Introduction to *metadeconfoundR*

Till Birkner[1]***, Sofia Kirke Forslund-Startceva***

[1] metadeconfoundr@till-birkner.de

Edited: June 5, 2024; Compiled: June 5, 2024

Contents

List of Abbreviations

BMI body mass index

lrt likelihood ratio test

1 Introduction

Package

metadeconfoundR 0.3.1

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. Instead of assuming 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, body mass index (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.

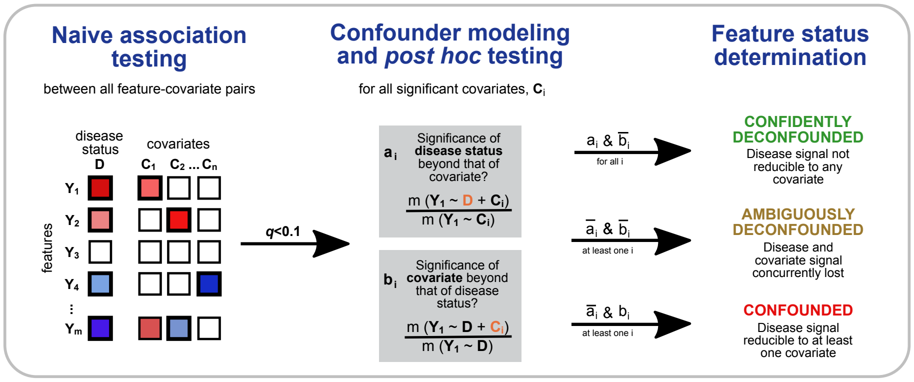
Naive association testing
between all feature-covariate pairs
disease status
D
covariates
C_1 C_2 ... C_n
features
Y_1
Y_2
Y_3
Y_4
⋮
Y_m
q<0.1
Confounder modeling and *post hoc* testing
for all significant covariates, C_i
a_i
Significance of **disease status** beyond that of covariate?
$\frac{m\,(Y_1 \sim D + C_i)}{m\,(Y_1 \sim C_i)}$
b_i
Significance of **covariate** beyond that of disease status?
$\frac{m\,(Y_1 \sim D + C_i)}{m\,(Y_1 \sim D)}$
a_i & $\bar{b}_i$
for all i
$\bar{a}_i$ & $\bar{b}_i$
at least one i
$\bar{a}_i$ & b_i
at least one i
Feature status determination
CONFIDENTLY DECONFOUNDED
Disease signal not reducible to any covariate
AMBIGUOUSLY DECONFOUNDED
Disease and covariate signal concurrently lost
CONFOUNDED
Disease signal reducible to at least one covariate

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.

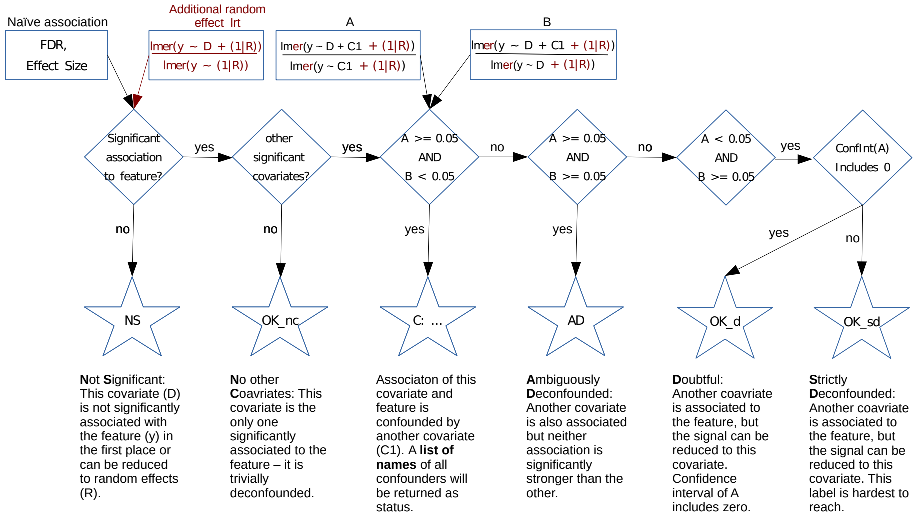
Naïve association
FDR,
Effect Size
Additional random effect lrt
lmer(y ~ D + (1|R))
lmer(y ~ (1|R))
A
lmer(y ~ D + C1 + (1|R))
lmer(y ~ C1 + (1|R))
B
lmer(y ~ D + C1 + (1|R))
lmer(y ~ D + (1|R))
Significant association to feature?
yes
other significant covariates?
yes
A >= 0.05 AND B < 0.05
no
A >= 0.05 AND B >= 0.05
no
A < 0.05 AND B >= 0.05
yes
ConfInt(A) Includes 0
no
no
yes
yes
yes
no
NS
OK_nc
C: ...
AD
OK_d
OK_sd
Not Significant: This covariate (D) is not significantly associated with the feature (y) in the first place or can be reduced to random effects (R).
No other Coavriates: This covariate is the only one significantly associated to the feature – it is trivially deconfounded.
Associaton of this covariate and feature is confounded by another covariate (C1). A list of names of all confounders will be returned as status.
Ambiguously Deconfounded: Another covariate is also associated but neither association is significantly stronger than the other.
Doubtful: Another coavriate is associated to the feature, but the signal can be reduced to this covariate. Confidence interval of A includes zero.
Strictly Deconfounded: Another coavriate is associated to the feature, but the signal can be reduced to this covariate. This label is hardest to reach.

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 reported from the first, naive part of the pipeline. 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 metadata variable associated to a certain feature, all features with only one significant metadata association are trivially deconfounded and get status “No Covariates (OK_nc)".

The actual confounder detection is done by performing a set of two likelihood ratio tests of nested linear models. For each possible combination of a feature and two of its associated metavariables, three models are fitted to the feature:

- lm(rank(feature) c̃ovariate1 + covariate2), the full model
- lm(rank(feature) c̃ovariate1), a model with only covariate1 as independent variable

- lm(rank(feature) c̃ovariate2), a model with only covariate2 as independent variable.

lrts reveal whether inclusion of covariate1 and/or covariate2 significantly improves the performance of the model.

Importantly, *metadeconfoundR* will always rank-transform the features during analysis.

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, only rows/columns with at least one significant association are plotted. The number of plotted associations can be further decreased by increasing cutoffs for effect size (`d_cutoff`) and significance (`q_cutoff`), and increased by explicitly specifying features (`keepFeature`) and metavariables (`keepMeta`) to be kept in the plot.

2 Quick start

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

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

3 Usage

3.1 `Metadeconfound()`

3.1.1 Minimal input

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.

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

3.1.2 data formatting

`Metadeconfound()` has built-in quality checks for formatting of the input data but it is best to check propper formatting beforehand.

Ensure that colnames and rownames do not contain any problematic characters by e.g running them through `make.names()` and check for same order of rows in both input data.frames.

```
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, returnLong = TRUE,
    logLevel = "ERROR")
```

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, randomVar = c("Dataset"),
    returnLong = TRUE, logLevel = "ERROR")
```

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

3.1.3 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)
grid.arrange(left, right, ncol = 2)
```

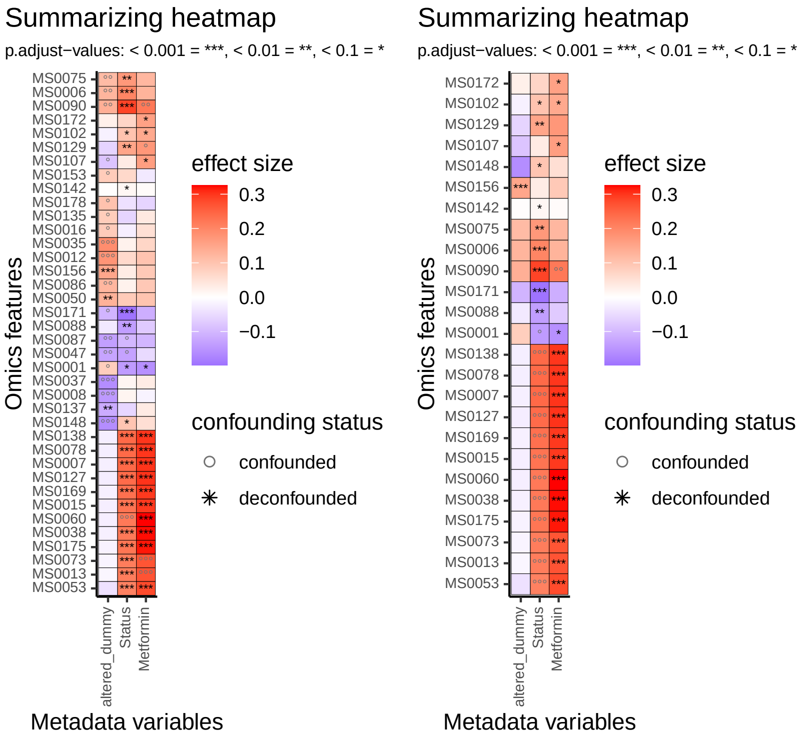

Figure 3: default output of the `BuildHeatmap()` function
A heatmap is returned, showing associations between individual omics features (y-axis) and meta-variables (x-axis). Color indicates effect size (Cliff's Delta or Spearman's Rho for binary or continuous meta-variables, respectively). Significance 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")
```

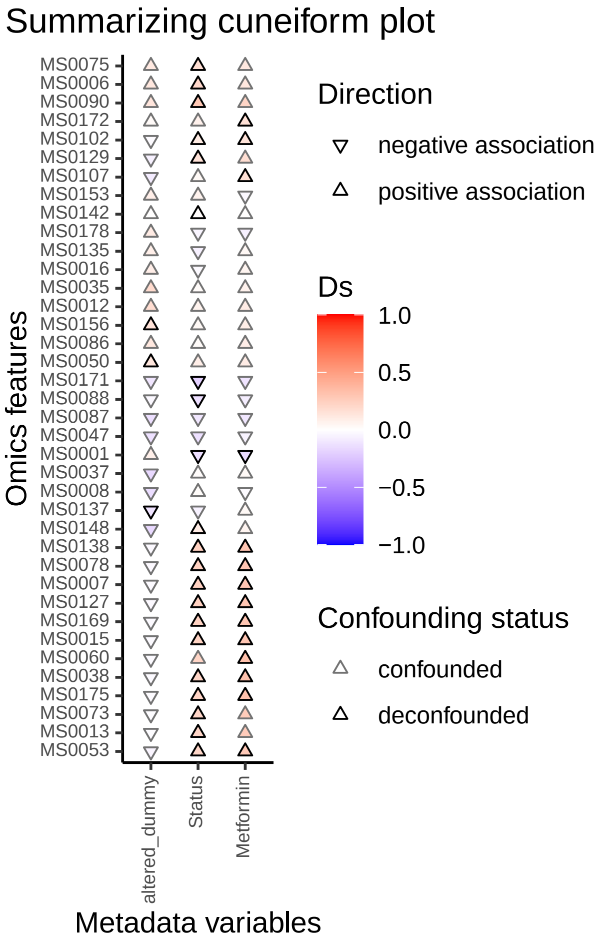

Figure 4: alternative output of the `BuildHeatmap()` function
A cuneiform plot is returned, showing associations between individual omics features (y-axis) and meta-variables (x-axis) as triangles. Fill color and direction of triangles indicate effect size (Cliff's Delta or Spearman's Rho for binary 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())
```

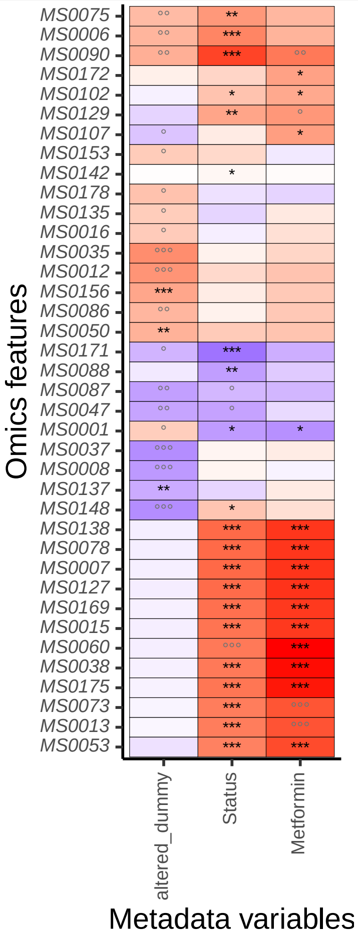

Figure 5: Manual changes to the default `BuildHeatmap()` ouptut
removal of legend, title, subtitle and italicization of feature names through `ggplot2::theme()` function.

4 Session info

- R version 4.3.0 (2023-04-21), `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`

- Time zone: `Europe/Berlin`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 22.04.4 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: detectseparation 0.3, ggplot2 3.4.2, gridExtra 2.3, kableExtra 1.3.4, metadeconfoundR 0.3.1
- Loaded via a namespace (and not attached): backports 1.4.1, BiocManager 1.30.20, BiocStyle 2.28.0, boot 1.3-28, checkmate 2.2.0, cli 3.6.1, codetools 0.2-19, colorspace 2.1-0, compiler 4.3.0, digest 0.6.31, doParallel 1.0.17, dplyr 1.1.2, evaluate 0.21, fansi 1.0.4, farver 2.1.1, fastmap 1.1.1, foreach 1.5.2, formatR 1.14, futile.logger 1.4.3, futile.options 1.0.1, generics 0.1.3, glue 1.6.2, grid 4.3.0, gtable 0.3.3, highr 0.10, htmltools 0.5.5, httr 1.4.6, iterators 1.0.14, knitr 1.43, labeling 0.4.2, lambda.r 1.2.4, lattice 0.21-8, lifecycle 1.0.3, lme4 1.1-33, lmtest 0.9-40, lpSolveAPI 5.5.2.0-17.9, magrittr 2.0.3, MASS 7.3-59, Matrix 1.5-1, minqa 1.2.5, munsell 0.5.0, nlme 3.1-162, nloptr 2.0.3, numDeriv 2016.8-1.1, parallel 4.3.0, pillar 1.9.0, pkgconfig 2.0.3, plyr 1.8.8, R6 2.5.1, Rcpp 1.0.10, registry 0.5-1, reshape2 1.4.4, rlang 1.1.1, rmarkdown 2.21, ROI 1.0-1, ROI.plugin.lpsolve 1.0-1, rstudioapi 0.14, rvest 1.0.3, scales 1.2.1, slam 0.1-50, splines 4.3.0, stringi 1.7.12, stringr 1.5.0, svglite 2.1.1, systemfonts 1.0.4, tibble 3.2.1, tidyselect 1.2.0, tools 4.3.0, utf8 1.2.3, vctrs 0.6.2, viridisLite 0.4.2, webshot 0.5.5, withr 2.5.0, xfun 0.39, xml2 1.3.4, yaml 2.3.7, zoo 1.8-12