SUPPLEMENTARY METHODS

Longitudinal metabolic and gut bacterial profiling of pregnant women with previous bariatric surgery

Study population, sampling and clinical data

The population is part of an ongoing prospective study investigating the impact of maternal BS on perinatal outcomes. The study was approved by the West London Research Ethics Committee (No: 14/LO/0592) and all women gave written, informed consent for their data and samples to be used. Pregnant women with and without previous BS were recruited from May 2015 to April 2017 at Chelsea & Westminster Hospital (London, UK) as previously described[1]. Women were seen at 5 time points during pregnancy (**T1**: 11^{+0}-14^{+0}, **T2**: 20^{+0}-24^{+0}, **T3**: 28^{+0}-30^{+0}, **T4**: 30^{+0}-33^{+0} and **T5**: 35^{+0}-37^{+6} weeks gestation) and within 72 h of delivery (**T6**). Maternal blood (serum) and urine samples were collected at each visit while fecal samples were requested at the T1, T2 and T4 visits (see online supplementary figure S1 and table S1). A full oral glucose tolerance test (2 h, 75 g) was conducted at T3 and maternal insulin resistance was calculated using the homeostatic model assessment for insulin resistance (HOMA-IR=fasting serum insulin (μU/L) x fasting glucose (mmol/L)/22.5)[2]. Estimated fetal weight was calculated by trans-abdominal ultrasound scans[3] at T2, T4 and T5. At T6, birth weight was recorded, percentiles for the gestation were calculated[4] and, where possible, neonatal samples (cord serum and urine) were collected. All samples were stored at -80^{o}C for future analysis. For the current study population, women with diagnosis of type 2 diabetes mellitus or GDM (due to the effect of diabetes on the metabolic profile) and those that had a miscarriage were excluded. Only NBS participants with a BMI of 25 to 50 kg/m^2 at T1 were included to match the BMI range of the included bariatric patients at T1.

Metabolic profiling of biofluid samples

Serum and urine samples were prepared according to an established protocol[5]. ^{1}H NMR spectra were acquired on a Bruker 600 MHz spectrometer (Bruker BioSpin, Karlsruhe, Germany) following a published method[5]. Briefly, experiments for serum samples were run at 310 K using a Carr-Purcell-Meiboom-Gill (CPMG) spin-echo pulse sequence with water presaturation (Bruker pulse program: cpmgpr1d). For urinary samples, experiments were run at 300 K using a standard one-dimensional (1D) spectroscopy pulse sequence with water presaturation (noesygppr1d). J-resolved (J-res) experiments (jresgpprqf) were also run for all samples to increase capacity for structural identification of molecules. The spectra were automatically baseline corrected, phased and referenced to sodium 3-(trimethylsilyl) propionate-2,2,3,3-d_4 (TSP) in the Bruker TopSpin 3.1 software. Raw data were processed in MATLAB version R2016b (The MathWorks, Inc., Natwick, MA) using scripts developed by Dr. T. Ebbels at Imperial College. Spectra were aligned using the recursive segment-wise peak alignment (RSPA) algorithm[6] and normalised with the probabilistic quotient normalisation (PQN) function[7]. Downstream multivariate modelling was performed in SIMCA 14.1 (Sartorius Stedim Biotech, Aubagne, France).

Statistical modelling and analysis of metabolic profiles

Unsupervised principal component analysis (PCA) models were used to assess variation in metabolic profile over all time points and to identify extreme outliers (based on Hotelling's T^2 statistic) to exclude from supervised models for each time point. Orthogonal partial least squares discriminant analysis (OPLS-DA)[8] was used to identify spectral variables that contributed to discrimination of clinical classes (pairwise comparisons between NBS, MAL and RES groups) at each time point. Valid models were determined by positive Q^2Y value and significant ($P<0.05$) ANOVA of the cross-validated residuals (CV-ANOVA[9]). NMR peaks were considered discriminatory if their correlation with the predictive component was 0.45 or greater. Relative concentrations of metabolites were calculated by integrating a representative peak of each identified metabolite. Downstream analysis was performed in the R software environment[10]. Time-series curves were generated for each discriminatory metabolite with "santaR". In this method, the measurements for each individual are condensed into a smooth, continuous function of time and mean curves are then calculated for each study group. Individuals with at least 5 data points were included. Significance of the distance between group mean trajectories is tested using a permutation-based method. Changes in metabolite concentrations at each time point were represented by $\log_2$(Fold Change) where Fold Change (FC) = mean(Case)/mean(Control) and significance was assessed by Mann-Whitney U test. Spearman's correlation coefficients (ρ) and *P* values were calculated for metabolite-clinical associations with the package "Hmisc". Partial correlations were calculated with "ppcor"[11] to adjust for confounding variables (maternal age and BMI on maternal measurements and maternal age, BMI and HOMA-IR on birthweight percentile). Biospecimens with missing data were removed from the analysis. All *P* values were adjusted (P_{adj}) where necessary to control the false discovery rate according to the Benjamini-Hochberg method[12]. An alpha of 0.05 was used for *P* and P_{adj} values. Plots were generated with "ggpubr" apart from the correlation network visualisation which was generated with Cytoscape version 3.5.1[13].

Metabolite identification

Discriminatory NMR peaks were used as driver peaks for statistical total correlation spectroscopy (STOCSY)[14] and subset optimization by reference matching (STORM)[15], two algorithms used to identify additional peaks corresponding to the same molecule as the driver peak. Chemical shifts, multiplicities, and J-couplings for each molecule were matched to an in-house reference database for annotation. Ambiguous annotations were confirmed with 2D NMR experiments (J-res, HSQC[16]) and by spiking in authentic standards[5].

Gut bacterial community profiling

Stool samples were randomised for processing and DNA was extracted from 250 mg stool using the PowerLyzer PowerSoil DNA Isolation Kit (Mo Bio, Carlsbad, CA, USA). Bead beating was carried out in a Bullet Blender Storm (Chembio Ltd., St. Albans, UK) for 3 min at speed 8. DNA was quantified using the Qubit fluorometric assay (Thermo Fisher Scientific, Carlsbad, CA, USA). Sample libraries amplifying the V1-V2 region of the 16S rRNA gene were prepared as previously described[17] and were sequenced on the Illumina MiSeq platform (San Diego, CA, USA) using the MiSeq Reagent Kit v3 and paired-end 300 bp chemistry. Primer sequences were removed from demultiplexed fastq files using cutadapt[18] and raw reads were processed in the R software environment[10] following a published workflow[19] which includes amplicon denoising[20]. The denoising algorithm "DADA2" uses error profiles generated during the sequencing run to infer real sequence variants; it allows the analysis of unique sequences rather than operational taxonomic units (OTUs), or clusters of similar sequences. Taxonomy was assigned with reference to the RDP[21] database and assignments of statistically significant taxa were confirmed with the SILVA[22] database.

Statistical analysis of microbiome data

Functions in the "vegan" R package were used to calculate Shannon Diversity Indices[23] on data rarefied to the minimum sequencing depth and Bray-Curtis dissimilarity[24] on log-transformed (pseudocount of 1 added to each value) data. Permutational multivariate analysis of variance (PERMANOVA)[25] was applied to the Bray-Curtis matrix to test whether the taxa distributions were different between the clinical classes. Changes in relative abundance were tested at each taxonomic rank from phylum to genus using the Mann-Whitney U test while differentially abundant 16S rRNA gene sequences were identified using "DESeq2" on raw counts[26]. DESeq2 implements a statistical model to account for the sparsity of the count matrix and over-dispersion of the counts, two features that are characteristic of 16S rRNA gene amplicon sequencing data. For "DESeq2" analysis, data were pooled for each individual rather than analysing distinct time points.

Integrative analysis of metabolic and taxonomic data

Relationships between the serum, urine and faecal datasets were modelled using the DIABLO method in "mixOmics"[27]. This is a multi-block latent variable-based approach which aims to identify concordance between multiple datasets. Metabolites significant throughout the time course (serum: leucine, isoleucine, isobutyrate, D-β-hydroxybutyrate; urine: PAG, PCS, IS, PHPA, unknown, α-ketoisovalerate, creatinine) and a subset of bacterial genera (log-transformed; selected using the LASSO penalization method implemented in "mixOmics") were modelled. Sampling points for each individual where matching microbiome and metabolite data were obtained (T1, T2 and T4) were included in the model.

All R code, packages and package versions used for data analysis can be found from page 7 of this document. The R Markdown file and data to reproduce the analysis are availale on GitHub (https://github.com/ka-west/PBS_manuscript).

REFERENCES

1 Maric T, Kanu C, Johnson MR, Savvidou MD. Maternal, neonatal insulin resistance and neonatal anthropometrics in pregnancies following bariatric surgery. Metabolism 2019;**97**:25-31.
2 Matthews DR, Hosker JR, Rudenski AS, Naylor BA, Treacher DF, Turner RC. Homeostasis model assessment: insulin resistance and fl-cell function from fasting plasma glucose and insulin concentrations in man Diabetologia 1985;**28** 412-9.
3 Maršál K, Persson P- H, Larsen T, Lilja H, Selbing A, Sultan B. Intrauterine growth curves based on ultrasonically estimated foetal weights. Acta Paediatrica 1996;**85**:843-8.
4 Poon LCY, Volpe N, Muto B, Syngelaki A, Nicolaides KH. Birthweight with Gestation and Maternal Characteristics in Live Births and Stillbirths. Fetal Diagnosis and Therapy 2012;**32**:156-65.
5 Dona AC, Jiménez B, Schäfer H, Humpfer E, Spraul M, Lewis MR*, et al.* Precision High-Throughput Proton NMR Spectroscopy of Human Urine, Serum, and Plasma for Large-Scale Metabolic Phenotyping. Analytical Chemistry 2014;**86**:9887-94.
6 Veselkov KA, Lindon JC, Ebbels TMD, Crockford D, Volynkin VV, Holmes E*, et al.* Recursive Segment-Wise Peak Alignment of Biological 1H NMR Spectra for Improved Metabolic Biomarker Recovery. Analytical Chemistry 2009;**81**:56-66.
7 Dieterle F, Ross A, Schlotterbeck G, Senn H. Probabilistic Quotient Normalization as Robust Method to Account for Dilution of Complex Biological Mixtures. Application in 1H NMR Metabonomics. Analytical Chemistry 2006;**78**:4281-90.
8 Bylesjö M, Rantalainen M, Cloarec O, Nicholson JK, Holmes E, Trygg J. OPLS discriminant analysis: combining the strengths of PLS-DA and SIMCA classification. Journal of Chemometrics 2006;**20**:341-51.
9 Eriksson L, Trygg J, Wold S. CV-ANOVA for significance testing of PLS and OPLS® models. Journal of Chemometrics 2008;**22**:594-600.
10 R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/. 2017.
11 Kim S. ppcor: An R Package for a Fast Calculation to Semi-partial Correlation Coefficients. Communications for statistical applications and methods 2015;**22**:665-74.
12 Benjamini Y, Hochberg Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society Series B (Methodological) 1995;**57**:289-300.
13 Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D*, et al.* Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. Genome Research 2003;**13**:2498-504.
14 Cloarec O, Dumas M-E, Craig A, Barton RH, Trygg J, Hudson J*, et al.* Statistical Total Correlation Spectroscopy: An Exploratory Approach for Latent Biomarker Identification from Metabolic 1H NMR Data Sets. Analytical Chemistry 2005;**77**:1282-9.
15 Posma JM, Garcia-Perez I, De Iorio M, Lindon JC, Elliott P, Holmes E*, et al.* Subset Optimization by Reference Matching (STORM): An Optimized Statistical Approach for Recovery of

Metabolic Biomarker Structural Information from 1H NMR Spectra of Biofluids. Analytical Chemistry 2012;**84**:10694-701.
16 Nicholson JK, Foxall PJD, Spraul M, Farrant RD, Lindon JC. 750 MHz 1H and 1H-13C NMR Spectroscopy of Human Blood Plasma. Analytical Chemistry 1995;**67**:793-811.
17 Mullish BH, Pechlivanis A, Barker GF, Thursz MR, Marchesi JR, McDonald JAK. Functional microbiomics: Evaluation of gut microbiota-bile acid metabolism interactions in health and disease. Methods (San Diego, Calif) 2018;**149**:49-58.
18 Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. Bioinformatics in Action 2012;**17**:10-2.
19 Callahan B, Sankaran K, Fukuyama J, McMurdie P, Holmes S. Bioconductor Workflow for Microbiome Data Analysis: from raw reads to community analyses [version 2; referees: 3 approved], 2016.
20 Callahan BJ, McMurdie PJ, Rosen MJ, Han AW, Johnson AJA, Holmes SP. DADA2: High-resolution sample inference from Illumina amplicon data. Nat Meth 2016;**13**:581-3.
21 Cole JR, Wang Q, Fish JA, Chai B, McGarrell DM, Sun Y*, et al.* Ribosomal Database Project: data and tools for high throughput rRNA analysis. Nucleic Acids Research 2014;**42**:D633-D42.
22 Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P*, et al.* The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucleic Acids Research 2013;**41**:D590-D6.
23 Shannon CE. A mathematical theory of communication. SIGMOBILE Mob Comput Commun Rev 2001;**5**:3-55.
24 Bray JR, Curtis JT. An Ordination of the Upland Forest Communities of Southern Wisconsin. Ecological Monographs 1957;**27**:325-49.
25 Anderson M. A new method for non-parametric multivariate analysis of variance. Austral Ecology 2001;**26**:32-46.
26 Love MI, Huber W, Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology 2014;**15**:550.
27 Rohart F, Gautier B, Singh A, Lê Cao K-A. mixOmics: An R package for ‘omics feature selection and multiple data integration. PLOS Computational Biology 2017;**13**:e1005752.

R script - report generated with "knitr"

R Markdown file and data to reproduce this analysis can be found on GitHub at https://github.com/ka-west/PBS_manuscript

---

All code used to generate results within the manuscript and the supplementary files is contained here

- For all *P* values, an alpha of 0.05 was used
- The Benjamini-Hochberg method was used to control for false discovery rate

---

Parameters

```r
save_dir <- "output" # where to save figures and tables
if (!dir.exists(save_dir)) { stop(paste("save_dir", save_dir, "does not exist")) }

color_pal <- list(NBS = "purple", RES = "coral2", MAL = "darkturquoise")
```

Load data

```r
load("PBS_data.Rdata")
ls()
```

```
## [1] "color_pal"       "integrals"       "med_specs_serum" "med_specs_urine"
## [5] "pkgs"            "ps_M"            "rdp.species"     "save_dir"
## [9] "silva.species"
```

Figure S1: Summary of sampling by individual

Contribution of samples by study group at each time point

```r
mb <- data.frame(Patient.TP = sample_names(ps_M),
                 faeces = TRUE)

# order individuals by number of time points where samples were collected
ord <- table(integrals$Study.no) %>%
  sort(decreasing = TRUE) %>%
  names()

# which samples were collected for each individual at each time point
all_samples <- integrals %>%
  filter(Time_point != 7) %>%
  mutate(urine = ifelse(!is.na(PAG), TRUE, NA),
         serum = ifelse(!is.na(isoleucine), TRUE, NA)) %>%
  full_join(mb, by = "Patient.TP") %>%
  mutate(Sampling = case_when(urine == TRUE & serum == TRUE & faeces == TRUE ~ "Serum/Urine/Faeces",
                              urine == TRUE & serum == TRUE & is.na(faeces) ~ "Serum/Urine",
                              urine == TRUE & is.na(serum) & faeces == TRUE ~ "Urine/Faeces",
                              is.na(urine) & serum == TRUE & faeces == TRUE ~ "Serum/Faeces",
                              urine == TRUE & is.na(serum) & is.na(faeces) ~ "Urine only",
                              is.na(urine) & serum == TRUE & is.na(faeces) ~ "Serum only"),
         Sampling = factor(Sampling, levels = c("Serum/Urine/Faeces", "Serum/Urine", "Urine/Faeces",
                                                "Serum/Faeces", "Urine only",  "Serum only")),
         Study.no = factor(Study.no, levels = ord))
```

Make plot

```r
p <- ggscatter(all_samples,
              x = "Study.no",
              xlab = "Individual",
              y = "Time_point_label",
              ylab = "Time point",
              color = "Sampling",
               palette = c("chartreuse3", "mediumorchid3", "dodgerblue",
                           "darkorange", "gold2", "red3"),
               size = "Obesity") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.spacing.x = unit(2, "mm"),
        legend.title = element_blank()) +
  grids(linetype = "dashed", axis = "x", color = "grey")

p <- facet(p, facet.by = "Group", scales = "free", ncol = 1)
```

Figures S2 and S5: Overlayed median 1H NMR spectra

Serum (Figure S2) and urine (Figure S5) median spectra were overlayed for the three study groups

```
plot_med_specs <- function(to_plot, save_dir, save_name) {

  p <- plot_ly(to_plot, x = ~ppm, y = ~MAL, name = "MAL", type = "scatter",
               mode = "lines", alpha = 0.8, color = color_pal$MAL) %>%
    add_trace(y = ~RES, name = "RES", mode = "lines", alpha = 0.8, color = color_pal$RES) %>%
    add_trace(y = ~NBS, name = "NBS", mode = "lines", alpha = 0.8, color = color_pal$NBS) %>%
    layout(xaxis = list(autorange = "reversed", size = 18),
           yaxis = list(title = "Integral (a.u.)", size = 18))

  htmlwidgets::saveWidget(p, file = file.path(getwd(), save_dir, save_name))
}

# serum
plot_med_specs(med_specs_serum, save_dir, "figureS2_median_spectra_serum.html")

# urine
plot_med_specs(med_specs_urine, save_dir, "figureS5_median_spectra_urine.html")
```

Figure 1: Longitudinal modelling of metabolite concentrations

Metabolites were considered discriminatory (malabsorptive vs control) if their NMR peaks were correlated (>=0.45) with the predictive component of a valid OPLS-DA model at any time point. A representative peak for each metabolite was integrated to calculate relative concentration.

The behavior of key discriminatory metabolites over time was assessed and compared between control and malabsorptive groups using `santaR` time-series analysis.

Abbreviations: PAG: phenylacetylglutamine, PCS: *p*-cresol sulfate, IS: indoxyl sulfate, PHPA: *p*-hydroxyphenylacetate, unkn: unknown, aKIV: a-ketoisovalerate, MM: methylmalonate, THB: D-B-hydroxybutyrate, NAGP: N-acetyl glycoprotein

```
################## DEFINED VARIABLES ##################

# remove restrictive group
input_data <- subset(integrals, integrals$Group != "Restrictive" & integrals$Time_point != 7)

# column names
metab_names <- c("leucine", "isoleucine", "isobutyrate", "NAGP", "glutamine", "THB")
tp_col <- "Time_point"
patient_col <- "Study.no"
group_col <- "Group"

tp_labs <- c("T1", "T2", "T3", "T4", "T5", "T6")

plot_titles <- c("Serum - Leucine",
                 "Serum - Isoleucine",
                 "Serum - Isobutyrate",
                 "Serum - GlycA",
                 "Serum - Glutamine",
                 "Serum - D-beta-hydroxubutyrate")

# see santaR vignette for choosing this value
degrees_freedom <- 5

# add custom locations (y axis) on plots for p values
pval_locs = c(27, 20, 2.1, 5.0, 8.0, 17)

###########################################################
```

Serum metabolites

```
# run santaR time series analysis
sp <- santaR_auto_fit(inputData = input_data[,metab_names], ind = input_data[[patient_col]],
                      time = as.numeric(input_data[[tp_col]]), group = input_data[[group_col]],
                      df = degrees_freedom)

# generate a plot for each metabolite
for (metab in 1:length(metab_names)) {

  p <- santaR_plot(sp[[metab_names[metab]]],
                   showIndPoint=FALSE,
                   showIndCurve=FALSE,
                   xlab = NULL,
                   ylab = NULL,
                   colorVect = unlist(color_pal[-2]),
                   title = plot_titles[metab]) +
    scale_x_continuous(breaks = 1:length(tp_labs),
                       labels = tp_labs) +
    theme(plot.title = element_text(size = 6),
          legend.position = "None",
          axis.text = element_text(size = 5),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank()) +
    # add p value
    annotate("text",
             size = 2,
             x = 5,
             y = pval_locs[metab],
             label = paste("P = ", round(sp[[metab_names[metab]]]$general$pval.dist, digits = 4)))

  assign(metab_names[metab], p)
}
```

Urine metabolites

```
################## DEFINED VARIABLES ##################

# remove restrictive group
input_data <- subset(integrals, integrals$Group != "Restrictive" & integrals$Time_point != 7)

# column names
metab_names <- c("PAG", "PCS", "IS", "PHPA", "unkn", "valine", "aKIV", "MM", "creatinine")
tp_col <- "Time_point"
patient_col <- "Study.no"
group_col <- "Group"

tp_labs <- c("T1", "T2", "T3", "T4", "T5", "T6")
```

```
plot_titles <- c("Urine - PAG",
                 "Urine - PCS",
                 "Urine - IS",
                 "Urine - PHPA",
                 "Urine - Unknown",
                 "Urine - Valine",
                 "Urine - alpha-Ketoisovalerate",
                 "Urine - Methylmalonate",
                 "Urine - Creatinine")

# see santaR vignette for choosing this value
degrees_freedom <- 5

# add custom locations (y axis) on plots for p values
pval_locs = c(28, 75, 6.5, 6.5, 13, 12, 22, 12, 550)

###########################################################
```

```
# run santaR time series analysis
sp <- santaR_auto_fit(inputData = input_data[,metab_names], ind = input_data[[patient_col]],
                      time = as.numeric(input_data[[tp_col]]), group = input_data[[group_col]],
                      df = degrees_freedom)

# generate a plot for each metabolite
for (metab in 1:length(metab_names)) {

  p <- santaR_plot(sp[[metab_names[metab]]],
                   showIndPoint=FALSE,
                   showIndCurve=FALSE,
                   xlab = NULL,
                   ylab = NULL,
                   colorVect = unlist(color_pal[-2]),
                   title = plot_titles[metab]) +
    scale_x_continuous(breaks = 1:length(tp_labs),
                       labels = tp_labs) +
    theme(plot.title = element_text(size = 6),
          legend.position = "None",
          axis.text = element_text(size = 5),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank()) +
    # add p value
    annotate("text",
             size = 2,
             x = 5,
             y = pval_locs[metab],
             label = paste("P = ", round(sp[[metab_names[metab]]]$general$pval.dist, digits = 4)))

  assign(metab_names[metab], p)
}
```

(1b) Relative concentration of PAG was also higher in neonatal urine from malabsorptive group.

```
pb <- ggboxplot(integrals,
                x = "Group_label",
                y = "PAG_B",
                color = "Group_label",
                ylim = c(-1,11),
                add = "jitter",
                size = 0.25,
                add.params = list(size = 0.5),
                palette = unlist(color_pal),
                legend = "none",
                xlab = FALSE,
                ylab = "Integral (a.u.)",
                title = "Neonatal Urine - PAG") +
  theme(axis.text=element_text(size = 5),
        axis.title = element_text(size = 6),
        plot.title = element_text(size = 8, hjust = 0.5)) +
  # add significance
  stat_compare_means(comparisons = list(c("NBS", "MAL"), c("RES", "MAL")),
                     label.y = c(7, 9.5),
                     label.y.npc = "bottom",
                     label =  "p.signif",
                     method = "wilcox.test")
```

Combine maternal (1a) serum, urine, and neonatal (1b) urine plots

```
p <- ggarrange(leucine, isoleucine, isobutyrate ,NAGP, glutamine, THB,
               PAG, PCS, IS, PHPA, unkn, valine, aKIV, MM, creatinine, pb)

fig1 <- annotate_figure(p,
                     left = text_grob("Integral (a.u.)",
                                      size = 10, rot = 90),
                     bottom = text_grob("Time point",
                                        size = 10))
```

Table S5: Maternal metabolic changes depend on type of bariatric surgery

Relative concentration changes for each discriminatory metabolite were detailed at all time points for both malabsorptive and restrictive groups.

```
################## DEFINED VARIABLES ###################

# input dataframe has one row for each sample, one column for each metabolite (with concentrations or integrals)
input_data <- integrals

# metabolites of interest (match column names in input data)
metab_names <- c("PAG", "PCS", "IS", "PHPA", "unkn", "creatinine", "aKIV", "MM", "valine",
                 "isoleucine", "leucine", "isobutyrate", "glutamine", "THB", "NAGP")

# column name for time point
tp_col <- "Time_point_label"

# time points to include
time_points <- c("T1", "T2", "T3", "T4", "T5", "T6")

# column name for group
group_col <- "Group"

# case group name
case_names <- c("Restrictive", "Malabsorptive") # function only takes one input group, use for loop

# control group name
con_name <- "No Bariatric Surgery"

############################################################
```

Generate table: log2(Fold Change) was calculated where Fold Change (FC) = mean(case)/mean(control). *P* values are from Mann-Whitney U test.

```
univariate_metabolites <- function(input_data,
                                   metab_names,
                                   tp_col,
                                    time_points,
                                    group_col,
                                    case_name,
                                    con_name) {
  # subset data to comparison groups
  input_data <- subset(input_data, input_data[[group_col]] %in% c(case_name, con_name))

  # create empty matrices to store results
  log2FC <- as.data.frame(matrix(nrow = length(time_points), ncol = length(metab_names))) %>%
    set_colnames(metab_names) %>%
    set_rownames(time_points)

  pval <- as.data.frame(matrix(nrow = length(time_points), ncol = length(metab_names))) %>%
    set_colnames(metab_names) %>%
    set_rownames(time_points)

  # Calculate log2(FC) for each metabolite at time points T1 - T6
  for (time in time_points){
    data <- subset(input_data, input_data[[tp_col]] == time)
    con <- subset(data, data[[group_col]] == con_name)
    case <- subset(data, data[[group_col]] == case_name)
    for (metab in metab_names){
      log2fc <- round(log2(mean(case[,metab], na.rm = T)/mean(con[,metab], na.rm = T)), digits = 3)
      log2FC[time,metab] <- log2fc
    }}

  # Calculate p value for each metabolite at time points T1 - T6
  for (time in time_points){
    for (metab in metab_names){
      data <- subset(input_data,
                     input_data[[tp_col]] == time & input_data[[group_col]] %in% c(case_name, con_name))
      form <- formula(paste(metab, "~", group_col))
      pval[time,metab] <- wilcox.test(form, data, exact = FALSE)$p.value
    }}

  # adjust p values
  log2FC.list <- data.frame(Time_point = rep(time_points, length(metab_names)),
                            bind_cols(gather(log2FC, "Metabolite", "log2FC"),
                                      gather(pval, "Metab", "p.val")))
  log2FC.list$p.adj.BH <- p.adjust(log2FC.list$p.val, method = "BH")
  log2FC.list$Group <- case_name
  return(log2FC.list[,-which(colnames(log2FC.list) == "Metab")])
}
```

Generate table for each case group

```
tableS5 <- NULL
for (gr in case_names) {
  idx <- univariate_metabolites(input_data,
                                metab_names,
                                tp_col,
                                time_points,
                                group_col,
                                case_name = gr,
                                con_name)
  tableS5 <- rbind(tableS5, idx)
}
```

Format table

```
# change p values to scientific notation
tableS5_formatted <- tableS5 %>%
  mutate(P = formatC(tableS5$p.val, format = "e", digits = 2),
            Padj = formatC(tableS5$p.adj.BH, format = "e", digits = 2))

# change non-significant p values to NS
tableS5_formatted$P[as.numeric(tableS5_formatted$P) > 0.05] <- "NS"
tableS5_formatted$Padj[as.numeric(tableS5_formatted$Padj) > 0.05] <- "NS"
```

Figure S3: Maternal metabolic changes depend on type of bariatric surgery

Graphical representation of the changes

```
################## DEFINED VARIABLES ##################

input_data <- tableS5_formatted %>%
  # label significant data
  mutate(plot_sig = ifelse(Padj == "NS", NA, "*"),
         Group_label = case_when(Group == "Malabsorptive" ~ "MAL",
                                 Group == "Restrictive" ~ "RES"),
         # Change names for plot titles
         Metabolite = plyr::mapvalues(Metabolite,
                                      from = c("THB", "NAGP", "unkn", "aKIV", "MM", "glutamine",
                                               "creatinine", "isobutyrate", "leucine", "isoleucine",
                                               "valine"),
                                      to = c("D-beta-hydroxybutyrate", "GlycA",
                                             "Unknown", "alpha-Ketoisovalerate", "Methylmalonate",
                                             "Glutamine", "Creatinine", "Isobutyrate", "Leucine",
                                             "Isoleucine", "Valine")),
         # order time points
         Time_point = factor(Time_point, levels = c("T6", "T5", "T4", "T3", "T2", "T1")),
         # add which biofluid metabolites were identified in
         biofluid = ifelse(Metabolite %in% c("PAG", "PCS", "IS", "PHPA", "Unknown", "Valine",
                                             "Creatinine", "alpha-Ketoisovalerate", "Methylmalonate"),
                           "urine",
                           "serum"))

# column names
tp_col <- "Time_point"
metab_col <- "Metabolite"
s_type_col <- "biofluid"
group_col <- "Group_label"
plot_val <- "log2FC"
label_sig <- "plot_sig"

figure_labels <- list(serum = "a", urine = "b")

###########################################################
```

Plot effect sizes and significance

```
for (s_type in names(figure_labels)) {

    # subset to appropriate biofluid type
    to_plot <- subset(input_data, input_data[[s_type_col]] == s_type)

    # plot log2FC for both groups
    p <- ggdotchart_not_ordered(to_plot,
                                x = tp_col,
                                xlab = "",
                                y = plot_val,
                                ylab = "",
                                rotate = TRUE,
                                color = group_col,
                                palette = unlist(color_pal),
                                dot.size = 1.5,
                                label = label_sig,
                                font.label = list(color = "black",
                                                  size = 10,
                                                  vjust = 0.72),
                                add = "segments",
                                add.param = list(color = "lightgray",
                                                 size = 0.4)) +
      ylim(c(min(to_plot$log2FC), max(to_plot$log2FC))) +
      geom_hline(yintercept = 0,
                 linetype = 2,
                 size = 0.4,
                 color = "lightgray") +
      theme(legend.title = element_blank(),
            legend.text=element_text(size = 6),
            legend.key.size = unit(3, "mm"),
            axis.text=element_text(size = 5),
            axis.line = element_line(size = 0.4),
            axis.ticks = element_line(size = 0.4)) +
      rotate_x_text(55)

    # separate plots for each metabolite
    p <- facet(p,
               facet.by = metab_col,
               nrow = 2,
               scales = "free_x",
               panel.labs.background = list(fill = "ivory2",
                                            color = "black"),
               panel.labs.font = list(size = 4,
                                      face ="bold"))

    # add figure labels and axis annotations
    p <- annotate_figure(p,
                         fig.lab.pos = "top.left",
                         fig.lab = figure_labels[[s_type]],
                         fig.lab.size = 10,
                         fig.lab.face = "bold",
                         left = text_grob("Time point",
                                          size = 6,
                                          rot = 90, vjust = 4),
                         bottom = text_grob("log2(Fold Change)",
                                            size = 6, vjust = -4))

    assign(paste0(s_type, "_dotplot"), p)
}
```

Combine serum and urine plots

```
figS3 <- ggarrange(serum_dotplot, urine_dotplot,
                  ncol = 2,
                  widths = c(1, 1.5))
```

Figure S7: Relative concentrations of urinary host-microbial co-metabolites vary depending on bariatric surgery sub-type

Comparison of key discriminatory metabolites between study sub-groups.

```
################## DEFINED VARIABLES ##################

input_data <- integrals %>%
  # separate restrictive group into band and sleeve
  mutate(Group_label = case_when(Group_label == "NBS" ~ "NBS",
                                 Group_label == "MAL" ~ "MAL",
                                 Group_label == "RES" & Bariatric.surgery == "BAND" ~ "BAND",
                                 Group_label == "RES" & Bariatric.surgery == "SLEEVE" ~ "SLEEVE")) %>%
  # order sub-groups
  mutate(Group_label = factor(Group_label, levels = c("NBS", "BAND", "SLEEVE", "MAL"))) %>%
  # remove neonatal samples
  filter(Time_point_label != "T7")

# column names
metab_names <- c("PAG", "PCS", "IS", "PHPA", "unkn")
tp_col <- "Time_point_label"
group_col <- "Group_label"

colors <- list(NBS = "purple", BAND = "coral2", SLEEVE = "coral2", MAL = "darkturquoise")

###########################################################
```

For each metabolite, plot relative concentrations at each time point.

```
for (metab in metab_names) {
  for (tp in unique(input_data[[tp_col]])) {
    p <- ggboxplot(subset(input_data, input_data[[tp_col]] == tp),
                   size = 0.2,
                   group_col,
                   metab,
                   color = group_col,
                   palette = unlist(colors),
                   add = "jitter",
                   add.params = list(alpha = 0.5,
                                     size = 0.3),
                   xlab = FALSE,
                   ylab = paste(metab, tp)) +
      theme(legend.position = "None",
            axis.text=element_text(size = 5),
            axis.title = element_text(size = 6),
            axis.line = element_line(size = 0.4),
            axis.ticks = element_line(size = 0.4)) +
      rotate_x_text(angle = 45)
    p$layers[[1]]$geom_params$outlier.size <- 0.3 # change size of outlier points
    assign(paste0(metab, "_", tp, "_boxplot"), p)
  }
}
```

Combine all plots together

```
figS7 <- ggarrange(PAG_T1_boxplot, PAG_T2_boxplot, PAG_T3_boxplot, PAG_T4_boxplot,
                   PAG_T5_boxplot, PAG_T6_boxplot, PCS_T1_boxplot, PCS_T2_boxplot,
                   PCS_T3_boxplot, PCS_T4_boxplot, PCS_T5_boxplot, PCS_T6_boxplot,
                   IS_T1_boxplot, IS_T2_boxplot, IS_T3_boxplot, IS_T4_boxplot,
                   IS_T5_boxplot, IS_T6_boxplot, PHPA_T1_boxplot, PHPA_T2_boxplot,
                   PHPA_T3_boxplot, PHPA_T4_boxplot, PHPA_T5_boxplot, PHPA_T6_boxplot,
                   unkn_T1_boxplot, unkn_T2_boxplot, unkn_T3_boxplot, unkn_T4_boxplot,
                   unkn_T5_boxplot, unkn_T6_boxplot)
```

Figure 2: Gut bacterial alterations in malabsorptive patients

```
################## DEFINED VARIABLES ##################

# input data is phyloseq object
input_data <- ps_M

# column names
tp_col <- "Time_point_label"
group_col <- "Group_label"

adiv_measure <- "Shannon"
bdiv_measure <- "bray"
ordination_meth <- "MDS"

set.seed(71)

###########################################################
```

Calculate alpha-diversity

```
# normalize by rarefying to minimum sequencing depth
data <- input_data %>%
  rarefy_even_depth(sample.size = min(sample_sums(input_data)), verbose = FALSE, replace = FALSE)

# calculate alpha diversity
alpha_div <- estimate_richness(data, split = TRUE, measures = adiv_measure)

# add variables needed for plot
alpha_div$Time_point <- sample_data(data)[[tp_col]]
alpha_div$Group <- sample_data(data)[[group_col]]
```

Calculate beta-diversity

```
# use log transformation normalization to overcome differences in sequencing depth
data <- input_data %>%
  transform_sample_counts(function(x) {log(x+1)})

bdiv <- phyloseq::distance(data, bdiv_measure)
MDS <- ordinate(data, ordination_meth, bdiv)

# extract % variance explained for axis labels
ord <- plot_ordination(data, MDS)[["labels"]]
```

(2a) Comparison of Shannon Index between malabsorptive and control groups

```
am <- facet(ggboxplot(alpha_div,
               x = "Group",
               y = adiv_measure,
               color = "Group",
               add = "jitter",
               add.params = list(alpha = 0.5),
               size = 0.5,
               palette = unlist(color_pal),
               legend = "none",
               ylim = c(3.3,5.2),
               xlab = FALSE,
               ylab = "Shannon's Diversity Index (H)") +
              theme(axis.text = element_text(size = 6),
                    axis.title = element_text(size = 6)) +
              # add significance
              stat_compare_means(comparisons = list(c("NBS", "MAL")),
                                   label =  "p.signif",
                                   method = "wilcox.test",
                                   label.y = 5.1,
                                   hide.ns = TRUE),
            facet.by = "Time_point",
            panel.labs.background = list(fill = "ivory2", color = "black"),
            panel.labs.font = list(size = 7, face ="bold"))
```

(2b) Bray-Curtis dissimilarity with PCoA ordination

```
# plot PCoA
to_plot <- plot_ordination(data, MDS, justDF = TRUE)
bm <- ggscatter(to_plot,
                x = "Axis.1",
                y = "Axis.2",
                color = "Group_label",
                alpha = 0.6,
                ggtheme = theme_bw(),
                shape = "Time_point_label",
                size = 1.5,
                ellipse = TRUE,
                ellipse.level = 0.95,
                ellipse.type = "norm",
                ellipse.alpha = 0) +
  scale_colour_manual(values = unlist(color_pal)) +
  labs(x = ord$x,
       y = ord$y) +
  theme(legend.position = "none",
        axis.title =element_text(size=6),
        axis.text = element_text(size = 6))

# permanova test
adonis_res <- adonis(bdiv ~ Group, data.frame(sample_data(data)))

# add permanova p value
bm <- bm +
  annotate("text",
           size = 2,
           x = 0.3,
           y = 0.32,
           label = paste("P = ", round(adonis_res$aov.tab$`Pr(>F)`[1], digits = 4)))

# put 2a and 2b together
plot1 <- ggarrange(am, bm, labels = c("a", "b"), widths = c(2,1))
```

(2c) Differential abundance (malabsorptive vs control) at each taxonomic level. Effect sizes are log2(FC) and *P* values are from Mann-Whitney U test.

```
################## DEFINED VARIABLES ###################

# input data is phyloseq object
data <- ps_M

tp_col <- "Time_point_label"
time_points <- c("T1", "T2", "T4")

group_col <- "Group_label"
case_name <- "MAL"
con_name <- "NBS"

# 16S data transformation parameters
# taxa without at least min_counts in at least prev_filt (%) of samples will be removed
min_counts <- 5
prev_filt <- 0.1

###########################################################
```

Generate differential abundance table (Table S6)

```
MWU <- NULL

for (rank in c("Phylum", "Class", "Order", "Family", "Genus")){
  for (time in time_points){

    ps_obj <- data %>%
      tax_glom(rank) %>%
      # prevalence filter
      filter_taxa(function(x) sum(x > min_counts) > (prev_filt*length(x)), TRUE) %>%
      # normalize by total sum scaling
      transform_sample_counts(function(x) x/sum(x)) %>%
      # subset to time point
      subset_samples(Time_point_label == time)

    tax <- data.frame(as(tax_table(ps_obj), "matrix"))
    abund <- data.frame(as(otu_table(ps_obj), "matrix")) %>%
      # change column names to names at current rank
      set_colnames(gsub("/", ".", tax[,which(colnames(tax)==rank)]))
    # add group column
    abund$Group <- sample_data(ps_obj)[[group_col]]

    # test differences between groups for every taxon
    for (i in c(1:(ncol(abund)-1))) {
      taxa <- colnames(abund)[1:(ncol(abund)-1)]
      form <- formula(paste(taxa[i], "~ Group"))
      mw <- wilcox.test(form, abund, exact = FALSE)
      idx <- tibble(Rank = rank,
                    Time_point = time,
                    Taxon = taxa[i],
                    Mean_rel_abund_case = mean(subset(abund, abund$Group == case_name)[,i]),
                    Mean_rel_abund_cont = mean(subset(abund, abund$Group == con_name)[,i]),
                    pval = mw$p.value,
                    log2FC = log2(mean(subset(abund, abund$Group == case_name)[,i])/
                                mean(subset(abund, abund$Group == con_name)[,i]))
      )
      MWU <- rbind(MWU, idx)
    }
  }
}

# multiple hypothesis testing correction
MWU$p.adj.BH <- p.adjust(MWU$pval, method = "BH")

# filter to significant taxa; excluding taxa that are not observed in a group (zero counts)
MWU_filt <- subset(MWU, MWU$p.adj.BH < 0.05 & abs(MWU$log2FC) != Inf)
```

Barplots for each rank

```
# color related taxa
palette <- list(Proteobacteria = "cyan4",
                Bacilli = "khaki4",
                Actinomycetales = "chocolate3",
                Anaerostipes = "thistle4",
                NS = "grey")

diff_abund <- MWU %>%
  # add x-axis plot label
  mutate(Label = str_replace(paste0(Taxon, "_", Time_point),
                             "Escherichia\\.", "Escherichia/"),
         # group taxa for coloring
         Taxa_group = case_when(p.adj.BH > 0.05 ~ "NS",
                                grepl("Proteobact|Gammaprot|Enterobacteria|Escherichia|Pasteurell|Haemophil",
                                      Taxon) ~ "Proteobacteria",
                                Taxon %in% c("Micrococcaceae", "Rothia", "Actinomycetales") ~ "Actinomycetales",
                                grepl("Bacill|Enteroc|Strepto|Lactobacill|Carnobact|Granulicat",
                                      Taxon) ~ "Bacilli",
                                grepl("Bacteroid|Butyricimonas", Taxon) ~ "Bacteroidetes",
                                Taxon == "Anaerostipes" ~ "Anaerostipes",
                                grepl("Holdeman|Solobact", Taxon) ~ "Erysipelotrichaceae",
                                grepl("Veillonella|Acidaminococcus", Taxon) ~ "Negativicutes",
                                grepl("Clostrid", Taxon) ~ "Clostridia"))

# factor for correct ordering on plot
diff_abund$Rank <- factor(diff_abund$Rank, levels = c("Phylum", "Class", "Order", "Family", "Genus"))
diff_abund$Taxa_group <- factor(diff_abund$Taxa_group, levels = c("Proteobacteria",
                                                                  "Bacilli",
                                                                  "Bacteroidetes",
                                                                  "Clostridia",
                                                                  "Negativicutes",
                                                                  "Actinomycetales",
                                                                  "Erysipelotrichaceae",
                                                                  "Anaerostipes",
                                                                  "NS"))

# remove taxa only significant at one time point
MWU_plot <- subset(diff_abund, diff_abund$Taxon %in%
                              names(which(table(MWU_filt$Taxon) > 1)))

# top barplots
p <- ggbarplot(subset(MWU_plot, MWU_plot$Rank %in% c("Phylum", "Class", "Order")),
              x = "Label",
              y = "log2FC",
              lab.size = 4,
              fill = "Taxa_group",
              color = "white",
              palette = unlist(palette),
              ylab = "log2(Fold Change)",
              lab.vjust = .5,
              xlab = FALSE,
              order = sort(MWU_plot$Label)) +
  theme(axis.title = element_text(size=6),
        axis.text = element_text(size = 6),
        legend.position = "none") +
  rotate_x_text(55)

p <- facet(p,
           facet.by = "Rank",
           scales = "free_x",
           nrow = 1,
           space = "free_x",
           panel.labs.background = list(fill = "ivory2", color = "black"),
           panel.labs.font = list(size = 8, face ="bold"))
```

```
# bottom barplots
q <- ggbarplot(subset(MWU_plot, MWU_plot$Rank %in% c("Family", "Genus")),
              x = "Label",
              y = "log2FC",
              lab.size = 4,
              fill = "Taxa_group",
              color = "white",
              palette = unlist(palette),
              ylab = "log2(Fold Change)",
              lab.vjust = .5,
              xlab = FALSE,
              order = sort(MWU_plot$Label)) +
  theme(axis.title = element_text(size=6),
        axis.text = element_text(size = 6),
        legend.position = "right",
        legend.text = element_text(size = 6),
        legend.title = element_text(size = 8),
        legend.key.size = unit(4, "mm")) +
  rotate_x_text(55) +
  geom_hline(yintercept = 0, linetype="dashed", size = 0.4)

q <- facet(q,
           facet.by = "Rank",
           scales = "free_x",
           nrow = 1,
           space = "free_x",
           panel.labs.background = list(fill = "ivory2", color = "black"),
           panel.labs.font = list(size = 7, face ="bold"))

# combine barplots
plot2 <- ggarrange(p, q, nrow = 2)
```

Combine (2c) with (2a) and (2b)

```
fig2 <- ggarrange(plot1, plot2,
                  nrow = 2,
                  heights = c(1.2,2),
                  labels = c("", "c"))
```

Table S7: Differentially abundant amplicon sequence variants

Differential abundance of amplicon sequence variants was assessed by DESeq2. Taxonomic assignments at species level were compared between SILVA and RDP databases.

```
################## DEFINED VARIABLES ##################

input_data <- merge_samples(ps_M, "Study.no")

# fix group labels - merging samples changes labels to numbers
sample_data(input_data)$Group <- str_replace_all(sample_data(input_data)$Group,
                                              c("1" = "Control",
                                              "2" = "Malabsorptive"))
sample_data(input_data)$Group <- factor(sample_data(input_data)$Group, levels = c("Control", "Malabsorptive"))

# 16S data transformation parameters
# taxa without at least min_counts in at least prev_filt (%) of samples will be removed
min_counts <- 5
prev_filt <- 0.1

alpha <- 0.05
```

```
gen <- c("Enterococcus", "Escherichia/Shigella", "Streptococcus", "Rothia")

############################################################
```

Make table with differential abundance and species assignments

```
# prevalence filter
ps_trim <- input_data %>%
  filter_taxa(function(x) sum(x > min_counts)> (prev_filt*length(x)), TRUE)

# run DESeq2
DDS <- phyloseq_to_deseq2(ps_trim, ~ Group)
DDS <- estimateSizeFactors(DDS, type = "poscounts")
DDS <- estimateDispersions(DDS, fitType = "local")
DDS <- DESeq(DDS, fitType = "local")

# results
res <- results(DDS)
res <- res[order(res$padj, na.last=NA), ]

sigtab <- res[(res$padj < alpha), ]
sigtab <- cbind(as(sigtab, "data.frame"), as(tax_table(ps_trim)[rownames(sigtab), ], "matrix"))

# keep only genera of interest
sigtab <- subset(sigtab, sigtab$Genus %in% gen)
keep <- which(row.names(rdp.species) %in% row.names(sigtab))

rdp.species <- data.frame(rdp.species)
rdp.species$seq <- row.names(rdp.species)
silva.species <- data.frame(silva.species)
silva.species$seq <- row.names(silva.species)

# combine RDP and SILVA species assignments
species_assign <- merge(rdp.species[keep, c(6:8)], silva.species[keep, c(6:8)], by = "seq")

# combine DESeq2 results with species assignments
sigtab$seq <- row.names(sigtab)
species_assign <- merge(sigtab[,c(1:6,13)], species_assign, by = "seq")
```

Figure 3: Integration of serum, urine and faecal profiles

Relative concentrations of key discriminatory metabolites were integrated with genera abundances using "DIABLO", a method implemented in `mixOmics`.

```
################## DEFINED VARIABLES ##################

# input data is phyloseq object
input_MB <- ps_M %>%
  tax_glom("Genus") %>%
  transform_sample_counts(function(x) {log(x+1)})

# remove neonatal samples & subset to individuals with microbiome data
input_metab <- integrals %>%
  filter(Time_point != 7)
input_metab <- left_join(data.frame(Patient.TP = sample_names(input_MB)),
                         input_metab,
                         by = "Patient.TP")

S_metab_names <- c("leucine", "isoleucine", "isobutyrate", "THB")
U_metab_names <- c("PAG", "PCS", "IS", "PHPA", "unkn", "aKIV", "creatinine")
```

```
# number of genera to keep in model
n_gen <- 10

###########################################################
```

Prepare datasets

```
tax <- as(tax_table(input_MB), "matrix") %>%
  data.frame(stringsAsFactors = F)

F_abund_mat <- as(otu_table(input_MB), "matrix") %>%
  data.frame() %>%
  set_colnames(tax$Genus)

S_NMR_mat <- input_metab %>%
  select(one_of(S_metab_names)) %>%
  set_rownames(input_metab$Patient.TP)

U_NMR_mat <- input_metab %>%
  select(one_of(U_metab_names)) %>%
  set_rownames(input_metab$Patient.TP)
```

Build model

```
X <- list(Urine = U_NMR_mat,
          Serum = S_NMR_mat,
          Faeces = F_abund_mat)
Y <- input_metab[, "Group_label"]

# only model top n taxa for each component
list.keepX <- list(Urine = ncol(U_NMR_mat), Serum = ncol(S_NMR_mat), Faeces = c(n_gen, n_gen))

# multi-block pls-da
m <- block.splsda(X, Y,
                  keepX = list.keepX,
                  near.zero.var = TRUE)
```

Plot sample scores for each dataset. Separation between classes is on Component 1.

```
ggsave(plotIndiv(m,
                 legend = TRUE,
                 X.label = "Component 1",
                 Y.label = "Component 2",
                 legend.title = '',
                 style = "lattice",
                 ind.names = FALSE,
                 col.per.group = unlist(color_pal)[-2],
                 ellipse = TRUE,
                 legend.position = "top",
                 pch = 20, alpha =  0.7,
                 size.axis = 0.5, size.xlabel = 0.9, size.ylabel = 0.9,
                 size.legend = 0.8) +
         theme(legend.box.background = element_rect()),
       filename = file.path(save_dir, "figure3a_integration_scores.pdf"),
       height = 3, width = 6)
```

Plot correlations between variables and components highlighting variables with high correlation on either component.

```
var_plot <- plotVar(m, plot = F) %>%
  mutate(Plot_label = plyr::mapvalues(names,
                                      from = c("THB", "unkn", "aKIV", "creatinine",
                                               "isobutyrate", "leucine", "isoleucine"),
                                      to = c("D-beta-HB", "Unknown",
                                             "alpha-Ketoisovalerate", "Creatinine",
                                             "Isobutyrate", "Leucine", "Isoleucine")),
         Plot_label = case_when(abs(x) > 0.5 | abs(y) >= 0.5 ~ Plot_label),
         Plot_alpha = ifelse(is.na(Plot_label), "|r|<0.5", "|r|>=0.5"))

ggsave(ggscatter(var_plot, x = "x", y = "y",
                 color = "Block", palette = c("darkolivegreen", "darkgoldenrod3", "rosybrown3"),
                 label = "Plot_label", repel = TRUE,
                 font.label = c(5, "plain"),
                 label.rectangle = FALSE,
                 alpha = "Plot_alpha",
                 xlab = "Component 1", ylab = "Component 2",
                 show.legend.text = FALSE) +
         xlim(c(-1, 1)) +
         ylim(c(-1,1)) +
         geom_hline(yintercept = 0,
                    linetype = 2,
                    size = 0.4,
                    color = "lightgray") +
         geom_vline(xintercept = 0,
                    linetype = 2,
                    size = 0.4,
                    color = "lightgray") +
         theme(legend.title = element_blank(),
               axis.text = element_text(size = 5),
               axis.title = element_text(size = 8),
               legend.text = element_text(size = 5)),
       filename = file.path(save_dir, "figure3b_integration_variables.pdf"),
       height = 3.5, width = 3.5)
```

Plot correlations between datasets for Component 1.

```
ggsave(circosPlot(m,
                  cutoff=0.5,
                  comp = 1,
                  color.cor = c("red", "blue"),
                  color.blocks = c("rosybrown3", "darkgoldenrod3", "darkolivegreen"),
                  showIntraLinks = FALSE,
                  legend = TRUE,
                  line = FALSE,
                  size.labels = 0.01,
                  size.variables = 0.5,
                  var.names = list(Urine = c("PAG","PCS","IS","PHPA","Unknown","alpha-KIV","Creatinine"),
                                   Serum = c("Leucine", "Isoleucine", "Isobutyrate", "D-beta-HB"),
                                   Faeces = m$names$colnames$Faeces),
                  size.legend = 0.5) +
         theme(legend.box.background = element_rect()),
       filename = file.path(save_dir, "figure3c_integration_circos.pdf"),
       height = 4, width = 4.5)
```

Figure 4: Relative concentrations of metabolites correlate with maternal insulin resistance and fetal/birth weight

Correlations (Spearman) were calculated between metabolite relative concentrations and clinical measures. Netowrk file was generated for Cytoscape input.

```
################## DEFINED VARIABLES ##################

time_points <- 1:6
input_data <- subset(integrals, integrals$Time_point %in% time_points)
```

```
tp_col <- "Time_point_label"

# column names
metab_names <- c("PAG", "PCS", "IS", "isoleucine", "leucine", "isobutyrate")
clin_varT1 <- c("Fast Maternal Insulin (microU/ml)",
                "Maternal HOMA-IR",
                "Fast Maternal Gluc (mmol/L)",
                "GA..days.",
                "BMI.1",
                "Age")
clin_varT2 <- c("Fast Maternal Insulin (microU/ml)",
                "Maternal HOMA-IR",
                "Fast Maternal Gluc (mmol/L)",
                "GA..days.",
                "BMI2",
                "EFW2-centile",
                "Age")
clin_varT3 <- c("Fast Maternal Insulin (microU/ml)",
                "Maternal HOMA-IR",
                "Fast Maternal Gluc (mmol/L)",
                "GA..days.",
                "BMI3",
                "Age")
clin_varT4 <- c("Fast Maternal Insulin (microU/ml)",
                "Maternal HOMA-IR",
                "Fast Maternal Gluc (mmol/L)",
                "GA..days.",
                "BMI4",
                "EFW4-centile",
                "Age")
clin_varT5 <- c("Fast Maternal Insulin (microU/ml)",
                "Maternal HOMA-IR",
                "Fast Maternal Gluc (mmol/L)",
                "GA..days.",
                "BMI5",
                "EFW5-centile",
                "Age")
clin_varT6 <- c("Fast Maternal Insulin (microU/ml)",
                "Maternal HOMA-IR",
                "Fast Maternal Gluc (mmol/L)",
                "GA..days.",
                "BMI6",
                "BW.centile",
                "Age")

############################################################
```

Generate network file

```
# use T5 BMI for T6 (visits are couple weeks apart, BMI not measured at T6)
input_data$BMI6 <- input_data$BMI5

# format data for correlation at each time point
for (time in unique(input_data[[tp_col]])) {
  idx <- subset(input_data, input_data[[tp_col]] == time)
  # keep only columns to correlate
  idx <- idx[,which(colnames(idx) %in% c(metab_names, get(grep(paste0("clin_var", time), ls(), value = T))))]
  # fix column names so they all match
  colnames(idx) <- str_replace_all(colnames(idx), c("BMI.*" = "BMI", ".*centile" = "EFW.BW"))
  if (!"EFW.BW" %in% colnames(idx)) { idx$EFW.BW <- NA }
  # tidy column names
  colnames(idx) <- make.names(colnames(idx))
  # arrange columns in the same order
  idx <- idx[,c(metab_names,
              "Age",
              "BMI",
              "GA..days.",
              "Fast.Maternal.Insulin..microU.ml.",
              "Fast.Maternal.Gluc..mmol.L.",
              "Maternal.HOMA.IR",
              "EFW.BW")]
  # make sure all columns are numeric
  idx <- sapply(idx, as.numeric)
  assign(paste(time), idx)
}

# remove confounders
for (time in unique(input_data[[tp_col]])) {
  idx <- get(paste(time))
  assign(paste0(time, "_to_corr"), idx[,-c(which(colnames(idx) %in% c("Age", "BMI")))])
}

# correlation at each time point
corr_all <- NULL
corr_pval <- NULL
for (cor in time_points) {
  idx <- get(grep(paste0(cor,"_to_corr"), ls(), value = T))
  rcorr <- rcorr(idx, type = "spearman")
  coef <- round(rcorr$r,2)
  coef <- coef[c(which(rownames(coef) %in% metab_names)), c(which(!colnames(coef) %in% metab_names))]
  row.names(coef) <- paste0(metab_names, "_T", cor)
  pval <- rcorr$P
  pval <- pval[c(which(rownames(pval) %in% metab_names)), c(which(!colnames(pval) %in% metab_names))]
  row.names(pval) <- paste0(metab_names, "_T", cor)
  corr_all <- rbind(corr_all, coef)
  corr_pval <- rbind(corr_pval, pval)
}

# change to long format and adjust p values
corr_all <- melt(corr_all, value.name = "spearman")
corr_all <- subset(corr_all, !is.na(corr_all$spearman))
corr_pval <- melt(corr_pval, value.name = "p.val")
corr_pval <- subset(corr_pval, !is.na(corr_pval$p.val))

corr_pval$p.adj.BH <- p.adjust(corr_pval$p.val, method = "BH")
corr_metaVmetab <- cbind(corr_all, corr_pval[,c(3:4)])
corr_metaVmetab_sig <- subset(corr_metaVmetab, corr_metaVmetab$p.adj.BH < 0.05)

# convert to graph for cytoscape visualisation
cytoscape <- corr_metaVmetab_sig[,c(1:3)] %>%
  mutate(Strength = case_when(abs(spearman) < 0.4 ~ "0.3-0.4",
                              abs(spearman) >= 0.4 ~ "0.4-0.5"),
         Direction = case_when(spearman > 0 ~ "pos",
                               spearman < 0 ~ "neg")) %>%
  rename(from = Var1, to = Var2, weight = spearman)

net <- graph.data.frame(cytoscape, directed = FALSE)
write_graph(net, file.path(save_dir, "cormat_cytoscape.gml"), format = "gml")
```

Table S8: Relative concentrations of metabolites correlate with maternal insulin resistance and fetal/birth weight

Includes partial correlations to adjust for the effect of maternal age and BMI as well as any effect of maternal insulin resistance on the baby's weight.

```
corr_metaVmetab_sig$partial <- NA
corr_metaVmetab_sig$part.pval <- NA

for (part_cor in c(1:nrow(corr_metaVmetab_sig))) {
  var1 <- gsub("_T.*", "", corr_metaVmetab_sig[part_cor,1])
  var2 <- corr_metaVmetab_sig[part_cor,2]
  idx <- get(paste0(gsub(".*_", "", corr_metaVmetab_sig[part_cor,1])))

  if (var2 == "EFW.BW") {
    conf <- which(colnames(idx) %in% c("Age", "BMI", "Maternal.HOMA.IR"))
    # remove any rows with missing data
    idx <- subset(idx,
                  !is.na(idx[,which(colnames(idx)==var1)]) &
                  !is.na(idx[,which(colnames(idx)==var2)]) &
                  !is.na(idx[,which(colnames(idx)=="Maternal.HOMA.IR")]) &
                  !is.na(idx[,which(colnames(idx)=="BMI")]) &
                  !is.na(idx[,which(colnames(idx)=="Age")]))
  } else {
    conf <- which(colnames(idx) %in% c("Age", "BMI"))
    # remove any rows with missing data
    idx <- subset(idx,
                  !is.na(idx[,which(colnames(idx)==var1)]) &
                  !is.na(idx[,which(colnames(idx)==var2)]) &
                  !is.na(idx[,which(colnames(idx)=="BMI")]) &
                  !is.na(idx[,which(colnames(idx)=="Age")]))
  }

  # partial correlations
  part_corr <- pcor.test(idx[,which(colnames(idx)==var1)],
                         idx[,which(colnames(idx)==var2)],
                         idx[,conf],
                         method = "spearman")
  corr_metaVmetab_sig$partial[part_cor] <- part_corr$estimate
  corr_metaVmetab_sig$part.pval[part_cor] <- part_corr$p.value

  corr_metaVmetab_sig$part.pval.adj <- p.adjust(corr_metaVmetab_sig$part.pval, method = "BH")
}

for (col in c("p.val","p.adj.BH","part.pval","part.pval.adj")) {
  corr_metaVmetab_sig[as.numeric(corr_metaVmetab_sig[[col]]) > 0.05, col] <- "NS"
}
```

Figure S8: Key metabolites are not associated with percentage of weight lost or length of time since surgery in malabsorptive patients

```
to_cor <- integrals %>%
  filter(Group_label == "MAL",
         Time_point != 7)

for (variab in c("perc_wl", "Months_betw_op_concep")) {
  for (metab in c("PAG", "PCS", "IS")) {
    p <- ggscatter(to_cor,
                   x = variab,
                   xlab = str_replace_all(variab,
                                          c("perc_wl" = "Post-surgery weight loss (%)",
                                            "Months_betw_op_concep" =
                                              "Time between operation and conception (months)")),
                   y = metab,
                   ylab = paste(metab, "(a.u.)"),
                   facet.by = "Time_point_label",
                   alpha = 0.7,
                   size = 1,
                   color = "darkturquoise",
                   cor.coef = TRUE,
                   cor.coeff.args = list(method = "spearman"),
                   cor.coef.size = 1.5) +
      theme(axis.text = element_text(size = 6),
            axis.title = element_text(size = 7))
    assign(paste0(metab, "_", variab), p)
  }
}

p <- ggarrange(PAG_perc_wl, PCS_perc_wl, IS_perc_wl,
               PAG_Months_betw_op_concep, PCS_Months_betw_op_concep, IS_Months_betw_op_concep)
```

Session Info

```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.15.1
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] plotly_4.8.0                reshape2_1.4.3
##  [3] ppcor_1.1                   igraph_1.2.4
##  [5] Hmisc_4.2-0                 Formula_1.2-3
##  [7] survival_2.44-1.1           mixOmics_6.6.2
##  [9] MASS_7.3-51.4               DESeq2_1.22.2
## [11] SummarizedExperiment_1.12.0 DelayedArray_0.8.0
## [13] BiocParallel_1.16.6         matrixStats_0.54.0
## [15] Biobase_2.42.0              GenomicRanges_1.34.0
## [17] GenomeInfoDb_1.18.2         IRanges_2.16.0
## [19] S4Vectors_0.20.1            BiocGenerics_0.28.0
## [21] vegan_2.5-4                 lattice_0.20-38
## [23] permute_0.9-5               ggpubr_0.2
## [25] magrittr_1.5                phyloseq_1.26.1
## [27] santaR_1.0                  forcats_0.4.0
## [29] stringr_1.4.0               dplyr_0.8.0.1
## [31] purrr_0.3.2                 readr_1.3.1
## [33] tidyr_0.8.3                 tibble_2.1.1
## [35] ggplot2_3.2.1               tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] colorspace_1.4-1       htmlTable_1.13.1       corpcor_1.6.9
##  [4] XVector_0.22.0         base64enc_0.1-3        rstudioapi_0.10
##  [7] bit64_0.9-7            RSpectra_0.14-0        AnnotationDbi_1.44.0
## [10] lubridate_1.7.4        xml2_1.2.0             codetools_0.2-16
## [13] splines_3.5.0          doParallel_1.0.14      geneplotter_1.60.0
## [16] knitr_1.22             shinythemes_1.1.2      ade4_1.7-13
## [19] jsonlite_1.6           annotate_1.60.1        broom_0.5.2
## [22] cluster_2.0.8          shiny_1.3.0            compiler_3.5.0
## [25] httr_1.4.0             backports_1.1.3        assertthat_0.2.1
## [28] Matrix_1.2-17          lazyeval_0.2.2         cli_1.1.0
## [31] later_0.8.0            acepack_1.4.1          htmltools_0.3.6
## [34] tools_3.5.0            gtable_0.3.0           glue_1.3.1
## [37] GenomeInfoDbData_1.2.0 Rcpp_1.0.1             cellranger_1.1.0
## [40] Biostrings_2.50.2      multtest_2.38.0        ape_5.3
## [43] nlme_3.1-138           iterators_1.0.10       xfun_0.6
## [46] rvest_0.3.2            mime_0.6               XML_3.98-1.19
## [49] zlibbioc_1.28.0        scales_1.0.0           hms_0.4.2
## [52] promises_1.0.1         biomformat_1.10.1      rhdf5_2.26.2
## [55] RColorBrewer_1.1-2     yaml_2.2.0             memoise_1.1.0
## [58] gridExtra_2.3          rpart_4.1-13           RSQLite_2.1.1
## [61] latticeExtra_0.6-28    stringi_1.4.3          genefilter_1.64.0
## [64] foreach_1.4.4          checkmate_1.9.1        rlang_0.3.4
## [67] pkgconfig_2.0.2        bitops_1.0-6           evaluate_0.13
## [70] Rhdf5lib_1.4.3         htmlwidgets_1.3        bit_1.1-14
## [73] tidyselect_0.2.5       plyr_1.8.4             R6_2.4.0
## [76] generics_0.0.2         DBI_1.0.0              pillar_1.3.1
## [79] haven_2.1.0            foreign_0.8-71         withr_2.1.2
## [82] mgcv_1.8-28            RCurl_1.95-4.12        nnet_7.3-12
## [85] modelr_0.1.4           crayon_1.3.4           rARPACK_0.11-0
## [88] ellipse_0.4.1          rmarkdown_1.12         locfit_1.5-9.1
## [91] grid_3.5.0             readxl_1.3.1           data.table_1.12.2
## [94] blob_1.1.1             digest_0.6.18          xtable_1.8-3
## [97] httpuv_1.5.1           munsell_0.5.0          viridisLite_0.3.0
```