# Rainplots Tutorial

*Mir Henglin*

*null*

To construct rainplots 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

Rainplots are useful for summarizing the results of multiple models at the same time. In order to plot those results in `ggplot2`, those results must formatted properly. Specifically, the data must be organized in a `data.frame` with columns indicating

- The model that the results came from
- The term that was evaluated
- The P-value
- The regression estimate

An example of data organized in this way can be seen below.

```
plot_data
```

```
## # A tibble: 168 x 5
##    response              term                    estimate  p.value  rank
##    <fct>                 <fct>                      <dbl>    <dbl> <int>
##  1 Body Mass Index       mzid_443.210809_1.7953       0.4 5.74e-52     1
##  2 Framingham Risk Score mzid_443.210809_1.7953       0.4 4.80e-46     1
##  3 Age                   mzid_349.202006_4.4691       0.3 1.87e-30     7
##  4 Female Sex            mzid_373.205458_6.3700       0.6 2.47e-25    37
##  5 Metabolic Syndrome    mzid_443.210809_1.7953       0.7 1.78e-23     1
##  6 Female Sex            mzid_443.210809_1.7953      -0.5 3.92e-18     1
##  7 Age                   mzid_373.205458_6.3700       0.2 2.96e-17    37
##  8 Framingham Risk Score mzid_349.202006_4.4691       0.2 8.91e-17     7
##  9 Age                   mzid_313.238624_5.2172       0.2 1.02e-15    43
## 10 Body Mass Index       mzid_361.236724_3.5879      -0.2 1.47e-13    67
## # ... with 158 more rows
```

In this dataset, `estimate` and `p.value` indicate the regression estimate and the p.value of that estimate. `term` indicates the ID of the metabolite included in the model. `response` indicates which model that a term corresponds to. For example if `response` = Body Mass Index, this indicates that the regression estimate and the p.value correspond to the model where Body Mass Index was the response variable.

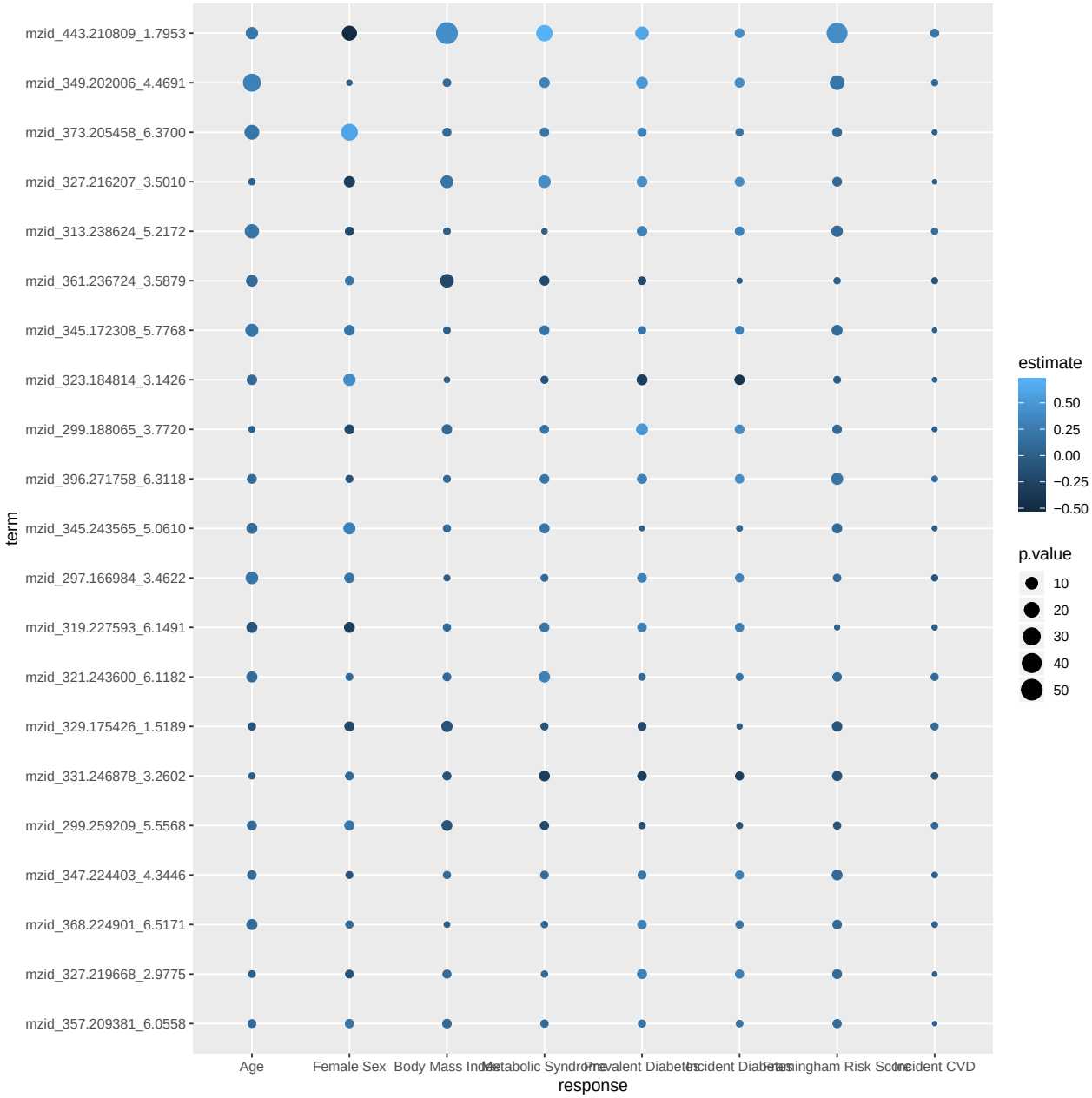
##Plotting

Before plotting, we will first transform `p.value` onto the negative log-scale. This allows smaller P-values, which are often of greater interest, to appear larger when plotted.

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

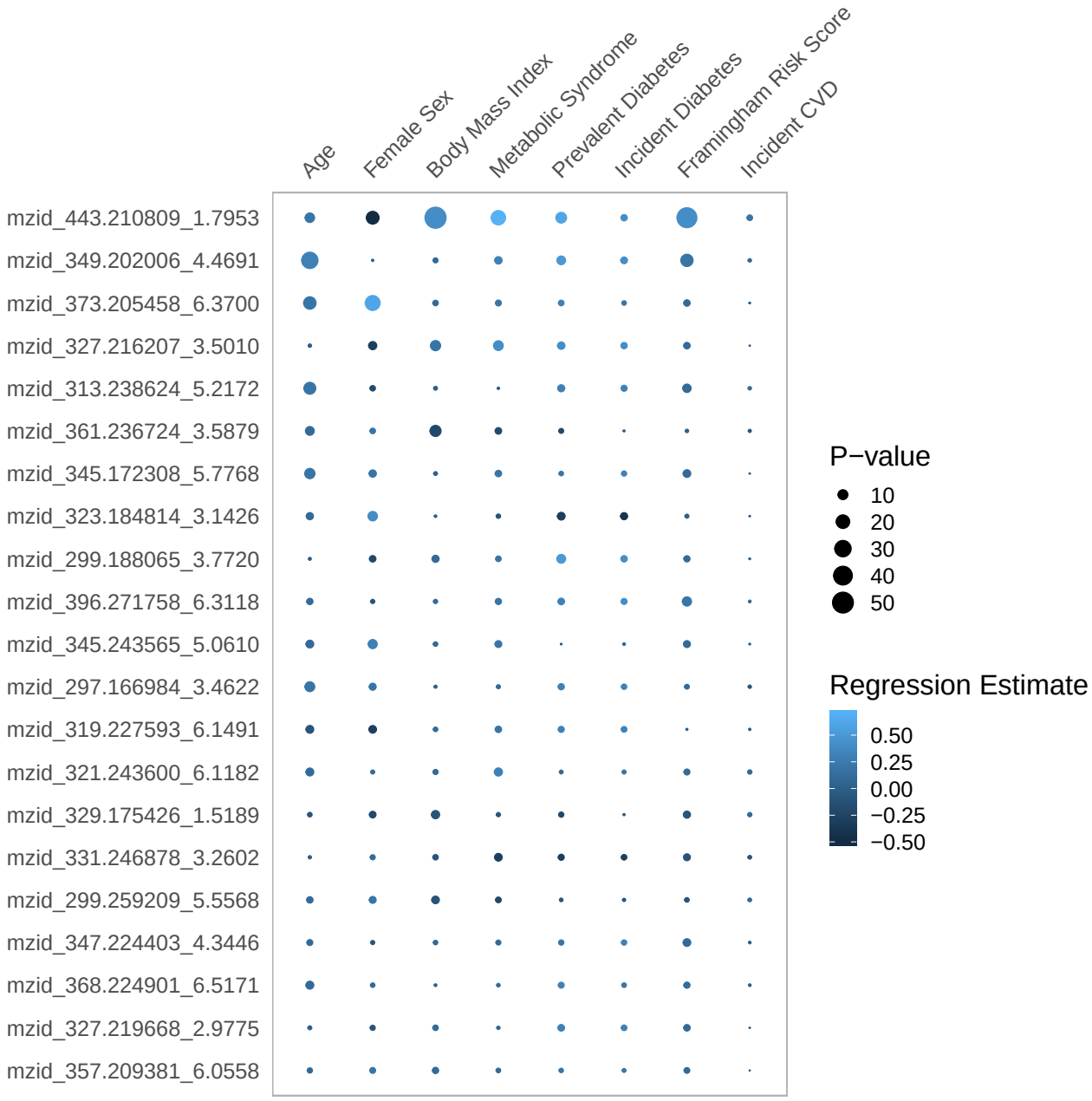
We can then construct a basic rainplot.

```
rainplot <-
  plot_data %>%
  ggplot(aes(x = response, y = term)) +
  geom_point(aes(colour = estimate, size = p.value))

rainplot
```




This is a good start, but we will want to clean up the visual presentation. We can do this by creating a custom `ggplot2` theme and adjusting legend titles. One thing to ensure is to represent P-values (the size of the plotted points) by area instead of radius. When comparing two points of different size, humans perceive the area of points, not their radius, when comparing them. Thus a value that is twice another should have twice as much area, not double the radius. This is ensured by using `scale_size_area`.

```r
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 Gridlines and boxes
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_blank(),
    legend.key = element_blank(),

    # White backgrounds
    panel.background = element_rect(fill = 'white'),
    plot.background = element_rect(fill = 'white'),
    legend.background = element_rect(fill = 'white'),

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

rainplot <-
  rainplot +
  thm +
  scale_x_discrete(position = 'top') +
  scale_size_area('P-value') +
  scale_color_continuous('Regression Estimate')

rainplot
```



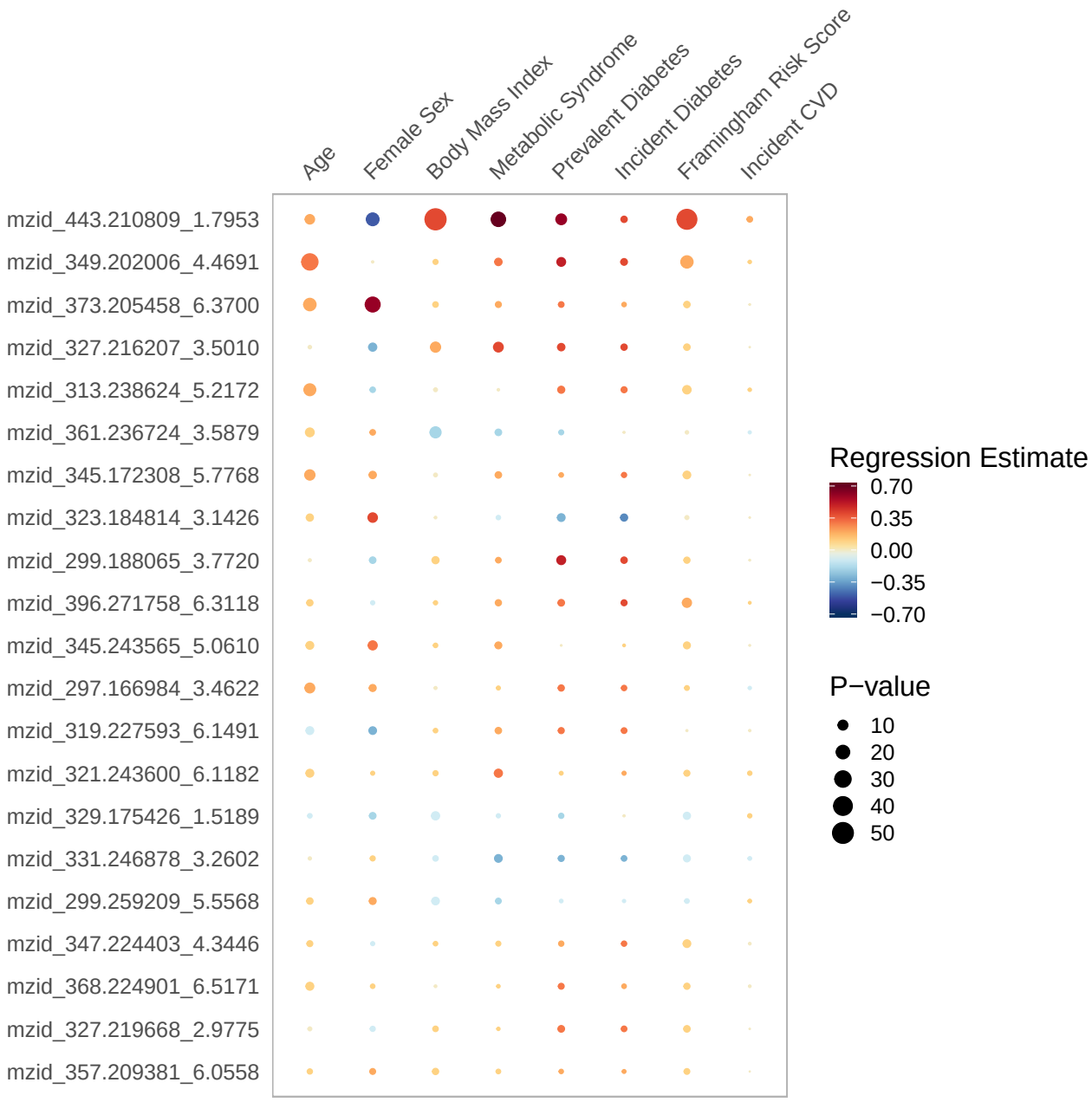

To make the presentation of the regression estimates clearer, we create a diverging color scale, and set positive and negative limits equidistant from 0.

```
palette <-
  c("#053061",
    "#313695",
    "#4575b4",
    "#74add1",
    "#abd9e9",
    "#e0f3f8",
    # "#ffffbf",
    "#fee090",
    "#fdae61",
    "#f46d43",
```

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

max_abs_estimate <- max(abs(plot_data$estimate))

max_lim <- max_abs_estimate
min_lim = -1 * max_lim

rainplot <- rainplot +
  scale_color_gradientn(
    'Regression Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim,  min_lim / 2, 0 , max_lim/2, max_lim)
  )
```

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

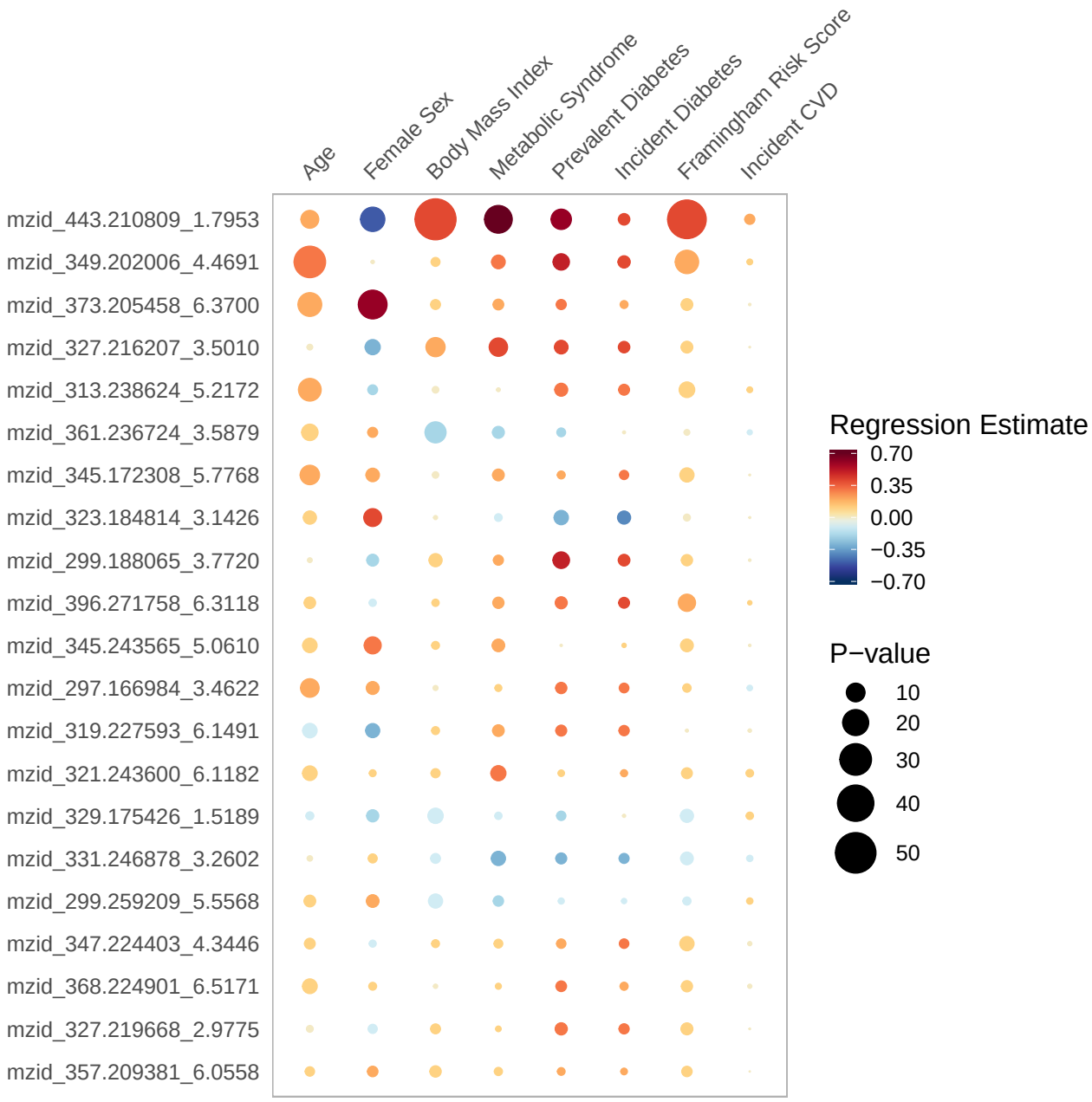
```
rainplot
```




Another step to improve presentation is to increase the maximum size of each point There will be a bit of trial and error here; if the size threshold is too large, the points will overlap.

```
rainplot +
  scale_size_area('P-value', max_size = 12)
```

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



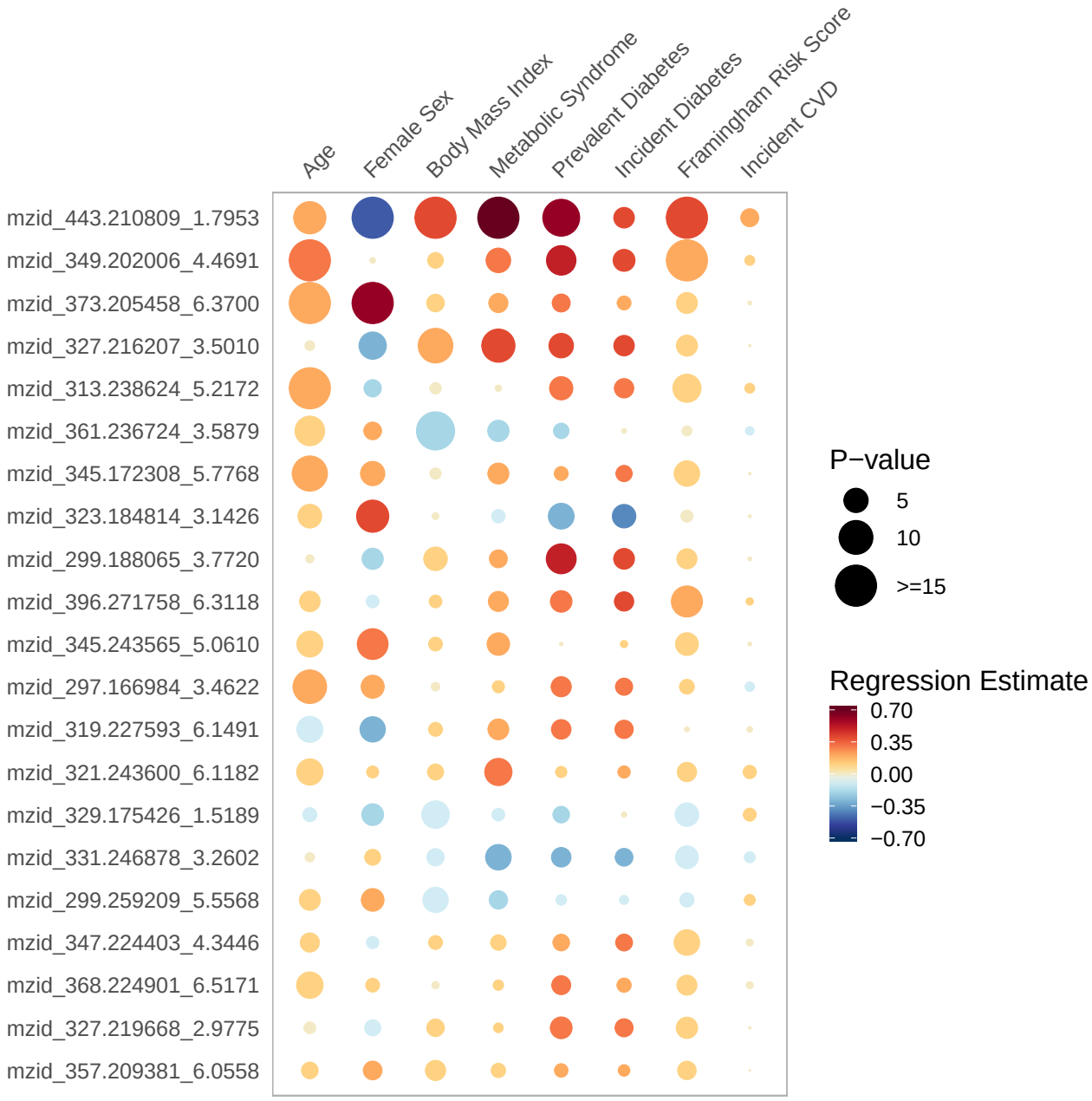

## Additional Plot Adjustments

### P-value Thresholding

When a few `p.values` are much smaller than the majority of the data the plot loses size 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))
```

```r
rainplot <-
  plot_data_thresholded %>%
  ggplot(aes(x = response, y = term)) +
  geom_point(aes(colour = estimate, size = p.value)) +
  scale_color_gradientn(
    'Regression Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim, min_lim / 2, 0 , max_lim / 2, max_lim)
  ) +
  scale_size_area('P-value', max_size = 12, breaks = c(5, 10, 15), labels = c('5', '10', '>=15')) +
  scale_x_discrete(position = 'top') +
  thm

rainplot
```



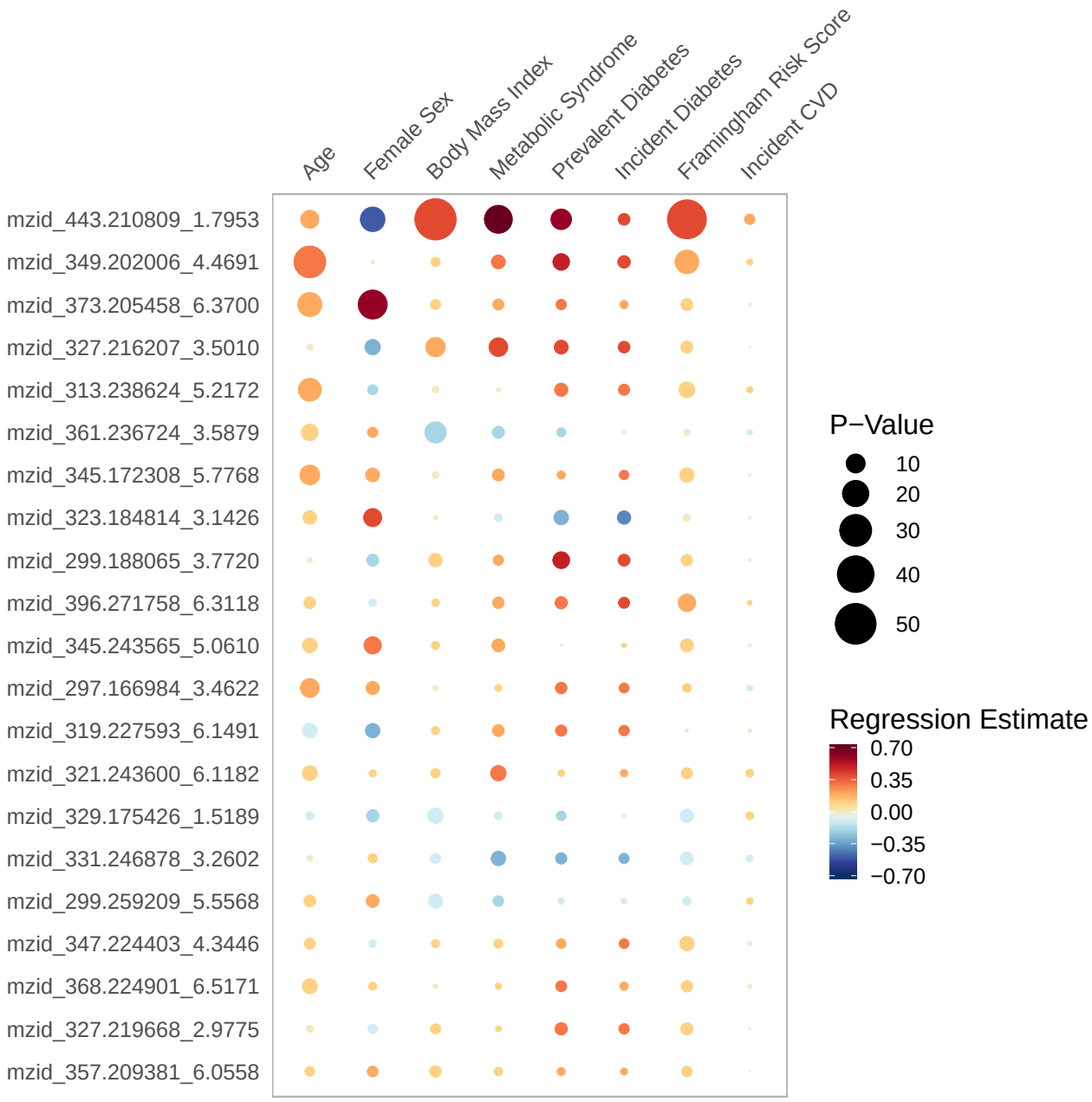

### Ordering by P-Value

To make it easier to identify the metabolites that had small P-values in multiple models, we will convert the `term` variable into a factor variable 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.

```
# Order metabolites by average p-value
term_order <-
  plot_data %>%
  group_by(term) %>%
  summarise(mpv = mean(p.value)) %>%
  arrange(mpv) %>%
```

```
  pull(term)

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

```
rainplot <-
  plot_data %>%
  ggplot(aes(x = response, y = term)) +
  geom_point(aes(colour = estimate, size = p.value)) +
  scale_color_gradientn(
    'Regression Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim, min_lim / 2, 0 , max_lim / 2, max_lim)
  ) +
  scale_size_area('P-Value', max_size = 12) +
  scale_x_discrete(position = 'top') +
  thm

rainplot
```




### Ordering by Cluster

Rainplots can be clustered like heatmaps. We will be using the `hculst` function to cluster the results by regression estimate. The `term` variable will be converted into an ordered factor, just as was done in `Ordering by P-Value`. In order to cluster the data, we will need to reshape it using the spread function from the `tidyr` package.

```
library(tidyr)
```

```
# Convert to matrix for clustering. `term` on the y-axis, `response` on the x-axis
cluster_data <-
  plot_data %>%
  select(response, term, estimate) %>%
```

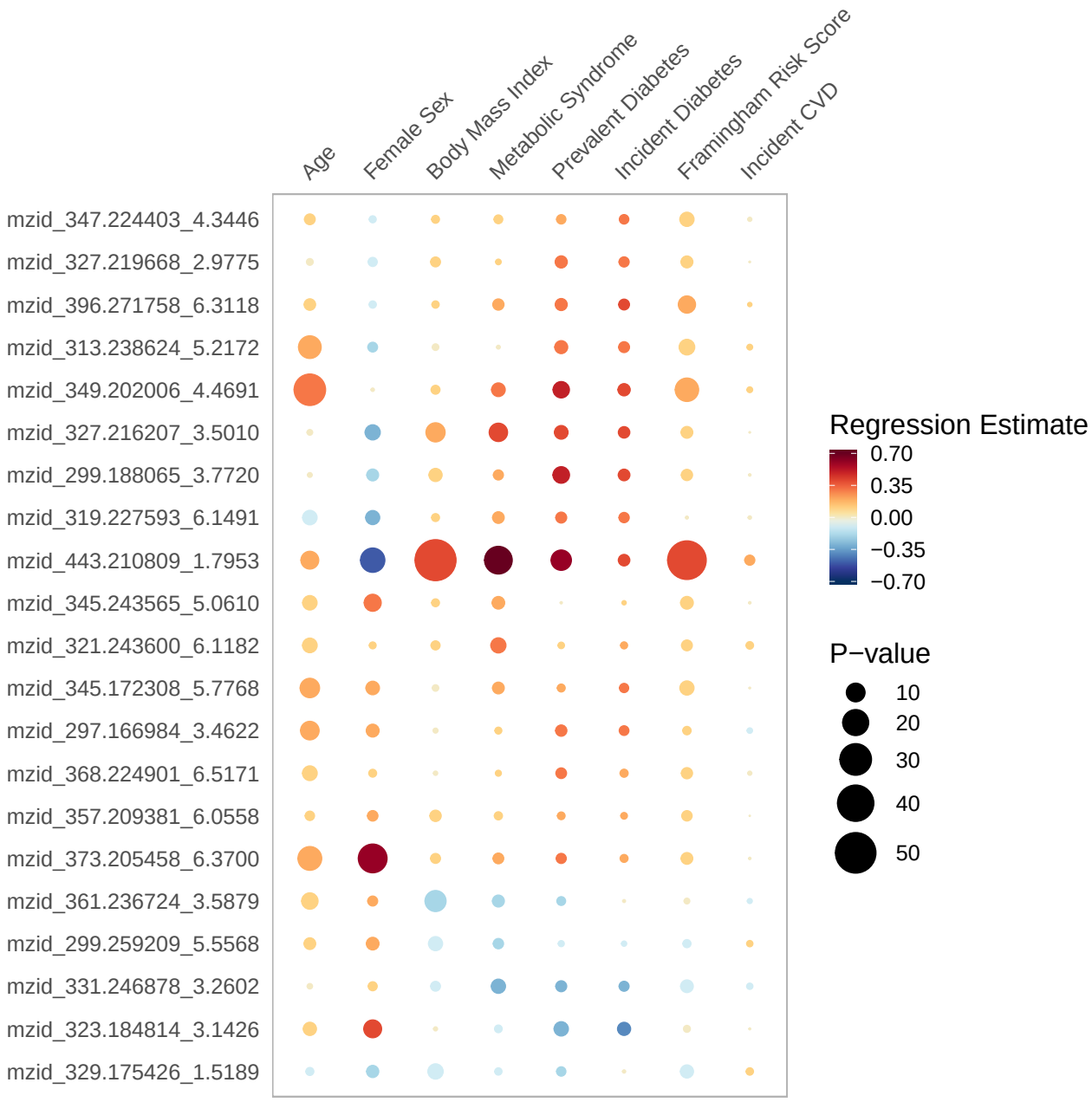
```
  spread(response, estimate)

rnms <-
  cluster_data$term

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

rownames(cluster_data) <- rnms

cluster_data[1:5, 1:5]
```

```
##                         Age Female Sex Body Mass Index Metabolic Syndrome
## mzid_357.209381_6.0558 0.1        0.2             0.1                 0.1
## mzid_327.219668_2.9775 0.0       -0.1             0.1                 0.1
## mzid_368.224901_6.5171 0.1        0.1             0.0                 0.1
## mzid_347.224403_4.3446 0.1       -0.1             0.1                 0.1
## mzid_299.259209_5.5568 0.1        0.2            -0.1                -0.2
##                         Prevalent Diabetes
## mzid_357.209381_6.0558                 0.2
## mzid_327.219668_2.9775                 0.3
## mzid_368.224901_6.5171                 0.3
## mzid_347.224403_4.3446                 0.2
## mzid_299.259209_5.5568                -0.1
```

```
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 <-
  plot_data %>%
  mutate(term = factor(term, levels = term_order))
```

```
rainplot <-
  plot_data %>%
  ggplot(aes(x = response, y = term)) +
  geom_point(aes(colour = estimate, size = p.value)) +
  scale_color_gradientn(
    'Regression Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim, min_lim / 2, 0 , max_lim / 2, max_lim)
  ) +
  scale_size_area('P-value', max_size = 12) +
  scale_x_discrete(position = 'top') +
  thm

rainplot
```




### Adding dendrograms

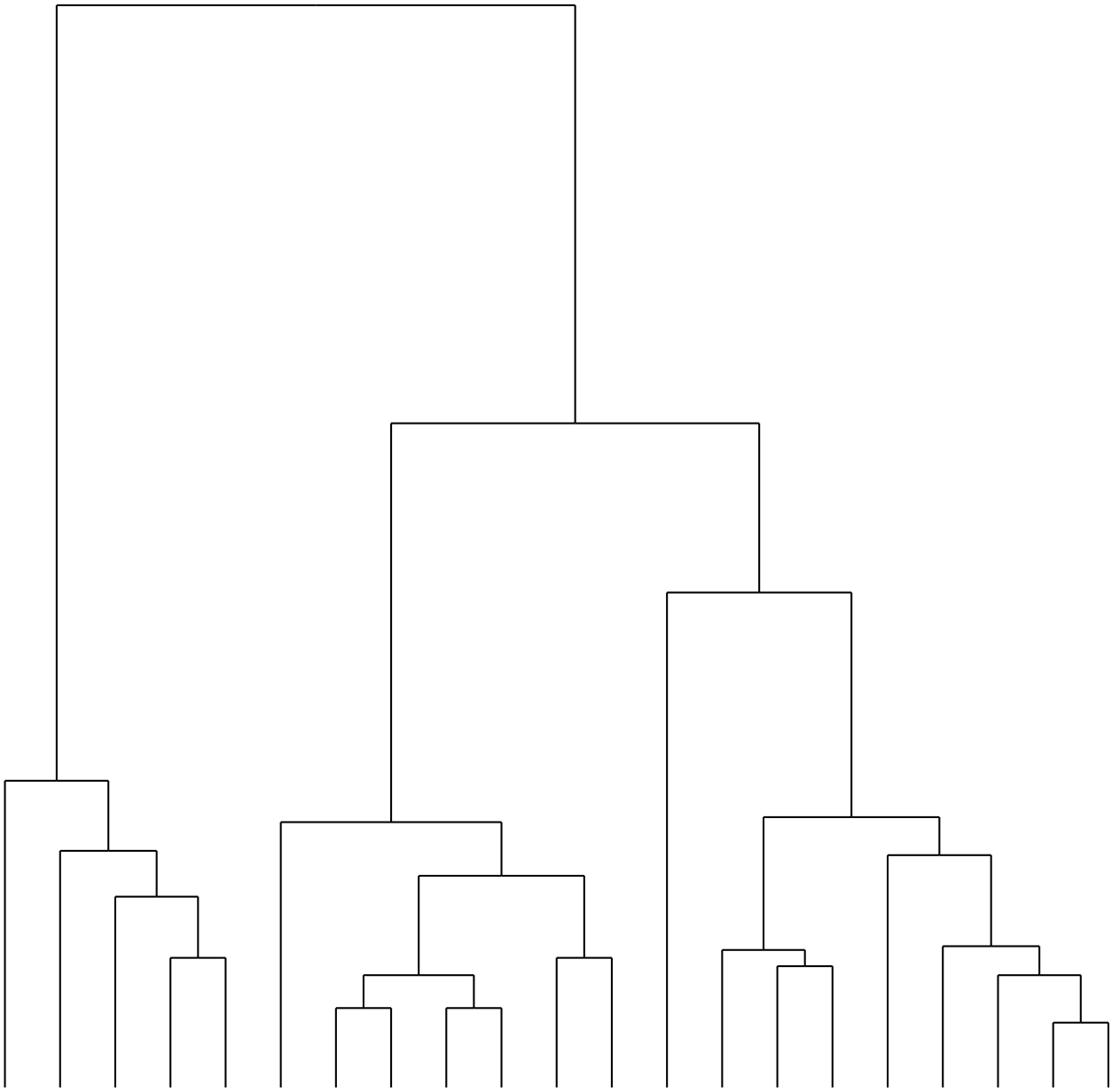
Dendrograms can be added to `ggplot2` plots, but it can be quite complicated. If one wants to add dendrograms to a clustered rainplot, there will be some trial and error to get everything properly aligned. Dendrograms will be created using the `ggdendro` package.

```
library(ggdendro)
```

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

```
# basic dendrogram

dendro <-
```

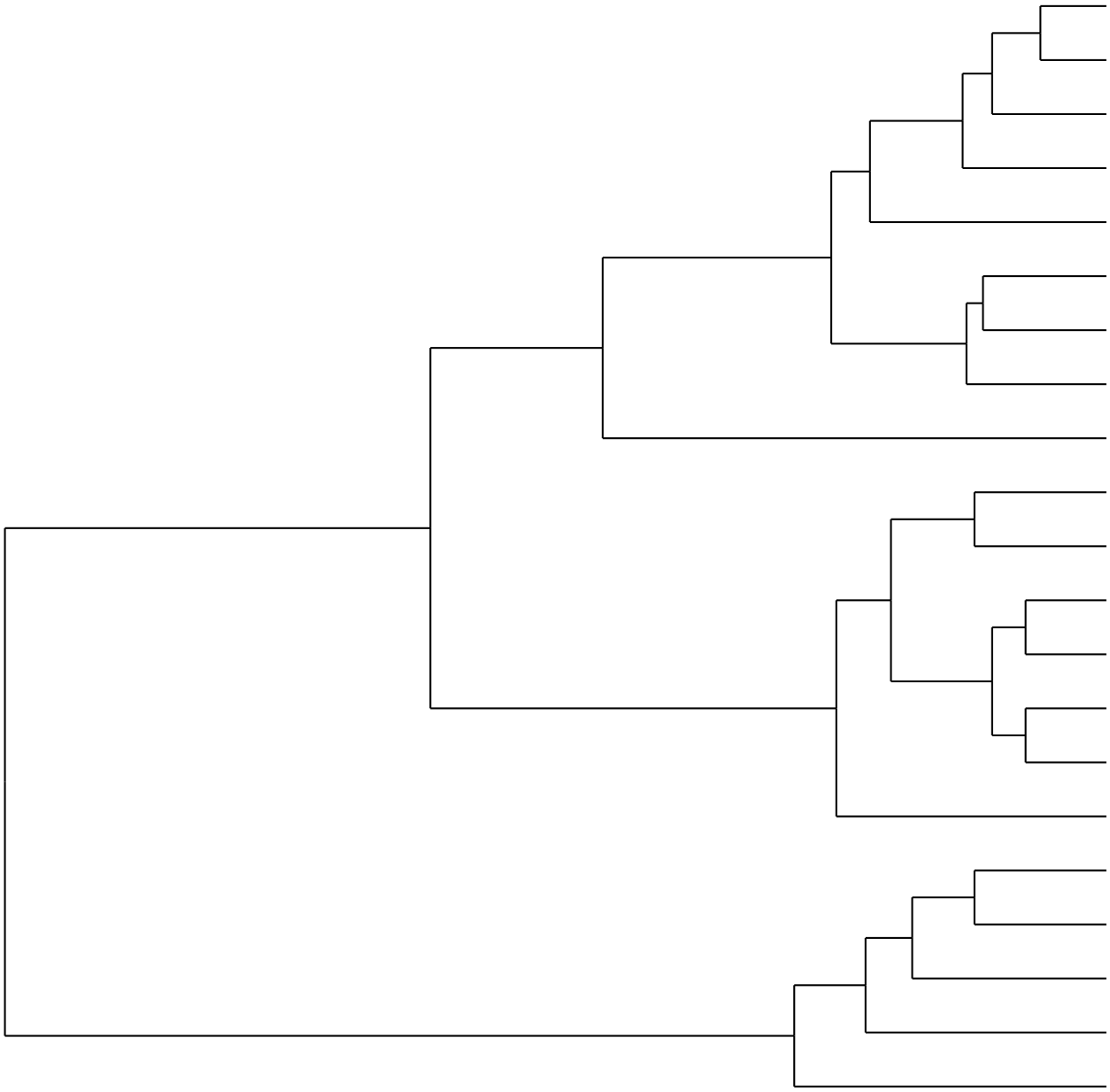
```
  ggplot(dendro_dat) +
  geom_segment(aes(x = x, y = y, xend=xend, yend=yend), colour = 'black') +
  theme_dendro()

dendro
```



The default dendrogram points down. To put the dendrogram on the left of our rain plot, we want it to point to the right.

```
dendro <-
  dendro +
  scale_y_reverse() +
  coord_flip()
```

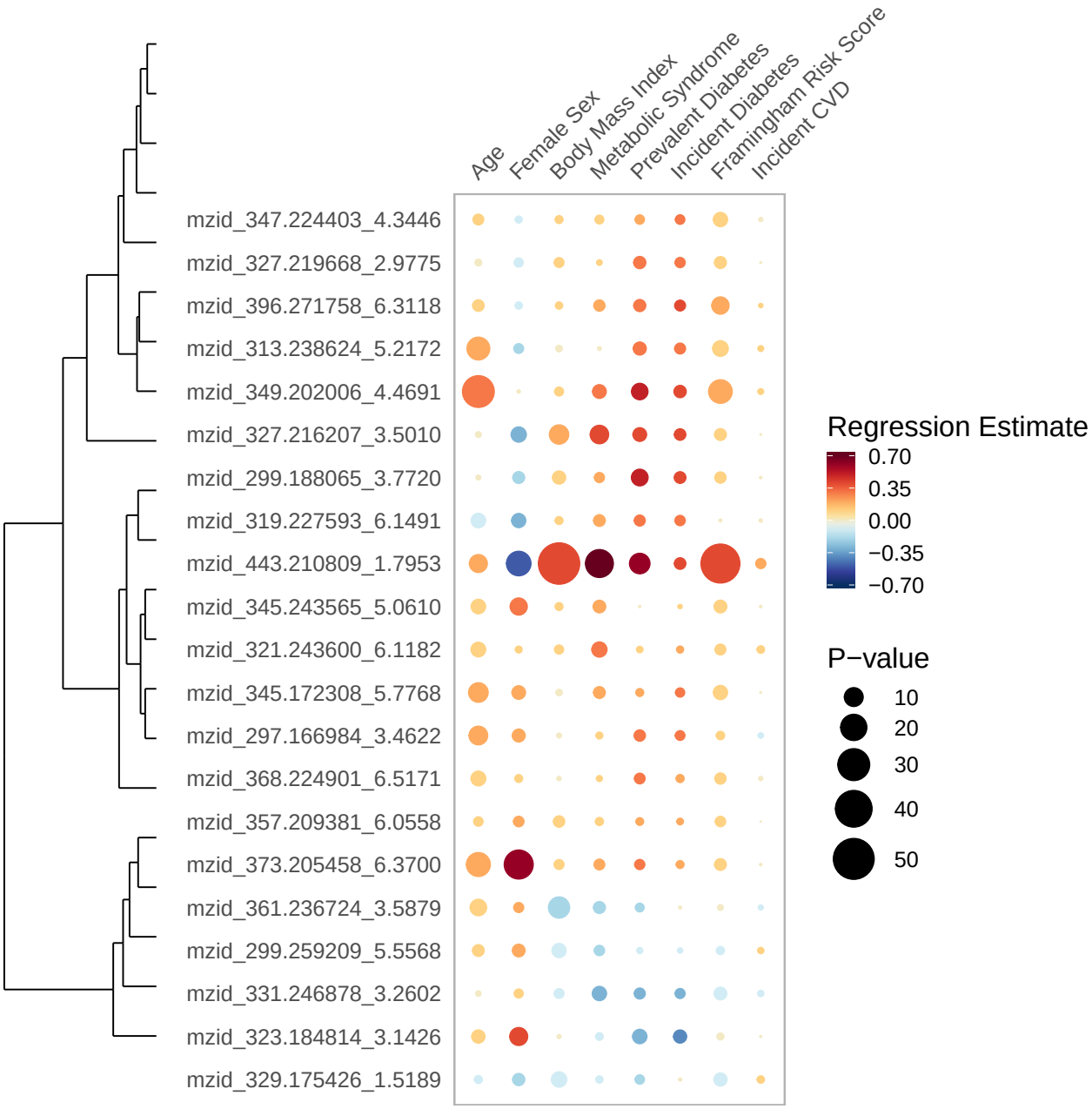
```
dendro
```



We will use the `gridExtra` package to display our plots together. Though our dendrogram looks okay, there will be alignment issues when we first try to juxtapose the plots.

```
library(gridExtra)
```

```
grid.arrange(dendro, rainplot, ncol = 2, widths = c(3, 15))
```



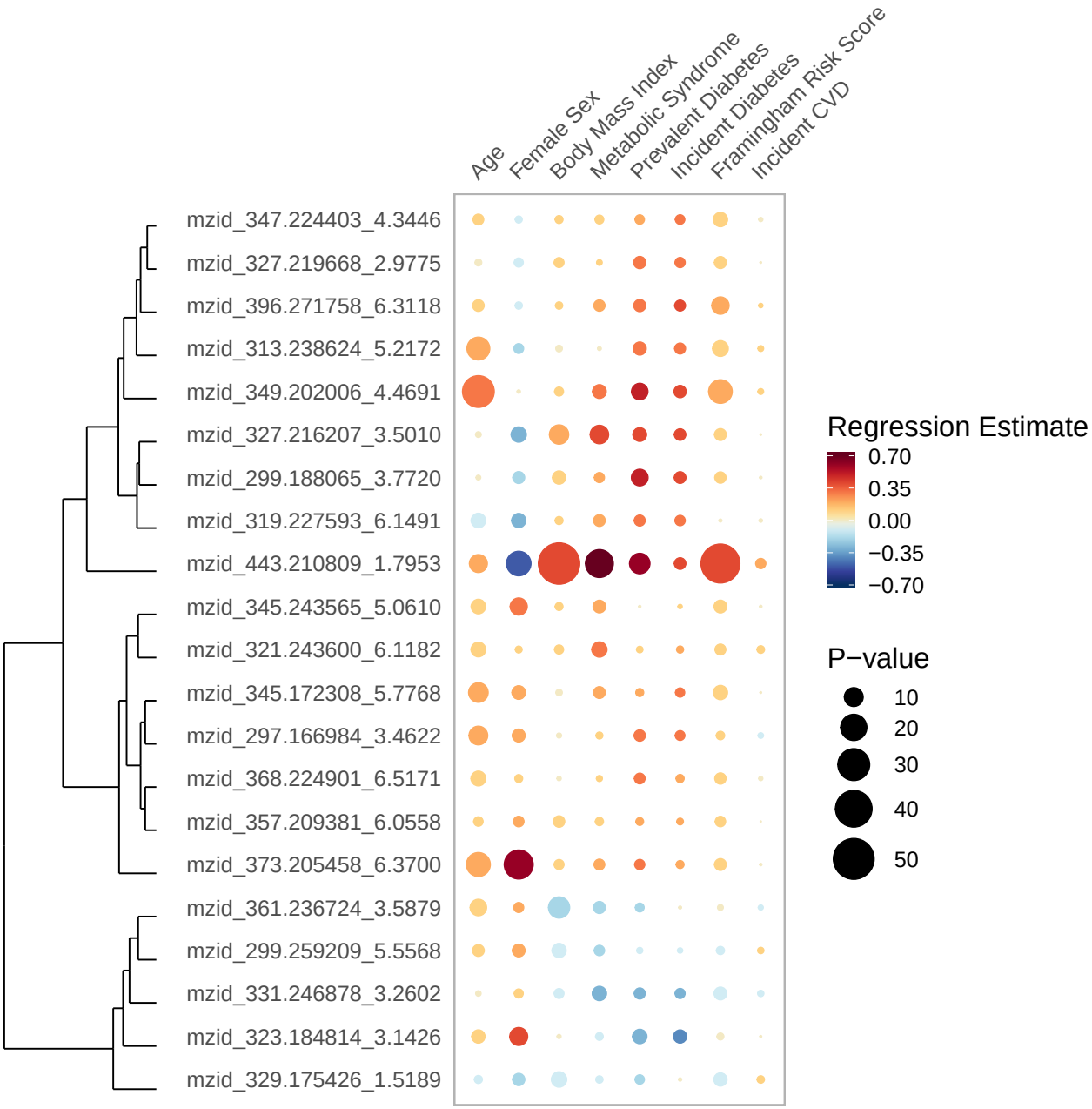

As we can see, the dendrogram is not aligned with the y-axis labels. This is due to the x-axis labels that are present in the rainplot, but not in the dendrogram. The lack of labels allows the top of the dendrogram to go all the way to the top of the plot area, causing misalignment. To get around this, we will use a hack. First, we will reduce the margin between the dendrogram, and the plot boundary. Second, We will add an x-axis label to the dendrogram, and make it invisible by matching the color of the label text to the background. By adjusting the length of this invisible label, we can make the dendrogram align with the rainplot.

```
# Reduce the margin between dendrogram and plot boundary
dendro <-
  dendro +
  scale_x_continuous(expand = c(0, 0.5))

# match label to background color, rotate 90 degrees so text points vertical
thme_invisible_text<- theme(
```

```
  axis.text.x = element_text(angle = 90, hjust = 0, colour = 'white'),
)

# Create a spacing label to align the dendrogram with the text label. The length
# of `labels` that works best will be discovered through trial and error. Add or
# remove '0' until the dendrogram is aligned.
dendro <-
  dendro +
  scale_y_reverse(breaks = 0, labels = '0000000000000000000000', position = 'right') +
  thme_invisible_text
```

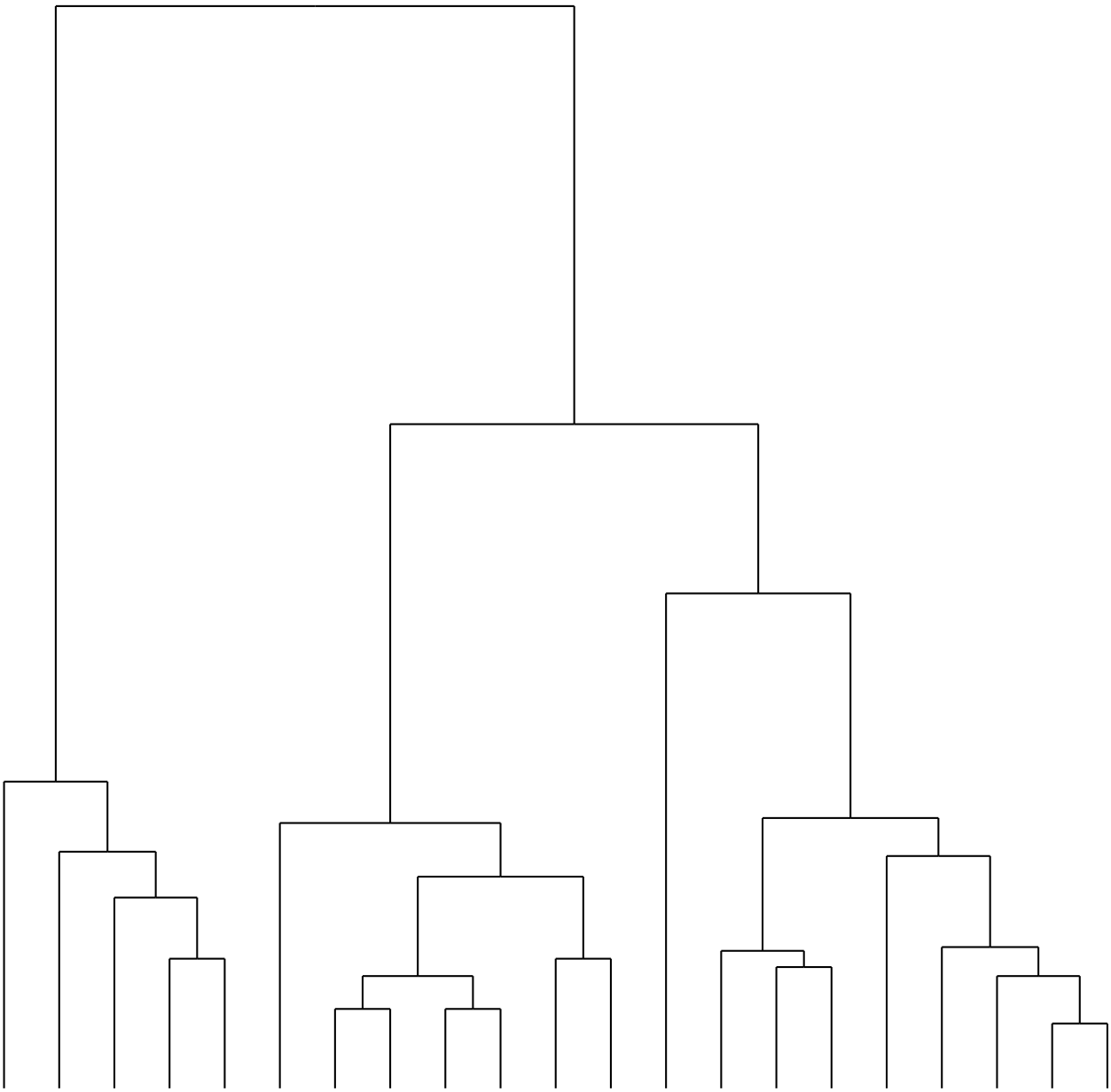
```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```

```
grid.arrange(dendro, rainplot, ncol = 2, widths = c(3, 15))
```



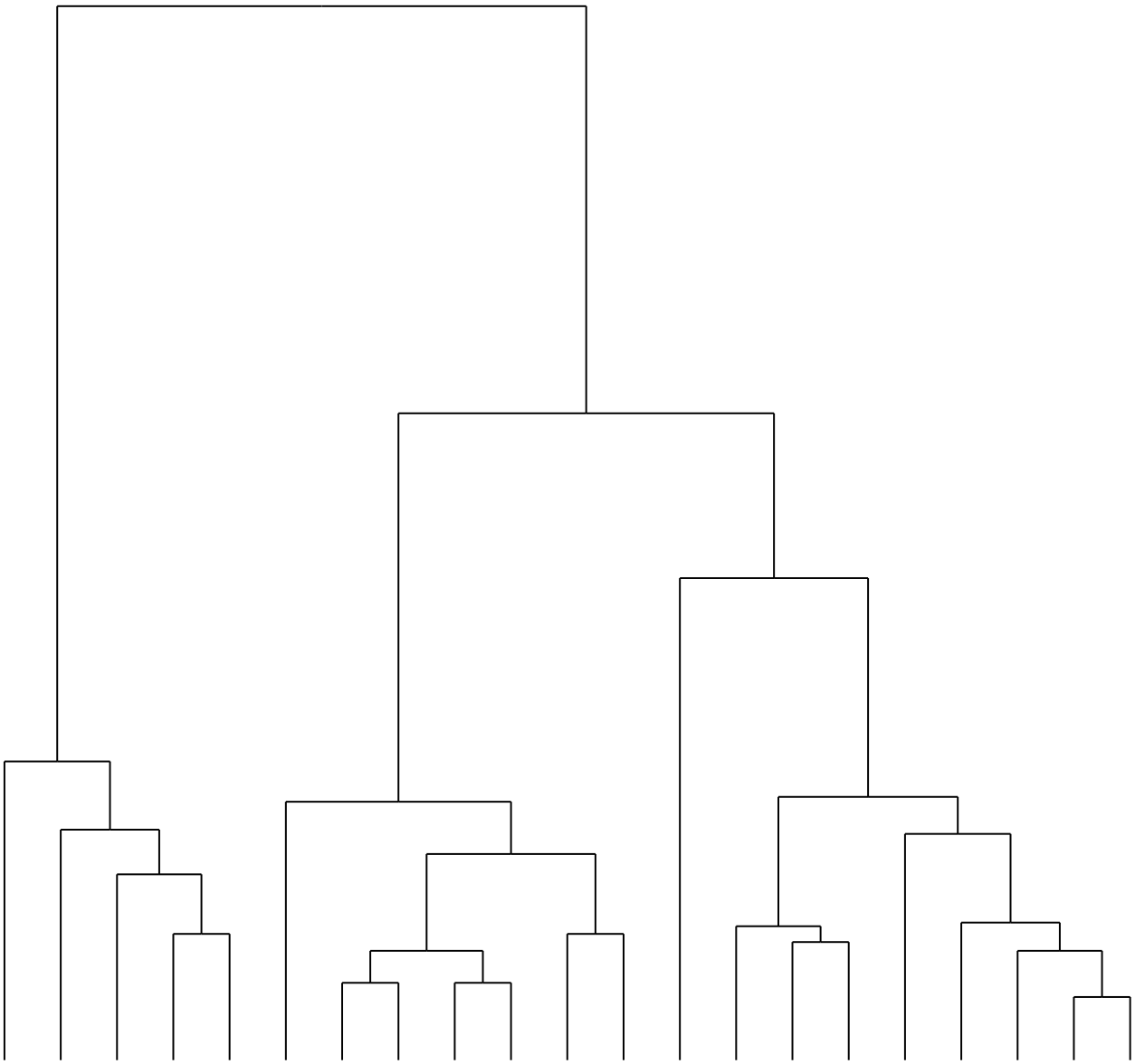

To better understand how the dendrogram was transformed, lets walk through it step by step.

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

dendro
```



As we can see, there is a margin between the edges of the plot and the outermost parts of the dendrogram. We first want to reduce that margin on the x-axis, but not totally eliminate it. When the `coord_flip` is applied, this will translate to a reduced margin on the y-axis. This will allow the bottom of the dendrogram to align with the bottom-most metabolite label, which is close to the bottom edge of the plot.

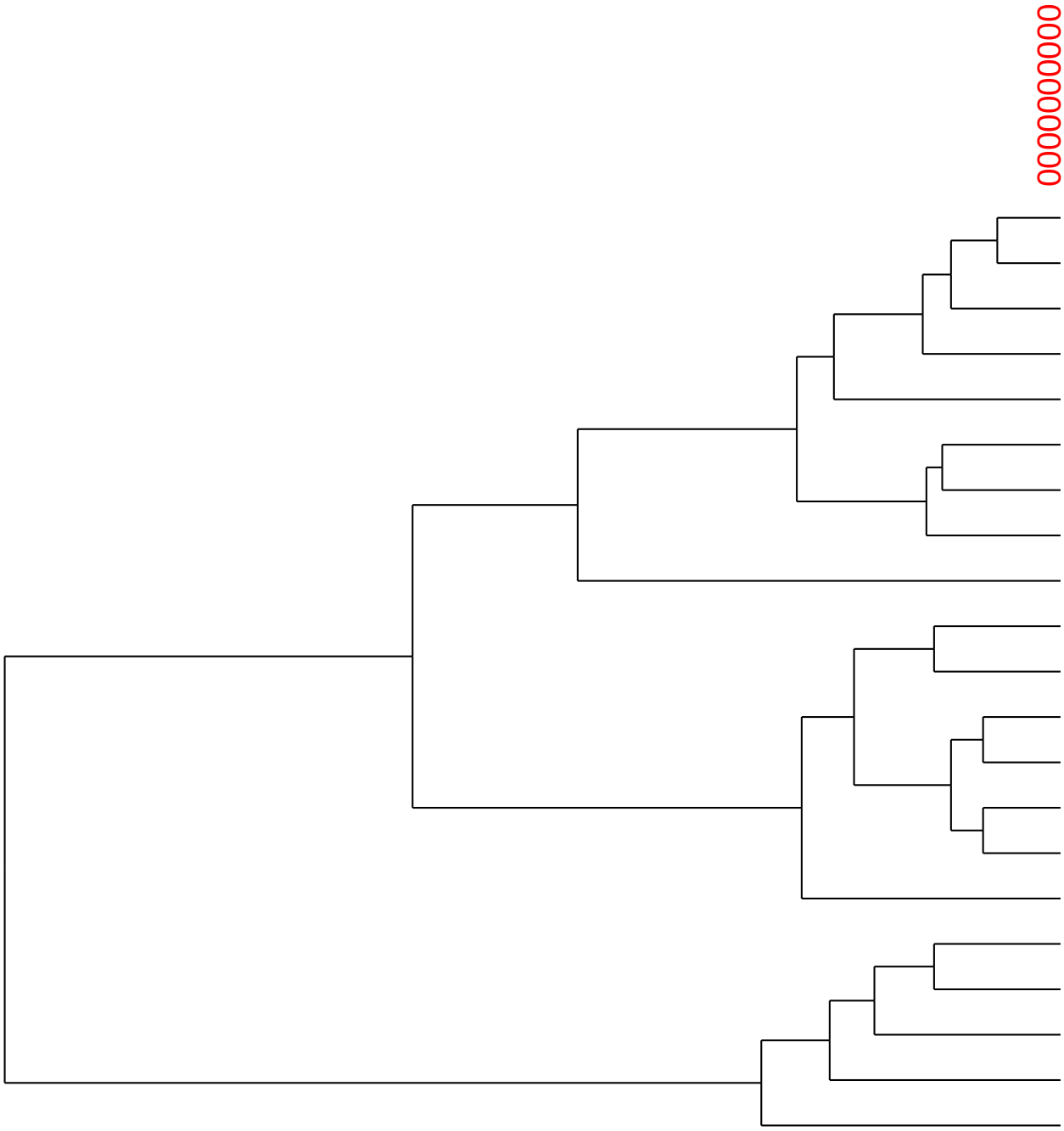
```
# Adjust the `expand` arguments to help the dendrogram align with the bottom
# metabolite label. I have found c(0, 0.5) to be a good general default.

dendro <-
  dendro +
  scale_x_continuous(expand = c(0, 0.5))

dendro
```



Next we want to add our spacing label. Normally, the text would match the color of the background, to provide an invisible effect. For this example, the label text will be visible to show how it affects the dendrogram. We use `scale_y_reverse` so that when `coord_flip` is applied, the dendrogram points right, not left, and `position = 'right'` so that the label, when rotated, is on top, not on bottom.

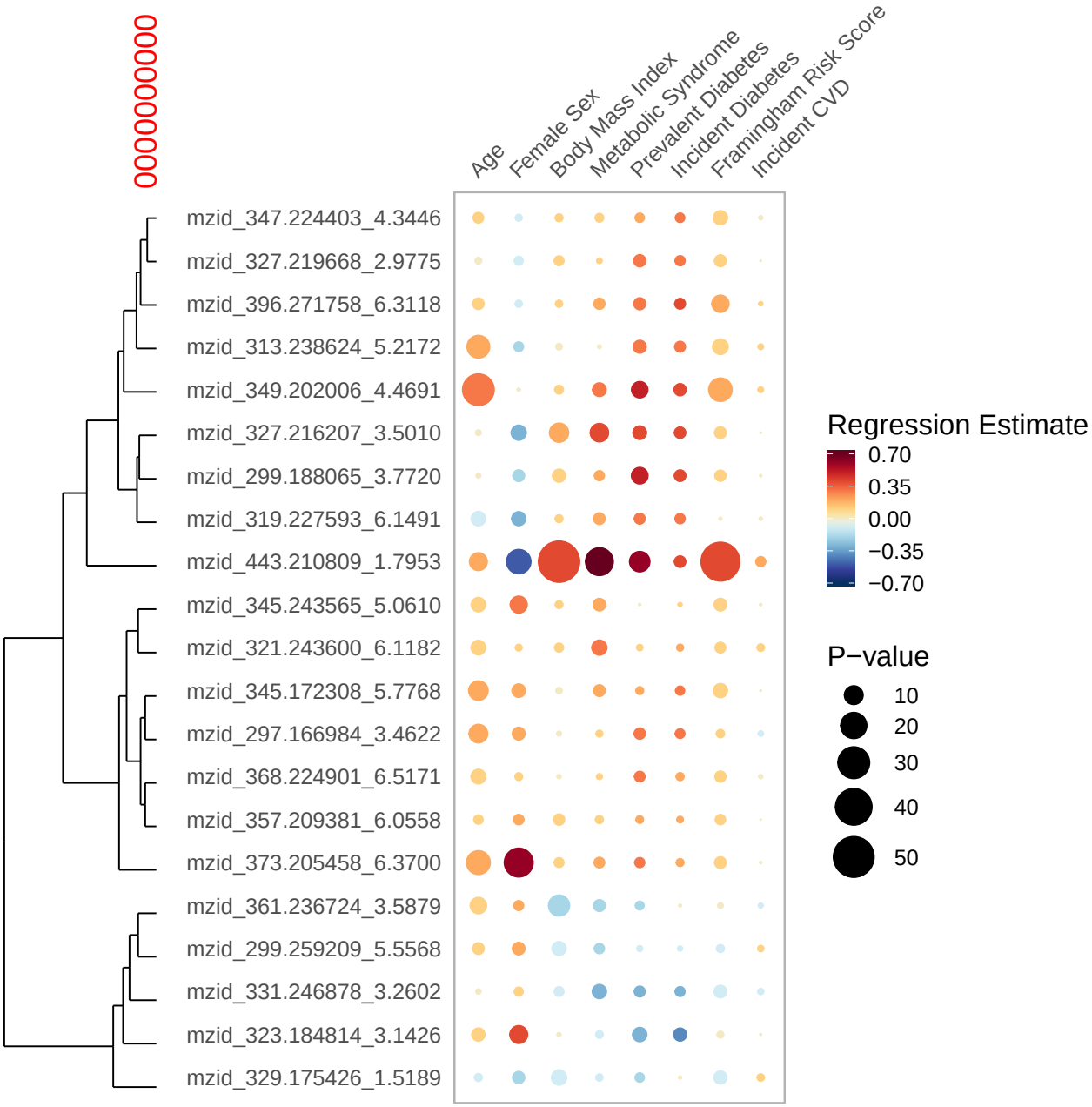
```
# Rotate text 90 degrees so text points perpendicular to axis. Text color is red
# to visualize effect.
thme_text <- theme(
  axis.text.x = element_text(angle = 90, hjust = 0, colour = 'red', size = 20),
)


# Adjust the length of `labels` to generally align the dendrogram and the rain
# plot. It can be any text you want. I just happened to use 0s.
dendro <-
  dendro +
  scale_y_reverse(breaks = 0, labels = '0000000000', position = 'right') +
```

```
  thme_text

dendro <-
  dendro + coord_flip()

dendro
```




When combined with the rainplot, we get:

```
grid.arrange(dendro, rainplot, ncol = 2, widths = c(3, 15))
```


0000000000
Age
Female Sex
Body Mass Index
Metabolic Syndrome
Prevalent Diabetes
Incident Diabetes
Framingham Risk Score
Incident CVD
mzid_347.224403_4.3446
mzid_327.219668_2.9775
mzid_396.271758_6.3118
mzid_313.238624_5.2172
mzid_349.202006_4.4691
mzid_327.216207_3.5010
mzid_299.188065_3.7720
mzid_319.227593_6.1491
mzid_443.210809_1.7953
mzid_345.243565_5.0610
mzid_321.243600_6.1182
mzid_345.172308_5.7768
mzid_297.166984_3.4622
mzid_368.224901_6.5171
mzid_357.209381_6.0558
mzid_373.205458_6.3700
mzid_361.236724_3.5879
mzid_299.259209_5.5568
mzid_331.246878_3.2602
mzid_323.184814_3.1426
mzid_329.175426_1.5189
Regression Estimate
0.70
0.35
0.00
−0.35
−0.70
P−value
10
20
30
40
50