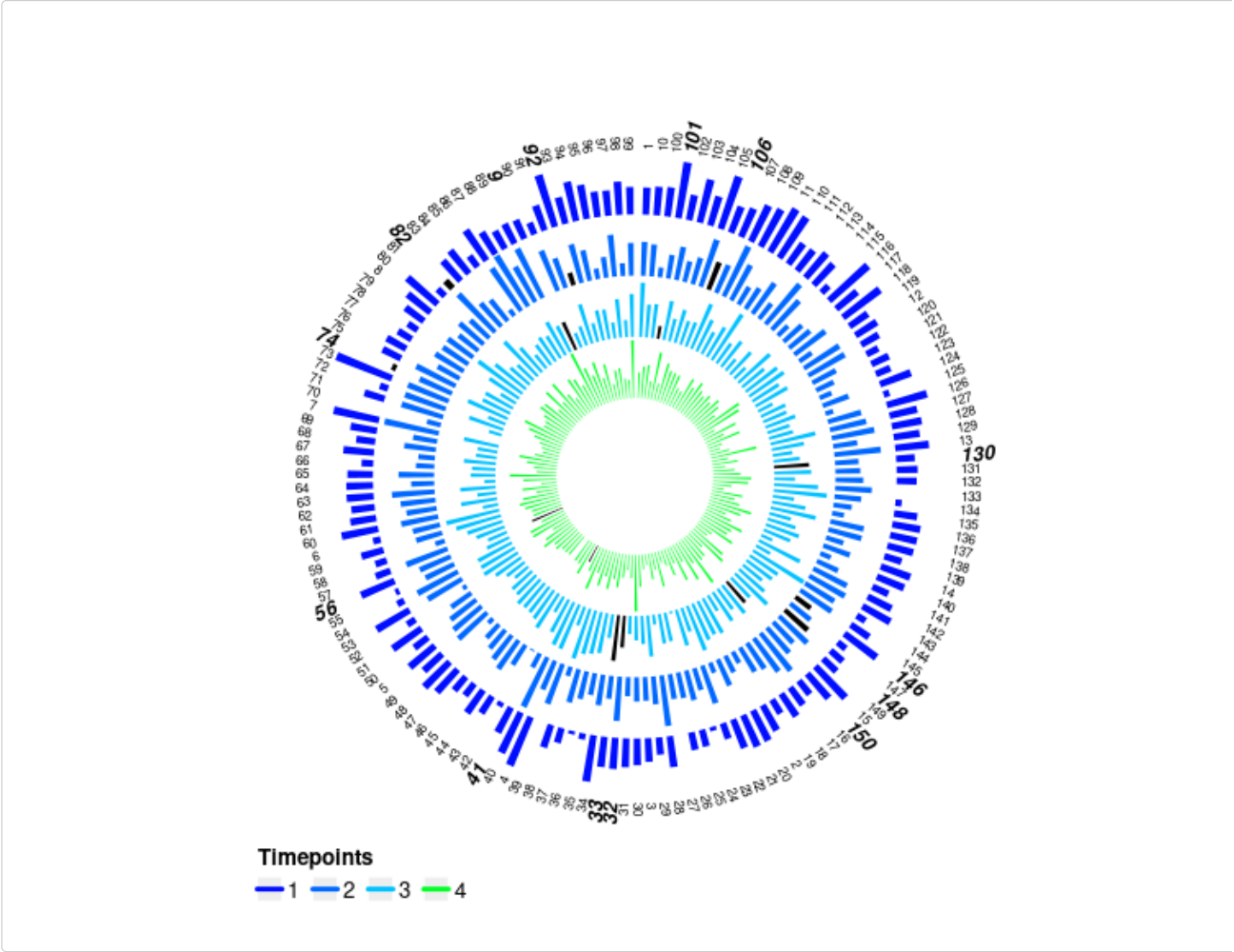


The heat map illustrates the number of individuals flagged for intra-individual variation across pairs of metabolites within a time point. Darker shades of red indicate higher numbers of flagged individuals. Notably, 15 individuals are flagged for the pair metE and metC while 5 individuals are flagged 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 the 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 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 IC, Brennan L. MetaboVariation: Exploring Individual Variation in Metabolite Levels. Metabolites. 2023; 13(2):164. https://doi.org/10.3390/metabo13020164

[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. https://doi.org/10.18637/jss.v033.i02

[3] Gelman A, Rubin DB. 1992 Inference from iterative simulation using multiple sequences. Stat. Sci. 7, 457–472. (doi:10.1214/ss/1177011136)