An Introduction Mixture Model Normalization with the `metabomxtr` Package

Michael Nodzenski, Anna C. Reisetter, Denise M. Scholtens

January 4, 2019

1 Introduction

Controlling technical variability in metabolite abundance, or normalization, is a critical step in the analysis and interpretation of non-targeted gas-chromatography/mass-spectrometry (GC/MS) data. In large scale metabolomics studies requiring sample processing in many analytic batches, technical artifacts due to batch and run-order within batch are common. In these cases, repeated assays of a set of control samples may be used to estimate and account for these artifacts. The `metabomxtr` package implements a mixture model normalization approach via the function `mixnorm` for studies implementing this quality control measure. Based on control sample variability, `mixnorm` allows for per-metabolite modeling of both batch and run-order effects, while allowing for bach specific thresholds of metabolite detectability.

2 Sample Mixture Model Normalization

The following commands demonstrate typical usage of `mixnorm`. First, load the package.

```
>  library(metabomxtr)
```

Next, load `euMetabData`, a sample data frame containing metabolite data for a total of 40 mother-baby pairs of Northern European ancestry. A total of 3 blood samples are included for each pair: mother fasting, mother 1-hour, and newborn cord blood. Mother samples were obtained during an oral glucose tolerance test (OGTT) at 28 weeks gestation, and baby samples were collected at birth. Sample types are indicated by row names, with 'mf' and 'm1' indicating maternal fasting and 1-hour samples, respectively, and 'bc' indicating baby samples. Note that while `euMetabData` is a data frame, `mixnorm` also accommodates metabolite data in matrix and ExpressionSet objects.

```
>  data(euMetabData)
>  class(euMetabData)

[1] "data.frame"

>  dim(euMetabData)

[1] 120   6

>  head(euMetabData)

        batch pheno betahydroxybutyrate pyruvic_acid malonic_acid aspartic_acid
MBP1_mf     1   MOM            20.14544     18.47593     15.52949      17.27488
MBP1_m1     1   MOM            19.30312     18.55794     16.89087      14.42220
MBP1_bc     1  BABY            22.83122     17.79843     14.77859            NA
MBP2_mf     1   MOM            20.55216     17.46991           NA            NA
MBP2_m1     1   MOM            19.76286     18.21836     16.13184            NA
MBP2_bc     1  BABY            21.62902     16.05125     14.58549            NA
```

Also load `euMetabCData`, a data frame containing GC/MS data from separate mom and baby control pools. Control pool aliquots were run at the beginning, middle and end of each batch with placement indicated by -1, -2 and -3 appended to the sample name, respectively.

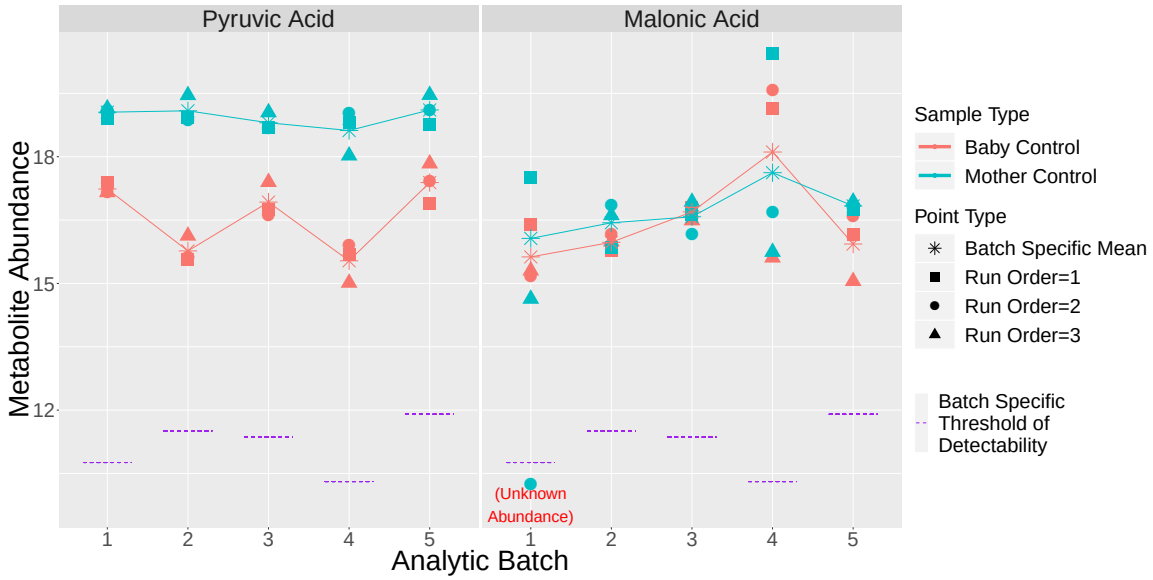
```
> data(euMetabCData)
> class(euMetabCData)

[1] "data.frame"

> dim(euMetabCData)

[1] 30  6

> head(euMetabCData)
```

	batch	pheno	betahydroxybutyrate	pyruvic_acid	malonic_acid	aspartic_acid
B-01-1	1	BABY	23.03775	17.38712	16.40161	NA
B-01-2	1	BABY	22.96725	17.16013	15.17785	NA
B-01-3	1	BABY	23.02668	17.15702	15.30278	NA
B-02-1	2	BABY	22.60945	15.55872	15.76774	NA
B-02-2	2	BABY	22.94031	15.62062	16.16459	12.98353
B-02-3	2	BABY	23.25504	16.12824	15.98942	NA

Pyruvic acid and malonic acid are included in both example data sets. We'll assume they are of analytical interest, and a define a character vector of the corresponding column names.

```
> ynames<-c('pyruvic_acid','malonic_acid')
```

Now we'll plot metabolite abundances from the control data set. In the absence of technical variability, we would expect to see constant mean abundance across batches for each metabolite. Also indicated in the plots are batch specific thresholds of metabolite detectability, based on experimental evidence not available here.



Both mother and baby control samples show considerable variability within and across batches, including one instance where abundance fell below the detectable threshold. To account for these technical artifacts, we will use mixture model normalization implemented in the function `mixnorm`. This function takes as required arguments a character vector of target metabolite column names, the name of the variable corresponding to analytic batch in the input data objects, a data object (data frame, matrix, or ExpressionSet) with quality control data, a data object with experimental data, and a numeric value corresponding to outlier criteria. More specifically, this numeric value indicates the maximum number of standard deviations from the mean metabolite abundance an observation may be and still be considered non-outlying. Any observations falling outside this threshold will not be used in estimating batch effects and other technical artifacts. In our experience, 2 standard deviations usually performs well, and the argument therefore defaults to 2. In the example data sets, the variable corresponding to analytic batch is 'batch', the target metabolite columns are 'pyruvic_acid' and 'malonic_acid' (specified previously), the control data set is `euMetabCData`, and the experimental data set is `euMetabData`. By default, `mixnorm` implements a mixture model with batch as the only covariate. For this analysis, we also want to account for sample phenotype (mother vs. baby), and can do this by specifying a mixture model formula including both batch and phenotype. Note that `mixnorm` will not run if mixture model covariates are missing values. Additionally, we will specify the experimentally determined thresholds of metabolite detectability in optional argument `batchTvals`. If not specified, the default detectable batch threshold is set to the minimum observed metabolite abundance for that batch, across all metabolites of analytic interest. Note this may result in obtaining different results for the same metabolite depending on the other metabolites entered as part of argument `ynames`. Most often, this manifests when running `mixnorm` on a subset of the full metabolite group. In these cases, the user needs to manually calculate the minimum observed metabolite abundance across all metabolites of interest for each batch, and enter that vector for `batchTvals`. Because of this, in general, we recommend running `mixnorm` on the full set of metabolites of interest.

```
> #execute normalization
> euMetabNorm <- mixnorm(ynames, batch="batch", mxtrModel=~pheno+batch|pheno+batch,
+                                batchTvals=c(10.76,11.51,11.36,10.31,11.90), cData=euMetabCData,
+                                data=euMetabData, qc.sd.outliers=2)
```

The output of `mixnorm` is a list of four data frames. The first, `normParamsZ`, contains parameter estimates for the variables included in the mixture model for each metabolite specified. All estimates except for the intercept are subtracted from the raw metabolite values to produce the normalized data.

```
> euMetabNorm$normParamsZ
```

	zInt	z_phenoMOM	z_batch2	z_batch3	z_batch4	z_batch5
pyruvic_acid	16.99942	2.290157	-0.7162666	-0.2801026	-0.8823561	0.1029673
malonic_acid	15.72895	0.184694	0.3834681	0.8148275	0.9723654	0.5622411

The second element of the output list, `ctlNorm`, contains normalized values for the control samples.

```
> head(euMetabNorm$ctlNorm)
```

	pyruvic_acid	malonic_acid
B-01-1	17.38712	16.40161
B-01-2	17.16013	15.17785
B-01-3	17.15702	15.30278
B-02-1	16.27499	15.38428
B-02-2	16.33689	15.78112
B-02-3	16.84450	15.60595

The third element of the output list, `obsNorm`, contains normalized values for the experimental samples. Note that when metabolite abundance falls below the detectable threshold, indicated by missing metabolite values, values will remain missing in the normalized data set.

```
> head(euMetabNorm$obsNorm)
```

	pyruvic_acid	malonic_acid
MBP1_mf	16.18577	15.34480
MBP1_m1	16.26779	16.70618
MBP1_bc	17.79843	14.77859
MBP2_mf	15.17976	NA
MBP2_m1	15.92820	15.94714
MBP2_bc	16.05125	14.58549

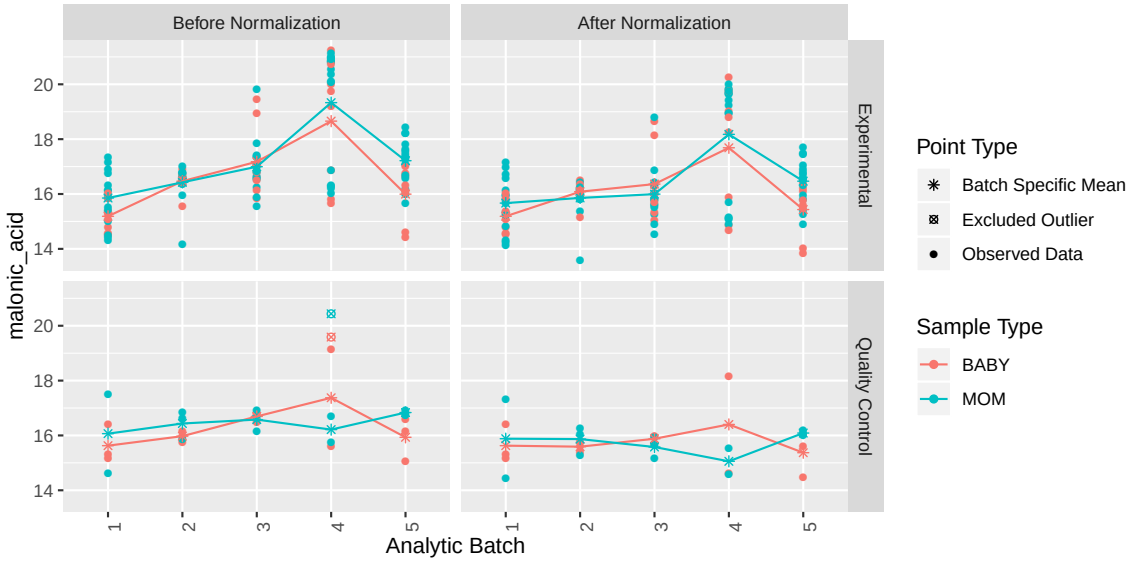
The fourth element of the output list, `conv`, contains information on whether models converged (indicated by a zero in the `conv` column) and whether effects for predictor variables could not be estimated.

```
> head(euMetabNorm$conv)
```

```
           .id conv
1 pyruvic_acid    0
2 malonic_acid    0
```

After normalization, the function `metabplot` can be used to assess how mixture model normalization performed. The function takes as arguments a metabolite column name (which must be present in all input data frames), a character indicating the name of the batch variable, and data frames of raw (non-normalized) experimental data, raw quality control data, normalized experimental data, and normalized quality control data. Optionally, the user can also specify a character indicating a variable to be used to group and color observations in plots. Last, the function requires numeric outlier thresholds for both the raw quality control data and the normalized experimental data. As with `mixnorm`, both arguments indicate the maximum number of standard deviations from the mean metabolite abundance considered to be non-outlying. For the argument indicating the quality control sample threshold (`cont.outlier.sd.thresh`), the same value used for `qc.sd.outliers` in `mixnorm` should be input and the argument defaults to 2. These points will be indicated in the plots as "Excluded Outliers" and are the observations that were not used in estimating batch effects. The argument indicating the normalized experimental data threshold defaults to 4. These points will appear in the normalized experimental data plot as "Potential Outliers". These observations may represent true outlying metabolite levels after control of technical variability, and the user may want to exclude these from downstream analysis (if using `mxtrmod` to analyze data, these points can be automatically excluded by specifying the argument `remove.outlier.sd`). In the example plots, following normalization, mean metabolite abundance values are much more stable across batches in the control samples. In the experimental data, mean abundances are more variable, even after normalization. This is expected, as characteristics of biological interest are not expected to be uniform across batches, and normalization aims to preserve this true biological variability.

```
> plot.list<-lapply(ynames, metabplot, batch="batch", raw.obs.data=euMetabData, raw.cont.data=euMetabCData,
+                           norm.obs.data=euMetabNorm$obsNorm, norm.cont.data=euMetabNorm$ctlNorm,
+                            color.var="pheno", cont.outlier.sd.thresh=2, norm.outlier.sd.thresh=4)
> #just show plot for one of the metabolites
> plot.list[[2]]
```



3 Function Options

3.1 Removing Model Corrections

By default, `mixnorm` subtracts the effects of all variables included in the mixture model from the raw data to produce the normalized data. However, in certain instances, it may be desirable to include covariates in the mixture model to accurately estimate batch effects, but not actually remove the effects of those covariates. For instance, in the plots above, mother samples tended to have higher levels of pyruvic acid than baby samples across batches. We can account for sample type (mom vs. baby) in estimating batch effects while preserving metabolite variability based on sample type by specifying the name of the covariate column (or a character vector of names) to optional argument `removeCorrection` as follows:

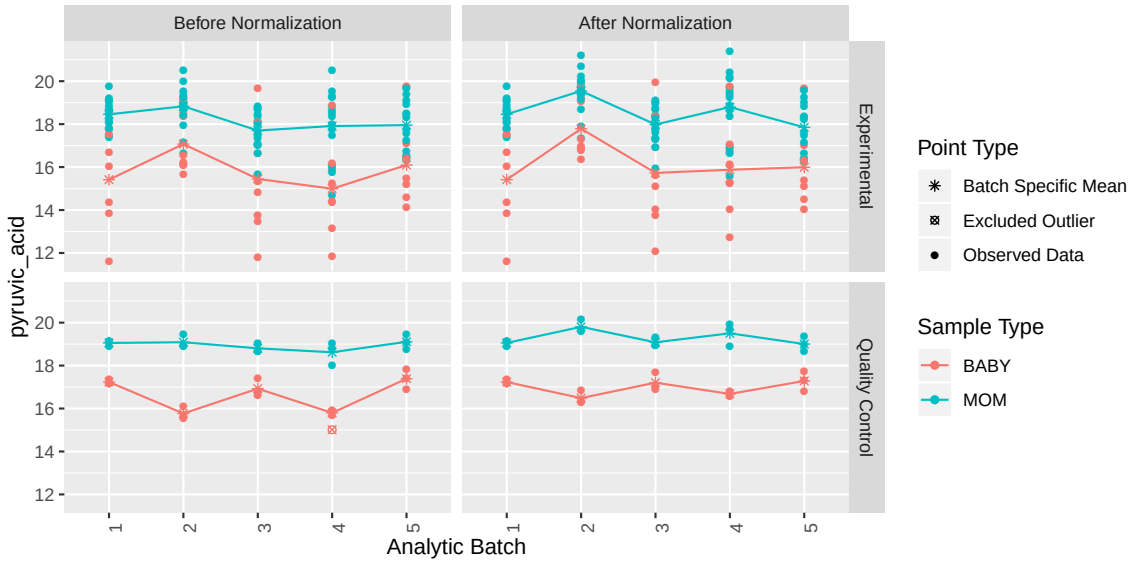
```
> euMetabNormRC <- mixnorm(ynames, batch="batch", mxtrModel="pheno+batch|pheno+batch,
+                                  batchTvals=c(10.76,11.51,11.36,10.31,11.90), cData=euMetabCData,
+                                  removeCorrection="pheno",data=euMetabData)
```

The parameter estimates in `normParamsZ` will be identical to those had `removeCorrection` not been specified:

```
> euMetabNormRC$normParamsZ[rownames(euMetabNormRC$normParamsZ)=="pyruvic_acid", ]

                zInt z_phenoMOM   z_batch2   z_batch3   z_batch4  z_batch5
pyruvic_acid 16.99942   2.290157 -0.7162666 -0.2801026 -0.8823561 0.1029673
```

However, the normalized data will not include a location shift for sample type. As seen below, the differences in pyruvic acid abundance between mother and baby samples are preserved.

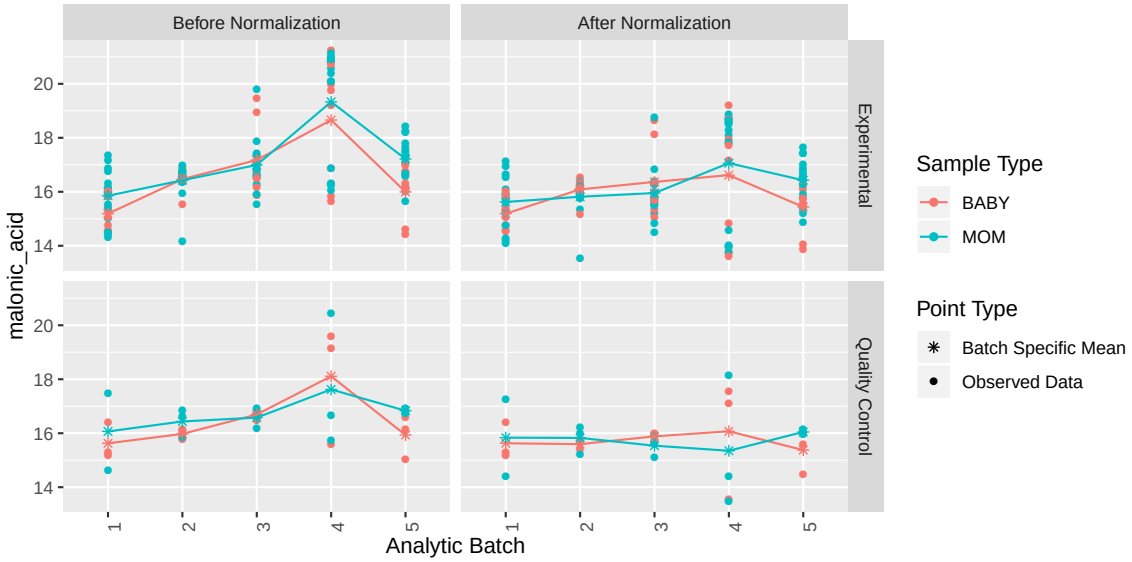
```
> metabplot("pyruvic_acid",  batch="batch", raw.obs.data=euMetabData, raw.cont.data=euMetabCData,
+                  norm.obs.data=euMetabNormRC$obsNorm, norm.cont.data=euMetabNormRC$ctlNorm,
+                  color.var="pheno", cont.outlier.sd.thresh=2, norm.outlier.sd.thresh=4)
```



3.2 Changing Outlier Criteria

Users may wish to change outlier criteria when executing normalization. In our experience, with only a small number of quality control samples per batch, an outlying sample may unduly influence mixture model results, yielding extreme batch effect estimates and poor normalization results. However, if users wish to not exclude data in estimating technical artifacts, they may do so by setting the `qc.sd.outliers` argument in `mixnorm` to Inf.

```
> norm.with.outliers <- mixnorm(ynames, batch="batch", mxtrModel=~pheno+batch|pheno+batch,
+                               batchTvals=c(10.76,11.51,11.36,10.31,11.90), cData=euMetabCData,
+                               data=euMetabData, qc.sd.outliers=Inf)

> metabplot("malonic_acid",  batch="batch", raw.obs.data=euMetabData, raw.cont.data=euMetabCData,
+                 norm.obs.data=norm.with.outliers$obsNorm, norm.cont.data=norm.with.outliers$ctlNorm,
+                 color.var="pheno", cont.outlier.sd.thresh=Inf, norm.outlier.sd.thresh=4)
```



3.3 Missing Batch Data

Users may encounter situations where quality control data are entirely missing for one or more batches. In these cases, the batch effect cannot be estimated.

```
> cData.missing.batch<-euMetabCData
> cData.missing.batch[cData.missing.batch$batch==2, "malonic_acid"]<-NA
> norm.with.missing.batch <- mixnorm(ynames, batch="batch", mxtrModel="~pheno+batch|pheno+batch,
+                                    batchTvals=c(10.76,11.51,11.36,10.31,11.90), cData=cData.missing.batch,
+                                    data=euMetabData)
```

These cases can be identified in several ways. First, in the `normParamsZ` output, the relevant batch effect will appear as NA.

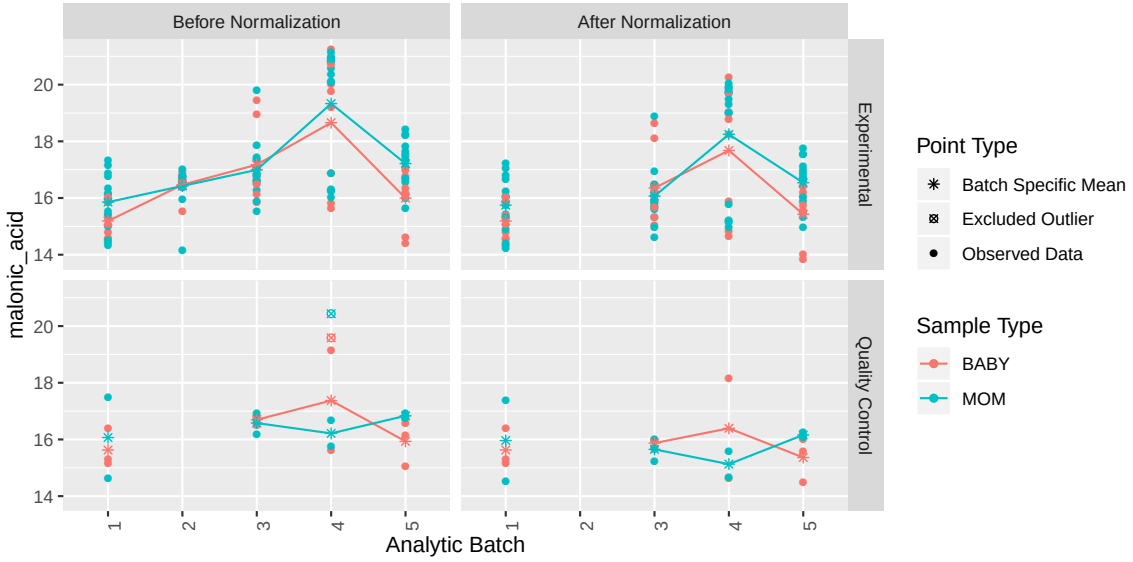
```
> norm.with.missing.batch$normParamsZ[rownames(norm.with.missing.batch$normParamsZ)=="malonic_acid", ]

                zInt z_phenoMOM z_batch2  z_batch3 z_batch4  z_batch5
malonic_acid 15.7609  0.1047759       NA 0.8228154 0.980348 0.5702425
```

Second, in the `conv` output, the missing batch will be identified in column `predictors_missing_levels`. Note that if this column is not present in the `conv` output, then there were no such instances.

```
> norm.with.missing.batch$conv

           .id conv predictors_missing_levels
1 pyruvic_acid    0                      <NA>
2 malonic_acid    0                   batch:2
```

Last, the missing data will be evident in output plots. Importantly, if quality control data are completely absent for a batch, in the `mixnorm` output, all non-missing values in the experimental data for that batch will be set to Inf, since normalization could not be performed for those observations. The missing batch data will therefore be evident in the quality control plots as well as the normalized experimental data plot.

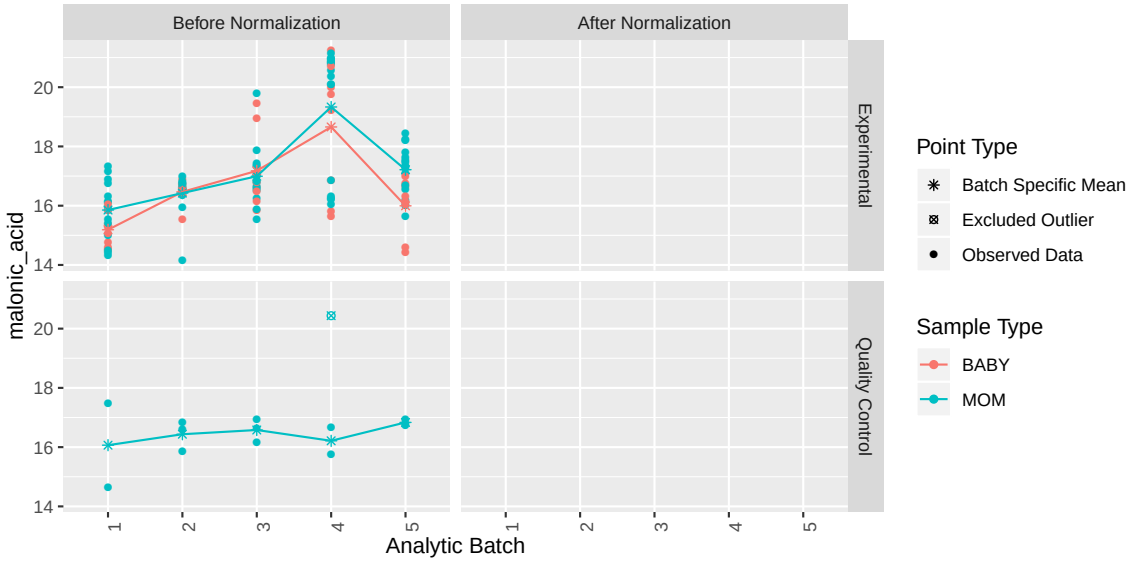
```
> metabplot("malonic_acid",  batch="batch", raw.obs.data=euMetabData, raw.cont.data=cData.missing.batch,
+                  norm.obs.data=norm.with.missing.batch$obsNorm,
+                  norm.cont.data=norm.with.missing.batch$ctlNorm,  color.var="pheno")
```



3.4 Missing Phenotype Data

In the example above, a multi-level factor predictor (batch) had entirely missing metabolite data for one batch, but still had data present for at least two other batches. Batch effects could therefore be estimated for those with sufficient data. However, there may be times that categorical mixture model predictors are missing too much metabolite data to be included in the model at all. For instance, in previous examples, `pheno` has been used as a predictor. This variable takes on one of two values and indicates whether a particular sample was obtained from a mother or baby. If quality control metabolite data were completely missing for either the mothers or the babies, we would be unable to obtain an estimate for the effect of `pheno` on metabolite abundance. In these situations, when predictors with completely inestimable effects are included in the mixture model, `mixnorm` does not output any data. Note these instances are also identified in the `conv` output of `mixnorm` in column `excluded_predictors`. Again, if this column is not present in the output, then these types of variables were not present. If users wish to normalize these metabolites, they must re-run the model excluding the appropriate predictor variable. If doing so, `batchTvals` should be manually specified such that batch specific thresholds of detectability match those of the full set of metabolites.

```
> cData.missing.pheno<-euMetabCData
> cData.missing.pheno[cData.missing.pheno$pheno=="BABY", "malonic_acid"]<-NA
> norm.missing.pheno <- mixnorm(ynames, batch="batch", mxtrModel=~pheno+batch|pheno+batch,
+                               batchTvals=c(10.76,11.51,11.36,10.31,11.90), cData=cData.missing.pheno,
+                               data=euMetabData)
> norm.missing.pheno$normParamsZ[rownames(norm.missing.pheno$normParamsZ)=="malonic_acid", ]

               zInt z_phenoMOM  z_batch2  z_batch3  z_batch4  z_batch5
malonic_acid 16.0659         NA 0.3697287 0.5123238 0.1478282 0.7683225

> norm.missing.pheno$conv

           .id conv excluded_predictors
1 pyruvic_acid    0               <NA>
2 malonic_acid    0             pheno

>

> metabplot("malonic_acid",  batch="batch", raw.obs.data=euMetabData, raw.cont.data=cData.missing.pheno,
+                 norm.obs.data=norm.missing.pheno$obsNorm, norm.cont.data=norm.missing.pheno$ctlNorm,
+                  color.var="pheno")
```



4 Session Information

- R version 3.5.2 (2018-12-20), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.5 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.8-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.8-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.42.0, BiocGenerics 0.28.0, ggplot2 3.1.0, metabomxtr 1.16.1, plyr 1.8.4, reshape2 1.4.3, xtable 1.8-3
- Loaded via a namespace (and not attached): BiocParallel 1.16.5, Formula 1.2-3, MASS 7.3-51.1, Matrix 1.2-15, R6 2.3.0, Rcpp 1.0.0, assertthat 0.2.0, bindr 0.1.1, bindrcpp 0.2.2, colorspace 1.3-2, compiler 3.5.2, crayon 1.3.4, digest 0.6.18, dplyr 0.7.8, glue 1.3.0, grid 3.5.2, gtable 0.2.0, labeling 0.3, lattice 0.20-38, lazyeval 0.2.1, magrittr 1.5, multtest 2.38.0, munsell 0.5.0, numDeriv 2016.8-1, optimx 2018-7.10, pillar 1.3.1, pkgconfig 2.0.2, purrr 0.2.5, rlang 0.3.0.1, scales 1.0.0, splines 3.5.2, stats4 3.5.2, stringi 1.2.4, stringr 1.3.1, survival 2.43-3, tibble 2.0.0, tidyselect 0.2.5, tools 3.5.2, withr 2.1.2

5 References

Nodzenski M, Muehlbauer MJ, Bain JR, Reisetter AC, Lowe WL Jr, Scholtens DM. Metabomxtr: an R package for mixture-model analysis of non-targeted metabolomics data. Bioinformatics. 2014 Nov 15;30(22):3287-8.

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8.