MetNet: Inferring metabolic networks from untargeted high-resolution mass spectrometry data

Thomas Naake [*]

Max Planck Institute of Molecular Plant Physiology 14476 Potsdam-Golm, Germany

[*]thomasnaake@googlemail.com

January 4, 2019

Abstract

A major bottleneck of mass spectrometry-based metabolomic analysis is still the rapid detection and annotation of unknown m/z features across biological matrices. Traditionally, the annotation was done manually imposing constraints in reproducibility and automatization. Furthermore, different analysis tools are typically used at different steps of analyses which requires parsing of data and changing of environments. I present here *MetNet*, a novel *R* package, that is compatible with the output of the *xcms*/*CAMERA* suite and that uses the data-rich output of mass spectrometry metabolomics to putatively link features on their relation to other features in the data set. *MetNet* uses both structural and quantitative information of metabolomics data for network inference that will guide metabolite annotation.

Package

MetNet 1.0.1

1 Introduction

Among the main challenges in mass spectrometric metabolomic analysis is the high-throughput analysis of metabolic features, their fast detection and annotation. By contrast to the screening of known, previously characterized, metabolic features in these data, the putative annotation of unknown features is often cumbersome and requires a lot of manual work, hindering the biological information retrieval of these data. High-resolution mass spectrometric data is often very rich in information content and metabolic conversions, and reactions can be derived from structural properties of features [1]. In addition to that, statistical associations between features (based on their intensity values) can be a valuable ressource to find co-synthesised or co-regulated metabolites, which are synthesised in the same biosynthetic pathways. Given that an analysis tool within the *R* framework is still lacking that is integrating the two features of mass spectrometric information commonly acquired with mass spectrometers (m/z and intensity values), I developed *MetNet* to close this gap. The *MetNet* package comprises functionalities to infer network topologies from high-resolution mass spectrometry data. *MetNet* combines information from both structural data (differences in m/z values of features) and statistical associations (intensity values of features per sample) to propose putative metabolic networks that can be used for further exploration.

The idea of using high-resolution mass spectrometry data for network construction was first proposed in [1] and followed soon afterwards by a Cytoscape plugin, MetaNetter [2], that is based on the inference of metabolic networks on molecular weight differences and correlation (Pearson correlation and partial correlation).

Inspired by the paper of [3] different algorithms for network were implemented in *MetNet* to account for biases that are inherent in these statistical methods, followed by the calculation of a consensus adjacency matrix using the differently computed individual adjacency matrices.

The two main functionalities of the package include the creation of an adjacency matrix from structual properties, based on losses/addition of functional groups defined by the user, and statistical associations. Currently, the following statistical models are implemented to infer a statistical adjacency matrix: Least absolute shrinkage and selection operator (LASSO, L1-norm regression, [4]), Random Forest [5], Pearson and Spearman correlation (including partial and semipartial correlation, see [6] for a discussion on correlation-based metabolic networks), context likelihood of relatedness (CLR, [7]), the algorithm for the reconstruction of accurate cellular networks (ARACNE, [8]) and constraint-based structure learning (Bayes, [9]). Since all of these methods have advantages and disadvantages, the user has the possibility to select several of these methods, compute adjacency matrices from these models and create a consensus matrix from the different statistical frameworks.

After creating the statistical and structural adjaceny matrices these two matrices can be combined to form a consensus matrix that has both information from structural and statistical properties of the data. This can be followed by further network analyses (e.g. calculation of topological parameters), integration with other data sources (e.g. genomic information or transcriptomic data) and/or visualization.

MetNet is currently under active development. If you discover any bugs, typos or develop ideas of improving *MetNet* feel free to raise an issue via GitHub or send a mail to the developer.

2 Prepare the environment and load the data

To install *MetNet* enter the following to the *R* console

```
install.packages("BiocManager")
BiocManager::install("MetNet")
```

Before starting with the analysis, load the *MetNet* package. This will also load the required packages *glmnet*, *stabs*, *randomForest*, *rfPermute*, *mpmi*, *parmigene*, *WGCNA* and *bnlearn* that are needed for functions in the statistical adjacency matrix inference.

```
library(MetNet)

##
```

The data format that is compatible with the *MetNet* framework is in the *xcms*/*CAMERA* output-like $m \ x \ n$ matrix, where columns denote the different samples n and where m features are present. In such a matrix, information about the masses of the features and quantitative information of the features (intensity or concentration values) are needed. The information about the m/z values has to be stored in a vector of length $|m|$ in the column `"mz"`.

MetNet does not impose any requirements for data normalization, filtering, etc. However, the user has to make sure that the data is properly preprocessed. These include division by internal standard, `log2` transformation, noise filtering, removal of features that do not represent mass features/metabolites, removal of isotopes, etc.

We will load here the object `x_test` that contains m/z values (in the column `"mz"`), together with the corresponding retention time (in the column `"rt"`) and intensity values. We will use here the object `x_test` for guidance through the workflow of *MetNet*.

```
data("x_test", package="MetNet")
x_test <- as.matrix(x_test)
```

3 Creating the structural matrix

The function `createStructuralAdjacency` will create the adjacency matrix based on structual properties (m/z values) of the features. The function expects a matrix with a column `"mz"` that contains the mass information of a feature (typically the m/z value). Furthermore, `createStructuralAdjacency` takes a `data.frame` object as argument `transformations` with the `colnames` `"mass"`, `"name"` and additional columns (e.g. `"formula"`). `createStructuralAdjacency` looks for transformation (in the sense of additions/losses of functional groups mediated by biochemical, enzymatic reactions) in the data using the mass information.

Following the work of [1] and [2], molecular weight difference w_X is defined by

$$w_X = |w_A - w_B| \tag{1}$$

where w_A is the molecular weight of substrate A, and w_B is the molecular weight of product B (typically, m/z values will be used as a proxy for the molecular weight since the molecular weight is not directly derivable from mass spectrometric data). As examplified in [2] specific enzymatic reactions refer to specific changes in the molecular weight, e.g. carboxylation reactions will result in a mass difference of 43.98983 (molecular weight of CO_2) between metabolic features.

The search space for these transformation is adjustable by the `transformation` argument in `createStructuralAdjacency` allowing to look for specific enzymatic transformations in mind. Hereby, `createStructuralAdjacency` will take into account the `ppm` value, to adjust for inaccuracies in m/z values due to technical reasons according to the formula

$$ppm = \frac{m_{exp} - m_{calc}}{m_{exp}} \cdot 10^{-6} \tag{2}$$

with m_{exp} the experimentally determined m/z value and m_{calc} the calculated accurate mass of a molecule. Within the function, a lower and upper range is calculated depending on the supplied `ppm` value, differences between the m/z feature values are calculated and matched against the `"mass"`es of the `transformations` argument. If any of the additions/losses defined in `transformations` is found in the data, it will be reported as an (unweighted) connection in the returned adjacency matrix. Together with the adjacency matrix the type of connection (derived from the column `"name"` in the `transformations`) will be written to a character matrix. These two matrices will be returned as a list (first entry: numerical adjacency matrix, second entry: character matrix) by the function `createStructuralAdjacency`.

Before calculating the structural matrix, one must define the search space, i.e. these transformation that will be looked for in the mass spectrometric data by creating the `transformations` object.

```
## define the search space for biochemical transformation
transformations <- rbind(
    c("Hydroxylation (-H)", "O", 15.9949146221, "-"),
    c("Malonyl group (-H2O)", "C3H2O3", 86.0003939305, "?"),
    c("C6H10O6", "C6H10O6", 178.0477380536, "-"),
    c("D-ribose (-H2O) (ribosylation)", "C5H8O4", 132.0422587452, "-"),
    c("Disaccharide (-H2O)", "C12H20O11", 340.1005614851, "-"),
    c("Glucuronic acid (-H2O)", "C6H8O6", 176.0320879894, "?"),
    c("Monosaccharide (-H2O)", "C6H10O5", 162.0528234315, "-"),
    c("Rhamnose (-H2O)", "C6H10O4", 146.057910, "-"),
    c("Trisaccharide (-H2O)", "C18H30O15", 486.1584702945, "-"),
    c("coumaroyl (-H2O)", "C9H6O2", 146.0367794368, "?"),
    c("feruloyl (-H2O)", "C9H6O2OCH2", 176.0473441231, "?"),
    c("sinapoyl (-H2O)", "C9H6O2OCH2OCH2", 206.0579088094, "?"),
    c("putrescine to spermidine (+C3H7N)", "C3H7N", 57.0578492299, "?"))

## convert to data frame
transformations <- data.frame(group=transformations[,1],
                            formula=transformations[,2],
                            mass=as.numeric(transformations[,3]),
                            rt=transformations[,4])
```

The function `createStructuralAdjacency` will then check for those m/z differences that are stored in the column `"mass"` in the object `transformations`. To create the adjacency matrix derived from these structural information we enter

```
struct_adj <- createStructuralAdjacency(x=x_test,
                                transformation=transformations, ppm=10)
```

in the *R* console.

Refining the structural adjacency matrix (optional)

Depending on the chemical group added the retention time will differ depending on the chemical group added, e.g. an addition of a glycosyl group will usually result in a lower retentiom time in reverse-phase chromatography- This information can be used in refining the adjacency matrix derived from the structural matrix. The `rtCorrection` does this checking, if predicted transformation correspond to the expected retention time shift, in an automated fashion. It requires information about the expected retention time shift in the `data.frame` passed to the `transformation` argument (in the `"rt"` column). Within this columns, information about retention time shifts is encoded by `"-"`, `"+"` and `"?"`, which means the feature with higher m/z value has lower, higher or unknown retention time than the feature with the lower m/z value. The values for m/z and retention time will be taken from the object passed to the `x` argument. In case there is a discrepancy between the transformation and the retention time shift the adjacency matrix at the specific position will be set to 0. `rtCorrection` will return the updated adjacency matrix and the updated character matrix with the descriptions of the transformation.

To account for retention time shifts we enter

```
struct_adj <- rtCorrection(struct_adj=struct_adj,
                                x=x_test, transformation=transformations)
```

in the *R* console.

4 Creating the statistical matrix

The function `createStatisticalAdjacency` will create the adjacency matrix based on statistical associations. `createStatisticalAdjacency` is a wrapper function for the functions `createStatisticalAdjacencyList` and `consensusAdjacency`. The former function will create a list of adjacency matrices using the statistical models defined by the `model` argument. Currently, the models LASSO (using *stabs*, [10, 11]), Random Forest (using *rfPermute*, CLR, ARACNE (the two latter using the package *mpmi* to calculate Mutual Information using a nonparametric bias correction by Bias Corrected Mutual Information, and the functions `clr` and `aracne.a` from the *parmigene* package), Pearson and Spearman correlation (based on the *WGCNA* package, [12]), partial and semipartial Pearson and Spearman correlation (using the *ppcor* package) and constraint-based structure learning based on the Fast Incremental Association (Fast-IAMB, algorithm from the *bnlearn* package, [9]).

For further information on the different models take a look on the respective help pages of `lasso`, `randomForest`, `clr`, `aracne`, `correlation` and/or `bayes`. Arguments that are accepted by the respective underlying functions can be passed directly to the `createStatisticalAdjacency` and `createStatisticalAdjacencyList` functions. In addition, arguments that are defined in the functions `lasso`, `randomForest`, `clr`, `aracne`, `correlation` and/or `bayes` can be passed to the functions.

From the list of adjacency matrices the function `consensusAdjacency` will create a consensus adjacency matrix using the employed statistical models. The reasoning behind this step is to circumvent disadvantages arising from each model and creating a statistically reliable topology that reflects the actual metabolic relations. To calculate the consensus adjacency matrix, the `consensus` function from the *sna* [13] is employed. The arguments that are accepted by this function can be passed to the `consensusAdjacency` and `createStatisticalAdjacency` function, respectively. Furthermore, in case a method other than `"central.graph"` is used the argument `threshold` will define if the value $a_{i,j}$ of the consensus adjacency matrix will be reported as a connection in the returned matrix (if $a_{i,j} \geq$ `threshold`) or not. `createStatisticalAdjacency` and `consensusAdjacency` will return an unweighted adjancency matrix with connections inferred from the respective models.

In the following example, we will create a consensus adjacency matrix using Pearson and Spearman correlation using the intensity values as input data. The p-values that will be used for assigning edges in the unweighted adjacency matrix will be adjusted by the Benjamini & Hochberg (False Discovery rate) method and the default q-value of 0.05.

```
x_int <- x_test[,3:dim(x_test)[2]]
stat_adj <- createStatisticalAdjacency(x_int,
                    model=c("pearson", "spearman"), correlation_adjust="BH")
```

```
## [1] "pearson finished."
## [1] "spearman finished."
```

To create the same adjacency matrix without using the wrapper function, one can call the two functions `createStatisticalAdjacencyList` and `consensusAdjacency` individually:

```
l <- createStatisticalAdjacencyList(x_int,
    model=c("pearson", "spearman", "aracne", "clr"), correlation_adjust="BH")

## [1] "clr finished."
## [1] "aracne finished."
## [1] "pearson finished."
## [1] "spearman finished."

stat_adj <- consensusAdjacency(l=l)
```

5 Combining the structural and statistical matrix

After creating the structural and statistical matrix, it is time to combine these two matrices. The function `combine_structural_statistical` will combine the matrices to the consensus matrix. The function accepts the arguments `structure` and `statistical` for the two matrices, respectively, and the argument `threshold`, that is a numerical value (default=1). After adding the matrices, the entries will be checked if they are greater or equal than `threshold` and 1 or 0 will be returned, respectively. The argument `threshold` needs to be adjusted by the user if another `method` than `"central.graph"` in `createStatisticalAdjacency`/`consensusAdjacency` is used.

```
cons_adj <- combineStructuralStatistical(structure=struct_adj[[1]],
                                         statistical=stat_adj)
```

6 Visualization and further analyses

To display the created consensus adjacency matrix, existing visualisation tools available in the *R* framework can be employed or any other visualisation tool after exporting the consensus matrix as a text file. In this example We will use the *igraph* [14] package to visualize the adjacency matrix.

```
g <- igraph::graph_from_adjacency_matrix(cons_adj, mode="undirected")
plot(g, edge.width=5, vertex.label.cex=0.5, edge.color="grey")
```

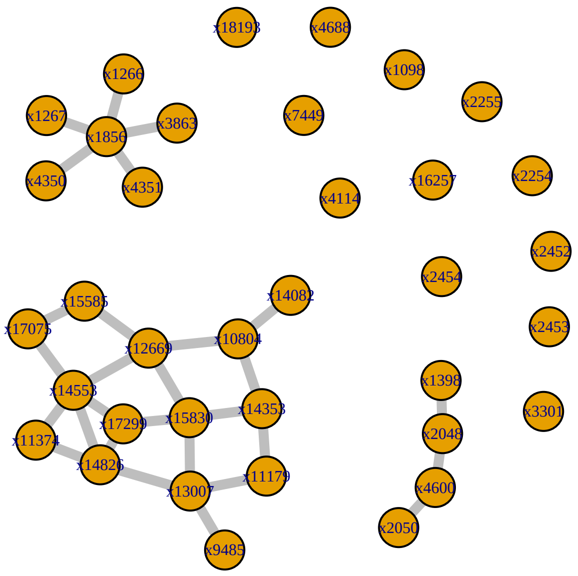

Figure 1: ***Ab initio* network inferred from structural and quantitative mass spectrometry data. Verteces are connected that are separated by given metabolic transformation and statistical association**

Furthermore, the network can be analysed by network analysis techniques (topological parameters such as centrality, degree, clustering indices) that are implemented in different packages in *R*(e.g. *igraph* or *sna*) or other software tools outside of the *R* environment.

References

[1] R. Breitling, S. Ritchie, D. Goodenowe, M.L. Stewart, and M.P. Barrett. *Ab initio* prediction of metabolic networks using fourier transform mass spectrometry data. *Metabolomics*, 2:155–164, 2006. doi:10.1007/s11306-006-0029-z.

[2] F. Jourdan, R. Breitling, M.P. Barrett, and D. Gilbert. Metanetter: inference and visualization of high-resolution metabolomic networks. *Bioinformatics*, 24:143–145, 2007. doi:10.1093/bioinformatics/btm536.

[3] D. Marbach, J.C. Costello, R. Küffner, N.M. Vega, R.J. Prill, D.M. Camacho, K.R. Allison, The DREAM5 Consortium, M. Kellis, J.J. Collins, and G. Stolovitzky. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9:796–804, 2012. doi:10.1038/nmeth.2016.

[4] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58:267–288, 1994.

[5] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. doi:10.1023/A:1010933404324.

[6] R. Steuer. On the analysis and interpretation of correlations in metabolomic data. *Briefings in Bioinformatics*, 7:151–158, 2006. doi:10.1093/bib/bbl009.

[7] J.J. Faith, B. Hayete, J.T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J.J. Collins, and T.S. Gardner. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *Plos Biology*, 5:e8, 2007. doi:10.1371/journal.pbio.0050008.

[8] A.A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7:S7, 2006. doi:10.1186/1471-2105-7-S1-S7.

[9] M. Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35:1–22, 2010. doi:10.18637/jss.v035.i03.

[10] B. Hofner, L. Boccuto, and M. Göker. Controlling false discoveries in high-dimensional situations: Boosting with stability selection. *BMC Bioinformatics*, 16:144, 2015. doi:10.1186/s12859-015-0575-3.

[11] J. Thomas, A. Mayr, B. Bischl, M. Schmid, A. Smith, and B. Hofner. Gradient boosting for distributional regression - faster tuning and improved variable selection via noncyclical updates. *Statistics and Computing*, 28:673–687, 2017. doi:10.1007/s11222-017-9754-6.

[12] P. Langfelder and S. Horvath. *WGCNA: an R package for weighted correlation network analysis*, 2008. doi:10.1186/1471-2105-9-559.

[13] Carter T. Butts. *sna: Tools for Social Network Analysis*, 2016. R package version 2.4. URL: https://CRAN.R-project.org/package=sna.

[14] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL: http://igraph.org.

Appendix

Session information

All software and respective versions to build this vignette are listed here:

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-pc-linux-gnu (64-bit)
## 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
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] MetNet_1.0.1 knitr_1.21
##
## loaded via a namespace (and not attached):
##   [1] nlme_3.1-137          matrixStats_0.54.0    robust_0.4-18
##   [4] fit.models_0.5-14     bit64_0.9-7           doParallel_1.0.14
##   [7] RColorBrewer_1.1-2    dynamicTreeCut_1.63-1 tools_3.5.2
##  [10] backports_1.1.3       R6_2.3.0              rpart_4.1-13
##  [13] KernSmooth_2.23-15    mgcv_1.8-26           Hmisc_4.1-1
##  [16] DBI_1.0.0             lazyeval_0.2.1        BiocGenerics_0.28.0
##  [19] colorspace_1.3-2      nnet_7.3-12           tidyselect_0.2.5
##  [22] gridExtra_2.3         bit_1.1-14            compiler_3.5.2
##  [25] preprocessCore_1.44.0 WGCNA_1.66            Biobase_2.42.0
##  [28] htmlTable_1.13        network_1.13.0.1      scales_1.0.0
##  [31] checkmate_1.8.5       spatstat.data_1.4-0   DEoptimR_1.0-8
##  [34] mvtnorm_1.0-8         robustbase_0.93-3     randomForest_4.6-14
##  [37] goftest_1.1-1         spatstat_1.57-1       stringr_1.3.1
##  [40] digest_0.6.18         spatstat.utils_1.13-0 foreign_0.8-71
##  [43] rmarkdown_1.11        rrcov_1.4-7           base64enc_0.1-3
## [46] pkgconfig_2.0.2      htmltools_0.3.6      mpmi_0.42
## [49] maps_3.3.0           highr_0.7            htmlwidgets_1.3
## [52] rlang_0.3.0.1        rstudioapi_0.8       RSQLite_2.1.1
## [55] impute_1.56.0        bindr_0.1.1          statnet.common_4.1.4
## [58] BiocParallel_1.16.5  acepack_1.4.1        dplyr_0.7.8
## [61] magrittr_1.5         GO.db_3.7.0          Formula_1.2-3
## [64] Matrix_1.2-15        Rcpp_1.0.0           munsell_0.5.0
## [67] S4Vectors_0.20.1     abind_1.4-5          stringi_1.2.4
## [70] bnlearn_4.4          yaml_2.2.0           MASS_7.3-51.1
## [73] plyr_1.8.4           grid_3.5.2           blob_1.1.1
## [76] parallel_3.5.2       crayon_1.3.4         ppcor_1.1
## [79] deldir_0.1-16        lattice_0.20-38      splines_3.5.2
## [82] tensor_1.5           sna_2.4              pillar_1.3.1
## [85] igraph_1.2.2         swfscMisc_1.2        fastcluster_1.1.25
## [88] reshape2_1.4.3       codetools_0.2-16     stats4_3.5.2
## [91] glue_1.3.0           evaluate_0.12        latticeExtra_0.6-28
## [94] mapdata_2.3.0        parmigene_1.0.2      data.table_1.11.8
## [97] BiocManager_1.30.4   rfPermute_2.1.6      foreach_1.4.4
## [100] polyclip_1.9-1      gtable_0.2.0         purrr_0.2.5
## [103] assertthat_0.2.0    ggplot2_3.1.0        xfun_0.4
## [106] coda_0.19-2         survival_2.43-3      pcaPP_1.9-73
## [109] tibble_2.0.0        iterators_1.0.10     AnnotationDbi_1.44.0
## [112] memoise_1.1.0       IRanges_2.16.0       bindrcpp_0.2.2
## [115] cluster_2.0.7-1     stabs_0.6-3          BiocStyle_2.10.0
```

Transformations

The list of transformations is taken from [1]. The numerical m/z values were calculated by using the structural formula and the Biological Magnetic Resonance Data Bank web tool.

```
transformations <- rbind(
    c("Alanine", "C3H5NO", "71.0371137878"),
    c("Arginine", "C6H12N4O", "156.1011110281"),
    c("Asparagine", "C4H6N2O2", "114.0429274472"),
    c("Guanosine 5-diphosphate (-H2O)", "C10H13N5O10P2", "425.0137646843"),
    c("Guanosine 5-monophosphate (-H2O)", "C10H12N5O7P", "345.0474342759"),
    c("Guanine (-H)", "C5H4N5O", "150.0415847765"),
    c("Aspartic acid", "C4H5NO3", "115.0269430320"),
    c("Guanosine (-H2O)", "C10H11N5O4", "265.0811038675"),
    c("Cysteine", "C3H5NOS", "103.0091844778"),
    c("Deoxythymidine 5'-diphosphate (-H2O)", "C10H14N2O10P2", "384.01236770"),
    c("Cystine", "C6H10N2O3S2", "222.0132835777"),
    c("Thymidine (-H2O)", "C10H12N2O4", "224.0797068840"),
    c("Glutamic acid", "C5H7NO3", "129.0425930962"),
    c("Thymine (-H)", "C5H5N2O2", "125.0351024151"),
    c("Glutamine", "C5H8N2O2", "128.0585775114"),
    c("Thymidine 5'-monophosphate (-H2O)", "C10H13N2O7P", "304.0460372924"),
    c("Glycine", "C2H3NO", "57.0214637236"),
    c("Uridine 5'-diphosphate (-H2O)", "C9H12N2O11P2", "385.9916322587"),
    c("Histidine", "C6H7N3O", "137.0589118624"),
    c("Uridine 5'-monophosphate (-H2O)", "C9H11N2O8P", "306.0253018503"),
    c("Isoleucine", "C6H11NO", "113.0840639804"),
    c("Uracil (-H)", "C4H3N2O2", "111.0194523509"),
    c("Leucine", "C6H11NO", "113.0840639804"),
    c("Uridine (-H2O)", "C9H10N2O5", "226.0589714419"),
    c("Lysine", "C6H12N2O", "128.0949630177"),
    c("Acetylation (-H)", "C2H3O2", "59.0133043405"),
    c("Methionine", "C5H9NOS", "131.0404846062"),
    c("Acetylation (-H2O)", "C2H2O",  "42.0105646863"),
    c("Phenylalanine", "C9H9NO",  "147.0684139162"),
    c("C2H2", "C2H2", "26.0156500642"),
    c("Proline", "C5H7NO", "97.0527638520"),
    c("Carboxylation", "CO2", "43.9898292442"),
    c("Serine", "C3H5NO2", "87.0320284099"),
    c("CHO2", "CHO2", "44.9976542763"),
    c("Threonine",  "C4H7NO2",  "101.0476784741"),
    c("Condensation/dehydration", "H2O", "18.0105646863"),
    c("Tryptophan", "C11H10N2O",  "186.0793129535"),
    c("Diphosphate", "H3O6P2", "160.9404858489"),
            c("Tyrosine", "C9H9NO2", "163.0633285383"),
            c("Ethyl addition (-H2O)", "C2H4", "28.0313001284"),
            c("Valine", "C5H9NO",  "99.0684139162"),
            c("Formic Acid (-H2O)", "CO", "27.9949146221"),
            c("Acetotacetate (-H2O)",  "C4H4O2", "84.0211293726"),
            c("Glyoxylate (-H2O)", "C2O2",  "55.9898292442"),
            c("Acetone (-H)", "C3H5O", "57.0340397826"),
            c("Hydrogenation/dehydrogenation", "H2", "2.0156500642"),
            c("Adenylate (-H2O)", "C10H12N5O6P", "329.0525196538"),
            c("Hydroxylation (-H)", "O", "15.9949146221"),
            c("Biotinyl (-H)", "C10H15N2O3S", "243.0803380482"),
            c("Inorganic phosphate", "P", "30.9737615100"),
            c("Biotinyl (-H2O)", "C10H14N2O2S", "226.0775983940"),
            c("Ketol group (-H2O)", "C2H2O", "42.0105646863"),
            c("Carbamoyl P transfer (-H2PO4)", "CH2ON", "44.0136386915"),
            c("Methanol (-H2O)", "CH2", "14.0156500642"),
            c("Co-enzyme A (-H)", "C21H34N7O16P3S", "765.0995583014"),
            c("Phosphate", "HPO3", "79.9663304084"),
            c("Co-enzyme A (-H2O)", "C21H33N7O15P3S", "748.0968186472"),
            c("Primary amine", "NH2", "16.0187240694"),
            c("Glutathione (-H2O)", "C10H15N3O5S", "289.0732412976"),
            c("Pyrophosphate", "PP", "61.9475230200"),
            c("Isoprene addition (-H)", "C5H7", "67.0547752247"),
            c("Secondary amine", "NH", "15.0108990373"),
            c("Malonyl group (-H2O)", "C3H2O3", "86.0003939305"),
            c("Sulfate (-H2O)", "SO3", "79.9568145563"),
            c("Palmitoylation (-H2O)", "C16H30O", "238.2296655851"),
            c("Tertiary amine", "N", "14.0030740052"),
            c("Pyridoxal phosphate (-H2O)", "C8H8NO5P", "229.0140088825"),
            c("C6H10O5", "C6H10O5", "162.0528234315"),
            c("Urea addition (-H)", "CH3N2O", "59.0245377288"),
            c("C6H10O6", "C6H10O6", "178.0477380536"),
            c("Adenine (-H)", "C5H4N5", "134.0466701544"),
            c("D-ribose (-H2O) (ribosylation)", "C5H8O4", "132.0422587452"),
            c("Adenosine (-H2O)", "C10H11N5O3", "249.0861892454"),
            c("Disaccharide (-H2O)", "C12H20O11", "340.1005614851"),
            c("Adenosine 5'-diphosphate (-H2O)", "C10H13N5O9P2", "409.0188500622"),
            c("Glucose-N-phosphate (-H2O)", "C6H11O8P", "242.0191538399"),
            c("Adenosine 5'-monophosphate (-H2O)", "C10H12N5O6P", "329.0525196538"),
            c("Glucuronic acid (-H2O)", "C6H8O6", "176.0320879894"),
            c("Cytidine 5'-diphosphate (-H2O)", "C9H13N3O10P2", "385.0076166739"),
            c("Monosaccharide (-H2O)", "C6H10O5", "162.0528234315"),
            c("Cytidine 5'-monophsophate (-H2O)", "C9H12N3O7P", "305.0412862655"),
            c("Trisaccharide (-H2O)", "C18H30O15", "486.1584702945"),
            c("Cytosine (-H)", "C4H4N3O",  "110.0354367661"))
```

```
transformations <- data.frame(name=transformations[,1],
            formula=transformations[,2],
            mass=as.numeric(transformations[,3]))
```