MetaboVariation

```
library(MetaboVariation)
```

Introduction

Most metabolomic biomarker research concentrates on disease biomarkers discovered by comparing patients with the illness to matched controls. This method is useful in the development of the diagnostic biomarkers. However, it has had only a 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. The objective of the **MetaboVariation** package is to facilitate implementation of a methodology for identifying individuals with significant intra individual variation in their metabolite levels using repeated measures data. A second objective is to provide visualisation of the results.

MetaboVariation employs a multivariate model using Bayesian generalised linear models (BGLM) that takes correlations among metabolites into account to better understand the variations in metabolite levels. The **MetaboVariation** package is built on the function MCMCglmm from the MCMCglmm package [1] that fits the BGLM. The approach allows us to flag individuals whose observed metabolite levels are outside their individual-level posterior predictive interval at a given time point. The approach is flexible, allowing for the consideration of multiple posterior predictive intervals, allowing a detailed interpretation 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 **MetaboVariaton** package has the following dependencies which will be installed along with the package if not already installed on the machine:

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

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 names.

```
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 id of individuals is 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 metabolite level of metabolite **X** at time point **Y**.

Extracting metabolite names

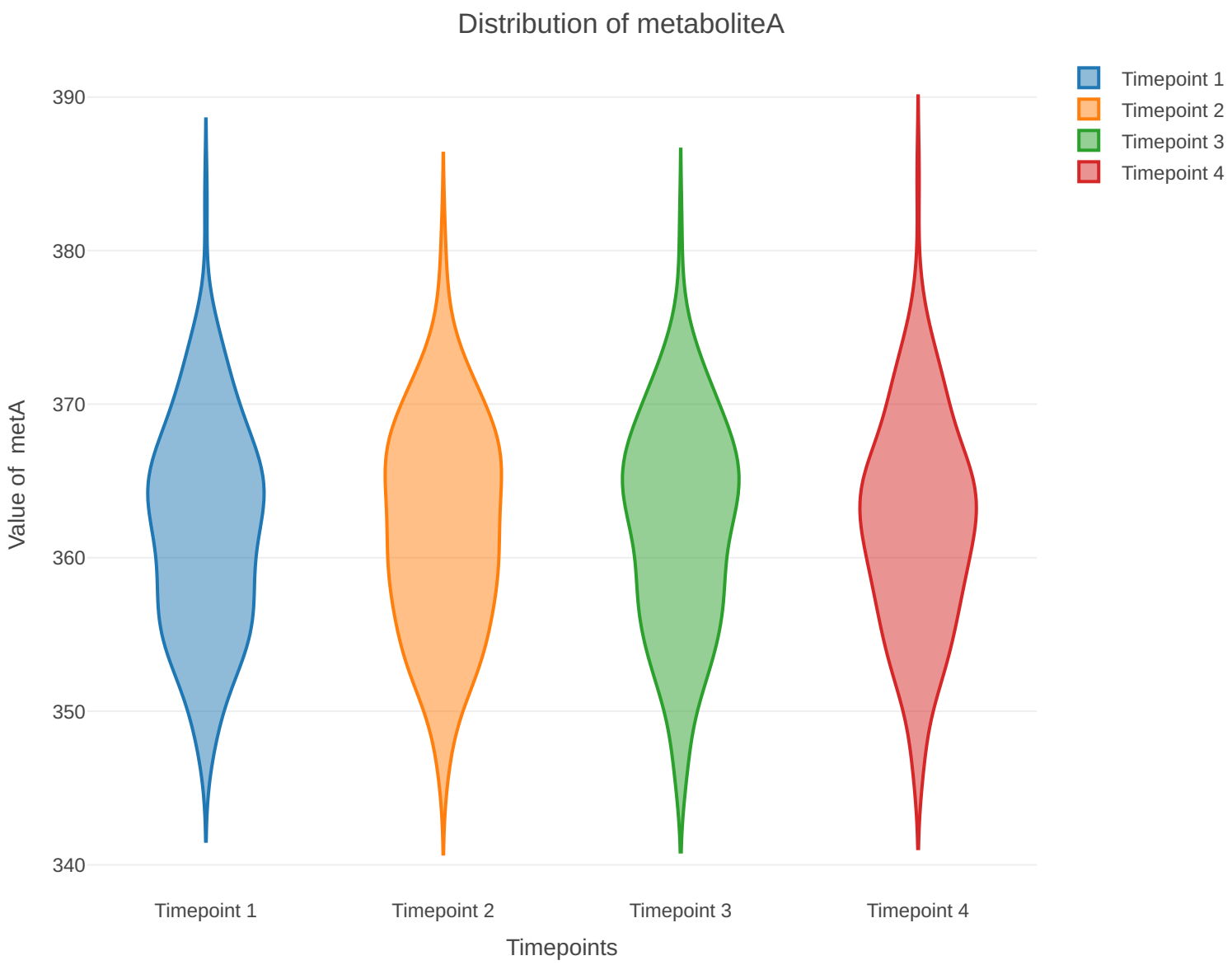
To extract the names of metabolites that are present in the data, the function `get.metabolites` reads the names of columns that contain metabolite values and provides a list of unique metabolites. The user should not pass any unnecessary column names (e.g. any column names that relate to covariate or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

```
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 distribution of metabolites

The user can plot the distributions of metabolites for each timepoint using the function `metabolite.plot`. The user can pass either a single metabolite or multiple metabolites to the function.

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



Modelling the data to flag individuals

The main function `MetaboVariation` performs the modelling. The function uses a Bayesian generalised linear model (BGLM) to identify individuals with significant variations in their metabolite levels using repeated measurements. As arguments, the function requires the data along with the names of covariate columns (if any) and the subject id column.

To find the individuals with significant variation, the BGLM provides the posterior predictive distribution for every individual at each time point, and 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 of the cutoff value passed. By putting `type="dependent"`, it make sure that `MetabVariation` models the data using multivaraite model that takes correlation between metabolites into account.

To execute the model on the data.

```
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 intermediate statistical models.
- type - shows which modelling is done. It can be either “dependent” or “independent”
- significant_covariates - a list of covariates that have a significant relationship (at fixed 95% level) with the metabolite levels in individuals.
- result - a summary of the posterior predictive distribution 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 [2]. 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.

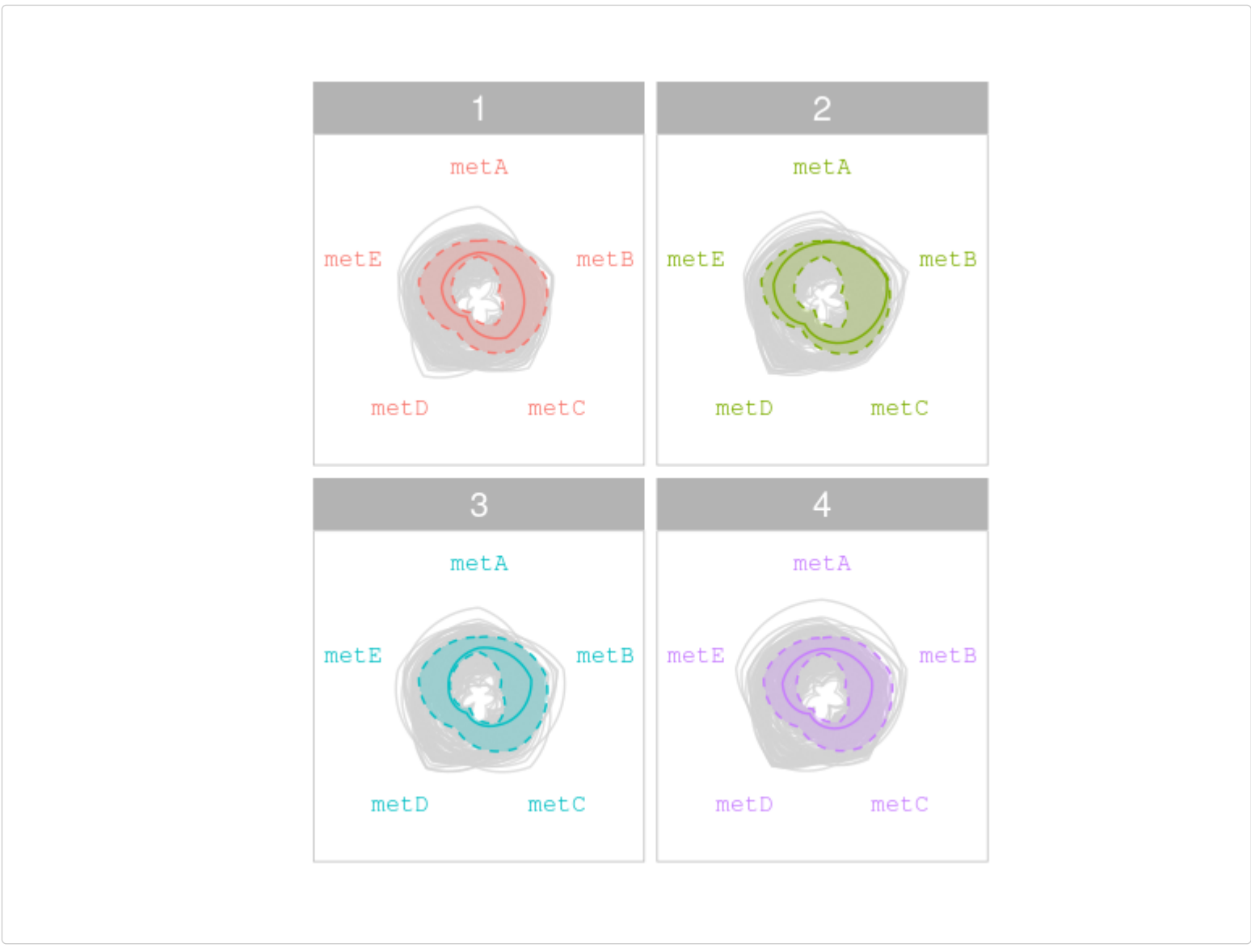
Plotting the results

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

The function `radar.plot` visualise the metabolic profile of an individual across all timepoints. This function requires the input of a **MetaboVariation** object, along with a specific `cutoff` value and the `individual` ID to generate the radar plot.

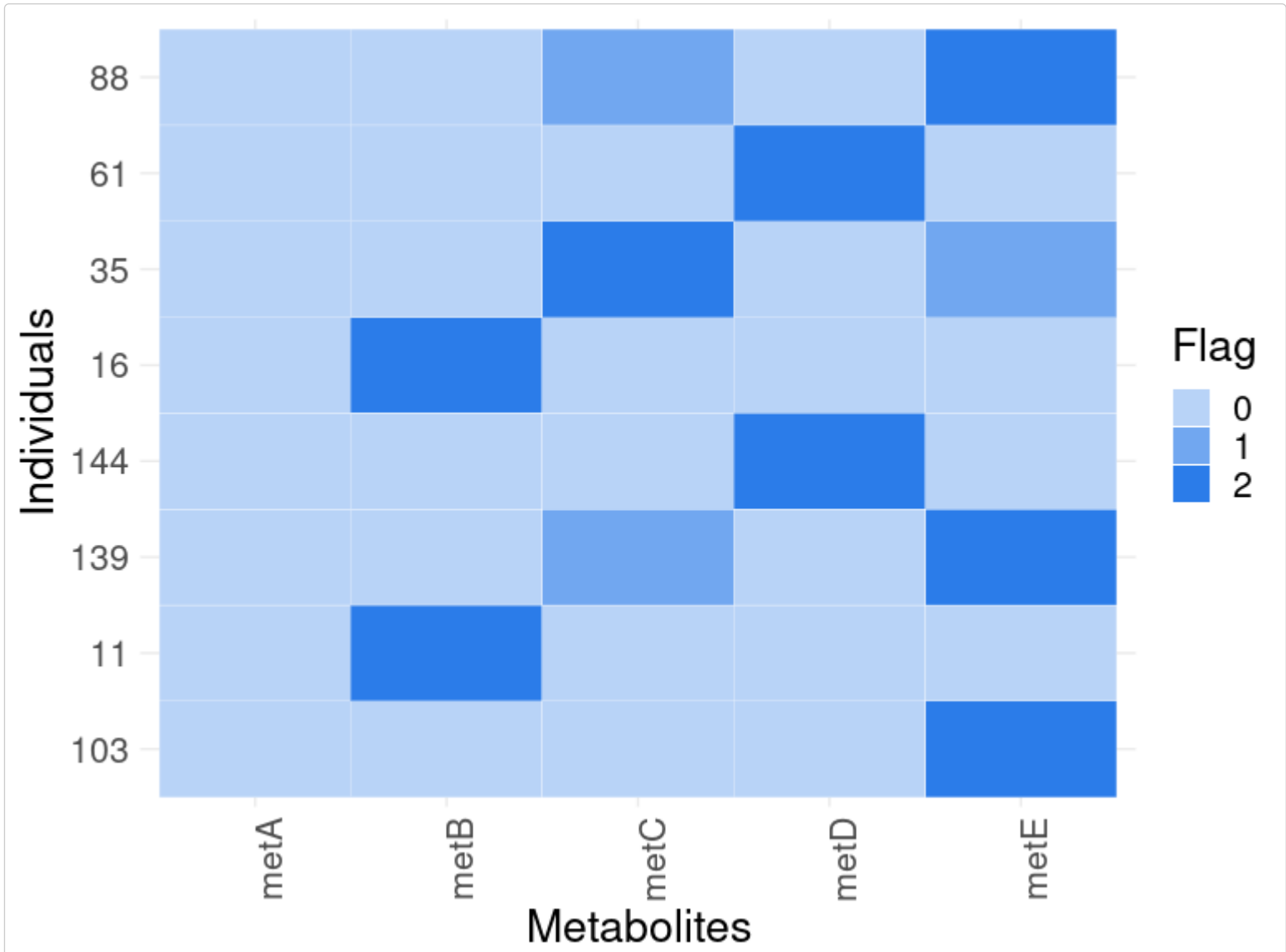
Here's how you can use the `radar.plot` function:

```
radar.plot(model,interval = 0.975,individual = 106)
```



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 a specific `cutoff` and `threshold` to generate the plot.

```
metabolite.heatmap(model,0.975,cutoff = 50,threshold = 5)
```



Independent modelling

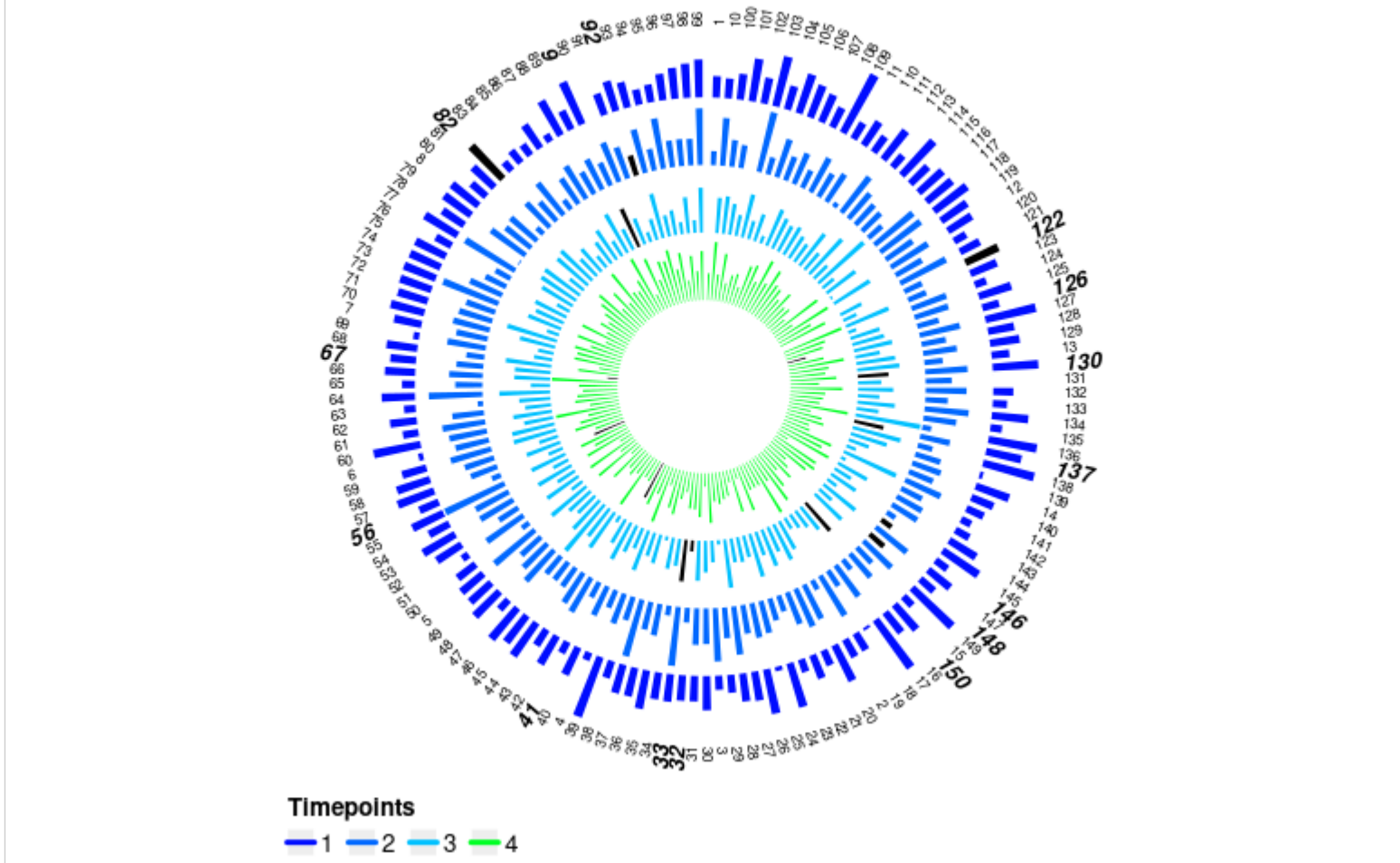
Once you have a list of certain metabolites you want to model or alternate model where you do not want to treat each metabolite independently, you can use the following code. By setting `type="independent"`, it tells `metaboVariation` to treat each metabolite independently.

```
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")
#> [1] "Calculating prior"
#> [1] "Building model"
#> [1] "Sampling Posterior predicive distribution"
```

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 `cutoff`% posterior predictive interval and a black bar indicates that a particular individual has been flagged.

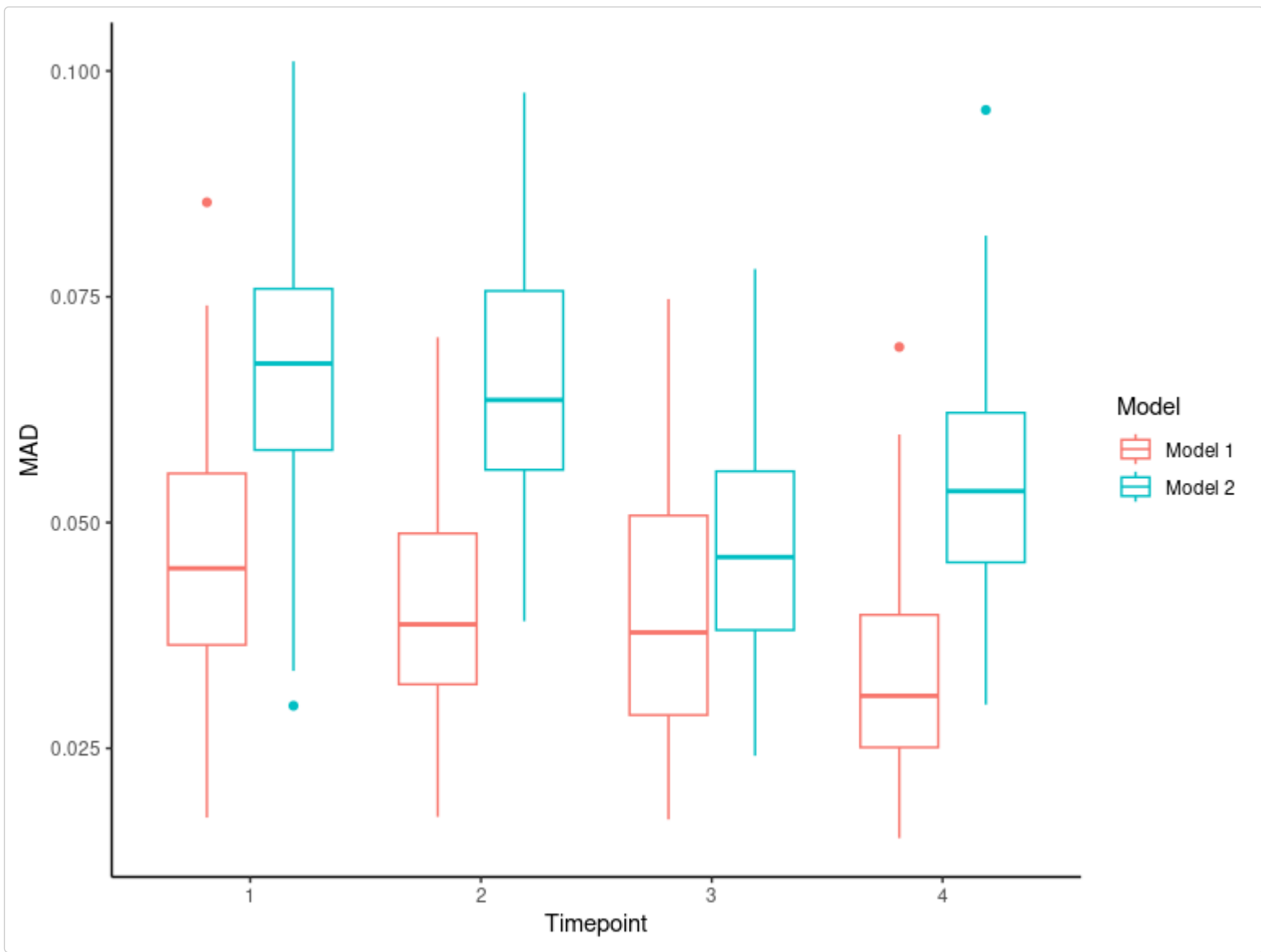
```
circos.plot(independent_model,metabolites[1],0.975)
```



Posterior predictive check

Now that we have two different models, one that takes correlation among metabolites into account and one that treats metabolites as independent, it is better to perform a posterior predictive check to see which model understands the data better. The function `predictive_check` performs a posterior predicve check between two models. It generates replicated data set using both models, default is 100. It then provides a box plot that shows mean absolute difference (MAD) between correlation matrices of the original dataset and those obtained from replicated datasets using model 1 (red) and model 2 (blue), over timepoints given.

```
predictive_check(metabol.data,model,independent_model,c(1,2,3,4),metabolites)
```



References

[1] 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

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