Rainplots Tutorial

Mir Henglin

June 25, 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 correspond to
- The dependent variable term
- The P-value
- The regression 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       mzid_443.210809_1.7953      0.4 5.74e-52
##  2 Framingham Risk Score mzid_443.210809_1.7953      0.4 4.80e-46
##  3 Age                   mzid_349.202006_4.4691      0.3 1.87e-30
##  4 Female Sex            mzid_373.205458_6.3700      0.6 2.47e-25
##  5 Metabolic Syndrome    mzid_443.210809_1.7953      0.7 1.78e-23
##  6 Female Sex            mzid_443.210809_1.7953     -0.5 3.92e-18
##  7 Age                   mzid_373.205458_6.3700      0.2 2.96e-17
##  8 Framingham Risk Score mzid_349.202006_4.4691      0.2 8.91e-17
##  9 Age                   mzid_313.238624_5.2172      0.2 1.02e-15
## 10 Body Mass Index       mzid_361.236724_3.5879     -0.2 1.47e-13
## # ... 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, 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 other control variables included in the model and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. `estimate` is the estimated value of β_1 and `p.value` is the corresponding P-value.

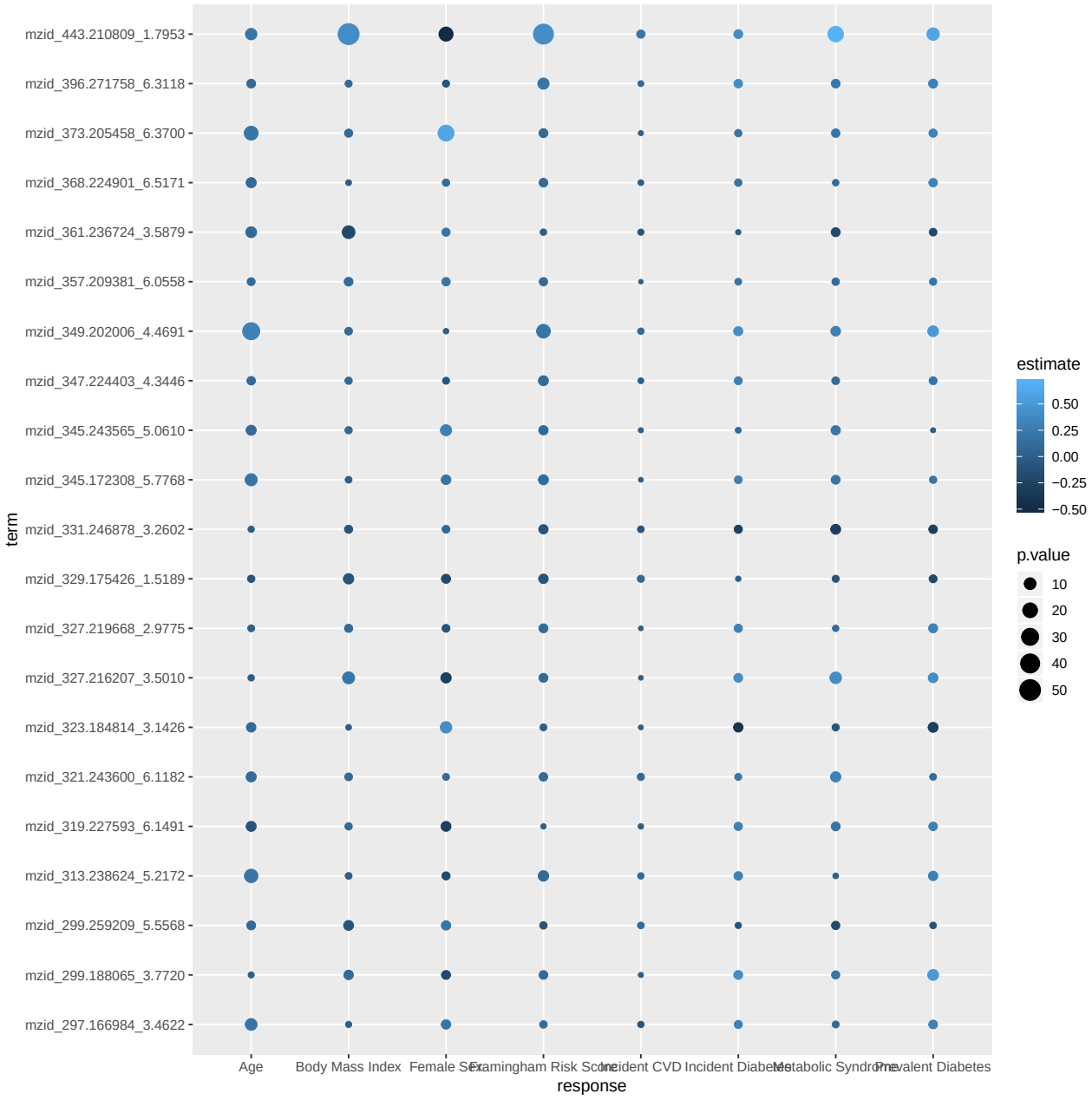
Before plotting, we will first transform `p.value` onto the negative-log-scale. This will allow the smallest P-values, which are often of greater interest, to have the largest values when plotted.

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

Plotting

A basic rainplot can be constructed in only two lines of `ggplot2` code!

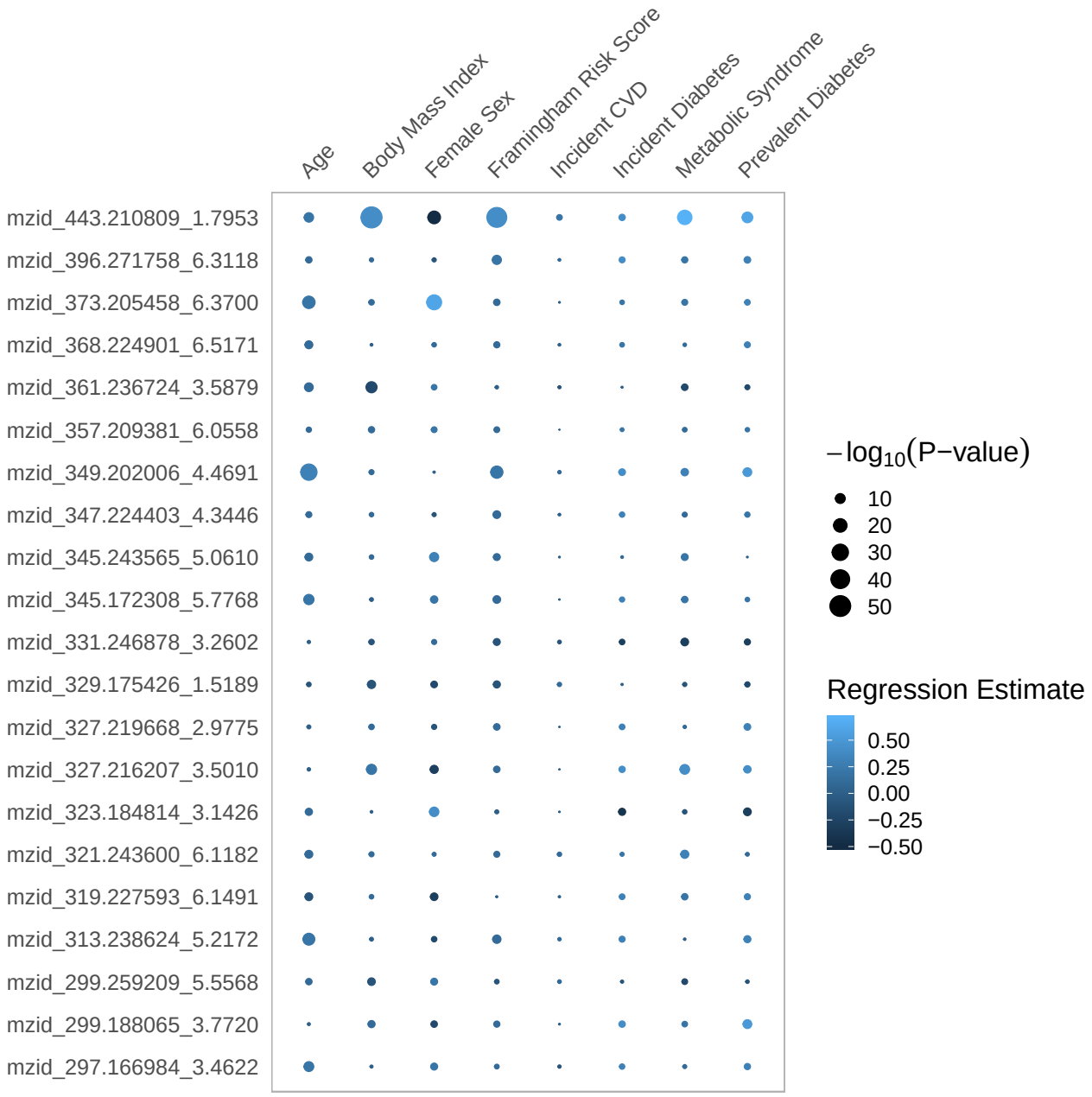
```
rainplot <-
  ggplot(plot_data) +
  geom_point(aes(x = response, y = term, colour = estimate, size = p.value))

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`.

```
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(expression(paste(-log[10]('P-value')))) +
  scale_color_continuous('Regression Estimate')

rainplot
```



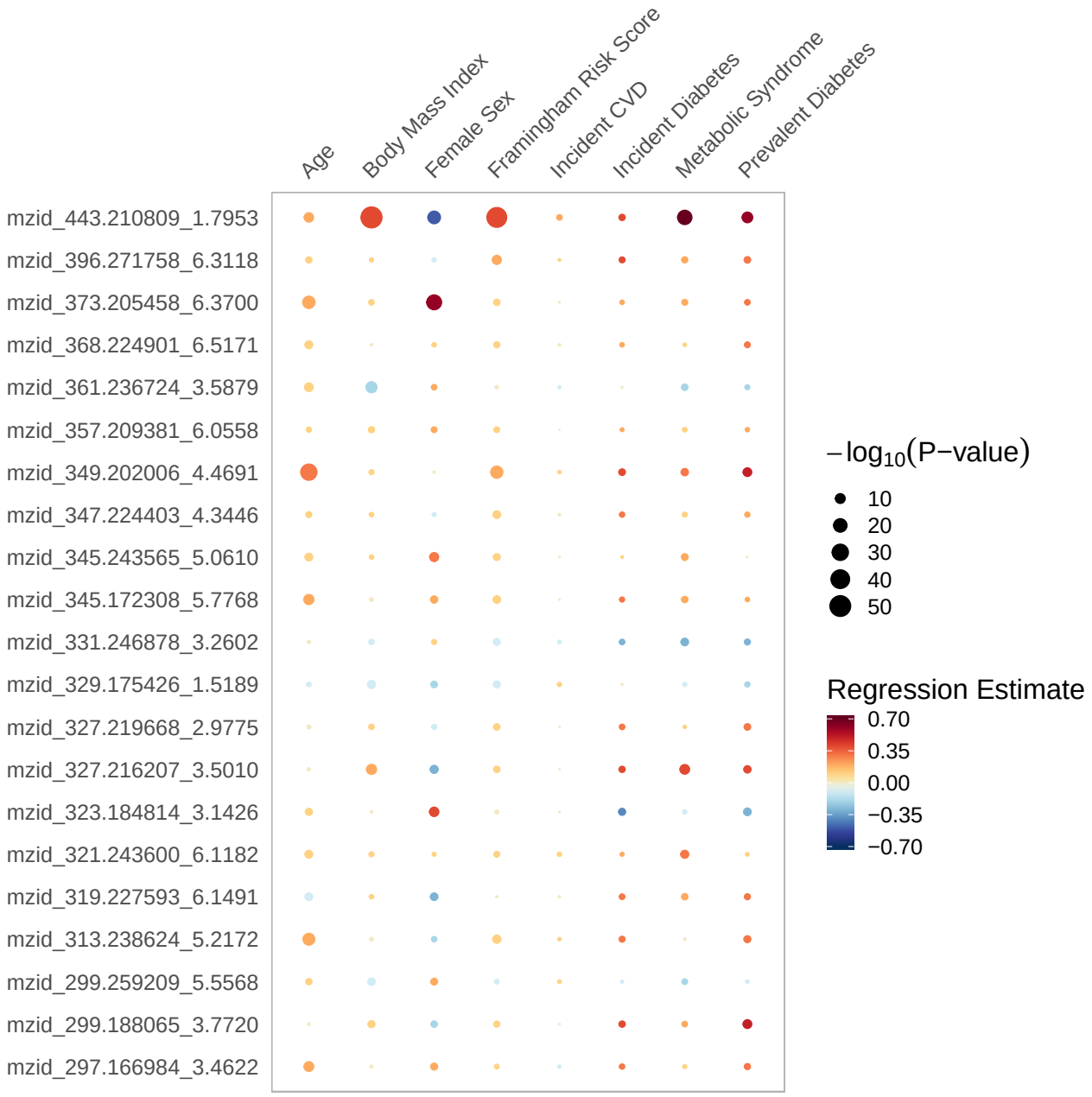
To make the presentation of the regression estimates clearer, we create a diverging color scale, and set positive and negative limits equidistant from 0.

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

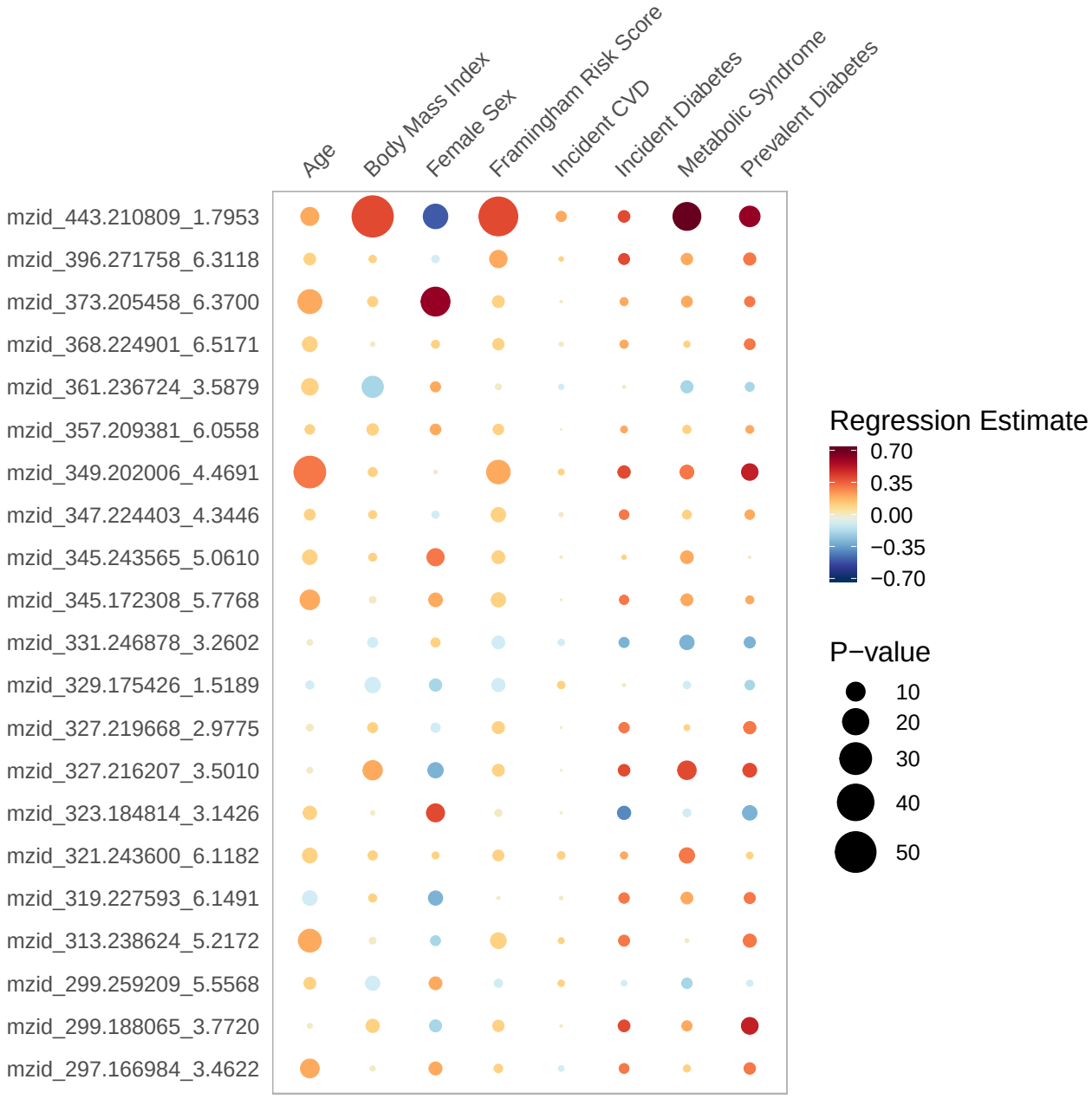
```
rainplot
```



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.
```



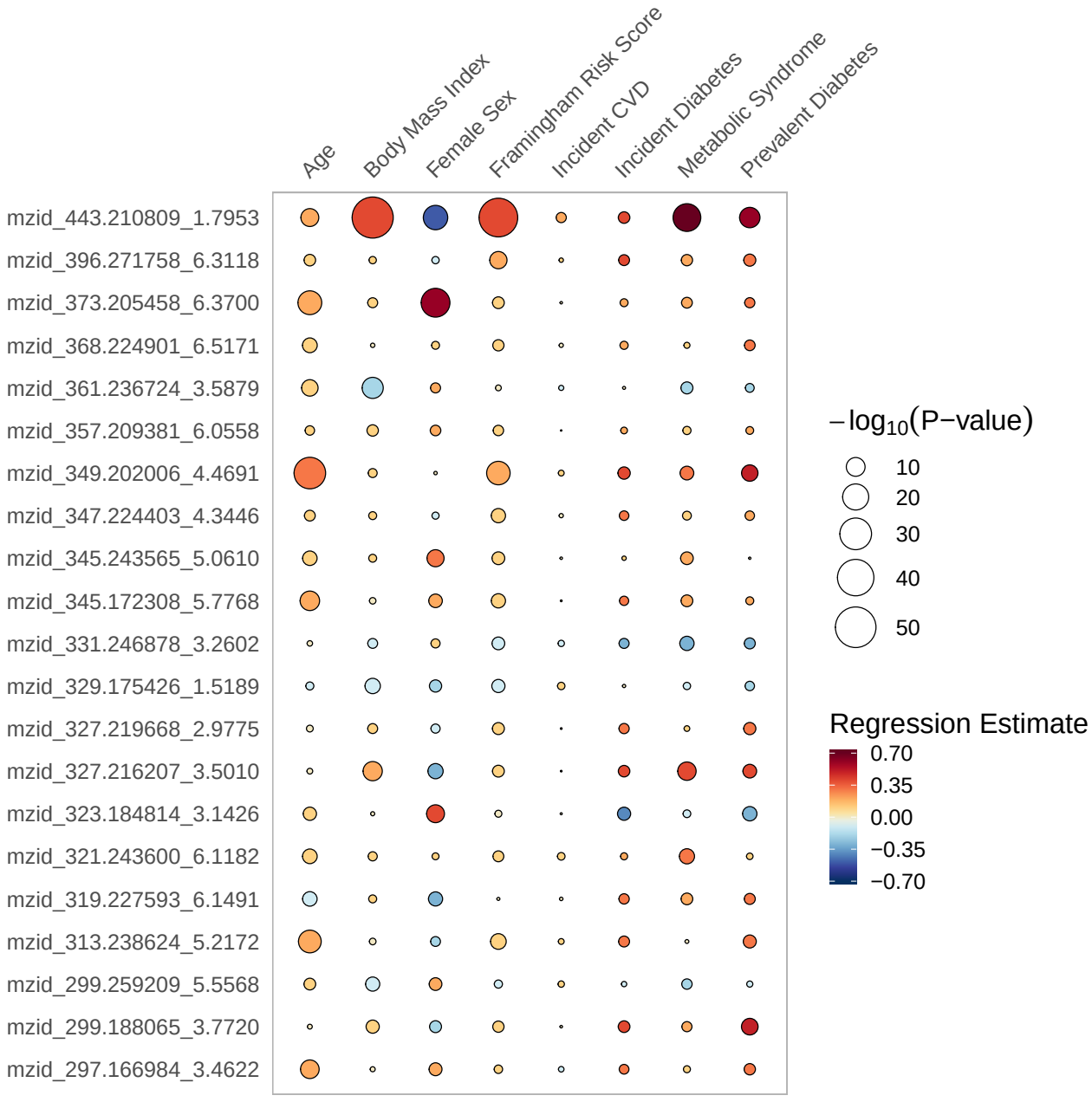
Additional Plot Adjustments

Outlines

The points on rainplots can be outlined in a color different from the color of the point. To get such a shape, we add the argument `shape = 21` to `geom_point`. Note that when we do this, the color of the point changes from `color` to `fill`.

```
rainplot <-
  ggplot(plot_data) +
  geom_point(aes(x = response, y = term, fill = estimate, size = p.value),
             shape = 21) +
  scale_fill_gradientn(
```

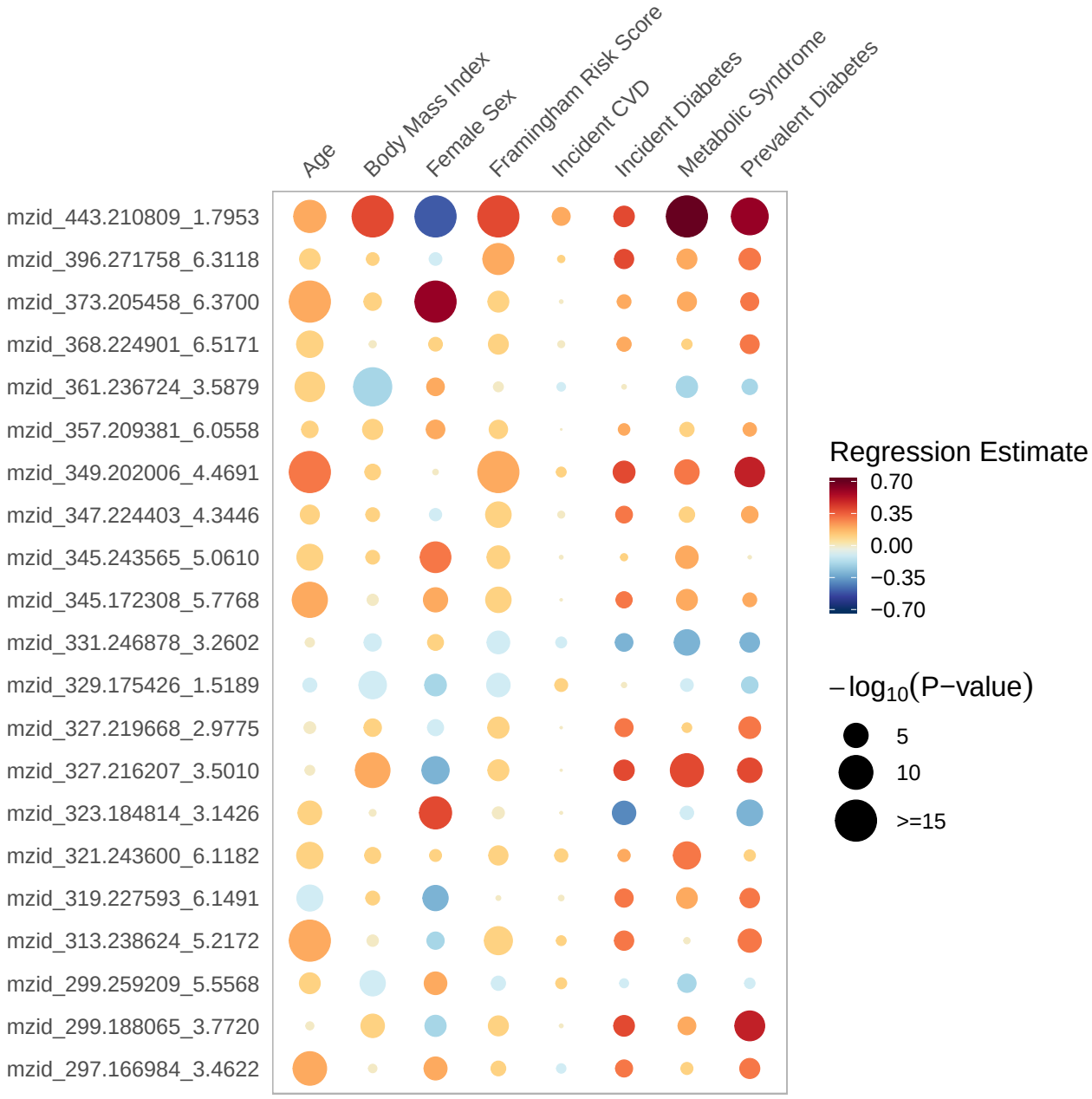
```
    'Regression Estimate',
    colors = palette,
    limits = c(min_lim, max_lim),
    breaks = c(min_lim, min_lim / 2, 0 , max_lim / 2, max_lim)
  ) +
  scale_size_area(expression(paste(-log[10]('P-value'))), max_size = 12) +
  scale_x_discrete(position = 'top') +
  thm

rainplot
```



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))

rainplot <-
  ggplot(plot_data_thresholded) +
  geom_point(aes(x = response, y = term, 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(
    expression(paste(-log[10]('P-value'))),
    max_size = 12,
    breaks = c(5, 10, 15),
    labels = c('5', '10', '>=15')) +
  scale_x_discrete(position = 'top') +
  thm

rainplot
```



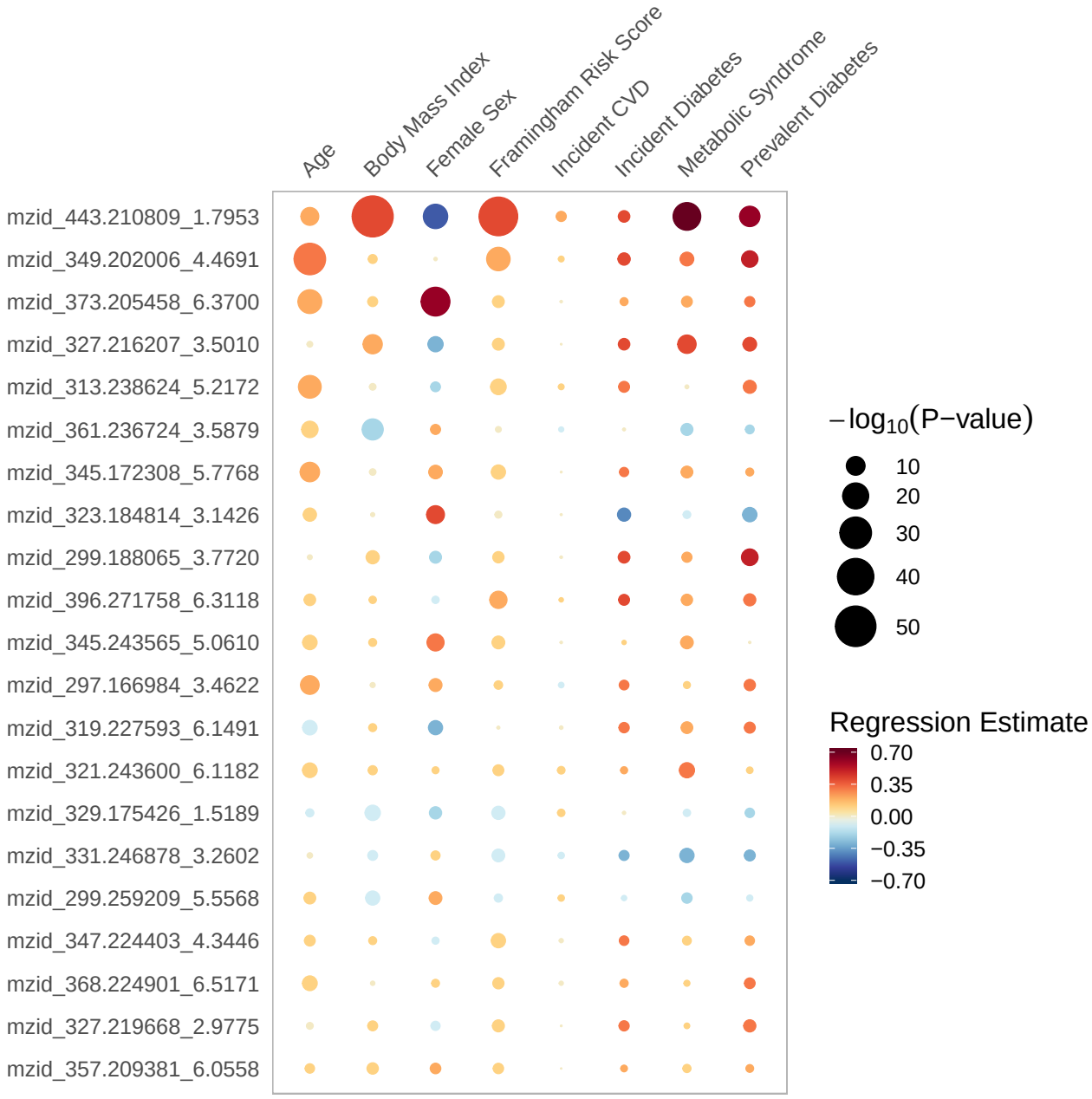
Ordering by P-Value

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 <-
  ggplot(plot_data) +
  geom_point(aes(x = response, y = term, 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(expression(paste(-log[10]('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) %>%
```

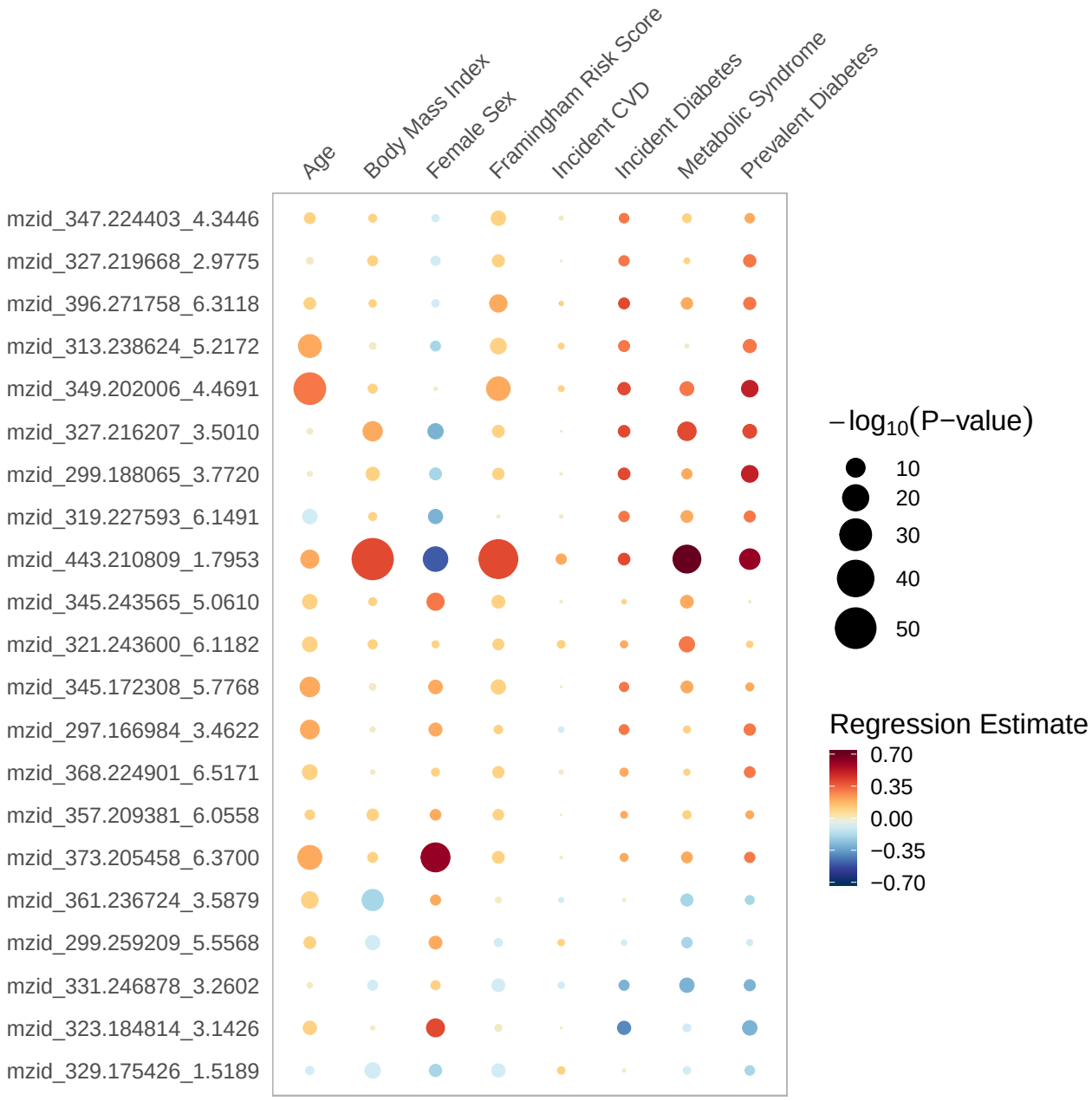
```
  spread(response, estimate)

rnms <-
  cluster_data$term

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

rownames(cluster_data) <- rnms

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

```
##                         Age Body Mass Index Female Sex
## mzid_357.209381_6.0558 0.1             0.1        0.2
## mzid_327.219668_2.9775 0.0             0.1       -0.1
## mzid_368.224901_6.5171 0.1             0.0        0.1
## mzid_347.224403_4.3446 0.1             0.1       -0.1
## mzid_299.259209_5.5568 0.1            -0.1        0.2
##                         Framingham Risk Score Incident CVD
## mzid_357.209381_6.0558                    0.1          0.0
## mzid_327.219668_2.9775                    0.1          0.0
## mzid_368.224901_6.5171                    0.1          0.0
## mzid_347.224403_4.3446                    0.1          0.0
## mzid_299.259209_5.5568                   -0.1          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 <-
  ggplot(plot_data) +
  geom_point(aes(x = response, y = term, 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(expression(paste(-log[10]('P-value'))), max_size = 12) +
  scale_x_discrete(position = 'top') +
  thm

rainplot
```



Adding dendrograms

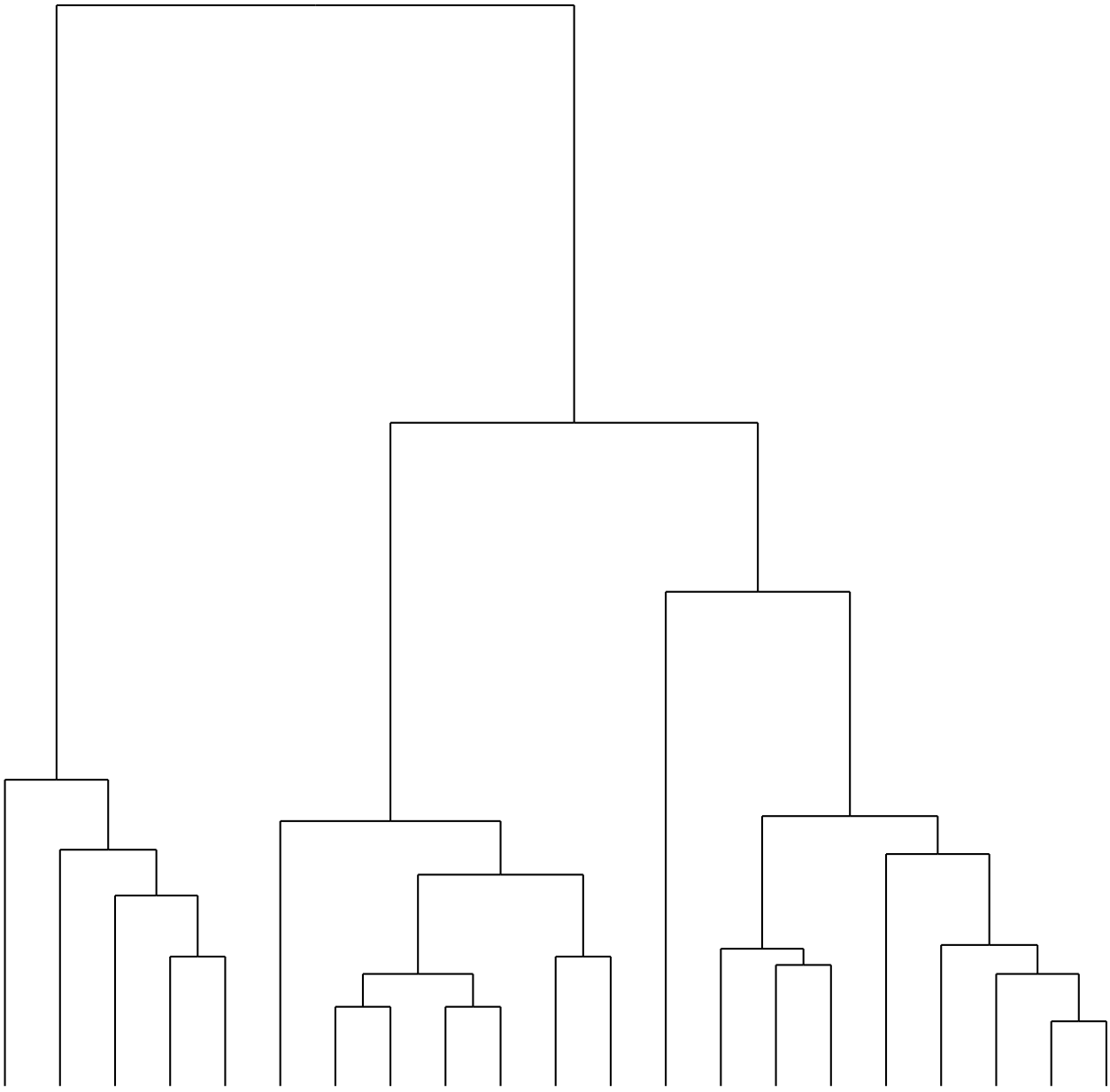
Dendrograms can be added to `ggplot2` plots, but it can be quite complicated. If one wants to add dendrograms to a clustered rainplot, there will be some trial and error to get everything properly aligned. Dendrograms will be created using the `ggdendro` package.

```
library(ggdendro)
```

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

```
# basic dendrogram

dendro <-
```

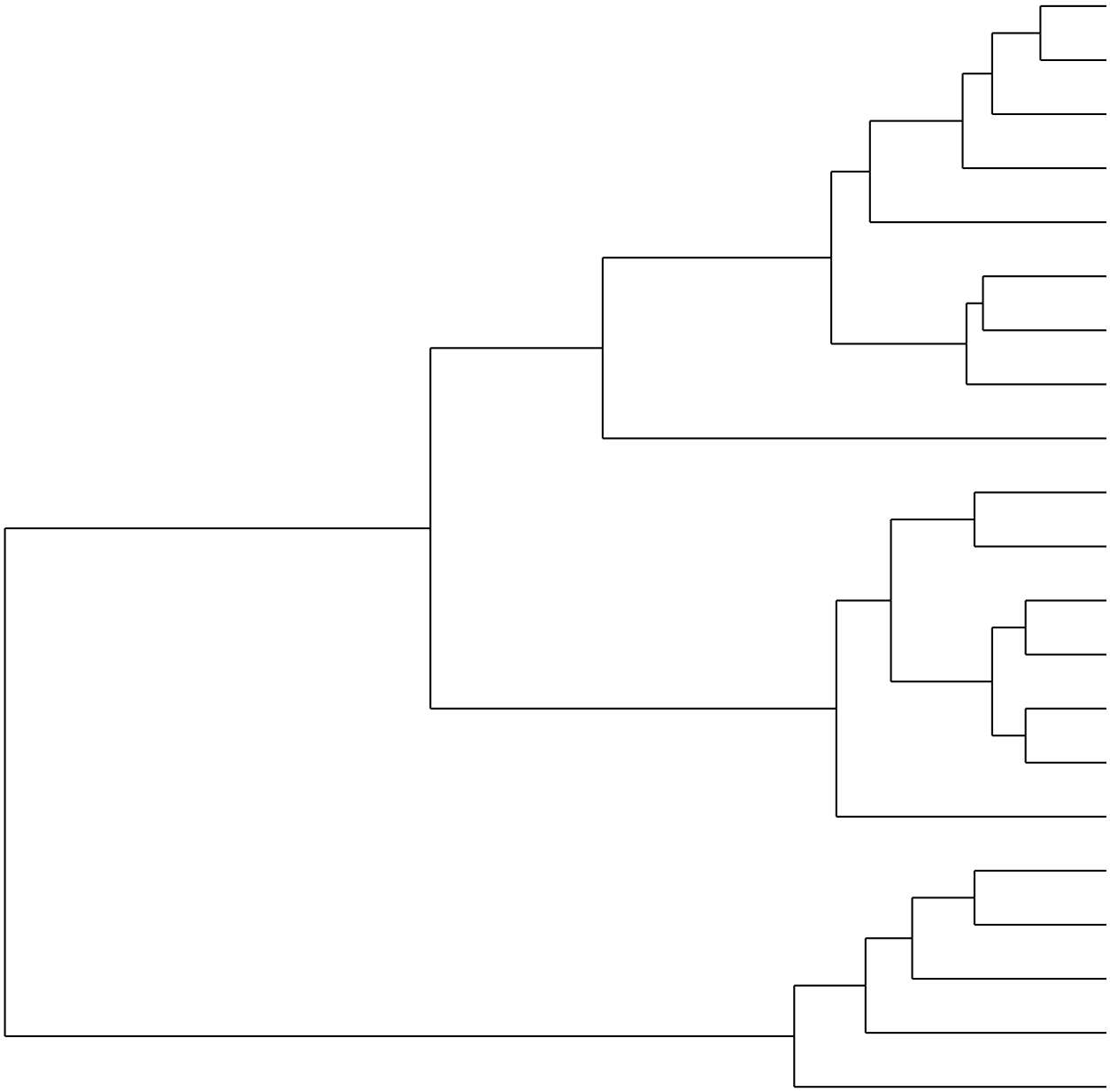
```
  ggplot(dendro_dat) +
  geom_segment(aes(x = x, y = y, xend=xend, yend=yend), colour = 'black') +
  theme_dendro()

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. 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') +
  theme_dendro()
```

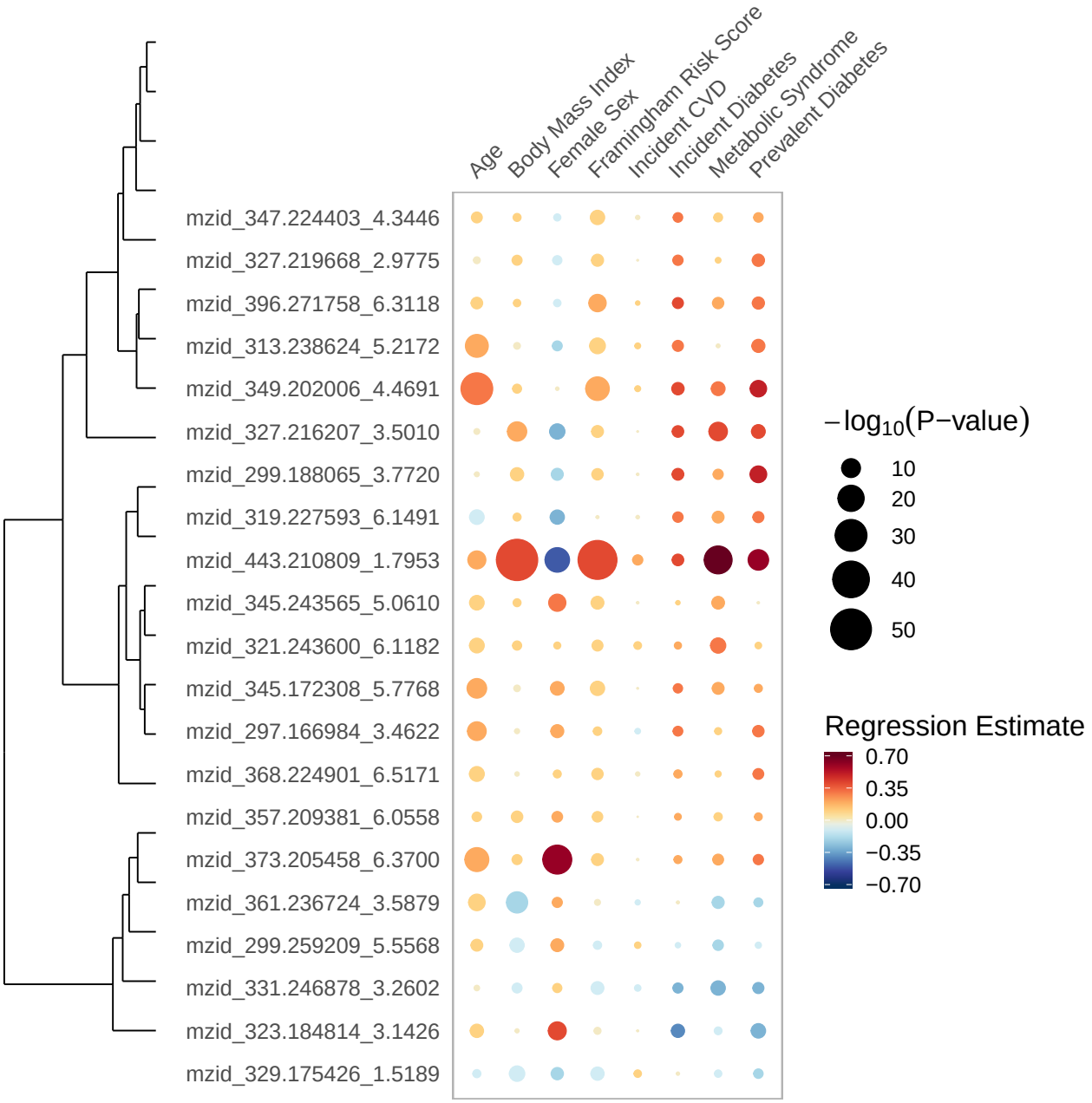
```
dendro
```



We will use the `gridExtra` package to display our plots together. Though our dendrogram looks okay, there will be alignment issues when we first plot the dendrogram and rainplot side by side.

```
library(gridExtra)
```

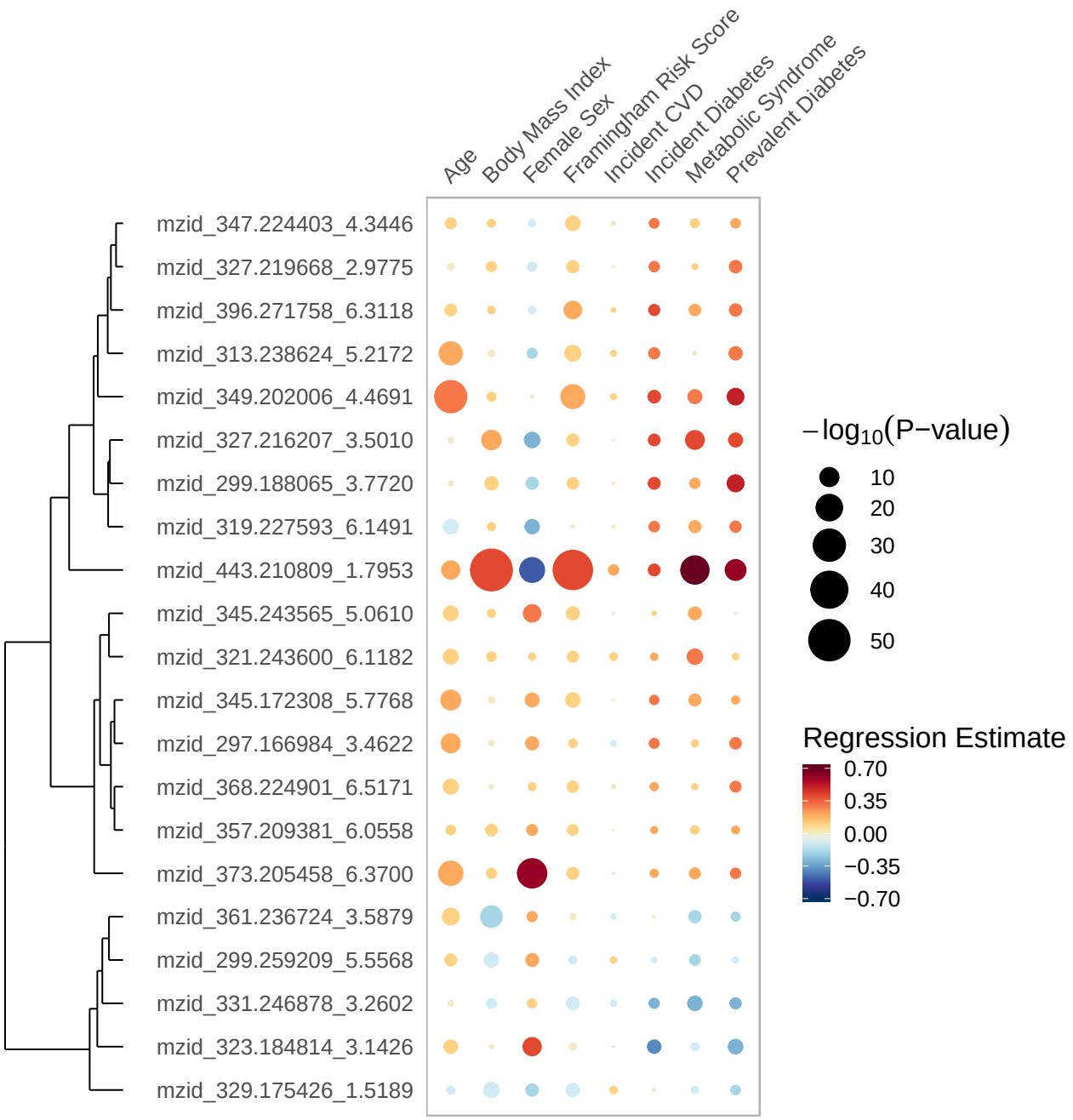
```
grid.arrange(dendro, rainplot, ncol = 2, widths = c(3, 15))
```



As we can see, the dendrogram is not aligned with the rainplot labels. To get around this, we will use a hack. First, we will plot a modified version of the rainplot, selecting only the column with the longest label, under the dendrogram. This will align the plot area between the dendrogram and the rainplot. Second, we will make the modified rainplot invisible, so that the dendrogram appears on its own.

```
x_labels <-
  plot_data$response %>%
  unique()

longest_x_label <-
  x_labels[[which.max(nchar(x_labels))]]


modified_rainplot_data <-
```

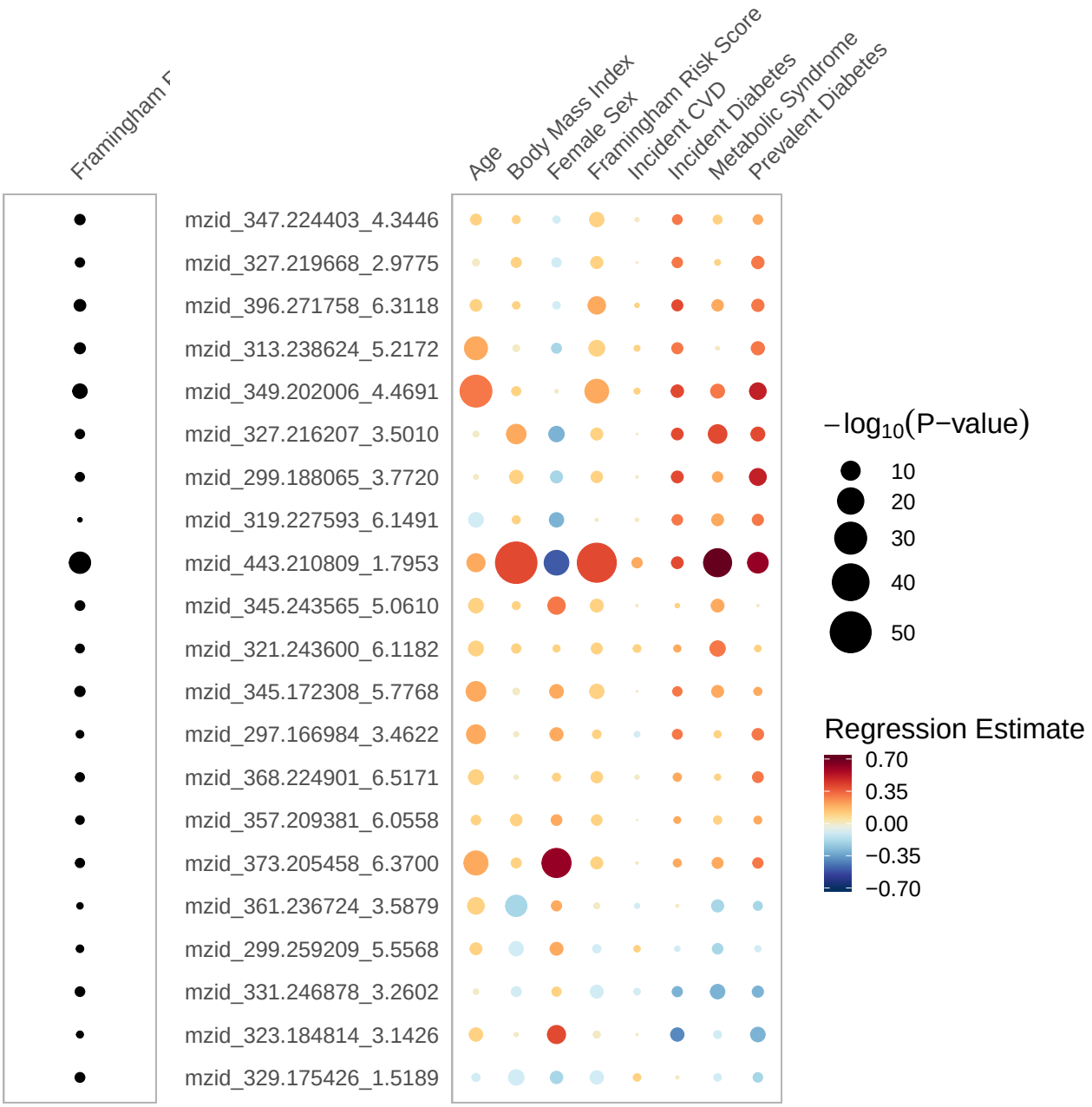
```
  plot_data %>%
  filter(response == longest_x_label)

dendro <-
  ggplot() +
  # One-column rainplot. White Points to be invisible.
  geom_point(aes(x = response, y = term, size = p.value),
             colour = 'white',
             data = modified_rainplot_data) +
  scale_x_discrete(position = 'top', expand = c(0, 3, 0, 0.1)) +
  thm +
  # Make rainplot invisible. Remove elements that don't affect alignemnt Make
  # invisible (match background) elemnts that do affect alignment.
  theme(legend.position = 'none',
        axis.text.y = element_blank(),
        axis.text.x = element_text(colour = 'white'),
        panel.border = element_rect(fill = NA, colour = 'white')

  )  +
  # Draw dendrogram
  geom_segment(aes(x = (-y + 1), y = x, xend=(-yend + 1), yend=xend),
               colour = 'black',
               data = dendro_dat)


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



To better understand how this works, lets walk through it step by step. To make what is going on clearer, many plot elements will initially be left visible. We start by plotting the modified rainplot next to the full rainplot.

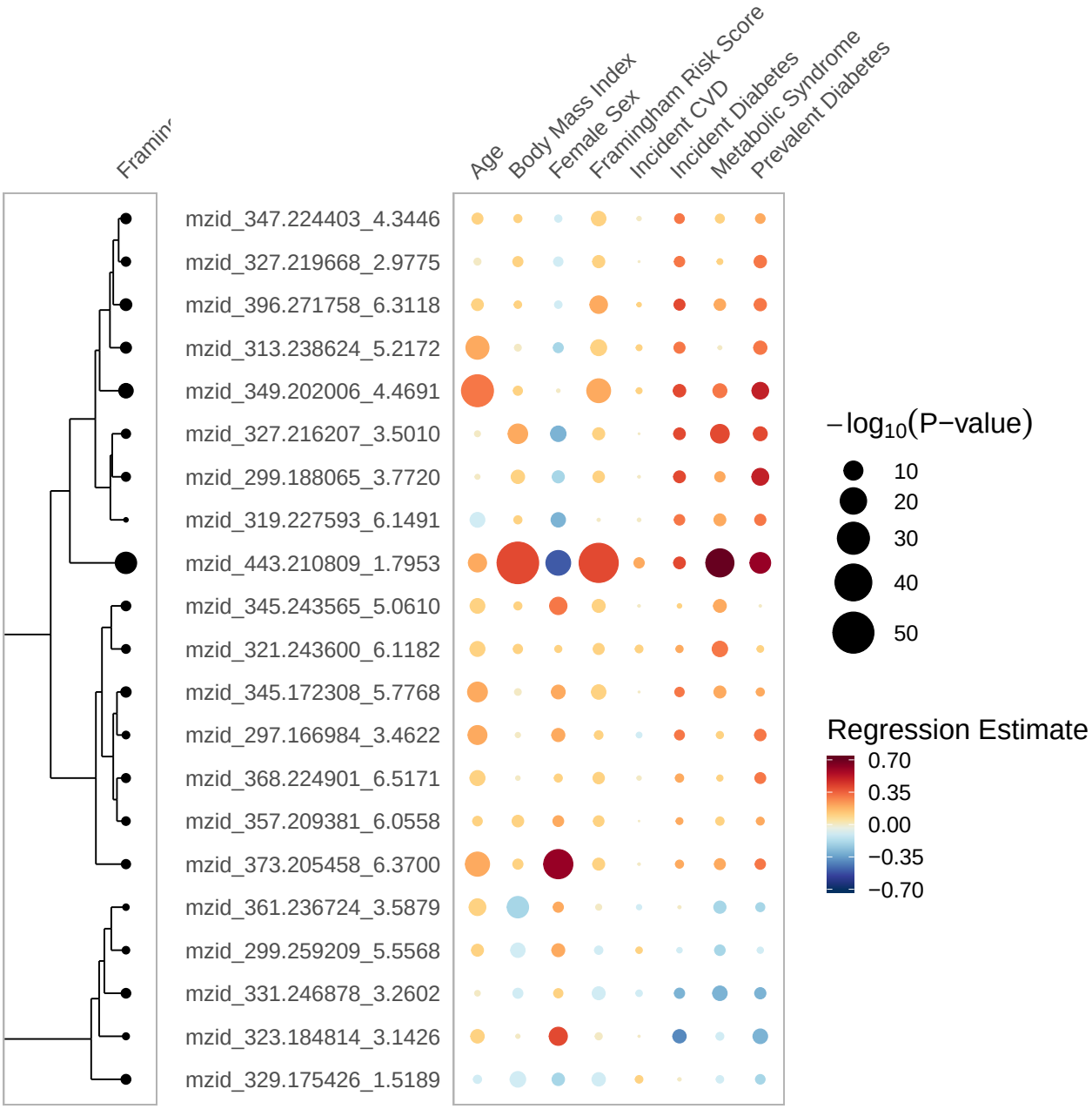
```
x_labels <-
  plot_data$response %>%
  unique()

longest_x_label <-
  x_labels[[which.max(nchar(x_labels))]]

modified_rainplot_data <-
  plot_data %>%
```

```
  filter(response == longest_x_label)

dendro <-
  ggplot() +
  # One-column rainplot.
  geom_point(aes(x = response, y = term, size = p.value),
             colour = 'black',
             data = modified_rainplot_data) +
  scale_x_discrete(position = 'top') +
  thm +
  # Remove legend and y axis text
  theme(legend.position = 'none',
        axis.text.y = element_blank()

  )

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



Looking at the plots side by side, we see that the plot areas are aligned. This will help align the dendrogram to the labels.

```
# The `+ 1` aligns the tips of the dendrograms to the plot points. This will be
# important when eliminating white space.
dendro <-
  dendro +
  geom_segment(aes(x = (-y + 1), y = x, xend=(-yend + 1), yend=xend),
               colour = 'black',
               data = dendro_dat)

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

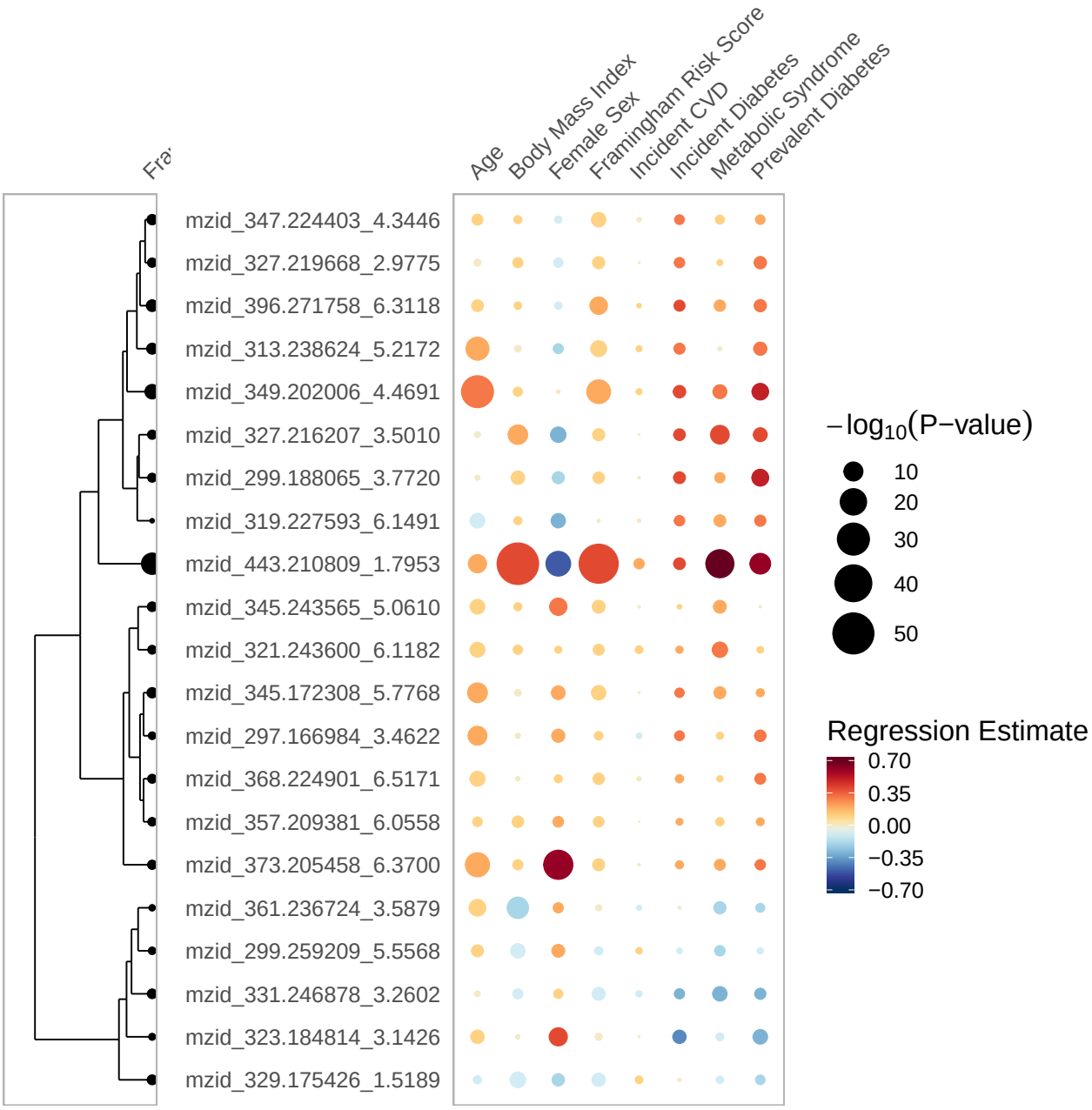


When we add the dendrogram, we see that it is perfectly aligned to the rainplot labels. However, some of the dendrogram is outside the plotted area to the left. Additionally, there is too much white space to the right of the plot. Both those issues can be rectified through the `expand` argument. The first non-zero argument to `expand` controls the space to the left of the plot, and needs to be large enough so that the whole dendrogram is visible. The best value may take some trial and error to discover. The second non-zero argument determines how close the right plot edge is to the bottom of the dendrogram. I have found 0.1 to be a good default, but it can be larger if more space between the dendrogram and rainplot is desired..

```
dendro <-
  dendro +
  scale_x_discrete(position = 'top', expand = c(0, 3, 0, 0.1))
```

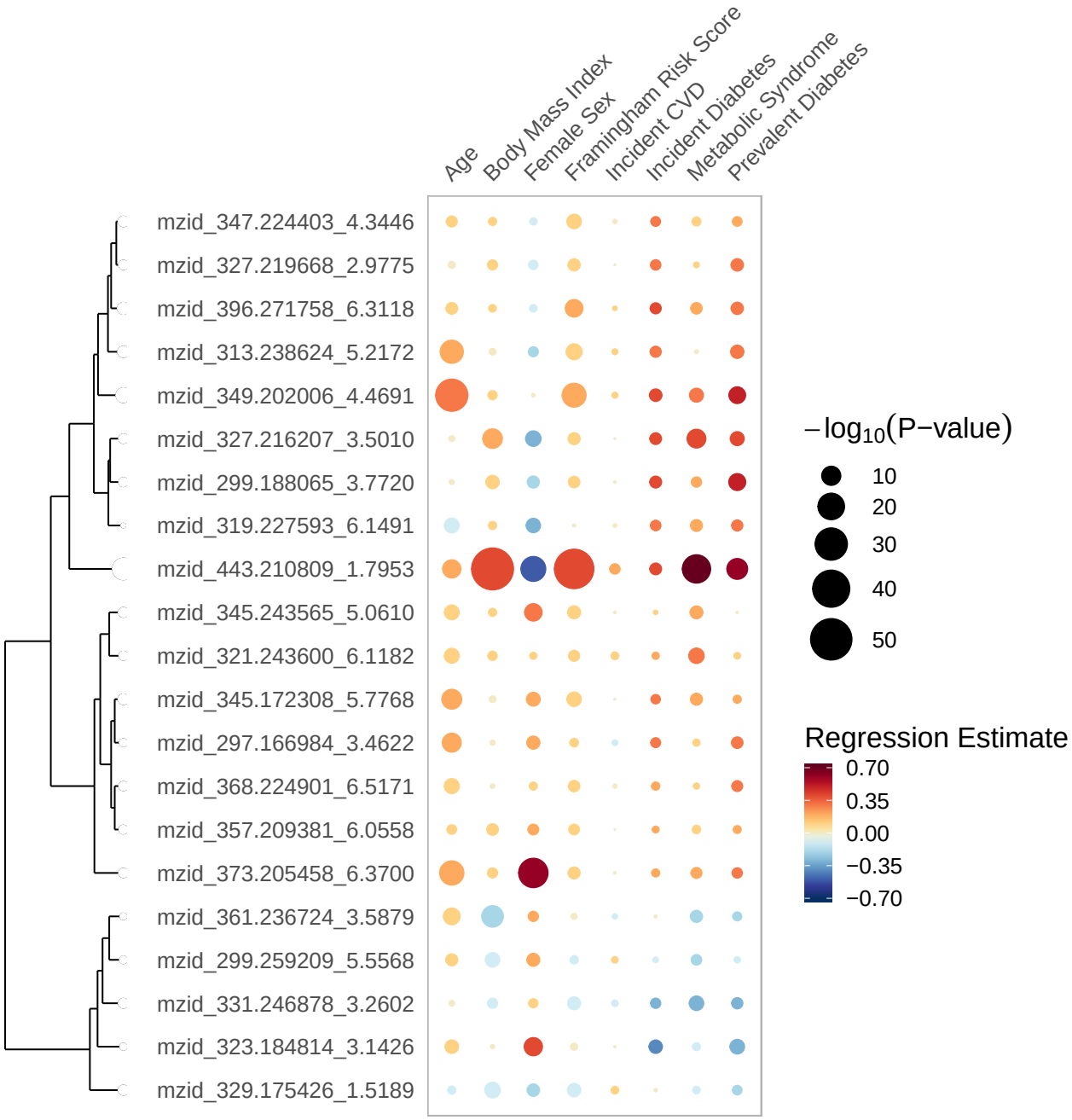
```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

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



To finalize the plot, we make invisible the remaining visual elements not a part of the dendrogram.

```
dendro <-
  dendro +
  geom_point(aes(x = response, y = term, size = p.value),
             colour = 'white',
             data = modified_rainplot_data) +
  theme(axis.text = element_text(color = 'white'),
        panel.border = element_rect(fill = NA, color = 'white'))

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


Age
Body Mass Index
Female Sex
Framingham Risk Score
Incident CVD
Incident Diabetes
Metabolic Syndrome
Prevalent Diabetes
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
−log10(P−value)
10
20
30
40
50
Regression Estimate
0.70
0.35
0.00
−0.35
−0.70