FOS 2019 METABOLOMICS

Erik van den Akker & Marian Beekman

29 October 2019

Abstract

Cardiovascular disease is one of the leading causes of death world-wide and its type and onset is strongly associated with age and gender. Interestingly, many of the known serum-based risc factors for cardiovascular disease display a is a NMR metabolomics platform targeting ~230 metabolite parameters and is optimized to measure serum-based risc factors for cardiovascular disease and diabetes. Its design is heavily biased towards lipid and fat parameters, but includes measurements targeting other branches of the metabolome as well, like the glucose metabolism or amino acid metabolism. During this practical you will investigate which of the Nightingale metabolites displays strong differences between sexes, using a cohort of Dutch middle-aged participants of the Leiden Longevity Study (LLS). Participants in this cohort are the children (and their partners) of the second cohort of participants you will have access to and which you will employ for investigating the consistency of observed gender associations across different ages. By the end of these practicals, you should have a basic understanding of data quality control procedures and should be able to identify metabolites that putatively underly the observed age and sex biases in cardiovascular risk using a metabolome wide association analysis (METABO-WAS). similar dependency on age and gender. The Nightingale health platform

Contents

We have prepared data of the Leiden Longevity Study and ready-to-use R functions for analyses for the practicals. Load them like this:

```
load(url("https://raw.githubusercontent.com/molepi/FOS2017/master/Metabolomics_practical/metabolomics_F
source(url("https://raw.githubusercontent.com/molepi/FOS2017/master/Metabolomics_practical/FOS2017_META
```

Load the libraries required for the practicals:

```
library(limma)
library(ggplot2)
library(pander)
```

```
## Warning: package 'pander' was built under R version 3.4.1
```

Note: Startup messages have been surpressed in this vignette for a clear overview.
Note: The package 'pander' might not have been installed yet. If receiving an error run: *install.packages("pander")* and try again.

DATA EXPLORATION

Have a look at the objects loaded to your workspace:

```
ls()
```

```
##  [1] "ANSWER"            "count.miss"        "dat_partoffs"
##  [4] "dat_sibs"          "do.metabowas"      "phen_partoffs"
##  [7] "phen_sibs"         "plot.beta_beta"    "plot.na.heatmap"
## [10] "print.flag.report" "print.miss.report" "RIN"
```

The **L**eiden **L**ongevity **S**tudy (LLS) consists of two cohorts, **LLS_SIBS** (elderly siblings) and **LLS_PARTOFFS** (their offspring and partners thereoff), for which Nigthingale metabolomics data has been generated. Explore the objects storing metabolomics data, **dat_partoffs** and **dat_sibs** and the objects storing the phenotypic data **phen_partoffs** and **phen_sibs** by using *dim* and looking at the first few entries, e.g:

```
dim(dat_partoffs)
```

```
## [1]  231 2313
```

```
dat_partoffs[1:4,1:4]
```

```
##       LLS_PARTOFFS-937 LLS_PARTOFFS-1831 LLS_PARTOFFS-935 LLS_PARTOFFS-716
## acace          0.02724           0.04410          0.00362          0.02829
## ace            0.02652           0.02117          0.02700          0.02716
## ala            0.25490           0.28930          0.41290          0.35190
## alb            0.08420           0.08649          0.09922          0.09055
```

Notice that the objects storing metabolomics data are matrices with numbers, and that the *columns* in each of the data matrices correspond to *samples* and the *rows* to *measurements*.

EXCERCISES

Q1: How many measurements are there in **dat_partoffs**? How is the first measurement called? And the first sample?
Q2: How many samples are there in **dat_sibs**?

Next to the metabolomics data, two data objects are given that hold phenotypic data: **sibs_partoffs** and **phen_sibs**. Again, explore these using *dim* and looking at the first few entries, e.g:

```
dim(phen_partoffs)
```

```
## [1] 2313   18
```

```
phen_partoffs[1:4,1:4]
```

```
##                   age   sex abnormal_macromolecule_a low_glucose
## LLS_PARTOFFS-937   53 FALSE                        0           0
## LLS_PARTOFFS-1831  59  TRUE                        0           0
## LLS_PARTOFFS-935   54 FALSE                        0           0
## LLS_PARTOFFS-716   49  TRUE                        0           0
```

Verify that the matrices containing the metabolomics measurements have the same ordering of samples as the matrices containing the phenotypic data, using:

```
all(rownames(phen_partoffs) == colnames(dat_partoffs))
```

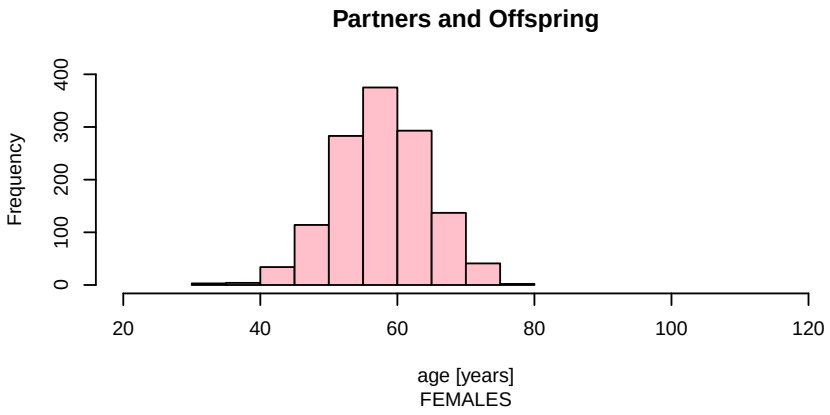
```
## [1] TRUE
```

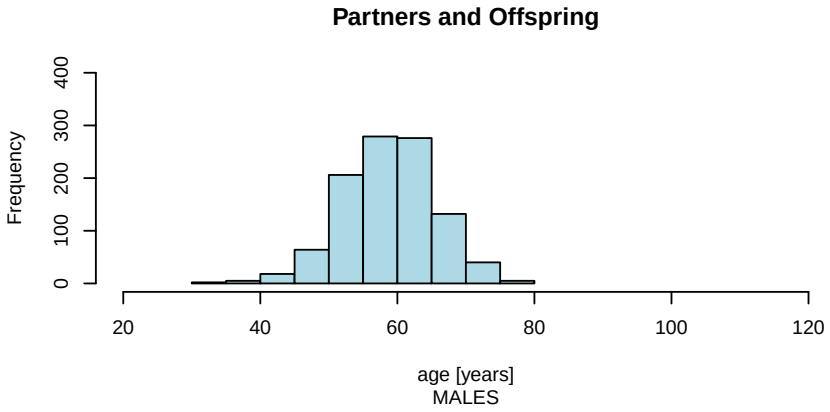
EXCERCISES
Q3: Verify this for **phen_sibs** too. What do the functions 'colnames', and 'rownames' do?
HINT: Rerun part of the code above, e.g: *rownames(phen_sibs)*

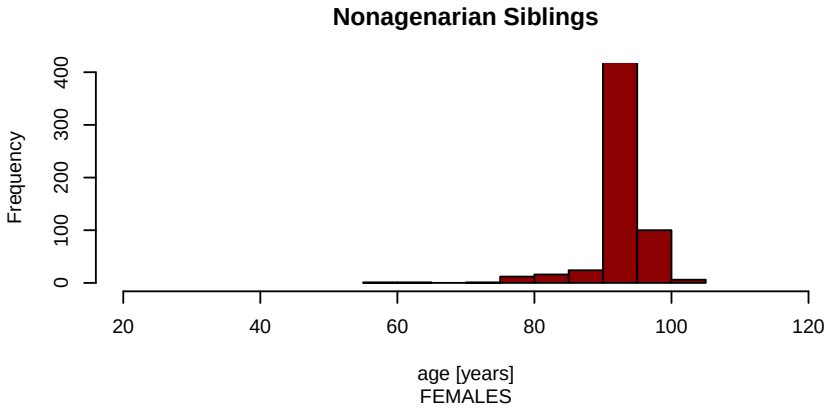
Phenotypes

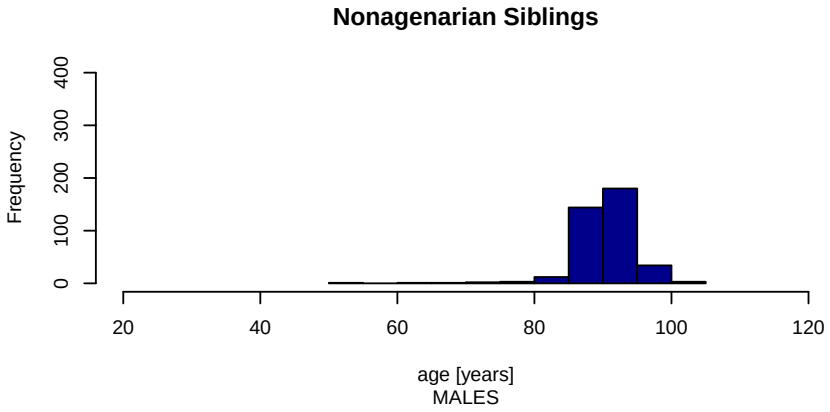
You will have access to data of two cohorts within the Leiden Longevity Study, each with its own age and sex distribution. To explore these distributions, repeatedly use histograms:

```
# Open a figure with 4 x 1 pannels:
par(mfrow=c(4,1))
# Set range in years equal for all histograms:
XLIM <- c(20,120)
# Set range in counts of histogram equal for all histograms:
YLIM <- c(0,400)
# Plot histograms on age:
# Ages of females in partoffs:
age_fpo <- phen_partoffs[which(phen_partoffs$sex==FALSE),"age"]
# Draw a histogram:
hist(age_fpo,xlim=XLIM,ylim=YLIM,col="pink",xlab="age [years]",main="Partners and Offspring",sub="FEMALE
# For males in partoffs:
age_mpo <- phen_partoffs[which(phen_partoffs$sex==TRUE),"age"]
hist(age_mpo,xlim=XLIM,ylim=YLIM,col="lightblue",xlab="age [years]",main="Partners and Offspring",sub="M
# For females in sibs:
age_fs <- phen_sibs[which(phen_sibs$sex==FALSE),"age"]
hist(age_fs,xlim=XLIM,ylim=YLIM,col="darkred",xlab="age [years]",main="Nonagenarian Siblings",sub="FEMA
# For males in sibs:
age_ms <- phen_sibs[which(phen_sibs$sex==TRUE),"age"]
hist(age_ms,xlim=XLIM,ylim=YLIM,col="darkblue",xlab="age [years]",main="Nonagenarian Siblings",sub="MALE
```


Partners and Offspring
Frequency
0
100
200
300
400
20
40
60
80
100
120
age [years]
FEMALES


Partners and Offspring
Frequency
0
100
200
300
400
20
40
60
80
100
120
age [years]
MALES


Nonagenarian Siblings
Frequency
0
100
200
300
400
20
40
60
80
100
120
age [years]
FEMALES

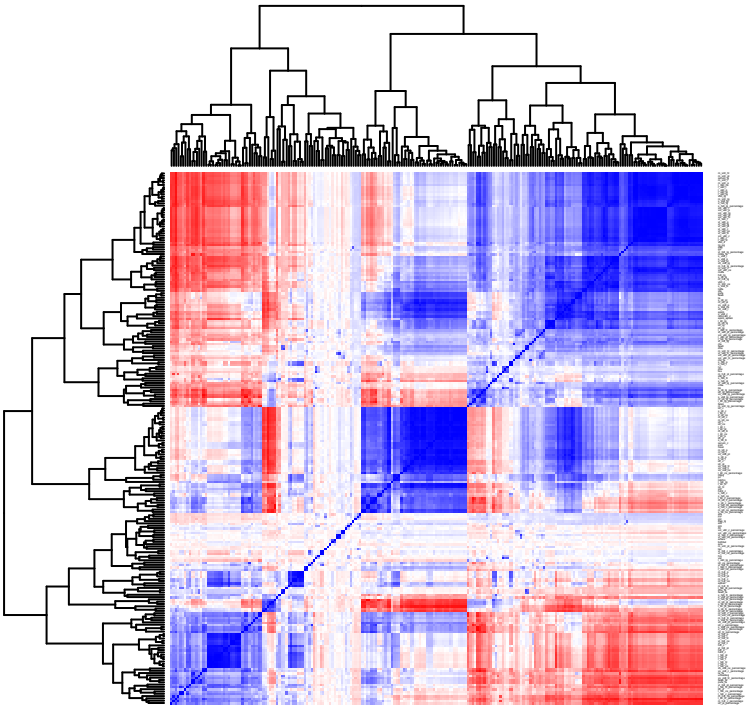

Nonagenarian Siblings
Frequency
0
100
200
300
400
20
40
60
80
100
120
age [years]
MALES

EXCERCISES
Q4: Viewing the four age distributions, what do you think what the mean age is of each of the four groups?
Q5: Verify this using the function *summary*, e.g.: *summary(age_fpo)*

Metabolomics

As previously stated, the Nightingale platform reports on different types of metabolites, but has a focus on lipid parameters. These lipid parameters measures are highly intercorrelated. To vizualize the correlation structure between metabolites draw a heatmap of of pairwise correlations of all metabolites (see function cor, use "pairwise.complete.obs").

```
cors <- cor(t(dat_partoffs), use="pairwise.complete.obs")
heatmap(cors,col=colorRampPalette(c("red", "white", "blue"))(256), scale="none", labCol=NA, cexRow=0.15)
```



On the rows and columns of this heatmap are metabolites. Strong positive correlations ("blue") are observed along the diagonal (including correlations with itself), whereas strong negative correlations ("red") are only observed off diagonal.

EXCERCISES
Q6: Where do you think are the lipids located in the heatmap?

QUALITY CONTROL

Discontinued metabolites

Nightingale discontinued reporting on 5 of the metabolites due to high technical variation. Make sure these are not these are removed from the data by typing:

```
IND <- which(rownames(dat_partoffs) %in% c("dag","dagtg","falen","cla","cla_fa"))
dat_partoffs <- dat_partoffs[-IND,]
dat_sibs <- dat_sibs[-IND,]
```

QC Flags

The final columns in the phen matrices are actually not phenotypes, but contain quality control flags provided by Nightingale. Print an overview of these flags using print.flag.report.

```
print.flag.report(phen_partoffs)
```

QCflag	counts
abnormal_macromolecule_a	0
low_glucose	0
low_glutamine_high_glutamate	0
low_protein_content	0
high_citrate	0
high_ethanol	36
high_lactate	0
high_pyruvate	0
serum_sample	0
unidentified_small_molecule_a	0
unidentified_small_molecule_b	0
unknown_acetylated_compound	0
isopropyl_alcohol	0
polysaccharides	0
aminocaproic_acid	0
fast	0
ANYFLAG	36
NOFLAG	2277

EXCERCISES
Q7: Some samples in partoffs have a warning of high ethanol contents. Where does this ethanol come from?
Q8: In **phen_sibs** next to cases with high ethanol flags, some other types of warnings for out of range measurements are given. What are they? To what could they point?

Lets exclude the samples having a "high_ethanol" QCflag in dat_partoffs. First find the rownumbers of samples having a QCflag:

```
IND <- which(phen_partoffs[,"high_ethanol"]==1)
IND
```

```
##  [1]    6   56   65  120  150  204  243  261  312  346  353  372  415  458
## [15]  492  495  697  712  735  743  756  757  834 1012 1040 1050 1097 1207
## [29] 1262 1458 1630 1670 1791 2071 2087 2150
```

Then remove these samples from **dat_partoffs** and **phen_partoffs**:

```
dat_partoffs <- dat_partoffs[,-IND]
phen_partoffs <- phen_partoffs[-IND,]
```

Check whether the two files still contain the same samples in the same ordering:

```
all(rownames(phen_partoffs) == colnames(dat_partoffs))
```

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

Rerun the QC overview:

```
print.flag.report(phen_partoffs)
```

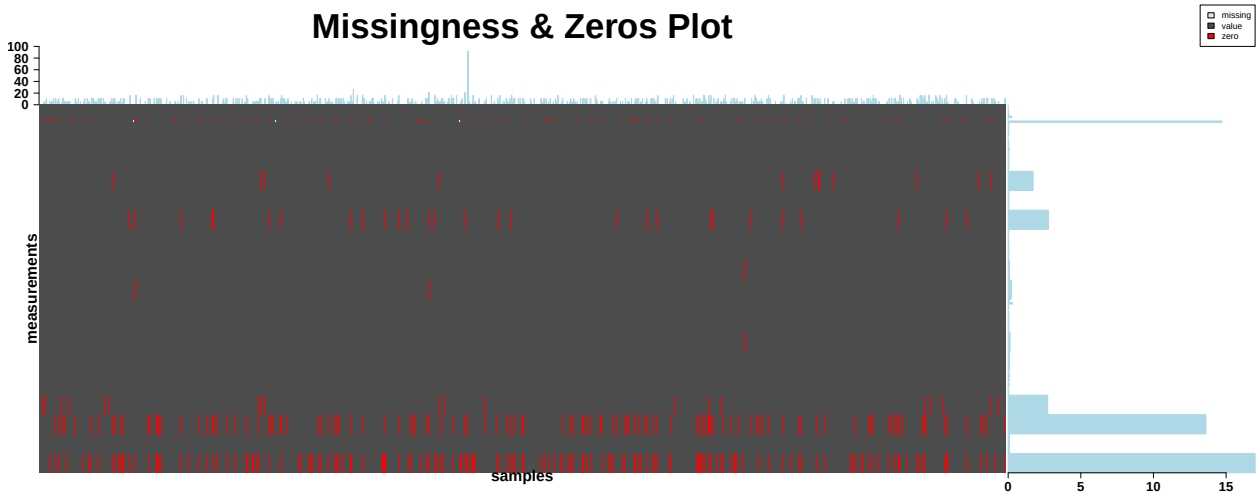
QCflag	counts
abnormal_macromolecule_a	0
low_glucose	0
low_glutamine_high_glutamate	0
low_protein_content	0
high_citrate	0
high_ethanol	0
high_lactate	0
high_pyruvate	0
serum_sample	0
unidentified_small_molecule_a	0
unidentified_small_molecule_b	0
unknown_acetylated_compound	0
isopropyl_alcohol	0
polysaccharides	0
aminocaproic_acid	0
fast	0
ANYFLAG	0
NOFLAG	2277

EXCERCISES
Q9: Perform this QC step for **dat_sibs** too and apply the exclusion of samples to both the data, -**dat_sibs**-, and phenotypes, -**phen_sibs**-.
HINT: You can filter for multiple QC flags by repeating the procedure you just performed.

Missingness & zeros

Another aspect of data quality control is to look into missing or zero measurements. In case a measurement could not be succesfully obtained, this is indicated with a missing value (in our case NA). Failing measurements may have many causes, and this missingness may become problematic if it somehow relates to the outcome of your study.
Likewise, we are especially careful with measurements with a zero result. Again, there may be many reasons why a measurement for a specific sample equals 0, for instance when the measured metabolite abundances are below the detection treshhold of the machine. Often the occurence of missing or zero measurements are not randomly distributed in a dataset. To get a feeling whether the missing and zero values are linked to particular samples or measurements, we plot a missingness heatmap:

```
plot.na.heatmap(dat_partoffs)
```

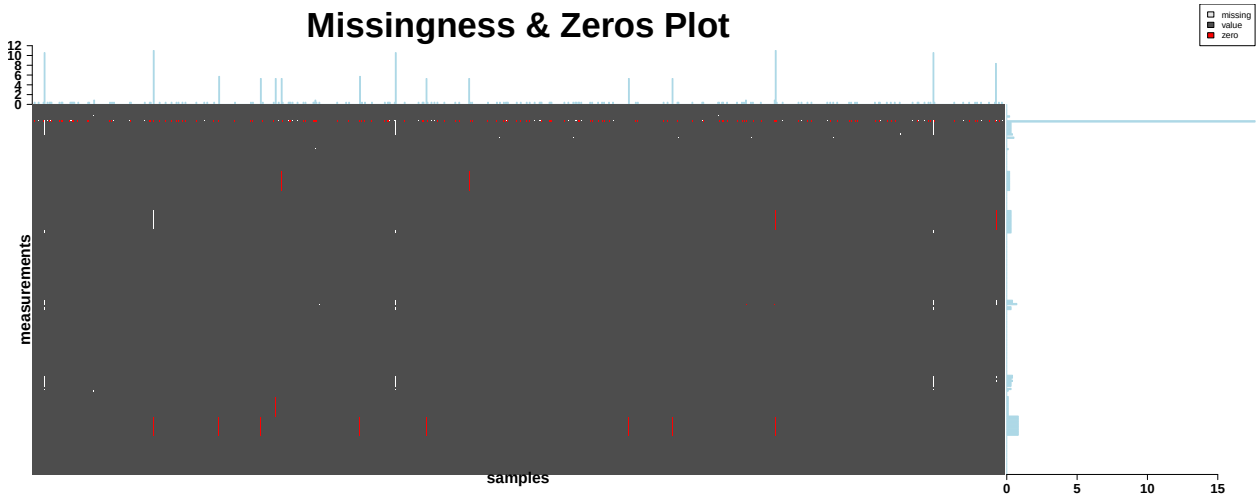


The colors in the heatmap indicate the status of a particular measurement (rows) for a particular sample (columns). The barplots on the sides indicate the percentage of missing + zero measurements per metabolite (right pannel) or per sample (top pannel). Note that a single sample almost completely failed, as indicated by the grey vertical bar. Also note that apparently some measurements are apparently more prone to fail than others and that missing and zero values co-occur in these particular measurements.

EXCERCISES

Q10: Draw the missingness heatmap for **dat_sibs**. Is the degree of missing and zero measurements comparable between the studies? What is different between the studies in terms of number of failed samples and problematic measurements?

HINT: Also have a a look at the scale of the axis!



As a first step we exclude problematic samples. To get an overview of the missingness in samples, we use print.miss.report:

```
print.miss.report(dat_partoffs,on_sample=TRUE,type="missingOrZero")
```

Table 3: missingOrZero in samples

# missMeas	[%]	# samp	[%]
0	0	1515	66.5
1	0.441	230	10.1
2	0.881	1	0.0439
3	1.32	1	0.0439

# missMeas	[%]	# samp	[%]
5	2.2	1	0.0439
12	5.29	202	8.87
13	5.73	46	2.02
24	10.6	178	7.82
25	11	49	2.15
36	15.9	32	1.41
37	16.3	18	0.791
48	21.1	2	0.0878
61	26.9	1	0.0439
209	92.1	1	0.0439

The first column describes the number of measurements missing for a sample; the second puts this as a percentage of all measurements; the third column states the number of samples having this missing number of measurements; the fourth again puts this as a percentage of all samples. Hence, viewing this output, we note that 1515 samples had all measurements complete (first row), whereas only a single sample almost completely failed (final row).

EXCERCISES
Q11: How many samples had at most two failed measurements?

Thresholds for a maximum number of missing values are always arbitrary and here we will stick to keeping samples having 5% or less zero or missing values. This corresponds to a maximum of 5 zero or missing values per sample.

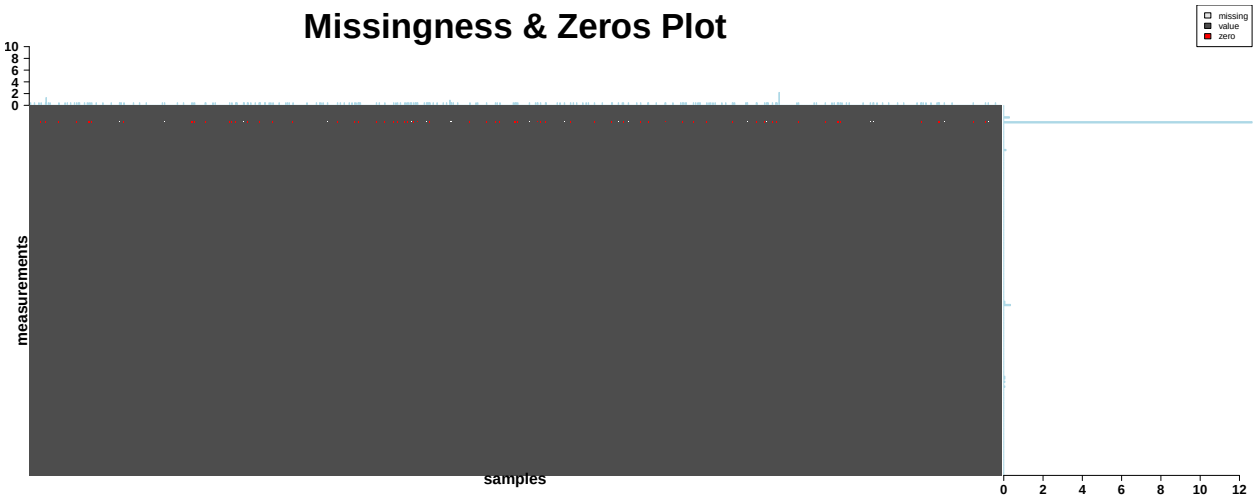
EXCERCISES
Q12: What percentage of samples had at most 5 zero or missing values per sample?

Excluding samples on this criterium can be done using:

```
## Getting the counts per sample:
counts <- count.miss(dat_partoffs,on_sample=TRUE)
## Finding the row numbers of samples to exclude:
IND <- which(counts$miss_or_zero>5)
## Subset data:
dat_partoffs <- dat_partoffs[,-IND]
phen_partoffs <- phen_partoffs[-IND,]
```

Lets have a look at the missingness heatmap again:

```
plot.na.heatmap(dat_partoffs)
```



Stringent exclusion of samples off course also alleviates some of the problems with problematic measurements. However, it appears that there is still a measurement left with a high number of missing or zero values. Again use print.miss.report for a detailed overview (with 'on_sample' set to FALSE):

```
print.miss.report(dat_partoffs,on_sample=FALSE,type="missingOrZero")
```

Table 4: missingOrZero in measurements

# missSamp	[%]	# meas	[%]
0	0	217	95.6
1	0.0572	6	2.64
2	0.114	1	0.441
5	0.286	1	0.441
6	0.343	1	0.441
221	12.6	1	0.441

Analogous to the previous report characterizing **samples** on the number of **missing measurements**, this overview now describes **measurements** in terms of numbers of **failed samples**. Hence, the first column describes the number of unsuccesfully measured samples ranging from 0 to 221; the second puts this as a percentage of all typed samples; the third column states the number of measurements having this missing number of failed samples; the fourth again puts this as a percentage of all measurements. Viewing this output, we note that 217 measurements were succesfully measured in all remaining samples (first row), whereas only one measurment failed in more than 12% of the samples (final row).

Thresholds for the maximum number of failed samples for a measurement is arbitrarely set to 10%, corresponding to a maximum of 6 failed samples:

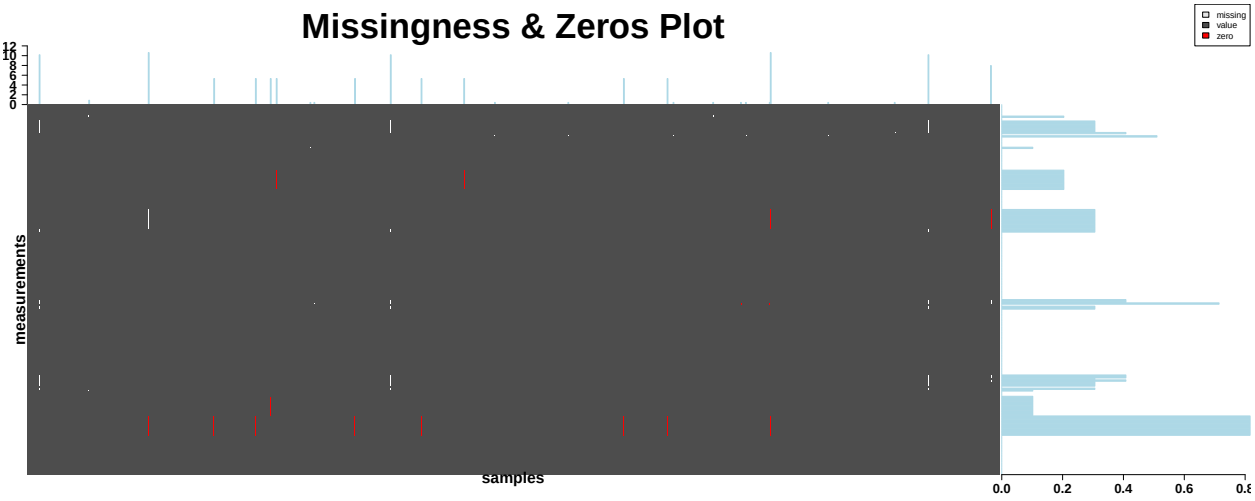
```
## Getting the counts per measurement:
counts <- count.miss(dat_partoffs,on_sample=FALSE)
## Finding the column numbers of measurements to exclude:
IND_meas <- which(counts$miss_or_zero>6)
## Subset data:
dat_partoffs <- dat_partoffs[-IND_meas,]
```

Since we want to compare results within **partoffs** and **sibs**, we discard the same measurements in **sibs**:

```
## Subset data:
dat_sibs <- dat_sibs[-IND_meas,]
```

To check whether there are problematic samples or measurements left in **dat_sibs** we run once more:

```
plot.na.heatmap(dat_sibs)
```



And observe that no other measurement exceeds the 10% treshhold and only a few samples exceed 5% threshold. To remove these samples too:

```
print.miss.report(dat_sibs,on_sample=TRUE,type="missingOrZero")
```

Table 5: missingOrZero in samples

# missMeas	[%]	# samp	[%]
0	0	953	97.2
1	0.442	11	1.12
2	0.885	1	0.102
12	5.31	9	0.918
18	7.96	1	0.102
23	10.2	3	0.306
24	10.6	2	0.204

```
## Getting the counts per sample:
counts <- count.miss(dat_sibs,on_sample=TRUE)
## Finding the row numbers of samples to exclude:
IND <- which(counts$miss_or_zero>2)
## Subset data:
dat_sibs <- dat_sibs[,-IND]
phen_sibs <- phen_sibs[-IND,]
```

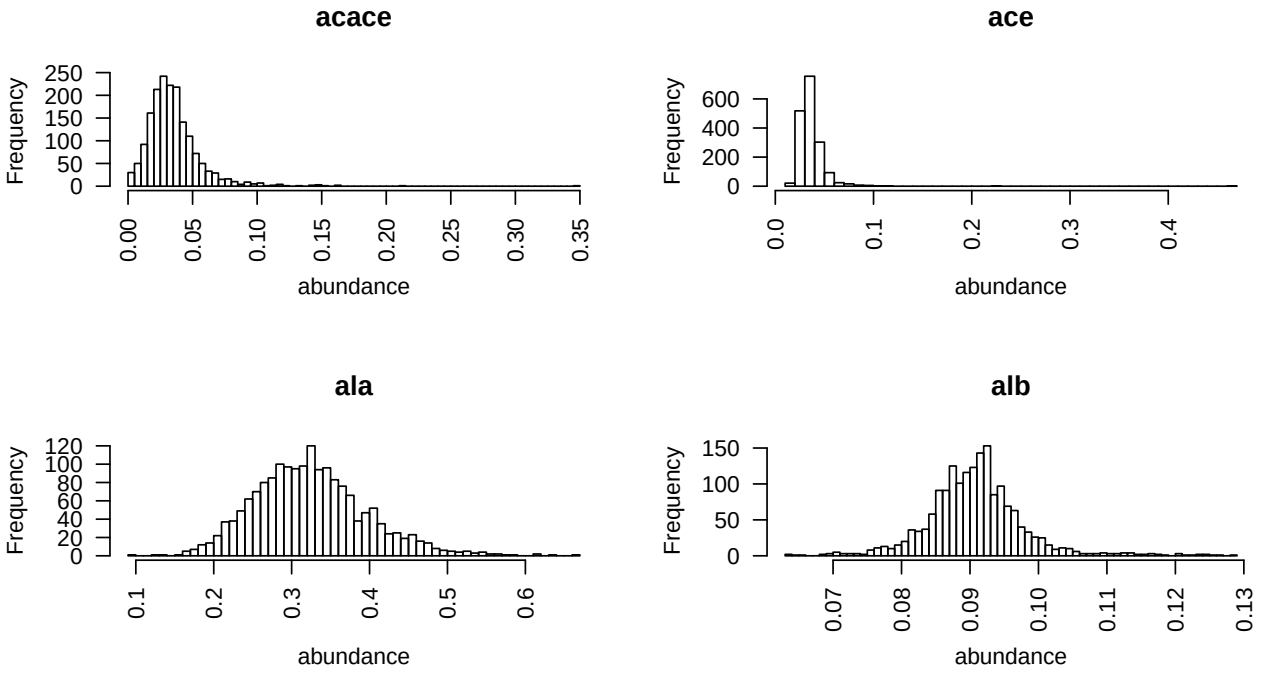
DATA TRANSFORMATION

Normality & standardization

In order to investigate whether metabolites may serve as a biomarker for partcular traits, we model the metabolite as the dependent variable and the trait as the independent variable, ergo: metabolite~age+sex+trait In this model, we require the metabolites to be normally distributed. However, some of the measured

metabolites have a skewed distribution instead. Have a look at the distributions of the first 4 metabolites, using:

```
par(mfrow=c(2,2))
for(i in 1:4){
  hist(dat_partoffs[i,],main=rownames(dat_partoffs)[i],50,xlab="abundance",las=2)
}
```

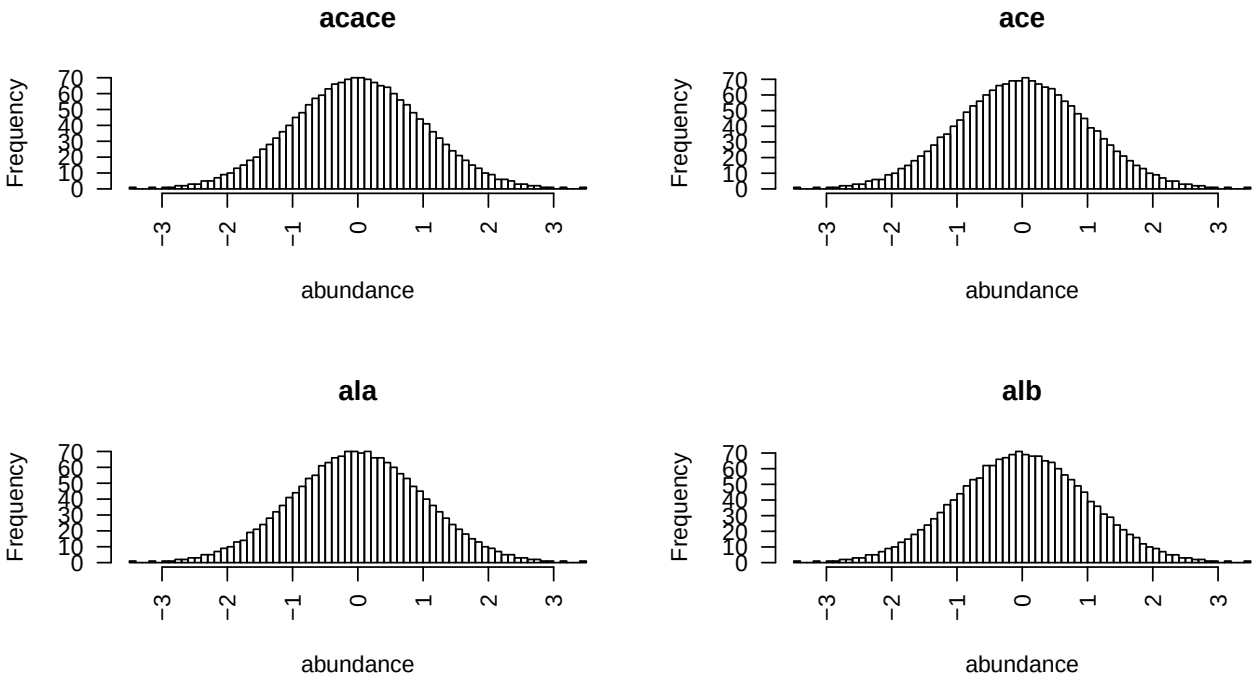


EXCERCISES

Q13: Which of the four metabolites has a skewed distribution, and which ones not?

Moreover, metabolites each have their own mean and variance. To be able to compare the effects of metabolites on traits, we usually standardize the distributions of metabolites, so that each metabolite is distributed with a comparable mean and variance. We can achieve normality, -'unskewing' the distribution-, and standardization, -metabolites having the same mean and variance-, in various ways and one method is the **R**ank **I**nverse **N**ormal (RIN) transformation. When applying this to the four metabolites above, we obtain:

```
par(mfrow=c(2,2))
for(i in 1:4){
  hist(RIN(dat_partoffs[i,]),main=rownames(dat_partoffs)[i],50,xlab="abundance",las=2)
}
```



Apply RIN transformation on **dat__partoffs** and **dat__sibs** using:

```
new_dat_partoffs <- RIN(dat_partoffs)
new_dat_sibs <- RIN(dat_sibs)
```

ANALYSIS

Data is now ready for analysis! In order to identify metabolites that have a different serum level in males versus females, we will subject all metabolite measurements within a cohort to a **M**etabolome **W**ide **A**ssociation **S**tudy (Metabo-WAS) on gender. Performing the Metabo-WAS on sex in both cohorts independently, will allow us to identify metabolites whos gender differences change with age.

Metabo-WAS

In the following analysis, we will associate each of the metabolite levels with sex, while adjusting for age:

```
res_partoffs <- do.metabowas(phen=phen_partoffs,dat=new_dat_partoffs,test_variable="sex",covariates=c("a
head(res_partoffs)
```

```
##             estimate     tstat          pval      pval.adj
## crea       1.0000928  23.05299 2.801413e-107 6.331193e-105
## l_hdl_p   -0.9256082 -20.90750  6.127402e-90  6.923964e-88
## l_hdl_l   -0.9199949 -20.76038  8.568649e-89  6.455049e-87
## l_hdl_pl  -0.9160153 -20.66006  5.137197e-88  2.902517e-86
## hdl_c     -0.8958867 -20.13086  5.876338e-84  2.656105e-82
## xl_hdl_pl -0.8938019 -20.07125  1.665231e-83  6.272370e-82
```

To find the number of significantly associated metabolites after correction for multiple testing (FDR), type:

```
length(which(res_partoffs$pval.adj<=0.05))
```

```
## [1] 189
```

Then repeat the procedure for **sibs**:

```
res_sibs <- do.metabowas(phen=phen_sibs,dat=new_dat_sibs,test_variable="sex",covariates=c("age"))
head(res_sibs)
```

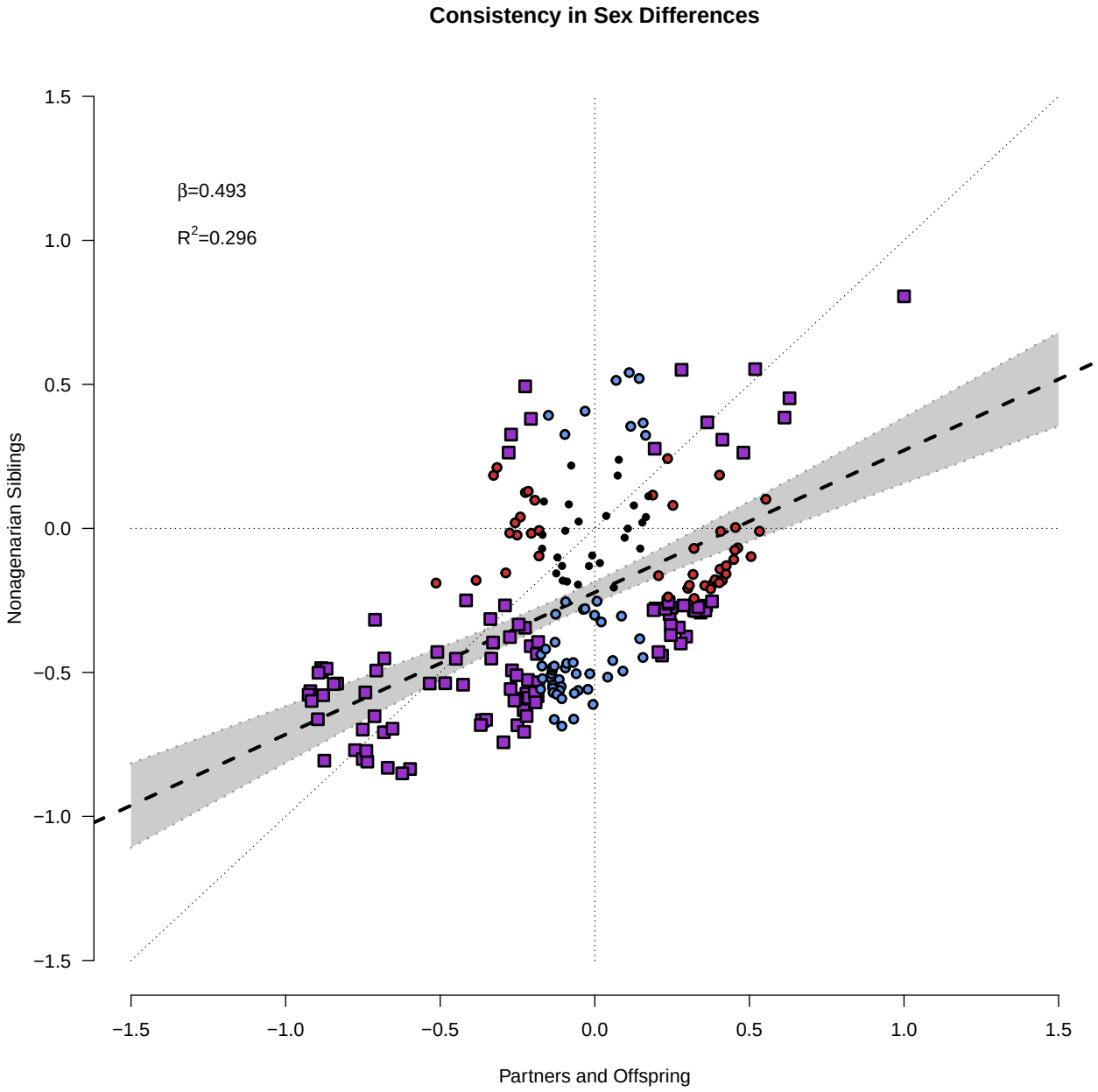
```
##           estimate     tstat         pval     pval.adj
## totpg   -0.8503664 -12.77677 2.280188e-37 5.153224e-35
## pc      -0.8352917 -12.53816 4.750808e-36 5.368412e-34
## totcho  -0.8310673 -12.48680 9.065435e-36 6.829294e-34
## m_hdl_p -0.8096374 -12.16482 4.908769e-34 2.773455e-32
## apoa1   -0.8063387 -12.11525 8.992077e-34 3.765195e-32
## crea     0.8057605  12.10657 9.996092e-34 3.765195e-32
```

EXCERCISES
Q14: How many metabolite parameters are significantly associated with gender in **sibs**?

beta's vs. beta's

To make a systematic comparisson between the gender associations in **partoffs** and **sibs**, we draw a scatterplot of observed effects in the middle-aged cohort versus the observed effects in the elderly cohort using:

```
plot.beta_beta(res1=res_partoffs,res2=res_sibs,main="Consistency in Sex Differences",
               xlab="Partners and Offspring",ylab="Nonagenarian Siblings")
```



Each of the dots represents a metabolite parameter. We have added a trendline with confidence intervals to illustrate the general trend. The top right and bottom left quadrants of the plot show the metabolites with consistent gender effects in the two groups, whereas the top left and bottom right show metabolites with opposite effects. The colors and shapes indicate the significance (after Bonferroni adjustment) in both cohorts. Purple squares indicate a significant effect in both cohorts; circles, -blue or red-, indicate significant effects in only one of the cohorts and solid dots indicates no significant effects in both cohorts. Notice the big group of purple squares in the bottom left quadrant. These indicate consistent significantly lowered levels within both age groups. Notice also the big group of purple squares in the bottom right corner. These indicate metabolites are significantly associated in the opposite direction across both age groups.

EXCERCISES

Q15: Browse your Metabo-WAS to find examples for which the sex-effect is significantly opposite between cohorts.

SESSIONINFO

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] pander_0.6.1    BiocStyle_2.4.1 ggplot2_3.1.0   limma_3.32.2
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.0       rstudioapi_0.9.0 bindr_0.1        knitr_1.20
##  [5] magrittr_1.5     munsell_0.5.0    colorspace_1.4-0 R6_2.3.0
##  [9] rlang_0.3.1      dplyr_0.7.2      stringr_1.3.1    plyr_1.8.4
## [13] tools_3.4.0      grid_3.4.0       gtable_0.2.0     withr_2.1.2
## [17] htmltools_0.3.6  assertthat_0.2.0 yaml_2.2.0       lazyeval_0.2.1
## [21] rprojroot_1.3-2  digest_0.6.18    tibble_2.0.1     crayon_1.3.4
## [25] bindrcpp_0.2     glue_1.2.0       evaluate_0.10.1  rmarkdown_1.10.2
## [29] stringi_1.2.2    compiler_3.4.0   pillar_1.3.1     scales_1.0.0
## [33] backports_1.1.3  pkgconfig_2.0.2
```