Introduction to *metadeconfoundR*

Till Birkner*[1]*, Sofia K. Forslund

[1] tillb@online.de

Edited: September 14, 2021; Compiled: February 8, 2023

Contents

List of Abbreviations

BMI body mass index

lrt likelihood ratio test

1 Introduction

Package

metadeconfoundR 0.3.0

- add actual tables of input data for a more direct example (instead of just linking to included data set)
- mention usage of ranked omics data (as differentiation towards other tools)

When analyzing multi omics datasets, the search for features that could serve as biomarkers is an important aspect. Because these biomarkers might be used in clinical settings for disease diagnosis etc., it is extremely important to minimize false positives. One possible error source are confounding variables: The biomarker is not directly linked to the disease but influenced by a third (confounding) variable, that in turn is linked to the disease.

The R package *metadeconfoundR* was developed to address this issue. It first uses univariate statistics to find associations between omics features and disease status or metadata. Using nested linear model comparison post hoc testing, those associations are checked for confounding effects from other covariates/metadata and a status label is returned. The tool is able to handle large scale multi-omics datasets in a reasonable time, by parallel processing suitable for high-performance computing clusters. In addition, results can be summarized by a range of plotting functions.

1.1 `Metadeconfound()`

The main (`metadeconfoundR::Metadeconfound()`) analysis is a two step process:

First, significant associations between single omics features (like gut microbial OTUs) and metadata (like disease status, drug administration, BMI) are identified *(Fig. 1, left)*. Based on the data type of the respective metadata, either `wilcox.test()` (for binary), `cor.test()` (for continuous numerical) or `kruskal.test()` (for neither numerical nor binary) is used. All three tests are rank-based to minimize assumptions about data distribution.

In addition to collecting p-values for all computed tests, effect size is measured as Cliff's Delta and Spearman's Rho for binary and continuous data, respectively. Since there is no suitable effect size metric for categorical data with more than 2 levels, no value is reported here. It is recommended to introduce binary pseudo-variables for each level of the categorical metadata to partially circumvent this drawback.

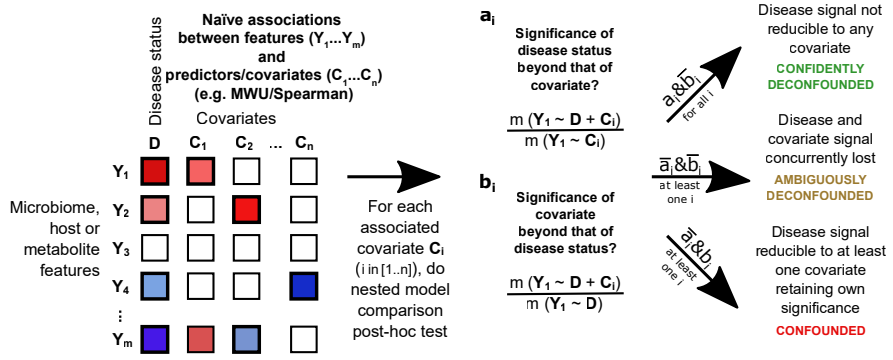

Figure 1: **Overview of main statistical approach used to determine confounding status of associations between omics features and the disease status**
(left) Each omics feature (Y) is independently tested for association to any of the predictors/covariates (D/C); Used test depends on data type of covariate. **(center)** For each identified Y ↔ D, sets of pairs of linear model likelihood ratio tests (lrts) are computed. Each set tests for confounding effects of an additional covariate (C) on the current Y ↔ D pair. **(right)** Based on significance of the lrts a status for the current feature (Y) ↔ disease status (D) pair is reported.
In additiona to only determining the confounding status of any associations between omics features and the disease status, also confounding between the different covariates is determined in the same way.

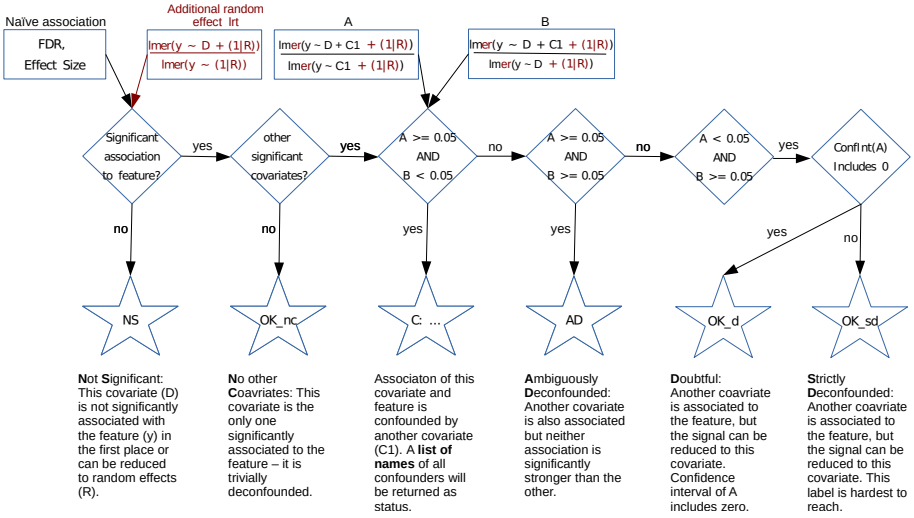

Figure 2: Status labeling process in more detail

For each possible feature ↔ covariate combination these steps are done. A and B are the linear model likelihood ratios. Should there be more than one "other significant covariate" (C1), linear model likelihood ratio comparison has to be repeated for every single one of them. Whenever CONF is reached, the name of C1 is returned as label. Only when SD is reached for all C1, this will be returned as label. (y = feature, D = current covariate, C1 = other significant covariates, NS = not significant, OK_nc = no covariates (i.e. trivially deconfounded), C: ...= confounded by variables listed after "C:", AD = Ambiguously deconfounded, OK_d = deconfounded, but doubtful since confidence interval for predictive difference in lrt A includes zero, OK_sd = strictly deconfounded and confidence interval for predictive difference in lrt A does not include zero.)
When random effect variables (like batch effects, cage, study center) are supplied via the `randomVar` parameter, parts in dark red are added to the labeling process: In addition to the naive association tests, an lrt is done to test whether the naive association can be reduced to the random effect. Later on, mixed effect models are used instead of linear models, enabling the inclusion of the random effect into A and B.

In the second step, all hits are checked for confounding effects *(Fig. 1, center and right)* and a status is reported for each feature ↔ metadata combination *(Fig. 2)*. A "hit" here is defined as a feature ↔ metadata association with small enough fdr-corrected p-value and big enough effect size. Thresholds for both parameters can be set via `QCutoff` and `DCutoff` when starting the analysis. Since confounding of signal can only happen with more than one different metadata associated to a certain feature, all features with only one significant metadata are trivially deconfounded and get status "No Covariates (OK_nc)".

1.2 `BuildHeatmap()`

In order to summarize and visualize the results of a deconfounding run, the `BuildHeatmap()` function supplies a set of predefined but customizable plots. Due to the typically large number of features in multi-omics data-sets, these plots usually only show a subset of the data representing high significance and effect size associations.

2 Quick start

```
library(devtools)
install_github("TillBirkner/metadeconfoundR")
library(metadeconfoundR)
```

```
## Lade nötiges Paket:  detectseparation
```

3 Usage

3.1 `Metadeconfound()`

Minimal input consists of two data.frames for feature data (*Tab. 1*) and metadata (*Tab. 2*), respectively. Both data.frames must have one row per sample (sample names as rownames) with matching order of sampleIDs and one feature/meta-variable per column. The first column of the metadata data.frame must be binary (i.e consist of only 0/1 entries.) Usually this is the control/case variable, but any other binary meta-variable will work as well. Ensure that colnames and rownames should not contain any problematic characters by e.g running them through `make.names()`. See *Fig. 3*, *left* for summarizing heatmap of output.

Table 1: included example feature data.frame `reduced_feature`

	MS0001	MS0006	MS0007	MS0008	MS0012
BGI003A	0	42.0	4.0	153.0	126.0
BGI089A	0	155.5	34.5	360.5	116.5
DLF001	3	67.0	6.0	443.0	40.0
DLF002	1	58.0	18.0	175.0	181.5
DLF003	45	43.0	0.0	66.0	74.0
DLF004	0	41.0	1.0	206.0	37.0

Table 2: included example metadata data.frame `metaMatMetformin`

	Status	Dataset	Metformin	continuous_dummy	altered_dummy
BGI003A	0	CHN	0	0.0617863	0.0617863
BGI089A	0	CHN	0	0.2059746	0.2059746
DLF001	1	CHN	0	0.1765568	0.1765568
DLF002	1	CHN	0	0.6870228	0.6870228
DLF003	1	CHN	1	0.3841037	0.3841037
DLF004	1	CHN	0	0.7698414	0.7698414

```
data(reduced_feature)
data(metaMatMetformin)

# check correct ordering
all(rownames(metaMatMetformin) == rownames(reduced_feature))
```

```
## [1] TRUE
```

```
all(order(rownames(metaMatMetformin)) == order(rownames(reduced_feature)))
```

```
## [1] TRUE
```

```
example_output <- MetaDeconfound(featureMat = reduced_feature,
    metaMat = metaMatMetformin, nnodes = 4, returnLong = TRUE)
```

Random effects can be included in the modeling process (as described in *Fig. 2*) by supplying the `randomVar` parameter *(Fig. 3, right)*.

```
RandDataset_output <- MetaDeconfound(featureMat = reduced_feature,
    metaMat = metaMatMetformin, nnodes = 4, randomVar = c("Dataset"),
    returnLong = TRUE)
```

For a full list of input parameters please refer to the help page.

3.1.1 output

Output can be returned either as a list of wide format data.frames (default) or as a single long format data.frame (*Tab. 3*). In both cases raw p-values (Ps), multiple testing corrected p-values (Qs), corresponding effect size (Ds), and confounding status (status) are reported for each possible combination of a feature to a meta-variable.

Table 3: output of a `MetaDeconfound()` run in long format

feature	metaVariable	Ps	Qs	Ds	status
MS0001	Status	0.0039510	0.0103973	-0.1396941	AD
MS0006	Status	0.0000321	0.0001147	0.2023848	OK_sd
MS0007	Status	0.0000001	0.0000016	0.2425731	OK_sd
MS0008	Status	0.7124524	0.7740977	0.0179492	NS
MS0012	Status	0.1847586	0.3079311	0.0645627	NS

3.2 `BuildHeatmap()`

3.2.1 input

Minimal input consists only of an output object from the main `Metadeconfound()` function either in long or wide format. This will result in a heatmap similar to .

```
left <- BuildHeatmap(example_output)
right <- BuildHeatmap(RandDataset_output)
plot_grid(left, right)
```

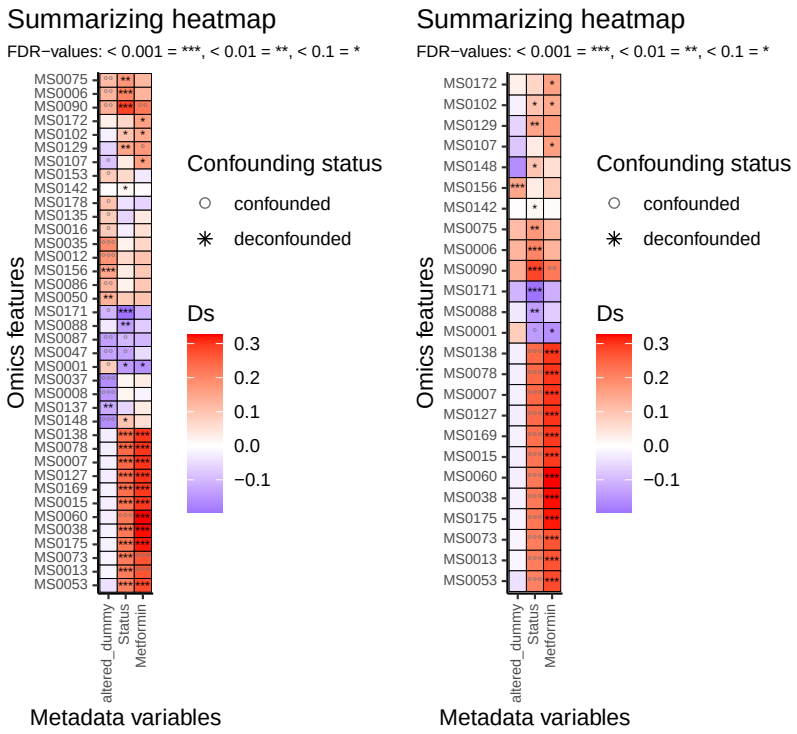

Figure 3: default output of the `BuildHeatmap()` function

A heatmap is returned, showing associations between individual omics features (y-axis) and meta-variables (x-axis). Color indicates effect size (Cliff's Delta or Spearman's Rho for binary or continuous meta-variables, respectively). Significance of shown associations indicated by black asterisks according to FDR adjusted p-values of naive tests. Naively significant but confounded associations are instead indicated by gray circles. The plot is clustered on both axes and features as well as meta-variables without any associations passing effect size and significance cutoffs (default: `q_cutoff = 0.1`, `d_cutoff = 0.01`) are removed. left: `example_output` as input. right: `RandDataset_output` as input.

For both this default heatmap, as well as the alternative cuneiform plot (`cuneiform = TRUE`), a range of customizations are available. In *Fig. 4* meta-variables not passing the effect size and significance are manually kept in the plot (`keepMeta`), the shown range of effect sizes is set from -1 to $+1$. For a full list of options, again, refer to the help page.

```
BuildHeatmap(example_output, cuneiform = TRUE,
    keepMeta = colnames(example_output$status),
    d_range = "full")
```

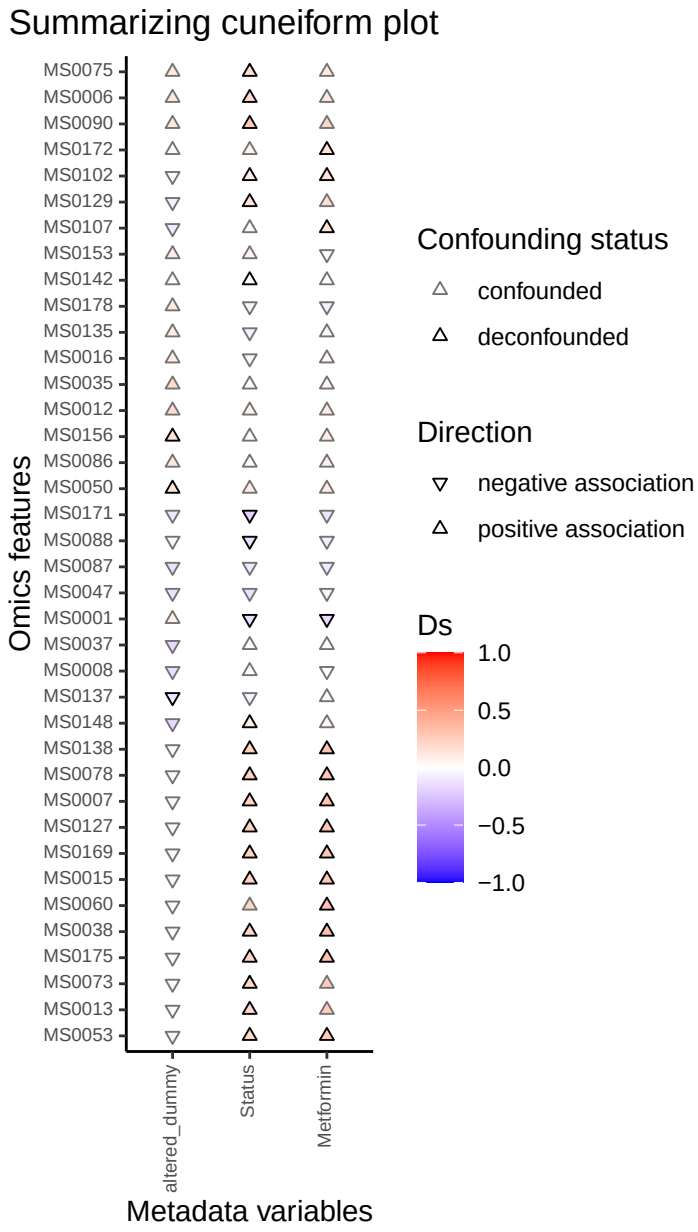

Figure 4: alternative output of the `BuildHeatmap()` function

A cuneiform plot is returned, showing associations between individual omics features (y-axis) and meta-variables (x-axis) as triangles. Fill color and direction of triangles indicate effect size (Cliff's Delta or Spearman's Rho for binary or continuous meta-variables, respectively). Significance of shown associations indicated by triangle line color: Both confounded and not significant associations are gray, while only robust (i.e. significant and not confounded) associations are black. The plot is clustered on both axes and features as well as meta-variables without any associations passing effect size and significance cutoffs (default: `q_cutoff = 0.1`, `d_cutoff = 0.01`) are removed.

3.2.2 output

The `BuildHeatmap()` function returns a ggplot2 object. This makes it possible to perform some easy alterations manually (*Fig. 5*)

```
BuildHeatmap(example_output) + theme(legend.position = "none",
    axis.text.y = element_text(face = "italic"),
    plot.title = element_blank(), plot.subtitle = element_blank())
```

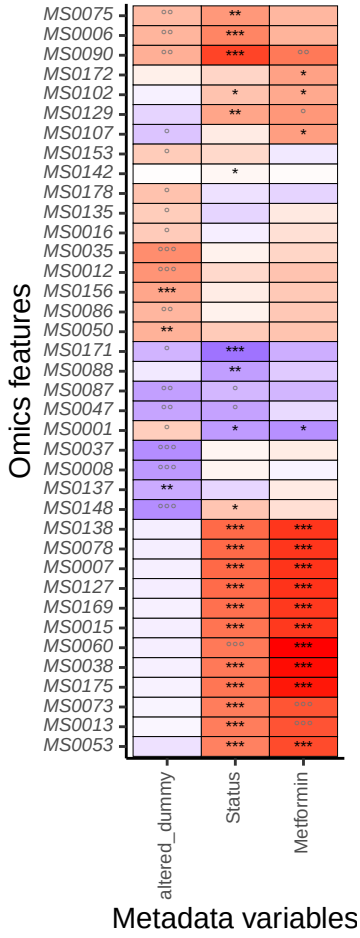

Figure 5: **Manual changes to the default `BuildHeatmap()` ouptut**
removal of legend, title, subtitle and italicization of feature names through `ggplot2::theme()` function.

4 `sessionInfo()`

```
toLatex(sessionInfo())
```

- R version 4.2.2 Patched (2022-11-10 r83330), `x86_64-pc-linux-gnu`

- Locale: `LC_CTYPE=de_DE.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=de_DE.UTF-8`, `LC_COLLATE=de_DE.UTF-8`, `LC_MONETARY=de_DE.UTF-8`, `LC_MESSAGES=de_DE.UTF-8`, `LC_PAPER=de_DE.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=de_DE.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04.5 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: cowplot 1.1.1, detectseparation 0.3, ggplot2 3.3.6, kableExtra 1.3.4, metadeconfoundR 0.3.0
- Loaded via a namespace (and not attached): assertthat 0.2.1, bigmemory 4.6.1, bigmemory.sri 0.1.3, BiocManager 1.30.18, BiocStyle 2.24.0, boot 1.3-28, cli 3.3.0, codetools 0.2-19, colorspace 2.0-3, compiler 4.2.2, crayon 1.5.1, DBI 1.1.2, digest 0.6.29, dplyr 1.0.9, ellipsis 0.3.2, evaluate 0.15, fansi 1.0.3, farver 2.1.0, fastmap 1.1.0, foreach 1.5.2, formatR 1.12, futile.logger 1.4.3, futile.options 1.0.1, generics 0.1.2, glue 1.6.2, grid 4.2.2, gtable 0.3.0, highr 0.9, htmltools 0.5.2, httr 1.4.3, iterators 1.0.14, knitr 1.39, labeling 0.4.2, lambda.r 1.2.4, lattice 0.20-45, lifecycle 1.0.1, lme4 1.1-29, lmtest 0.9-40, lpSolveAPI 5.5.2.0-17.9, magrittr 2.0.3, MASS 7.3-58.2, Matrix 1.5-1, minqa 1.2.4, munsell 0.5.0, nlme 3.1-162, nloptr 2.0.3, numDeriv 2016.8-1.1, pillar 1.7.0, pkgconfig 2.0.3, plyr 1.8.7, purrr 0.3.4, R6 2.5.1, Rcpp 1.0.8.3, registry 0.5-1, reshape2 1.4.4, rlang 1.0.3, rmarkdown 2.14, ROI 1.0-0, ROI.plugin.lpsolve 1.0-1, rstudioapi 0.13, rvest 1.0.3, scales 1.2.0, slam 0.1-50, splines 4.2.2, stringi 1.7.6, stringr 1.4.0, svglite 2.1.0, systemfonts 1.0.4, tibble 3.1.7, tidyselect 1.1.2, tools 4.2.2, utf8 1.2.2, uuid 1.1-0, vctrs 0.4.1, viridisLite 0.4.0, webshot 0.5.4, withr 2.5.0, xfun 0.31, xml2 1.3.3, yaml 2.3.5, zoo 1.8-10