Plotting Heatmaps

Mir Henglin

October 18, 2019

To construct out plots in R, we will be using the `ggplot2` package. To perform data manipulation, we will be using the `dplyr` package.

```
library(dplyr)
library(ggplot2)
```

Data Format

The plots we will make will summarize the results of multiple models at the same time. In order to plot those results in `ggplot2`, they must be properly formatted. Specifically, the data must be organized in a `data.frame` with columns indicating:

- The model that the results a row correspond to
- The dependent variable term
- The P-value
- The effect size estimate

Other columns can be included in the data, but for the purpose of this tutorial we will focus on the four necessary columns.

Tutorial Dataset

The dataset we will be using contains regression coefficents and P-values for models evaluating the relationship between the measured levels of metabolites found in the blood and biological measures of interest. A small portion of that dataset is shown below.

```
plot_data
```

```
## # A tibble: 168 x 4
##    response              term          estimate  p.value
##    <chr>                 <chr>            <dbl>    <dbl>
##  1 Body Mass Index       Metabolite 17      0.4 5.74e-52
##  2 Framingham Risk Score Metabolite 17      0.4 4.80e-46
##  3 Age                   Metabolite 09      0.3 1.87e-30
##  4 Female Sex            Metabolite 07      0.6 2.47e-25
##  5 Metabolic Syndrome    Metabolite 17      0.7 1.78e-23
##  6 Female Sex            Metabolite 17     -0.5 3.92e-18
##  7 Age                   Metabolite 07      0.2 2.96e-17
##  8 Framingham Risk Score Metabolite 09      0.2 8.91e-17
##  9 Age                   Metabolite 04      0.2 1.02e-15
## 10 Body Mass Index       Metabolite 14     -0.2 1.47e-13
## # ... with 158 more rows
```

In this dataset, `estimate` and `p.value` indicate the effect size estimate and the P-value of that estimate, `term` indicates the ID of the metabolite, and`response` indicates the biological measure of interest. For example, if `response` = Body Mass Index and `term` = mzid_396.271758_6.3118, that row in the dataset corresponds to the model;

$$BodyMassIndex = \beta_0 + \beta_1 * mzid\_396.271758\_6.3118 + X\beta + \epsilon$$

Where $X\beta$ corresponds to control variables included in the model and ϵ is the error term. `estimate` is the estimated value of β_1 and `p.value` is the corresponding P-value.

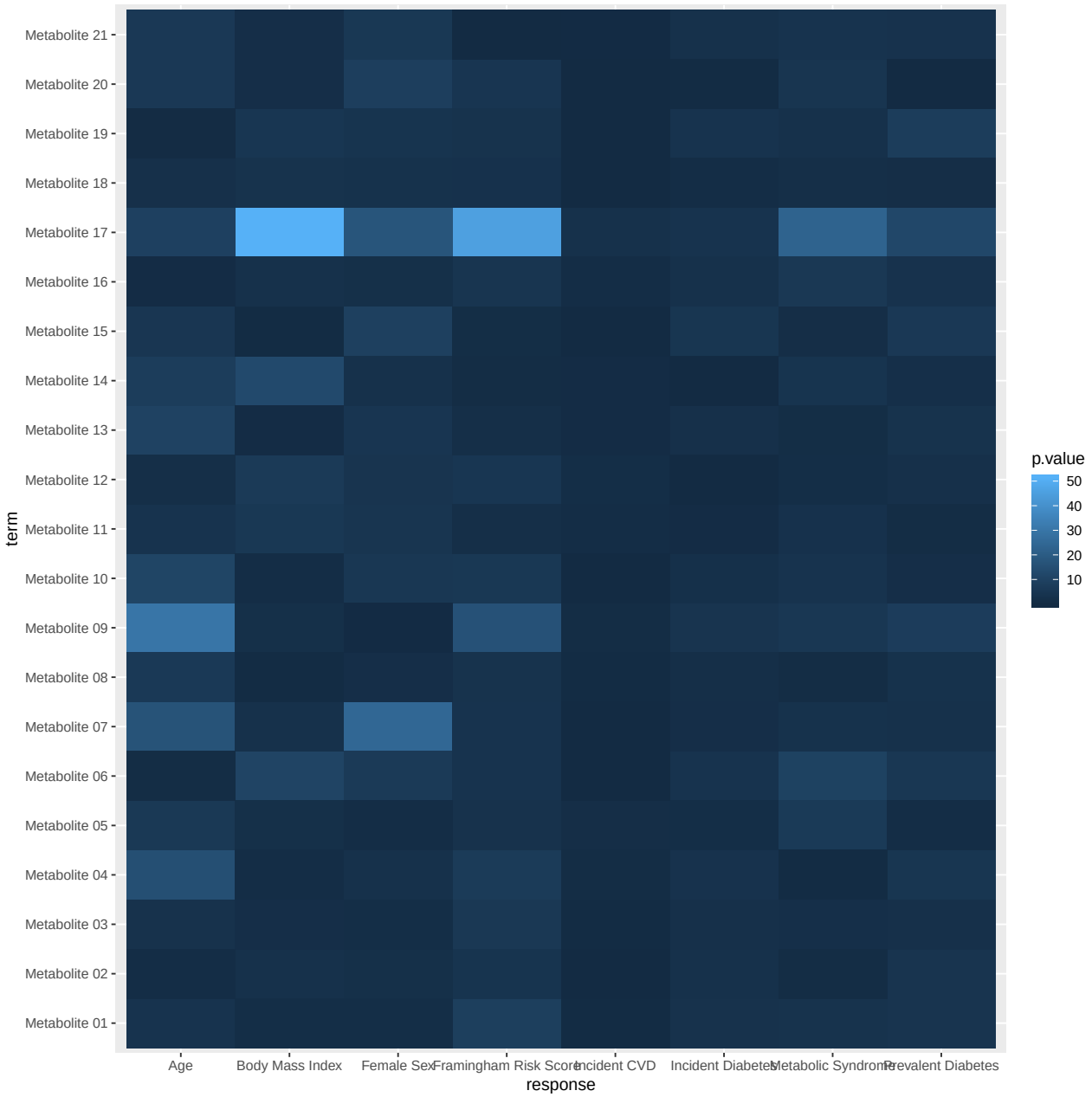
Before plotting, we will first transform `p.value` onto the negative-log scale. This will allow the smallest P-values, which are often of greater interest, to have the largest values when plotted.

```
plot_data <-
  plot_data %>%
  mutate(p.value = -1 * log10(p.value))
```

Plotting

A basic heatmap can be constructed in just two lines of `ggplot2` code!

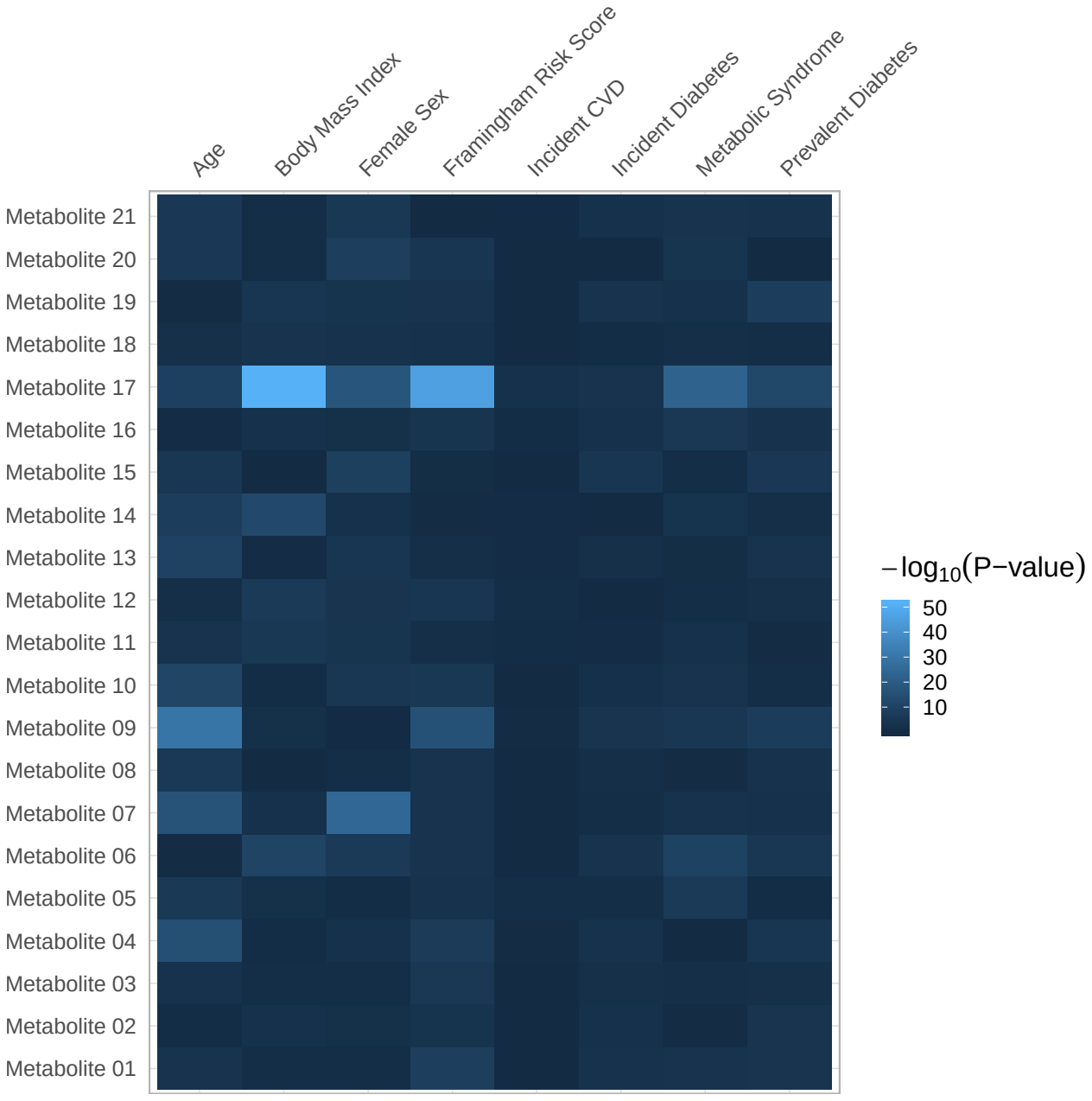
```
heatmap <-
  ggplot(plot_data) +
  geom_tile(aes(x = response, y = term, fill = p.value))

heatmap
```



This is a good start, but we want to clean up layout and presentation. We can do this by creating a custom `ggplot2` theme and adjusting scales and layout.

```
thm <-
  # Good starting theme + set text size
  theme_light(base_size = 18) +
  theme(
    # Remove axis ticks and titles
    axis.title.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.ticks.y = element_blank(),

    # Remove boxes
```

```
    axis.line = element_blank(),
    legend.key = element_blank(),

    # Angle text
    axis.text.x.top  = element_text(angle = 45, hjust = 0)
  )

heatmap <-
  heatmap +
  thm +
  # Move x-axis label to top of plot
  scale_x_discrete(position = 'top') +
  # Better legend title
  scale_fill_continuous(expression(paste(-log[10]('P-value'))))


heatmap
```

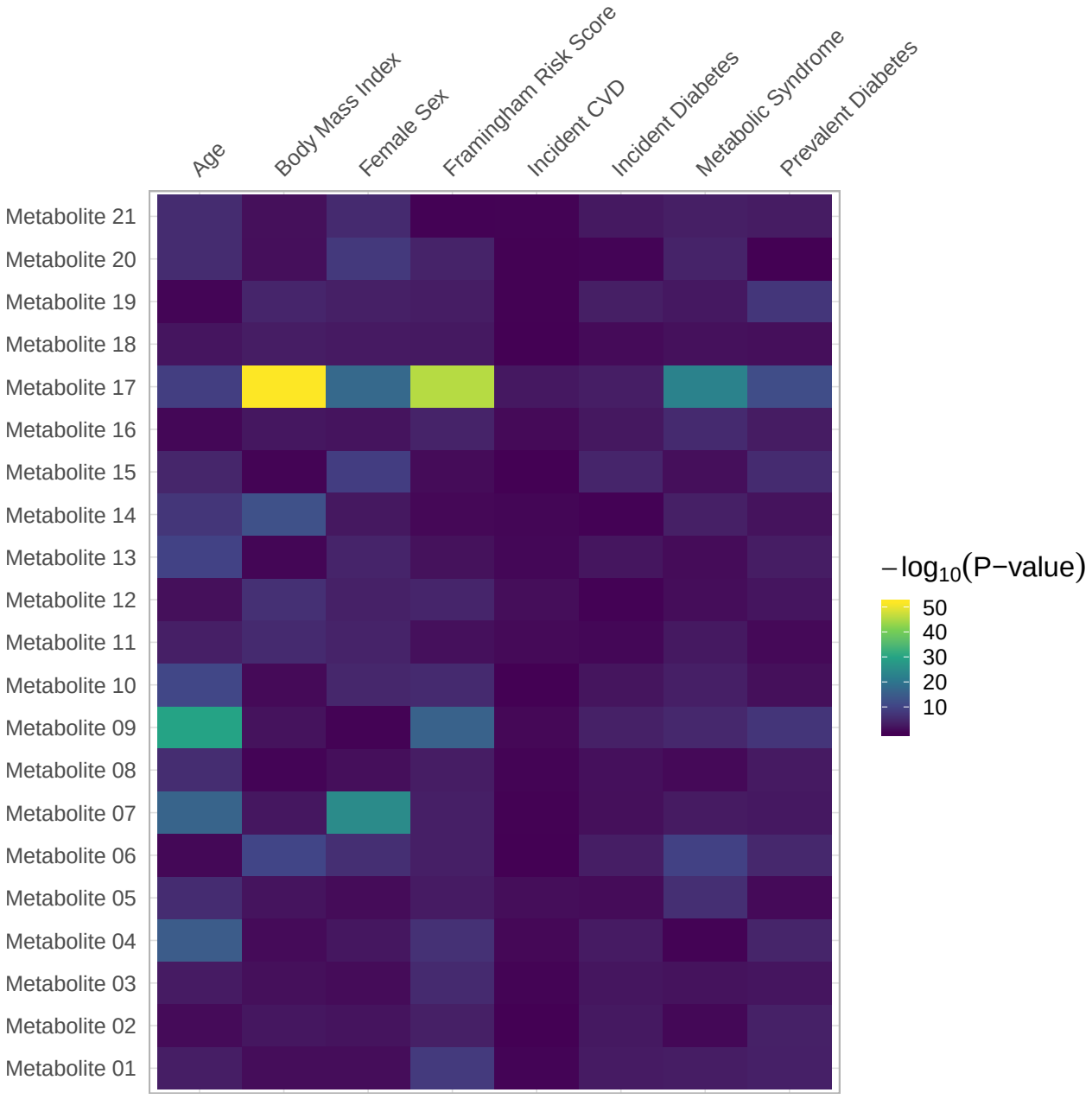


We can further improve the plot by using a 'viridis' color palette.

```
heatmap <-
  heatmap +
  scale_fill_viridis_c(
    expression(paste(-log[10]('P-value')))
  )
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```

```
heatmap
```

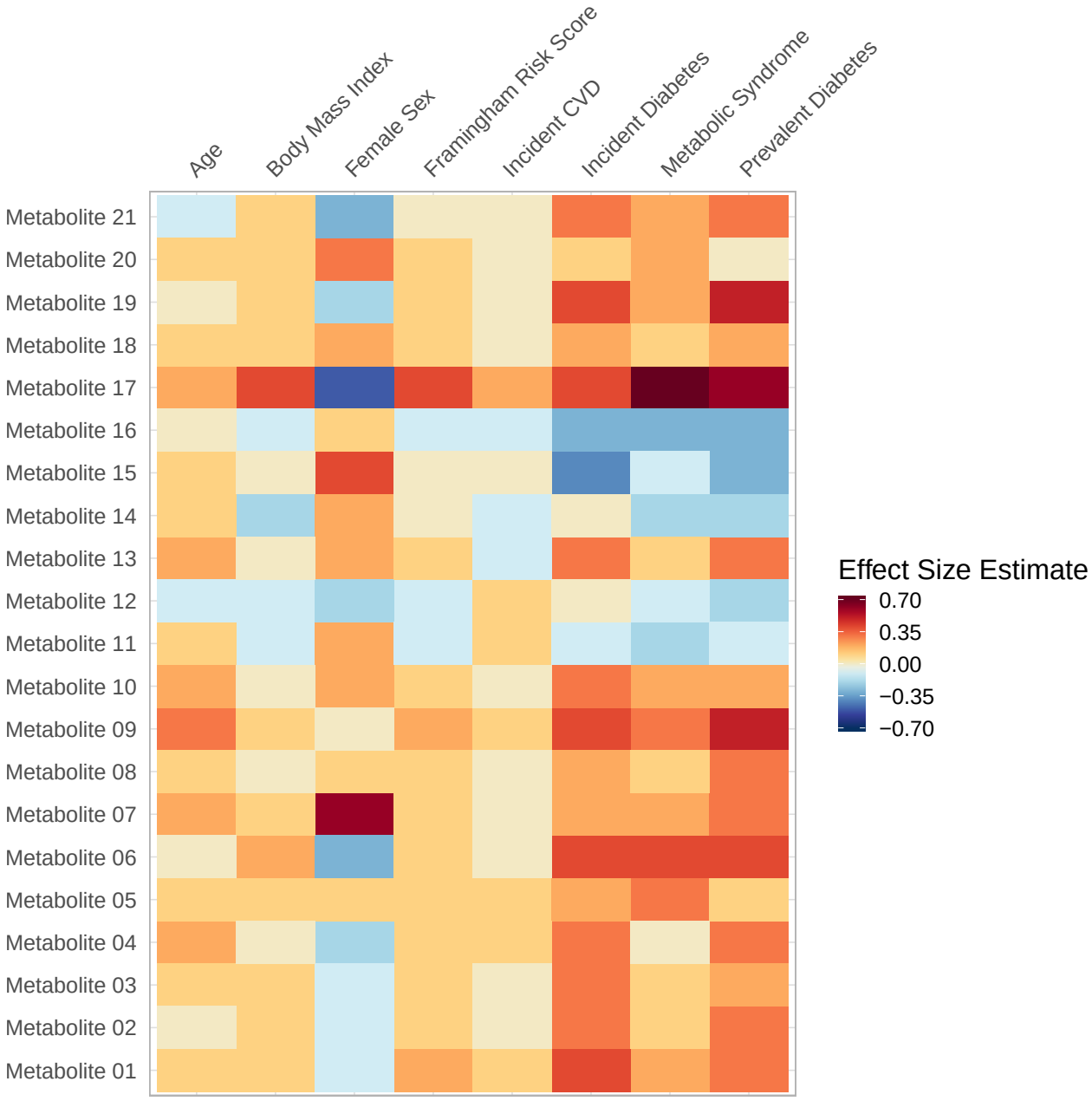


A heatmap can also be used to display information about effect size estimates. For the effect size heatmap, we use a diverging red-blue palette. In order to guarantee a symmetric palette, we set the palette limits based on our data.

```
palette <-
  # Blue
  c("#053061",
    "#313695",
    "#4575b4",
    "#74add1",
    "#abd9e9",
    "#e0f3f8",
    "#fee090",
    "#fdae61",
```

```
    "#f46d43",
    "#d73027",
    "#a50026",
    '#67001f')
# Red

# Calculate symmetric limits based on most extreme value
max_abs_estimate <- max(abs(plot_data$estimate))

max_lim <- max_abs_estimate
min_lim = -1 * max_lim
```

```
heatmap <-
  ggplot(plot_data) +
  geom_tile(aes(x = response, y = term, fill = estimate)) +
  thm +
  scale_x_discrete(position = 'top') +
  scale_fill_gradientn(
    'Effect Size Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim,  min_lim / 2, 0 , max_lim/2, max_lim)
  )

heatmap
```



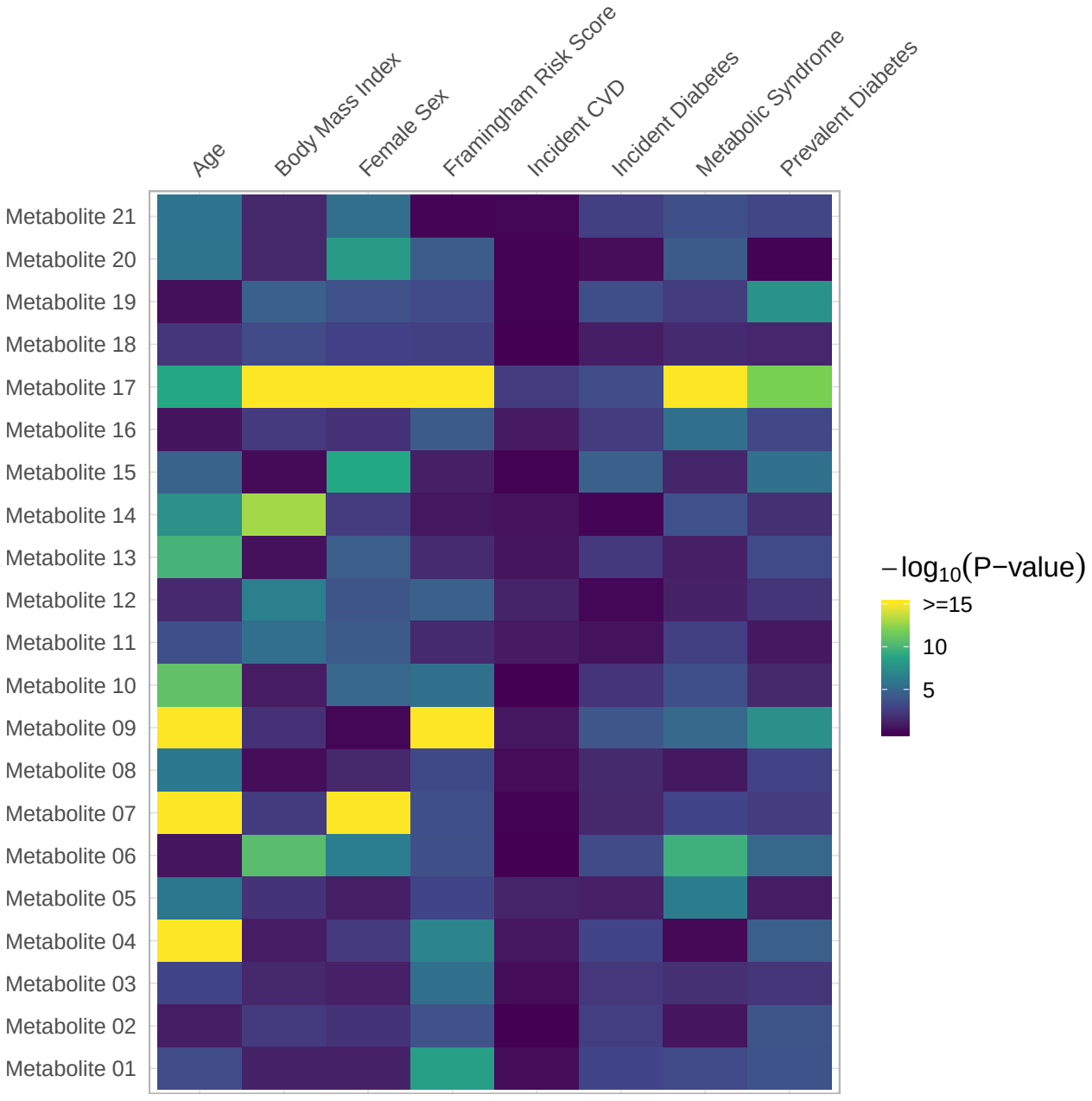
Additional Plot Adjustments

P-value Thresholding

When a few P-values are much smaller than the majority of the data, a P-value heatmap loses color resolution in the range where most of the data lies. One possible solution is to set all P-values above some ceiling, here chosen to be 15, to the value of the ceiling. The threshold can be set at a level where one considers all P-values more extreme than the threshold to be 'of interest'.

```
plot_data_thresholded <-
  plot_data %>%
  mutate(p.value = ifelse(p.value > 15, 15, p.value))

heatmap_thresh <-
```

```r
  # Use the thresholded data
  ggplot(plot_data_thresholded) +
  geom_tile(aes(x = response, y = term, fill = p.value)) +
  scale_x_discrete(position = 'top') +
  # Set the legend breaks and labels to account for the thresholding
  scale_fill_viridis_c(
    expression(paste(-log[10]('P-value'))),
    breaks = c(0, 5, 10, 15),
    labels = c('0', '5', '10', '>=15')
  ) +
  thm


heatmap_thresh
```



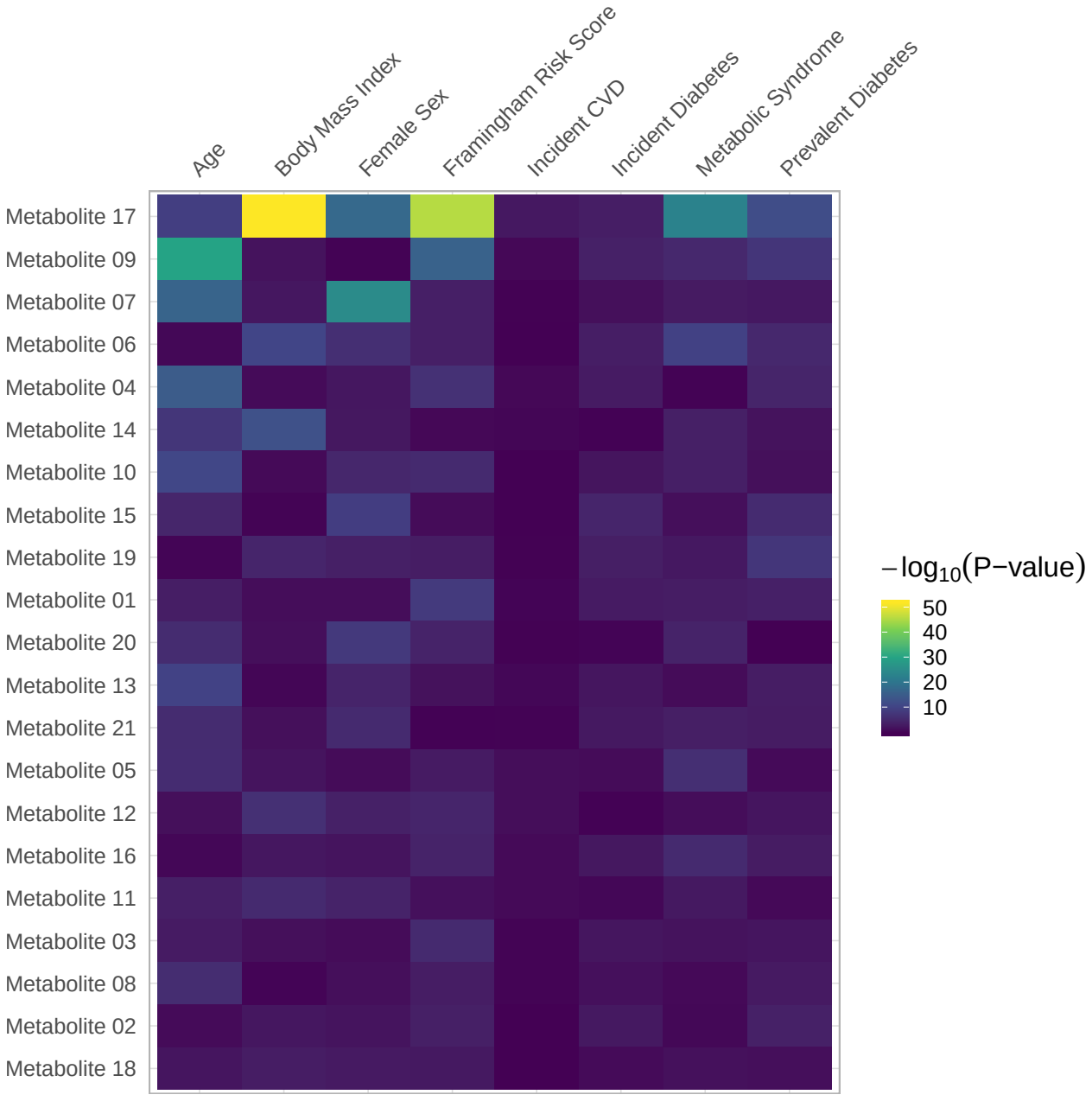
Ordering by P-Value

To make it easier to identify the metabolites that had small P-values in multiple models, we will convert the `term` column into an 'ordered factor' ordered by the average P-value across all models. This will put metabolites with small P-values in multiples models at the top of the plot, and metabolites with large P-values in multiple models at the bottom of the plot. Once the `term` column has been converted, both P-value and effect estimate heatmaps will have the same order.

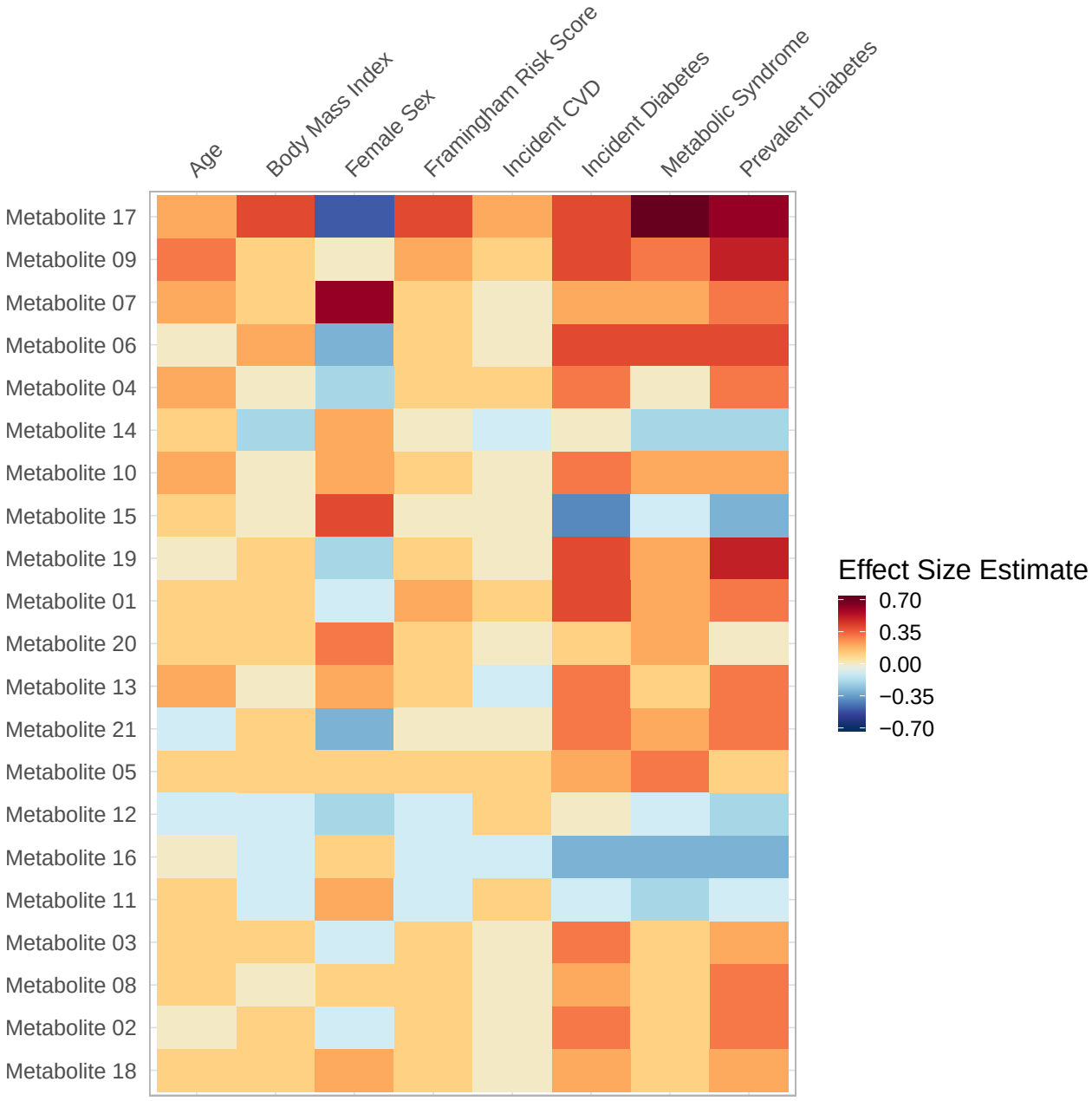
```
# Calculate mean P-value for each metabolite
mpv <-
  plot_data %>%
  group_by(term) %>%
  summarise(mean_pv = mean(p.value))
```

```r
# Order metabolites by average p-value
term_order <-
  mpv %>%
  arrange(mean_pv) %>%
  pull(term)

# Convert term to a factor, ordered by `term_order`
plot_data_pvo <-
  plot_data %>%
  mutate(term = factor(term, levels = term_order))
```

```r
heatmap_base <-
  # Use the data with the term column ordered by mean P-value
  ggplot(plot_data_pvo) +
  scale_x_discrete(position = 'top') +
  thm

pv_heatmap_pvo <-
  heatmap_base +
  geom_tile(aes(x = response, y = term, fill = p.value)) +
  scale_fill_viridis_c(
    expression(paste(-log[10]('P-value')))
  )


pv_heatmap_pvo
```




```
es_heatmap_pvo <-
  heatmap_base +
  geom_tile(aes(x = response, y = term, fill = estimate)) +
  scale_fill_gradientn(
    'Effect Size Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim,  min_lim / 2, 0 , max_lim/2, max_lim)
  )

es_heatmap_pvo
```



Ordering by Cluster

Heatmaps can be ordered by cluster such that similar dependent variable terms are plotted next to one another. We will be using the `hculst` function to cluster the results by effect size estimate. The `term` variable will be converted into an ordered factor, such that clustered terms are plotted next to one another. In order to cluster the data, we will need to reshape it using the spread function from the `tidyr` package. Once the `term` column has been converted, both P-value and effect size estimate heatmaps will have the same order.

```
library(tidyr)
```

```
# Convert to matrix and reshape for clustering.
cluster_data <-
  plot_data %>%
  select(response, term, estimate) %>%
```

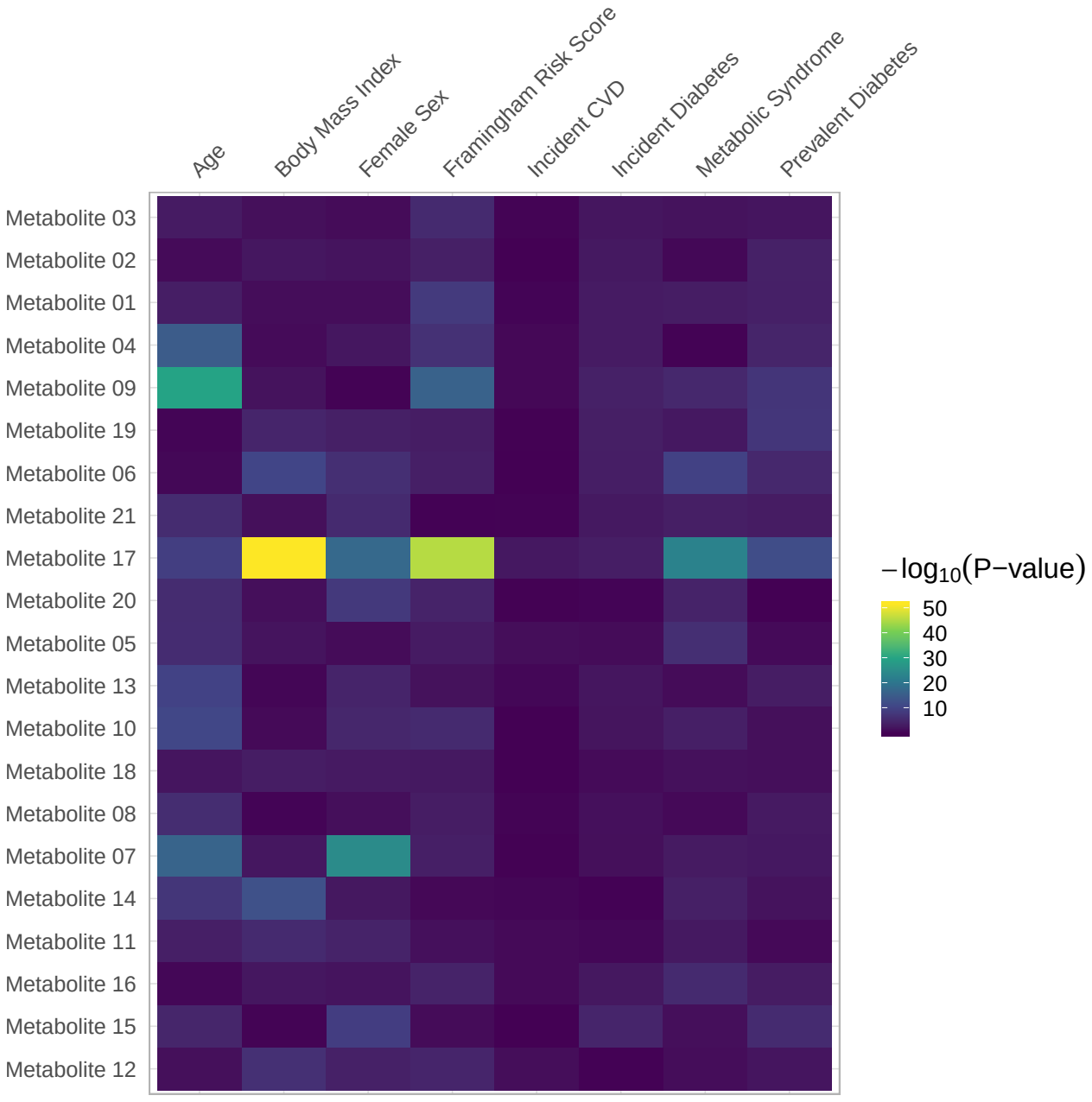
```
  spread(response, estimate)

rnms <-
  cluster_data$term

cluster_data <-
  cluster_data %>%
  select(-term) %>%
  as.matrix()

rownames(cluster_data) <- rnms

# Quick peek to see structure of data
cluster_data[1:5, 1:5]
```

```
##                Age Body Mass Index Female Sex Framingham Risk Score
## Metabolite 01 0.1             0.1       -0.1                    0.2
## Metabolite 02 0.0             0.1       -0.1                    0.1
## Metabolite 03 0.1             0.1       -0.1                    0.1
## Metabolite 04 0.2             0.0       -0.2                    0.1
## Metabolite 05 0.1             0.1        0.1                    0.1
##                Incident CVD
## Metabolite 01           0.1
## Metabolite 02           0.0
## Metabolite 03           0.0
## Metabolite 04           0.1
## Metabolite 05           0.1
```

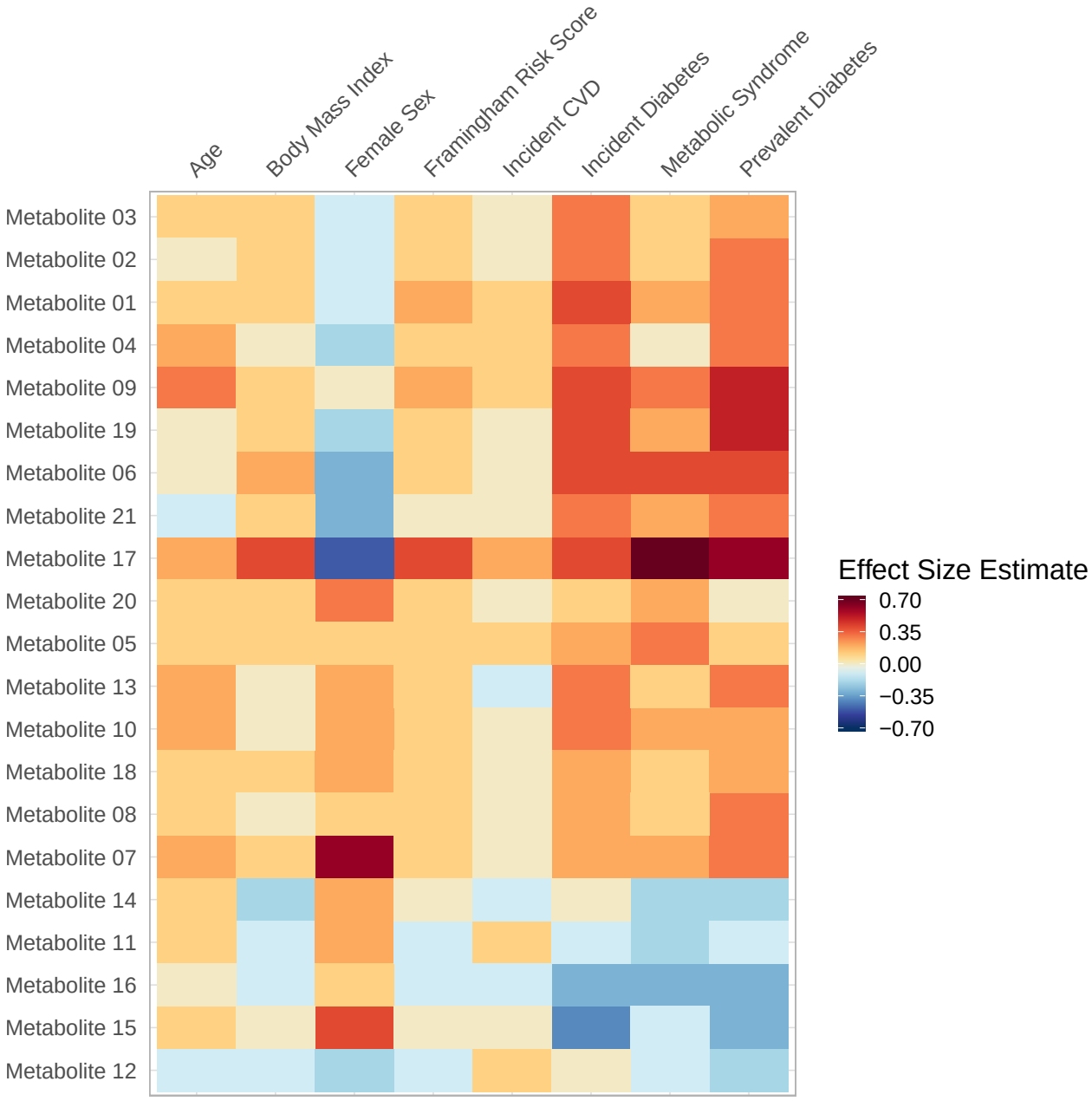
```
# cluster dependent variable terms
clust <- hclust(dist(cluster_data), method = 'ward.D2')
```

```
# `clust$order` orders `term` into clusters
term_order <-
  clust$labels[clust$order]

# Convert term to a factor, ordered by `term_order`
plot_data_clo <-
  plot_data %>%
  mutate(term = factor(term, levels = term_order))
```

```
heatmap_base <-
  # Use cluter ordered data
  ggplot(plot_data_clo) +
  scale_x_discrete(position = 'top') +
  thm

pv_heatmap_clo <-
  heatmap_base +
  geom_tile(aes(x = response, y = term, fill = p.value)) +
  scale_fill_viridis_c(
    expression(paste(-log[10]('P-value')))
  )


pv_heatmap_clo
```




```
es_heatmap_clo <-
  heatmap_base +
  geom_tile(aes(x = response, y = term, fill = estimate)) +
  scale_fill_gradientn(
    'Effect Size Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim,  min_lim / 2, 0 , max_lim/2, max_lim)
  )

es_heatmap_clo
```



Side-by-side plots

Often, one will want to plot multiple graphs together. We will use the `ggarrange` function from the `egg` (extensions for ggplot2) package to accomplish this. By plotting multiple graphs together, we can easily scale up the information presented in our plots.

A common problem when composing multiple graphs in the same plot is alignment. Graphs will often share common features that need to be aligned, and differences in scale can break that alignemnt when the graphs are plotted together. Thus, an important consideration when creating our graphs plots will be to match the scale of every graph we want to align.

```
library(egg)
```

Adding dendrograms

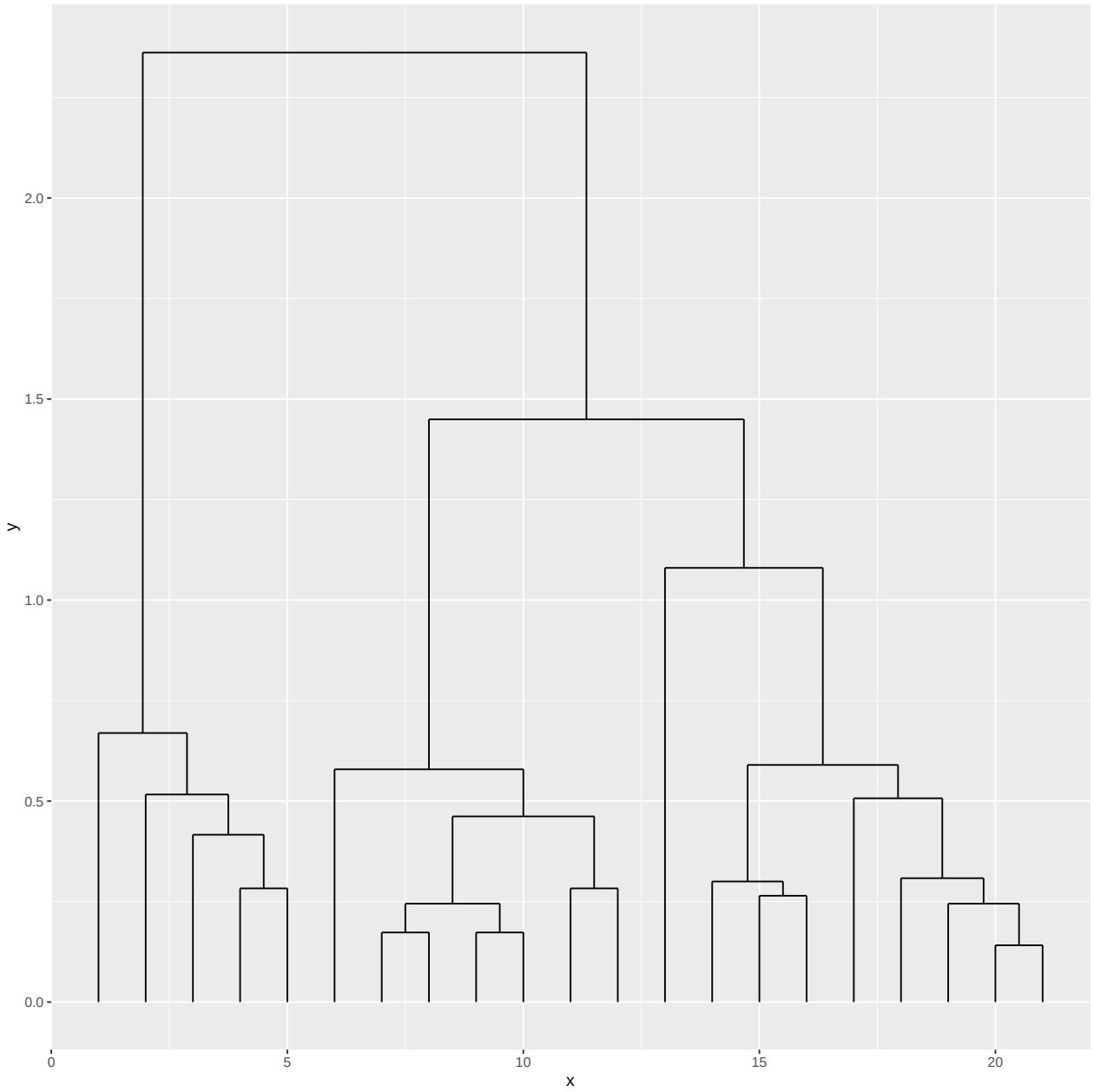
Dendrograms can be added to cluster-ordered `ggplot2` plots using the `ggdendro` package.

```
library(ggdendro)
```

```
# Extract dendrogram data from previous cluster results
dendro_dat <- segment(dendro_data(clust))
```

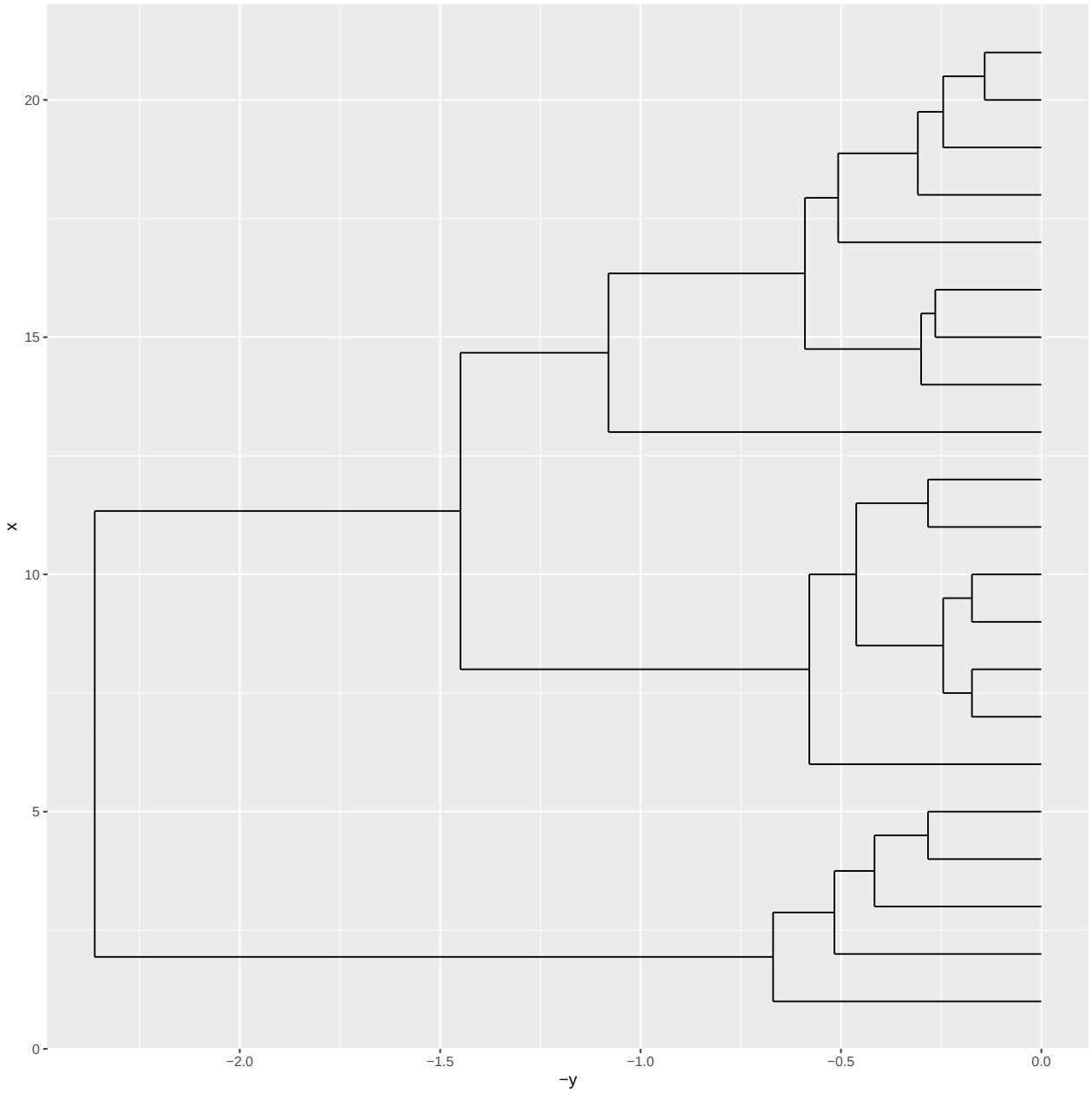
A basic dendrogram can be plotted quite easily.

```
dendro <-
  ggplot(dendro_dat) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend), colour = 'black')

dendro
```



The default dendrogram points down. To put the dendrogram on the left of our plot, we want it to point to the right. We can do this by switching the x and y coordinates.

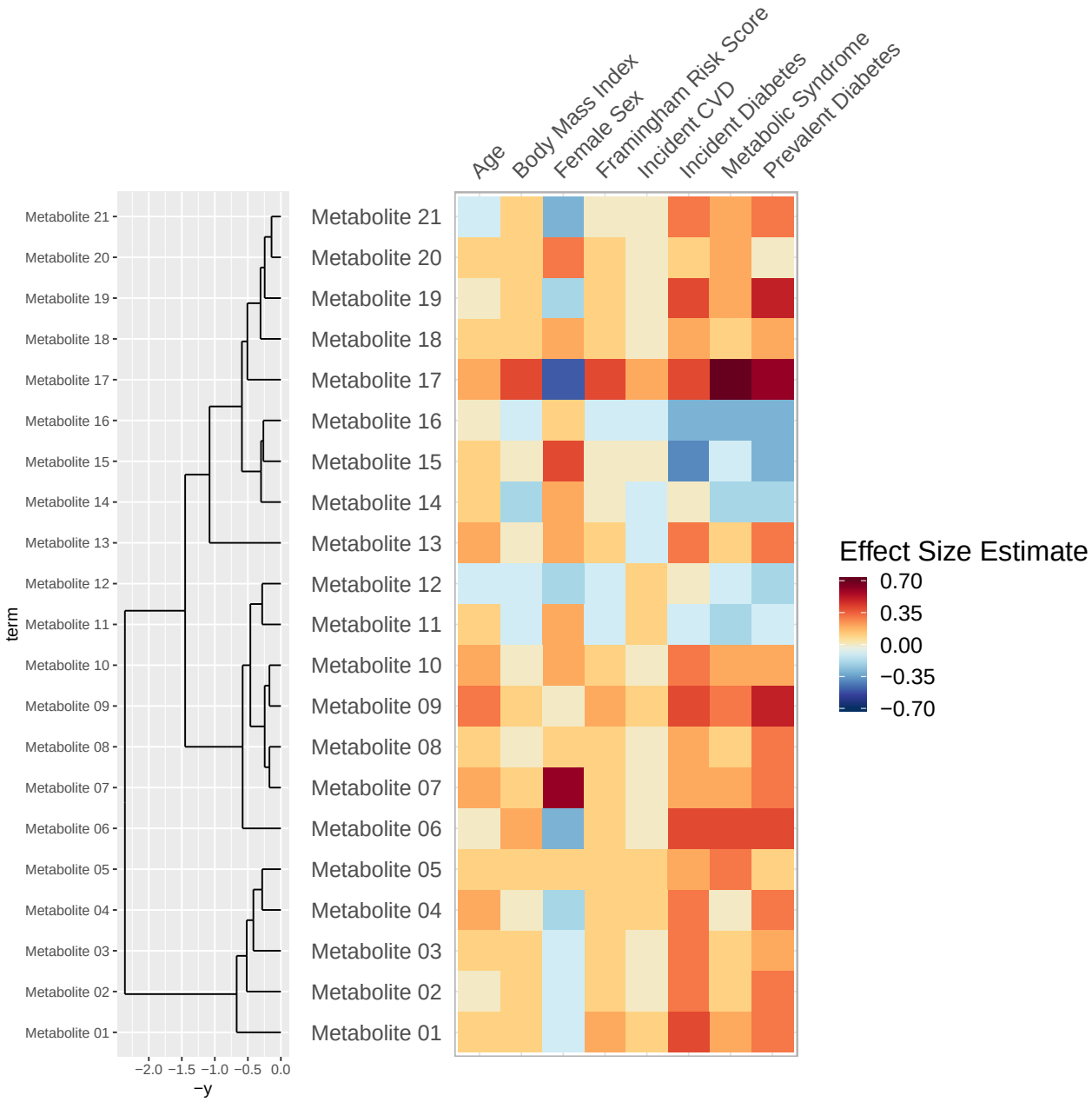
```
dendro <-
  ggplot(dendro_dat) +
  geom_segment(aes(x = -y, y = x, xend = -yend, yend = xend), colour = 'black')

dendro
```



Before plotting our dendrogram and rainplot side-by-side, we need to first create a version of our dendrogram with the same scale as our heatmap

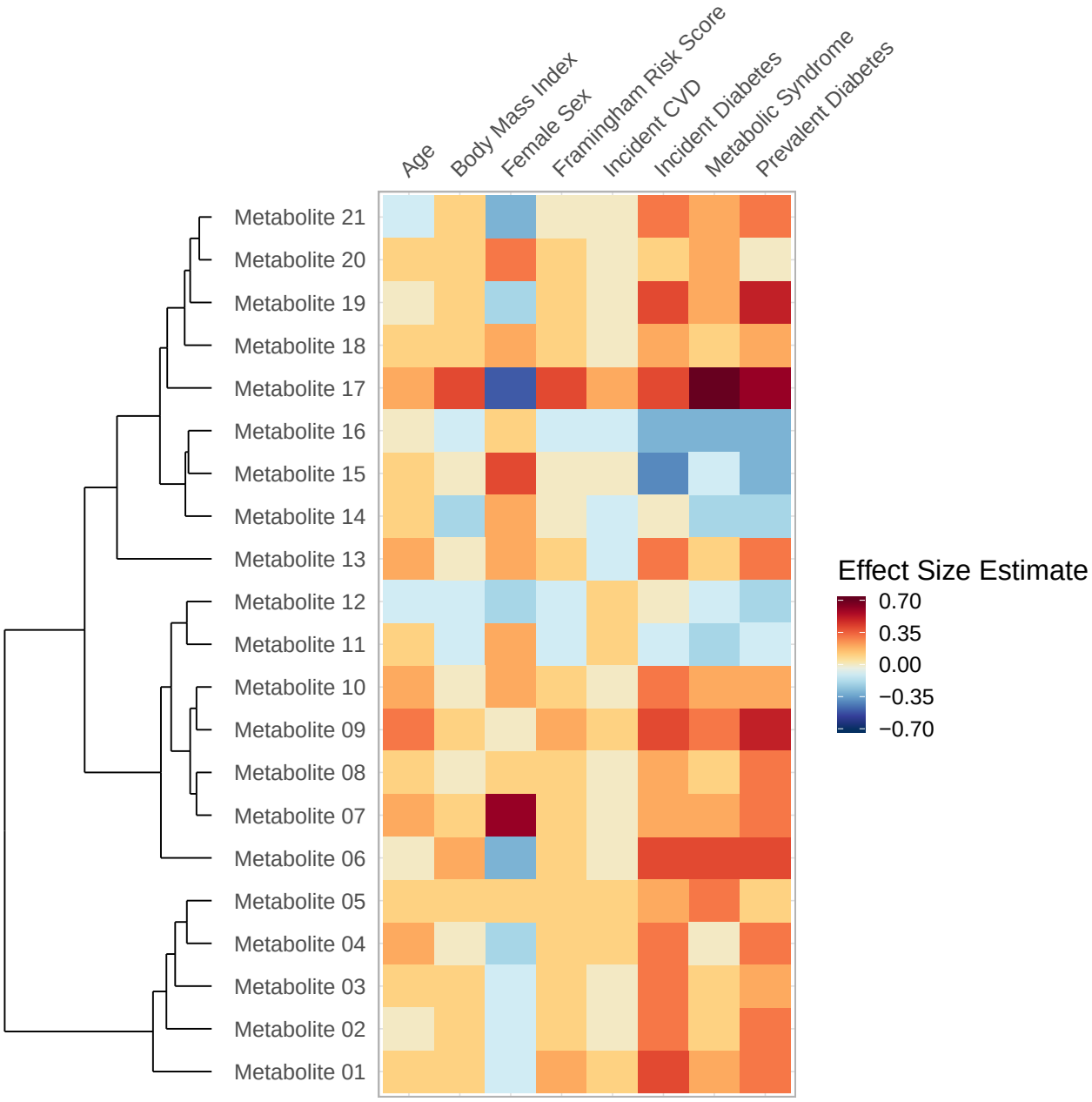
```
dendro <-
  ggplot(dendro_dat) +
  # Empty ggplot with same y-scale as rainplot
  geom_blank(aes(y = term), data = plot_data) +
  geom_segment(aes(x = -y, y = x, xend = -yend, yend = xend), colour = 'black')
```

```
ggarrange(dendro, heatmap, ncol = 2, widths = c(1, 2))
```



Finally, we cleanup the dendrogram using `theme_dendro` and removing excess blank space.

```
dendro <-
  dendro +
  theme_dendro() +
  # 'expand' controls whitespace around the dendrogram. The non-zero argument
  # may need to be increasesed if the line thickness of the dendrogram is
  # increased to make sure the entire dendrogram is plotted
  scale_x_discrete(position = 'top', expand = c(0, 0.02, 0, 0))

ggarrange(dendro, heatmap, ncol = 2, widths = c(1, 2))
```



Side-by-side heatmaps

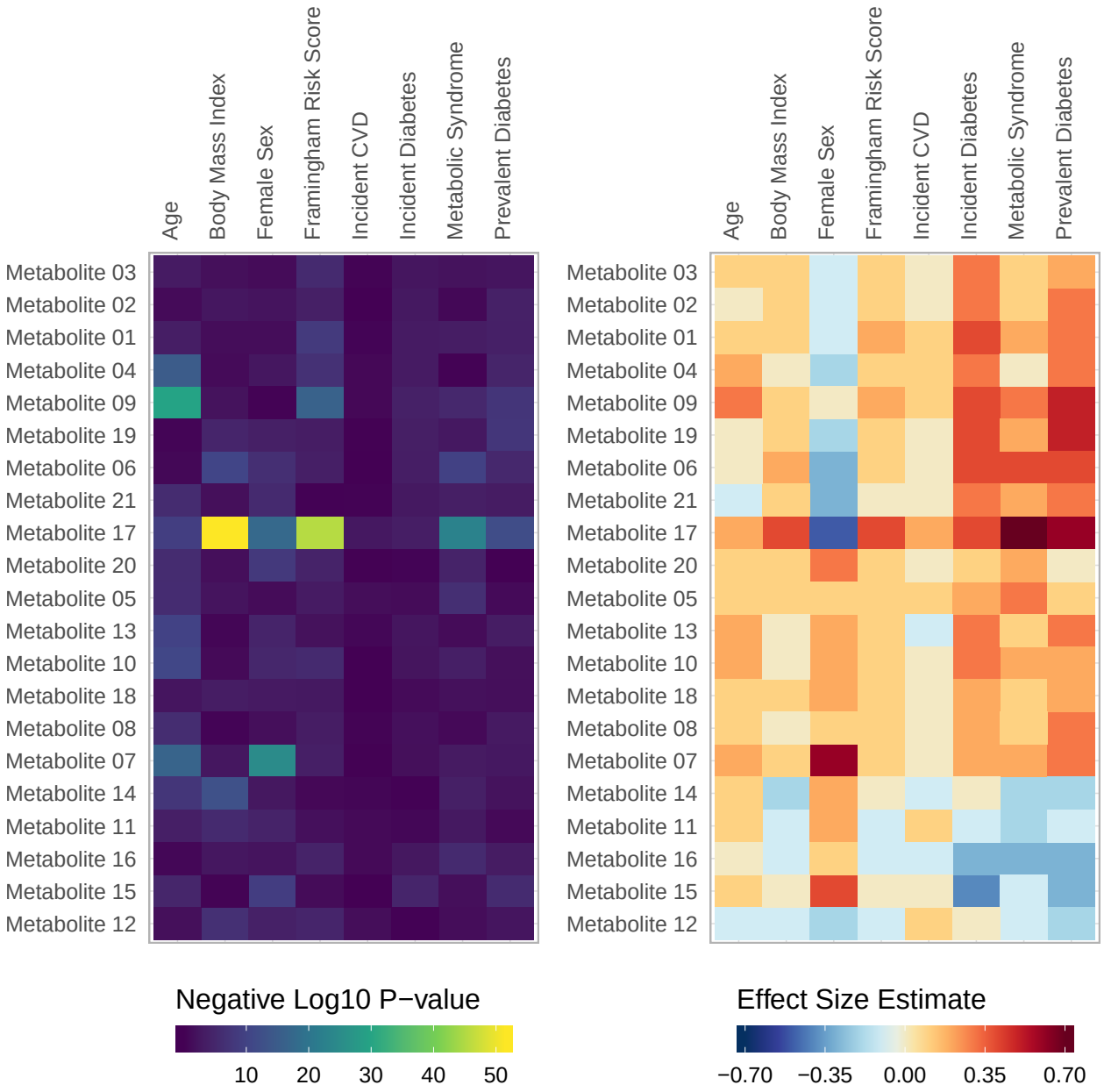
One might find it desirable to show a heatmap for both P-values and effect sizes at the same time. When displaying heatmaps side-by-side, some visual elements will need to be changed or rearranged for optimal presentation.

```
# 90 degree text and move legend below plot
sbs_heatmap_thm <-
  thm +
  theme(axis.text.x.top  = element_text(angle = 90),
        legend.position = 'bottom')

# Put legend title on top of bar
gcb <-  guides(fill = guide_colorbar(title.position = 'top', barwidth = 15))
```

```
pv_heatmap_clo <-
  pv_heatmap_clo +
  sbs_heatmap_thm +
  gcb +
  # Can only use regular text for legend title
  scale_fill_viridis_c('Negative Log10 P-value')
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```

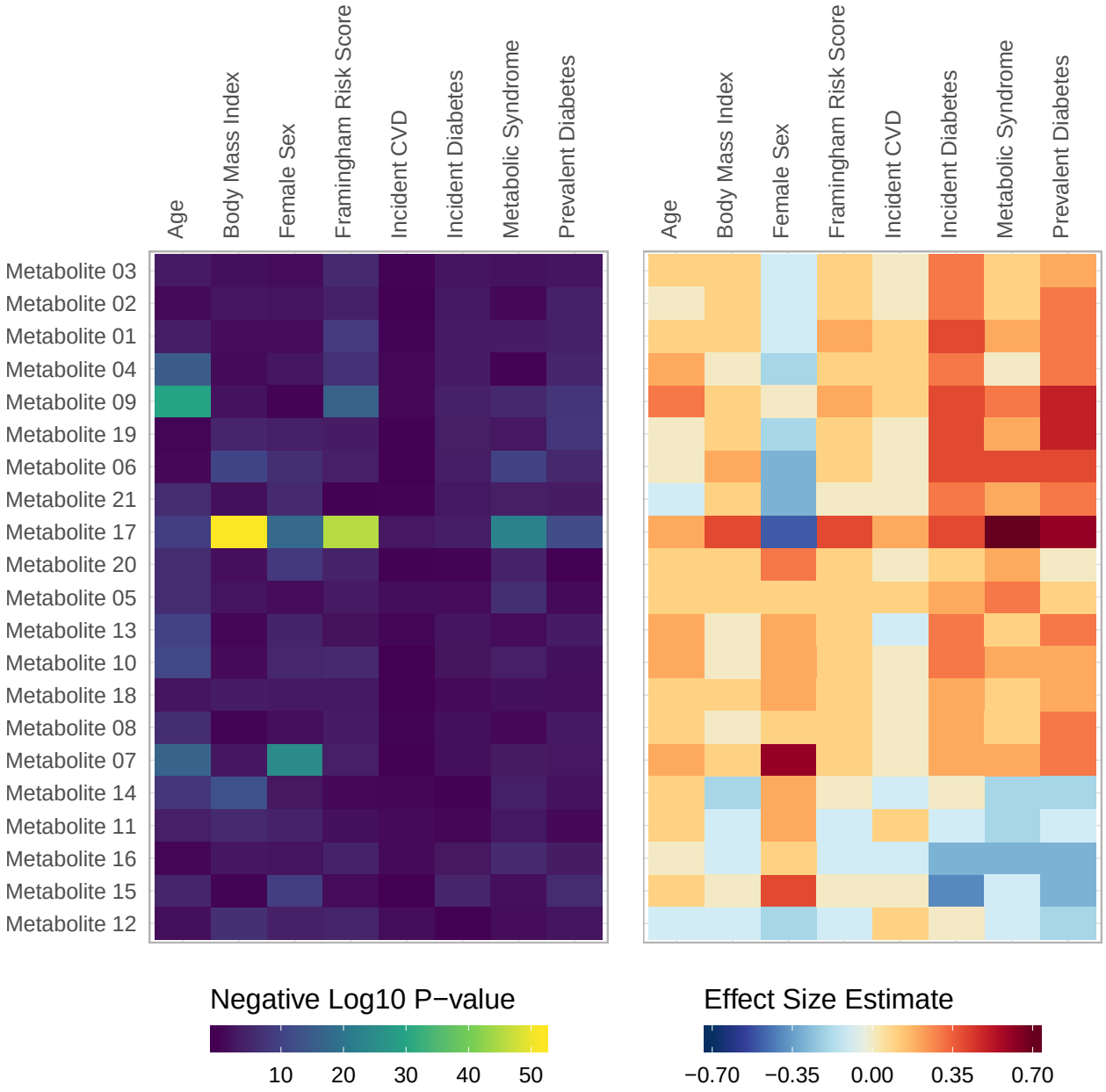
```
es_heatmap_clo <-
  es_heatmap_clo +
  sbs_heatmap_thm +
  gcb

ggarrange(pv_heatmap_clo, es_heatmap_clo, ncol = 2)
```



One set of Y-axis labels

If one wants a single set of y-axis labels for both heatmaps, we simply remove the y-axis labels from the right heatmap.

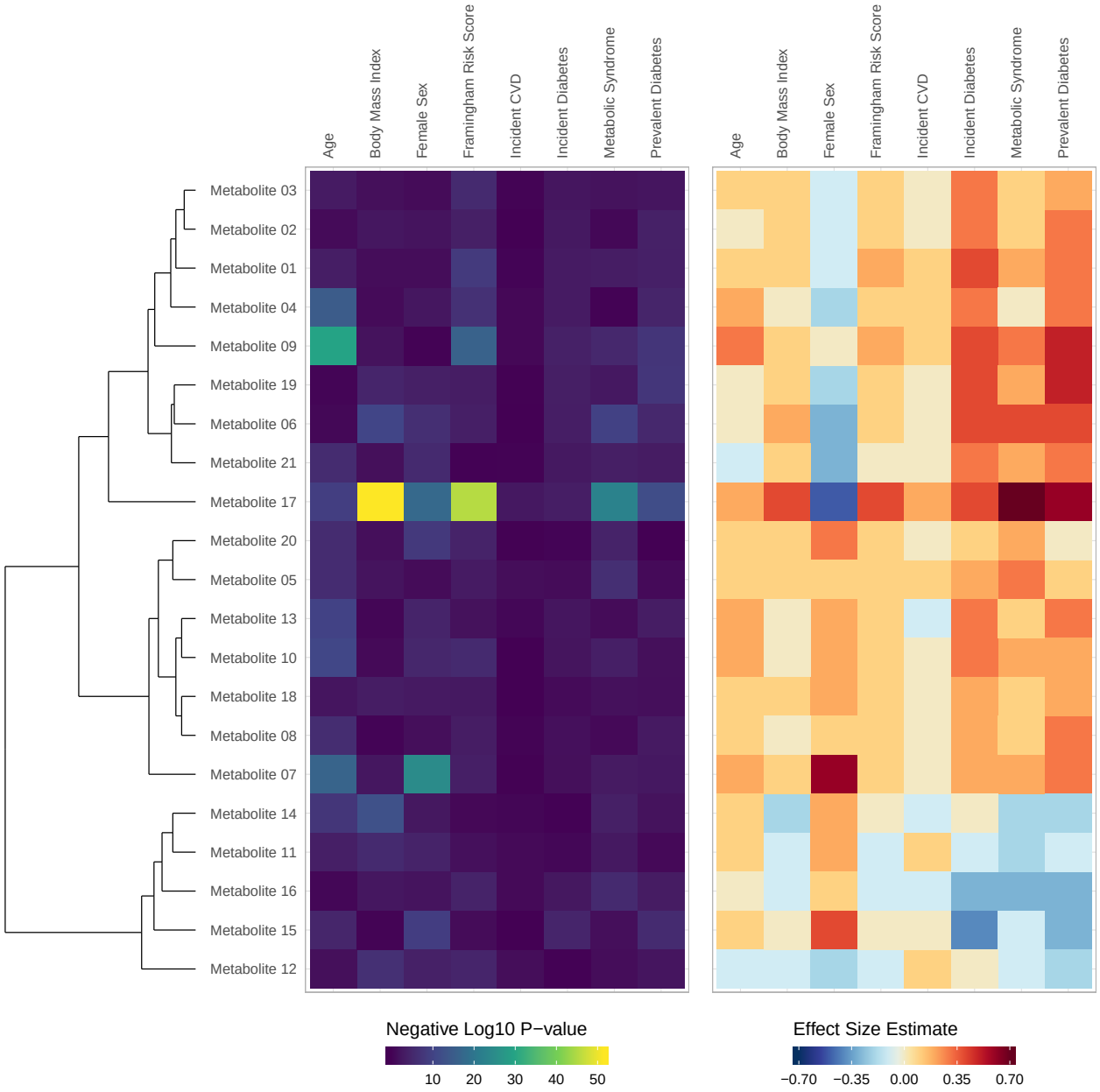
```
es_heatmap_clo_nl <-
  es_heatmap_clo +
  theme(axis.text.y = element_blank())

ggarrange(pv_heatmap_clo, es_heatmap_clo_nl, ncol = 2)
```



Adding dendrograms

We can add a dendrogram to the left of side by side heatmaps with one set of Y-axis labels

```
ggarrange(dendro, pv_heatmap_clo, es_heatmap_clo_nl,
              ncol = 3,
              widths = c(1, 2, 2))
```


Age
Body Mass Index
Female Sex
Framingham Risk Score
Incident CVD
Incident Diabetes
Metabolic Syndrome
Prevalent Diabetes
Metabolite 03
Metabolite 02
Metabolite 01
Metabolite 04
Metabolite 09
Metabolite 19
Metabolite 06
Metabolite 21
Metabolite 17
Metabolite 20
Metabolite 05
Metabolite 13
Metabolite 10
Metabolite 18
Metabolite 08
Metabolite 07
Metabolite 14
Metabolite 11
Metabolite 16
Metabolite 15
Metabolite 12
Negative Log10 P−value
10
20
30
40
50
Effect Size Estimate
−0.70
−0.35
0.00
0.35
0.70