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 pvalue
- The regression estimate

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

```
plot_data
```

```
## # A tibble: 168 x 4
##    response              term                    estimate  p.value
##    <fct>                 <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 Metabolic Syndrome    mzid_443.210809_1.7953       0.7 1.78e-23
##  4 Female Sex            mzid_379.248022_5.0532       0.6 1.01e-22
##  5 Female Sex            mzid_443.210809_1.7953      -0.5 3.92e-18
##  6 Female Sex            mzid_361.161884_4.4617       0.6 1.08e-17
##  7 Female Sex            mzid_279.158850_4.7261       0.5 1.17e-16
##  8 Female Sex            mzid_379.248814_4.7785       0.4 1.30e-14
##  9 Age                   mzid_311.223343_4.4913       0.2 2.04e-14
## 10 Framingham Risk Score mzid_293.208327_5.9709       0.2 3.93e-14
## # ... 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.

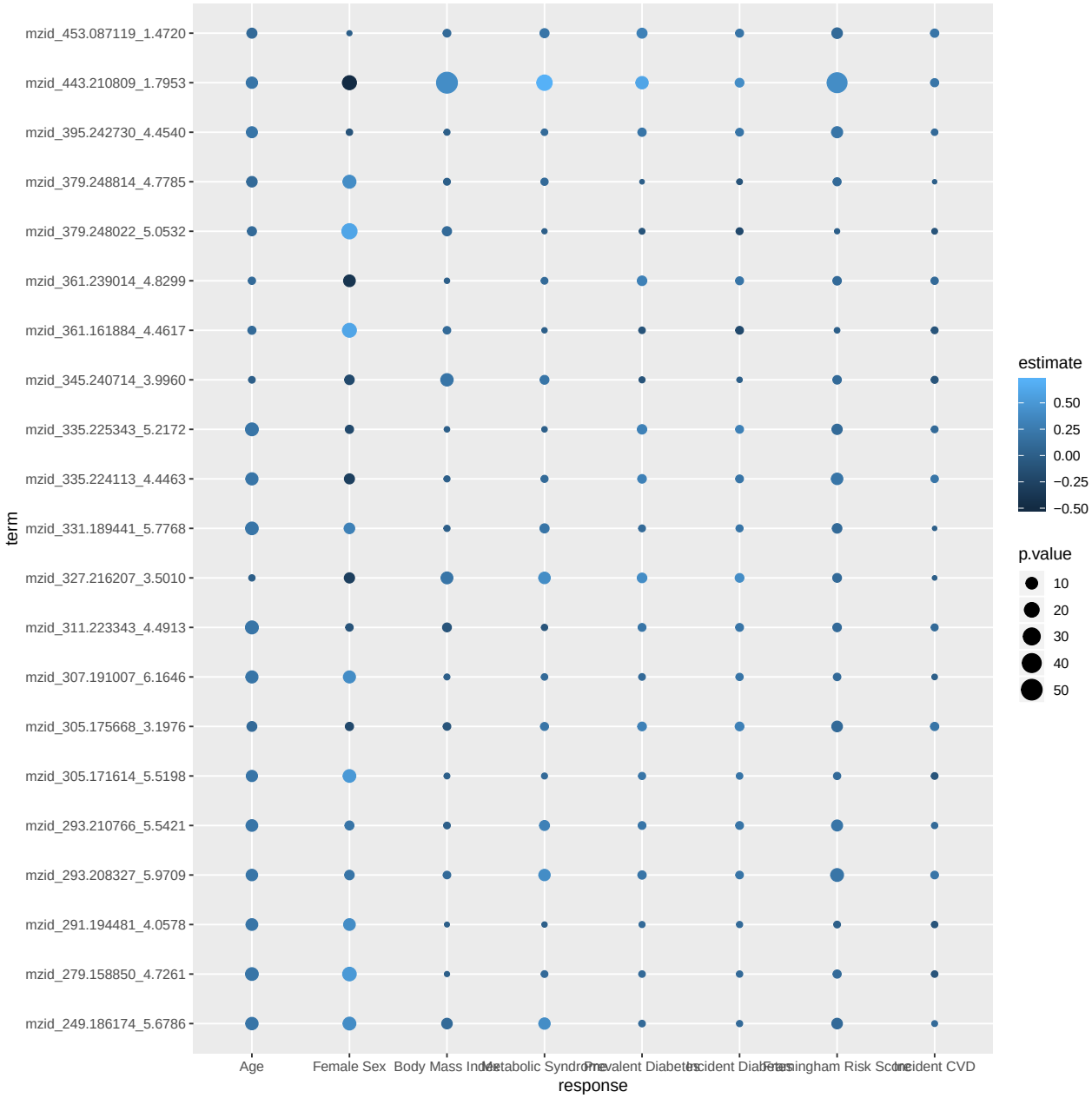
Plotting

Before plotting, we will first transform the p.value onto the log-scale.

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

We can then construct a basic rainplot.

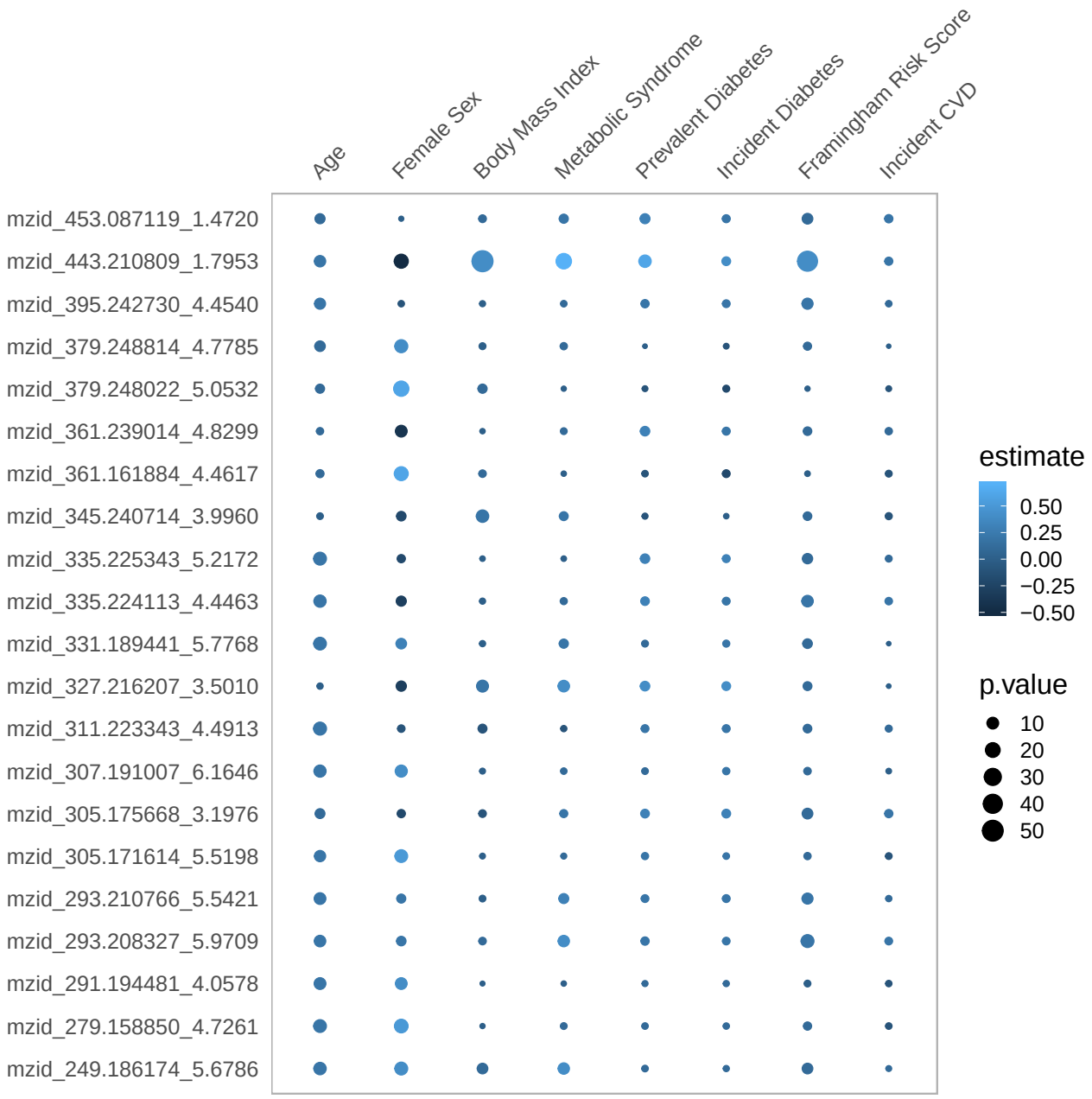
```
rainplot <-
    plot_data %>%
    ggplot(aes(x = response, y = term)) +
    geom_point(aes(colour = estimate, size = p.value))

rainplot
```



This is a good start, but we will want to clean it up. We can do this by creating a custom ggplot2 theme.

```
thm <-
    theme_light(base_size = 18) +
    theme(
```

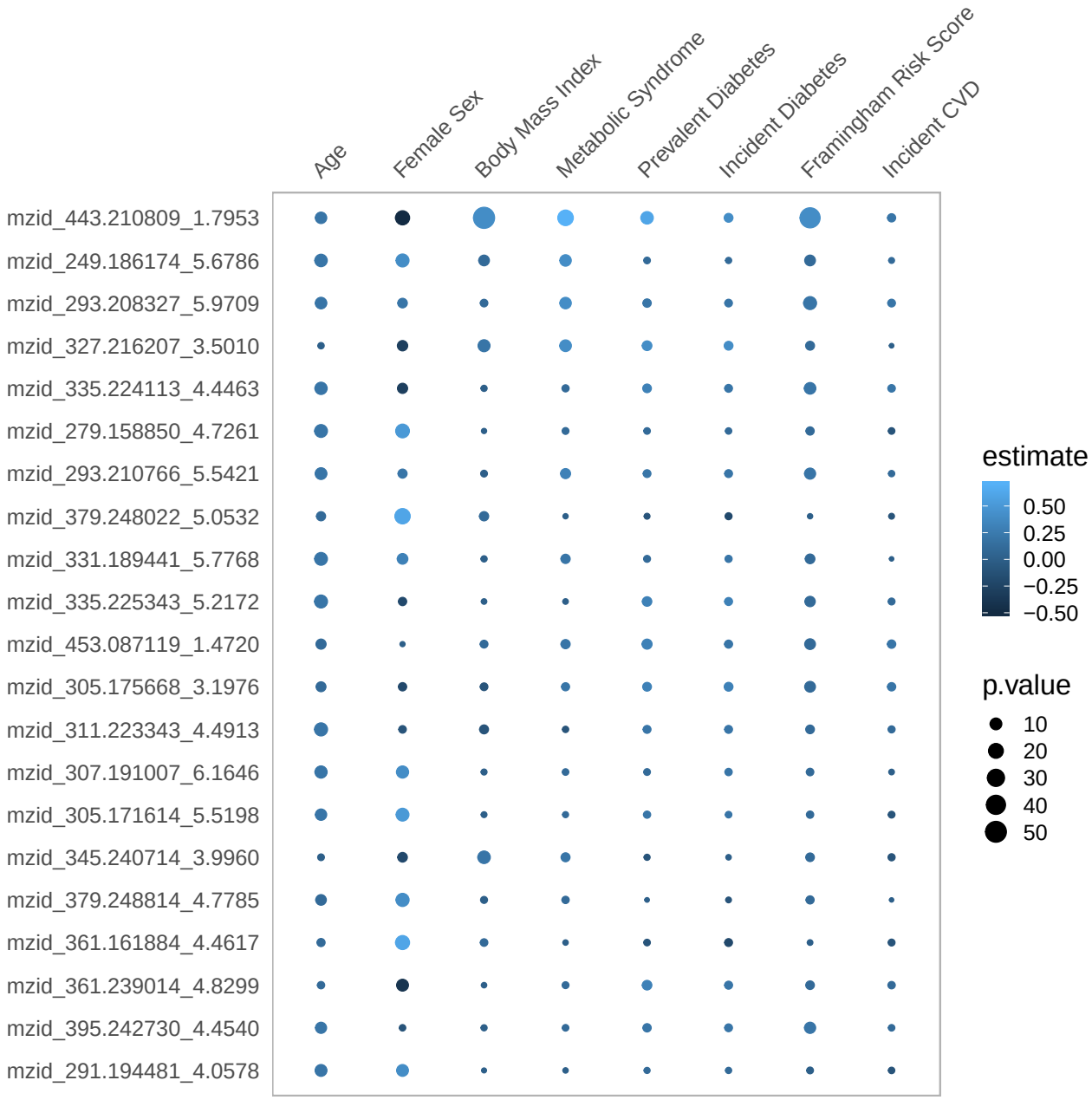
```
        axis.title.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_blank(),
        legend.key = element_blank(),
        panel.background = element_rect(fill = 'white'),
        plot.background = element_rect(fill = 'white'),
        legend.background = element_rect(fill = 'white'),
        axis.text.x.top  = element_text(angle = 45, hjust = 0)
    )

rainplot <- rainplot + thm + scale_x_discrete(position = 'top')
rainplot
```



To make it easier to identify the terms that had small p.values in multiple models, we will convert the `term` variable into an ordered factor.

```
term_order <-
    plot_data %>%
    group_by(term) %>%
    summarise(mpv = mean(p.value)) %>%
    arrange(mpv) %>%
    pull(term)

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

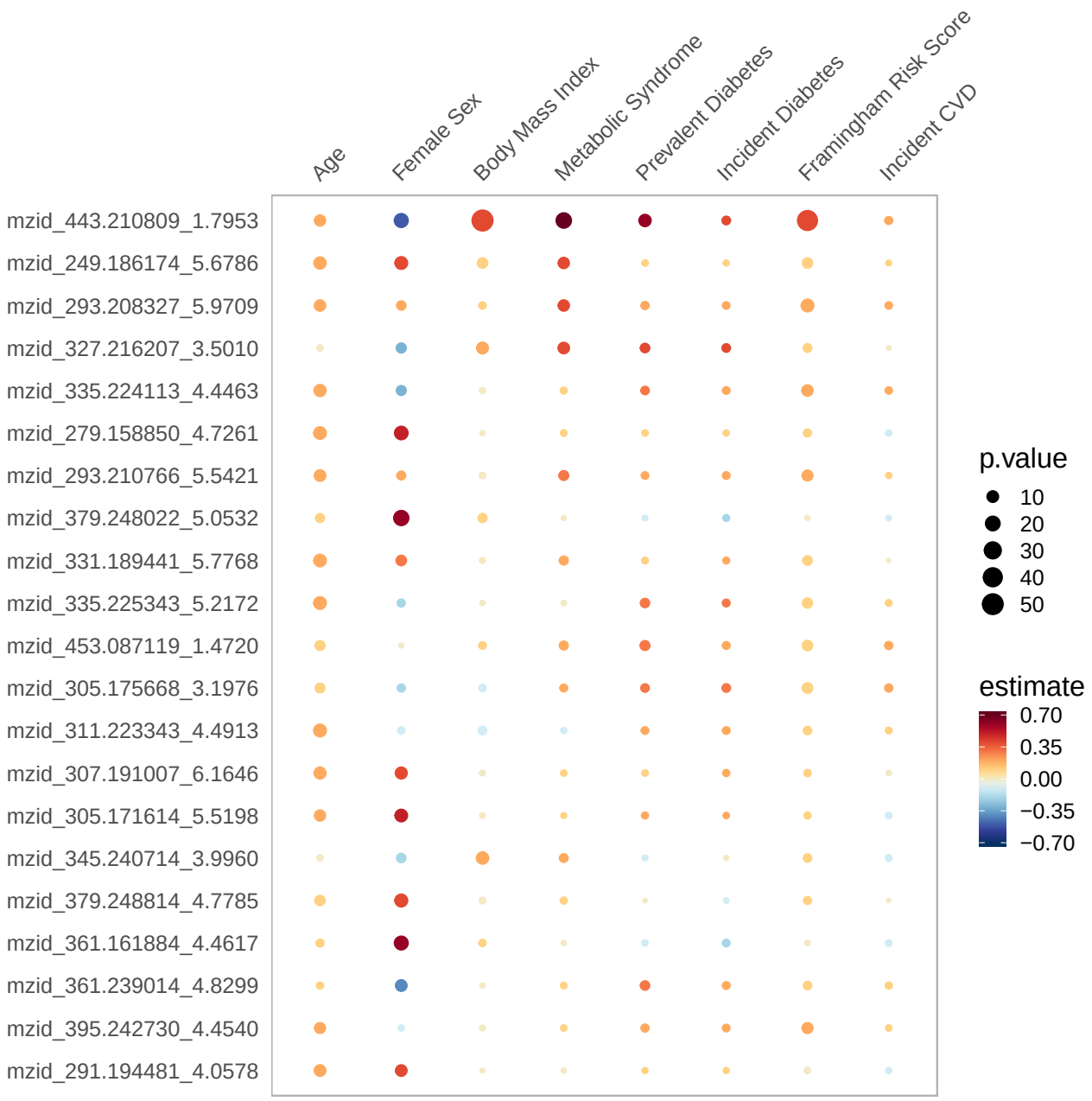
```
rainplot <-
    plot_data %>%
    ggplot(aes(x = response, y = term)) +
    geom_point(aes(colour = estimate, size = p.value)) +
    scale_x_discrete(position = 'top') +
    thm

rainplot
```



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

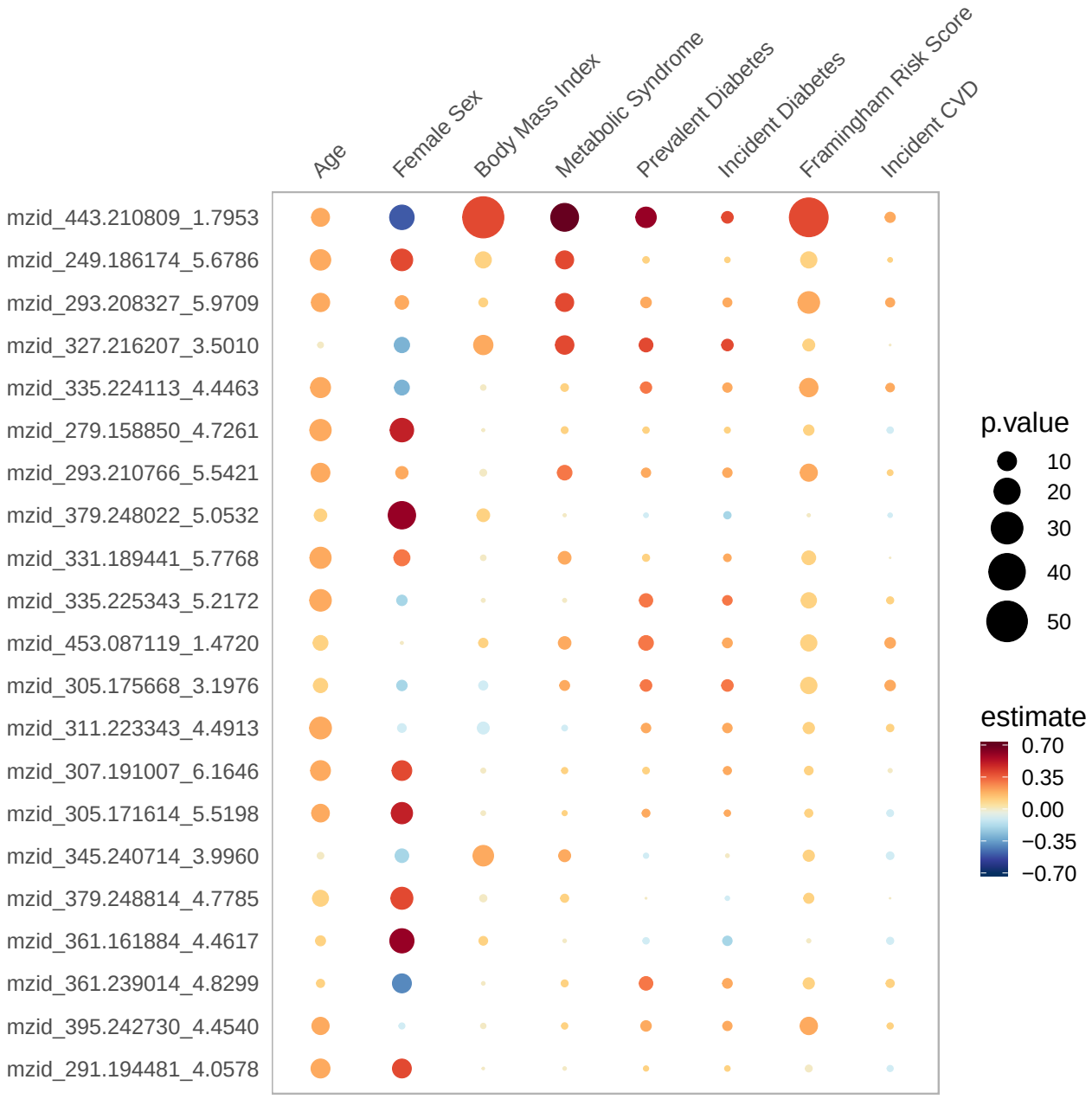
```
palette <-
    c("#053061",
```

```
      "#313695",
      "#4575b4",
      "#74add1",
      "#abd9e9",
      "#e0f3f8",
      # "#ffffbf",
      "#fee090",
      "#fdae61",
      "#f46d43",
      "#d73027",
      "#a50026",
      '#67001f')

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

rainplot <- rainplot +
    scale_color_gradientn(
        colors = palette,
        limits = c(-1 * max_abs, max_abs),
        breaks = c(-1 * max_abs, -1 * max_abs / 2, 0 , max_abs/2, max_abs)
    )

rainplot
```



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

```
rainplot +
    scale_size_area(max_size = 12)
```



Onoe possible issue is that we have a few `p.values` that are much larger than the others. When we plot the full range of the data, we lose resolution in the range where most of the data lies. One possible solution is to set all values above some ceiling, here chosen to be 15, to the value of the ceiling.

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

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

```
        limits = c(-1 * max_abs, max_abs),
        breaks = c(-1 * max_abs, -1 * max_abs / 2, 0 , max_abs/2, max_abs)
    ) +
    scale_size_area(max_size = 12, breaks = c(5, 10, 15), labels = c('5', '10', '>=15')) +
    scale_x_discrete(position = 'top') +
    thm

rainplot
```