Plotting Manhattan Plots

Mir Henglin

7/15/2019

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.

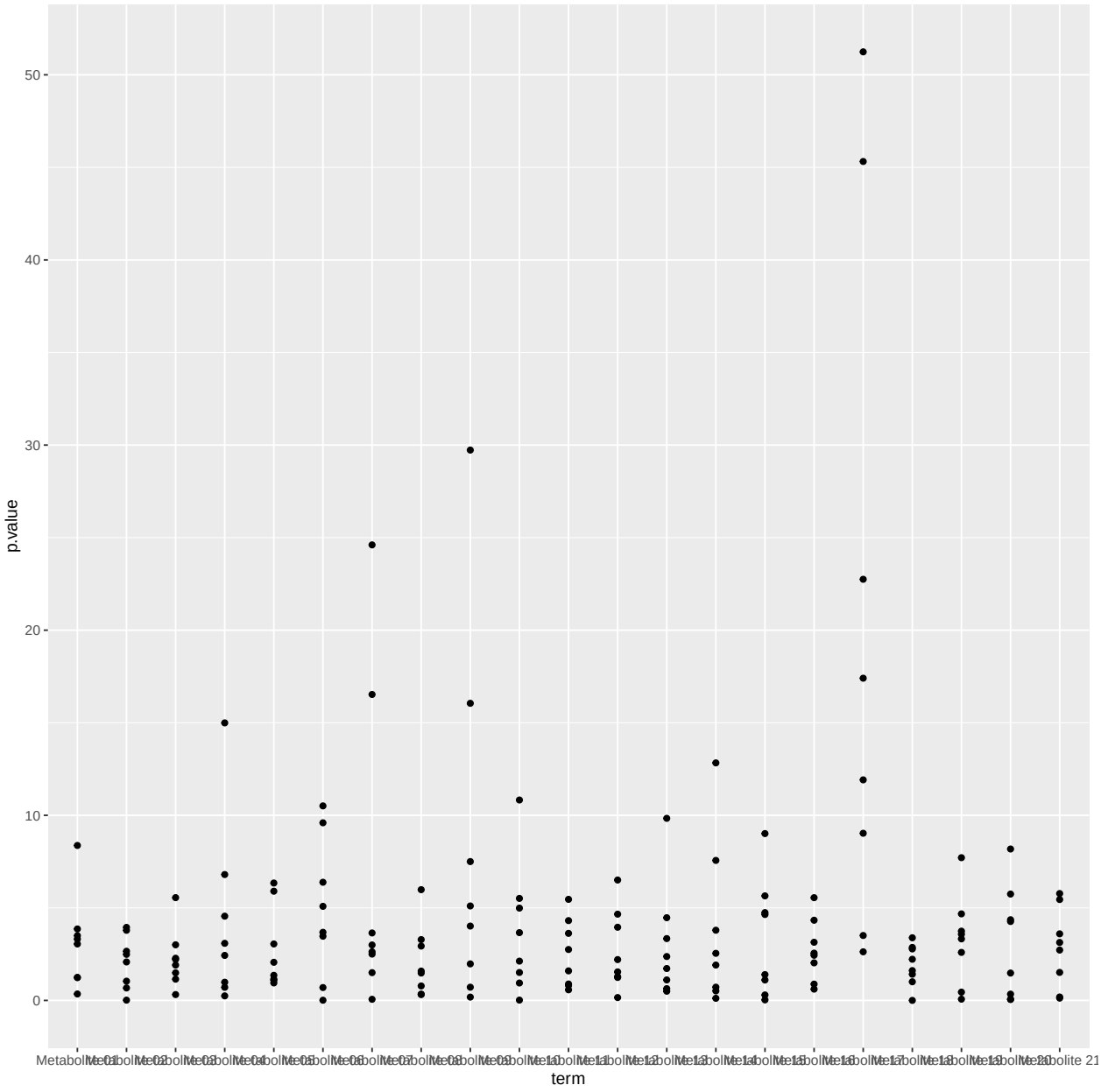
Before plotting, we will first transform `p.value` onto the negative-log scale. This will allow the smallest P-values, which are often of greater interest, to have the largest values when plotted.

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

Plotting

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

```
manhattan_plot <-
  ggplot(plot_data) +
  geom_point(aes(x = term, y = p.value))

manhattan_plot
```

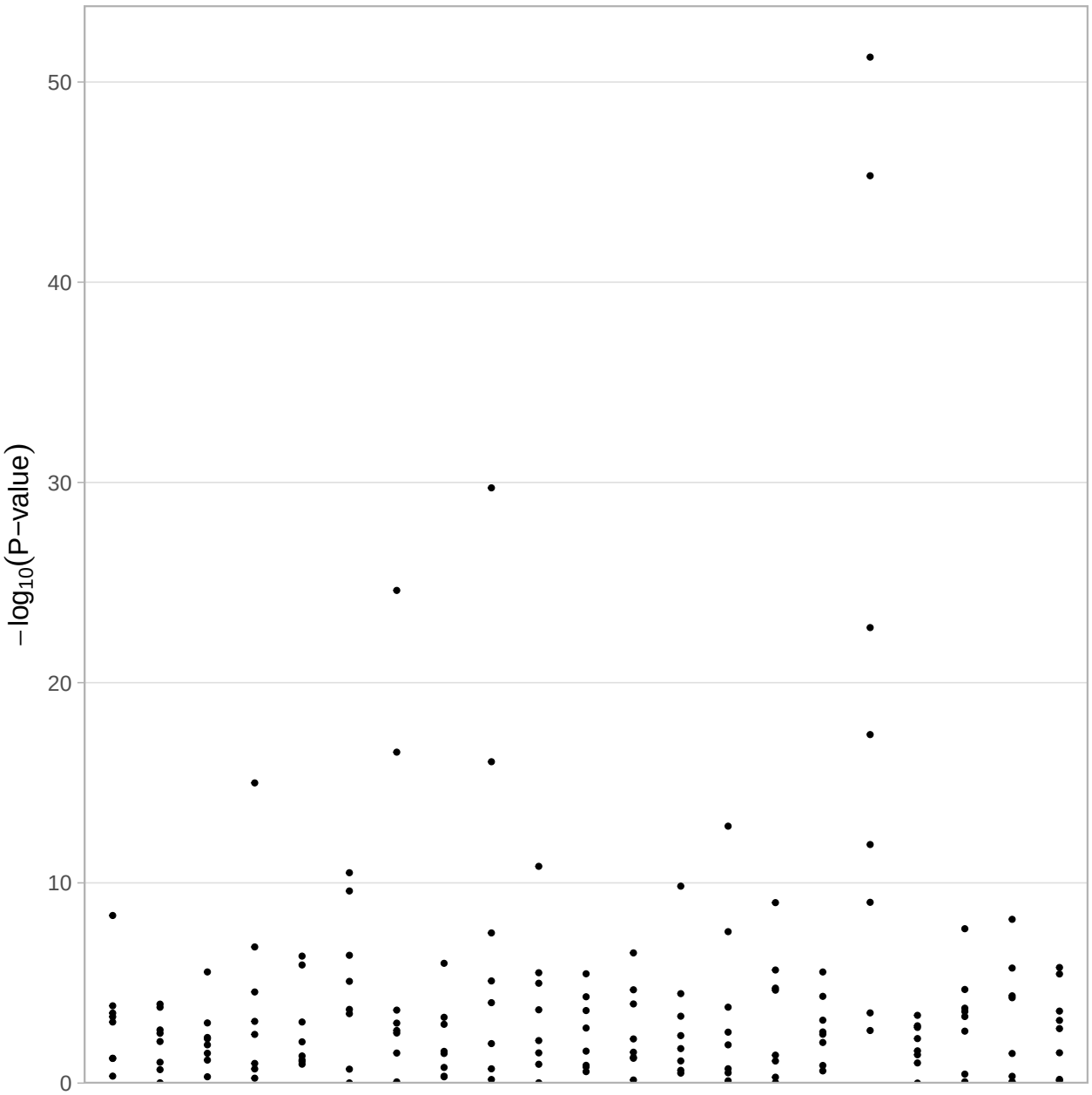


This is a good start, but we want to clean up layout and presentation. We can do this by creating a custom `ggplot2` theme and adjusting scales and layout.

```
thm <-
  # Good starting theme + set text size
  theme_light(base_size = 18) +
  theme(
    # Remove x-axis text
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    # Remove Vertical Grid Lines
    panel.grid.major.x = element_blank(),
    panel.grid.minor = element_blank(),
```

```
    axis.line = element_blank(),
    # White background
    panel.background = element_rect(fill = 'white'),
    plot.background = element_rect(fill = 'white')
  )
```

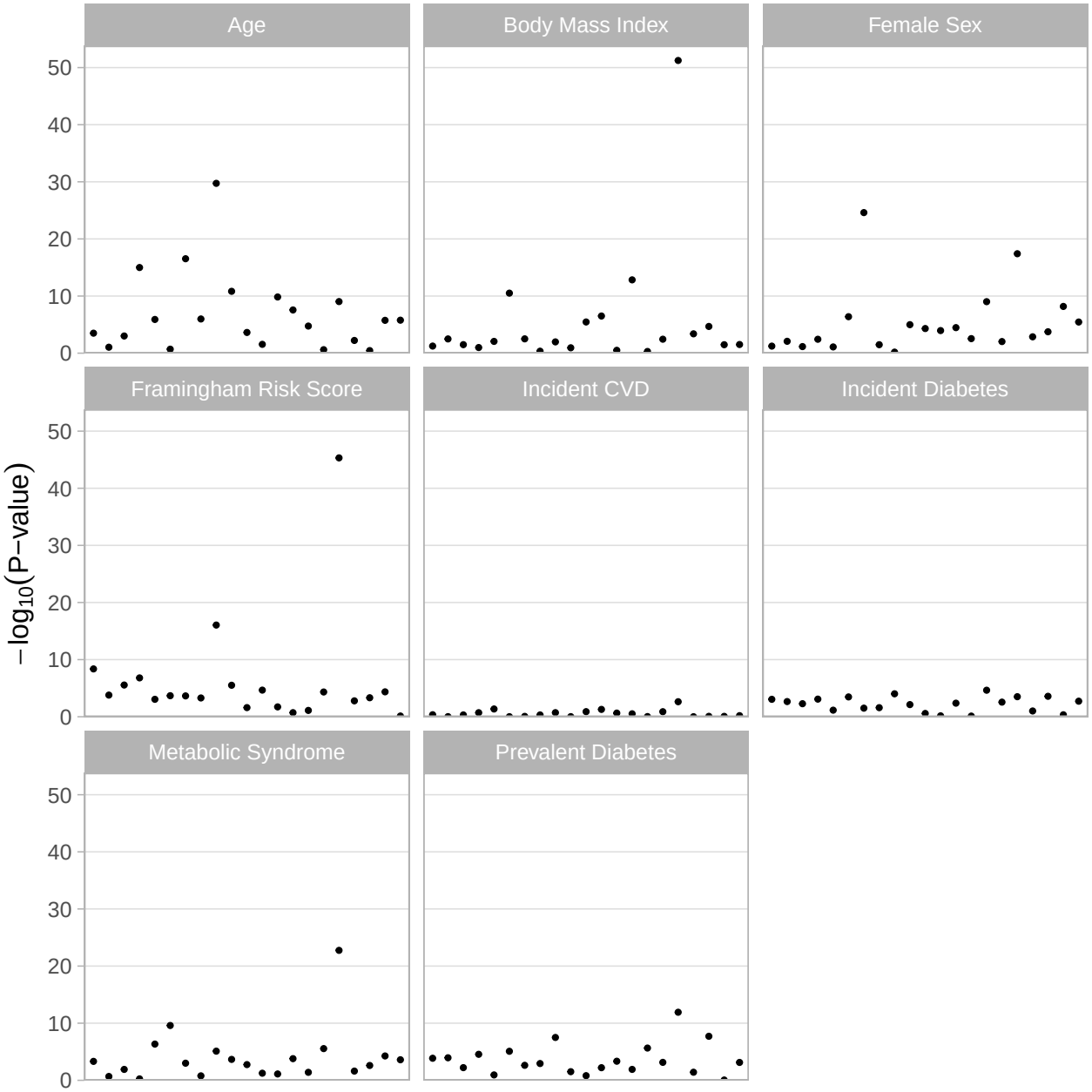
```
manhattan_plot <-
  manhattan_plot +
  scale_y_continuous(
    expression(paste(-log[10]('P-value'))),
    # These `limits` and `expand` set the bottom of the plot to be '0' while
    # maintaining spacing at the top of the plot.
    limits = c(0, NA),
    expand = expand_scale(mult = c(0, 0.05))
  ) +
  thm

manhattan_plot
```



Faceting

The above code creates a single Manhattan plot. However, our data consists of results obtained by comparing a set of metabolites against multiple biological measures of interest. A single plot therefore contains the same `term` plotted multiple times, once for each biological measure of interest To remove this possibly confusing duplication, we can create a Manhattan plot for each different biological measure of interest in our data. Luckily, `ggplot2` makes this very easy through 'faceting'.

```
manhattan_plot_facet <-
  manhattan_plot +
  facet_wrap(~response)

manhattan_plot_facet
```



Despite the simplicity of this technique, this is not the most common way that Manhattan plots are presented. Typically, each set of results is plotted adjacent to every other set of results. Adjacent sets of results are colored in a way that makes them distinct. We can create this style of Manhattan plot, but it will need some manual calculation and fiddling.

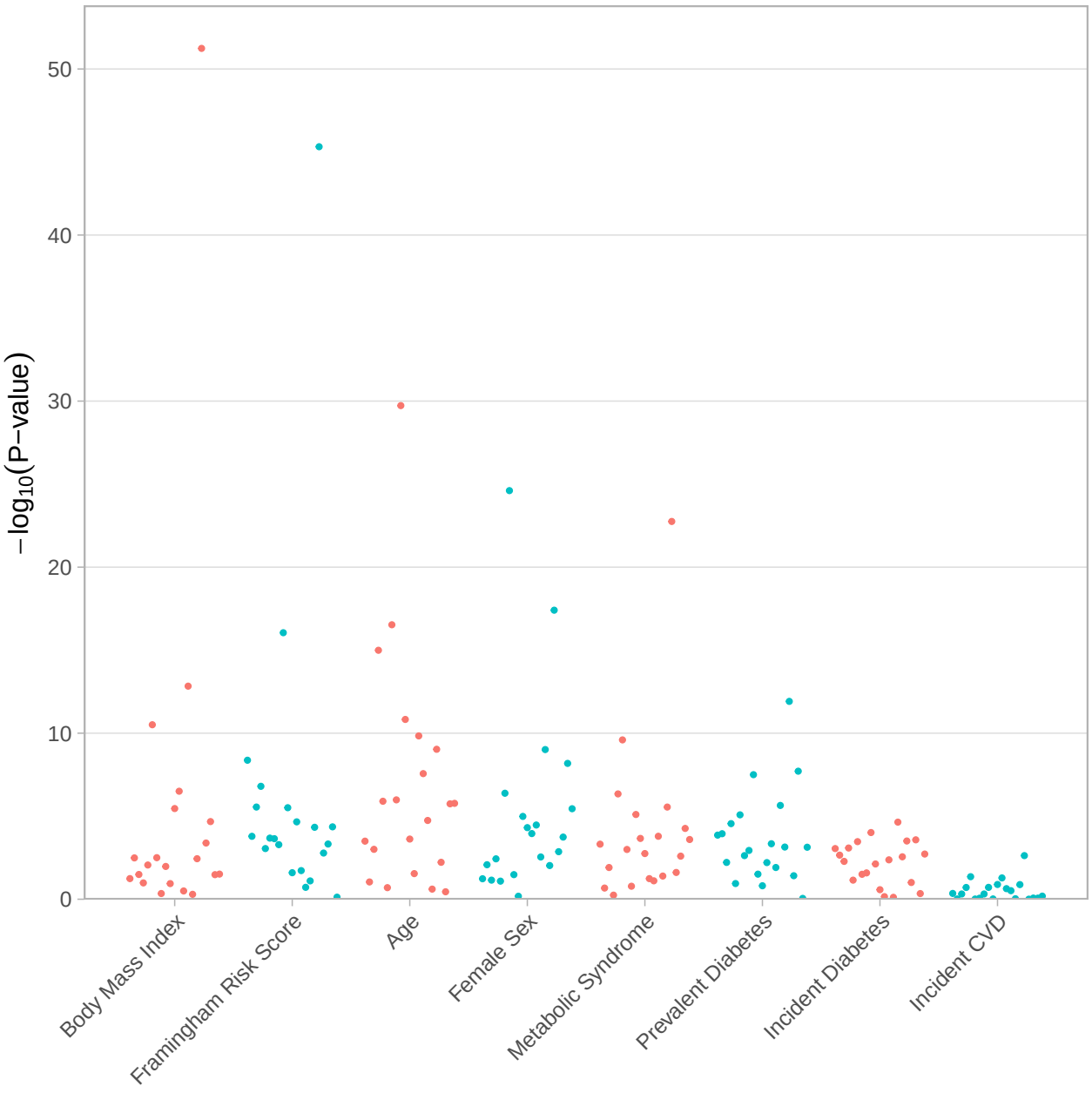
```
n_terms <-
  n_distinct(plot_data$term)

# Space between sets of results
buffer <- n_terms/4

model_set_spacing <-
  plot_data %>%
  distinct(response) %>%
```

```
  # Assign each set of results to a block
  mutate(block = 1:n() - 1) %>%
  # Calculate the center location of each set of results
  mutate(center = block * n_terms + block * buffer + median(1:n_terms)) %>%
  # Assign each set of results to a color. Here, using alternating colors
  mutate(color_group = as.factor(rep_len(c(0, 1), length.out = n())))

# Calculate the position of each `term` relative to the median
# (median set to be 0)
term_position <-
  plot_data %>%
  distinct(term) %>%
  arrange(term) %>%
  mutate(x = 1:n() - 1) %>%
  mutate(x = x - median(x))

plot_data <-
  plot_data %>%
  left_join(model_set_spacing, by = 'response') %>%
  left_join(term_position, by = 'term')

# Combine relative positioning of each term with the center of each set of
# results to calculate each final term's positioning
plot_data <-
  plot_data %>%
  mutate(x = x + center)

head(plot_data)
```

```
## # A tibble: 6 x 8
##   response        term      estimate p.value block center color_group     x
##   <chr>           <chr>        <dbl>   <dbl> <dbl>  <dbl> <fct>       <dbl>
## 1 Body Mass Index Metaboli~      0.4    51.2     0   11   0            17
## 2 Framingham Ris~ Metaboli~      0.4    45.3     1   37.2 1            43.2
## 3 Age             Metaboli~      0.3    29.7     2   63.5 0            61.5
## 4 Female Sex      Metaboli~      0.6    24.6     3   89.8 1            85.8
## 5 Metabolic Synd~ Metaboli~      0.7    22.8     4  116   0           122
## 6 Female Sex      Metaboli~     -0.5    17.4     3   89.8 1            95.8
```

```
# This theme is different from the previous theme only in that it keeps x-axis
# text and ticks
thm <-
  theme_light(base_size = 18) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_text(angle = 45, hjust = 1),
    # Remove Vertical Grid Lines
    panel.grid.major.x = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_blank(),
    # White background
    panel.background = element_rect(fill = 'white'),
    plot.background = element_rect(fill = 'white')
  )
```

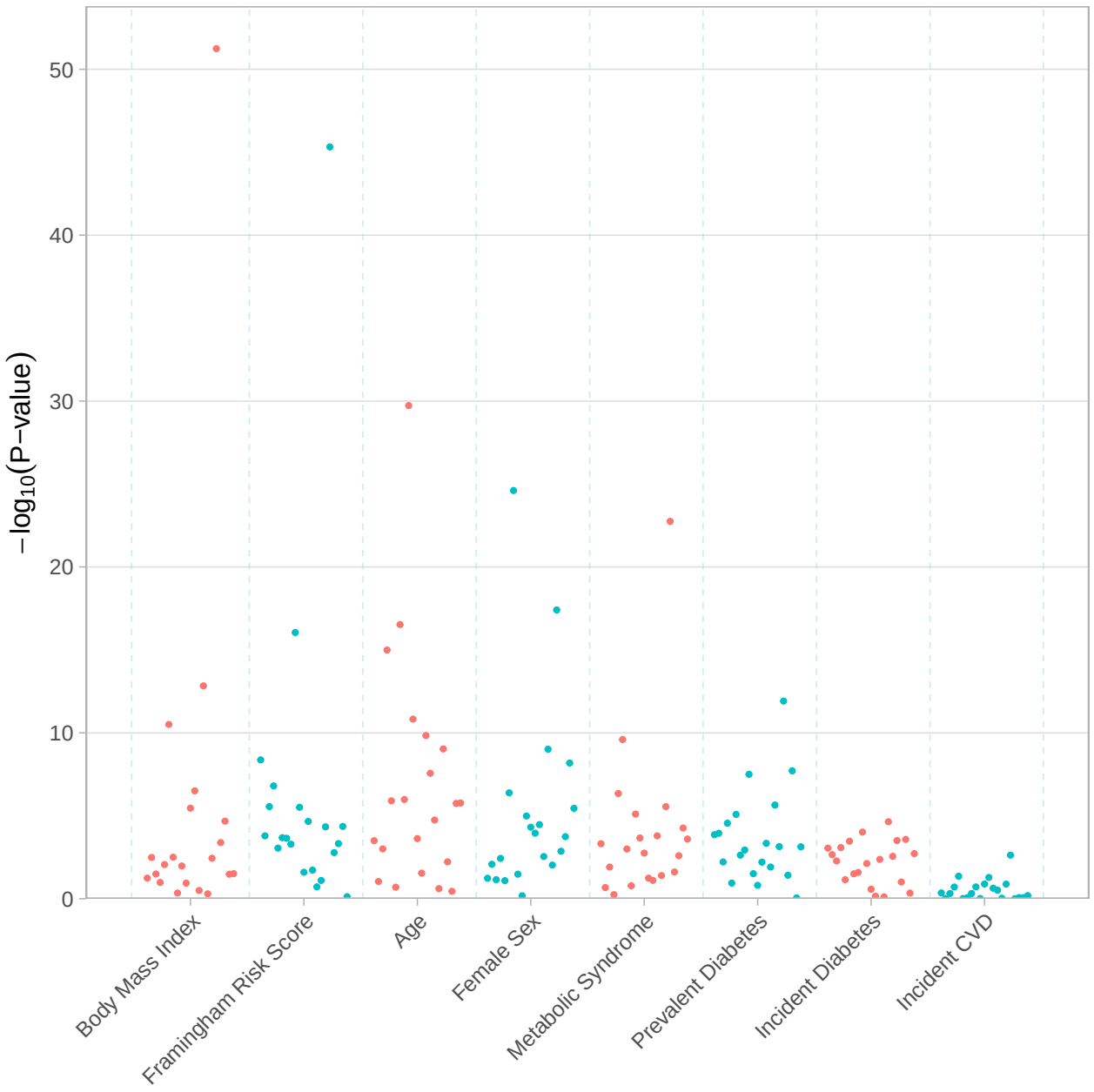
```r
manhattan_plot <-
  ggplot(plot_data) +
  # Use the calculated x position and color
  geom_point(aes(x = x, y = p.value, color = color_group)) +
  scale_y_continuous(
    expression(paste(-log[10]('P-value'))),
    limits = c(0, NA),
    expand = expand_scale(mult = c(0, 0.05))) +
  # Use the x labels to mark the `response` each set of results corresponds to
  scale_x_continuous(
    breaks = model_set_spacing$center,
    labels = model_set_spacing$response
  ) +
  # Remove color legend
  scale_color_discrete(guide = FALSE) +
  thm


manhattan_plot
```



We can also add vertical lines to better group the points corresponding to each set of results.

```
manhattan_plot <-
  manhattan_plot +
  # Add the first vertical line
  geom_vline(
    xintercept = -buffer/2,
    colour = 'lightblue',
    alpha = 0.5,
    linetype = 2
  ) +
  # Add every other one
  geom_vline(
    aes(xintercept = c(center + median(1:n_terms) + buffer/2)),
    colour = 'lightblue',
```

```
    alpha = 0.5,
    linetype = 2
  )

manhattan_plot
```



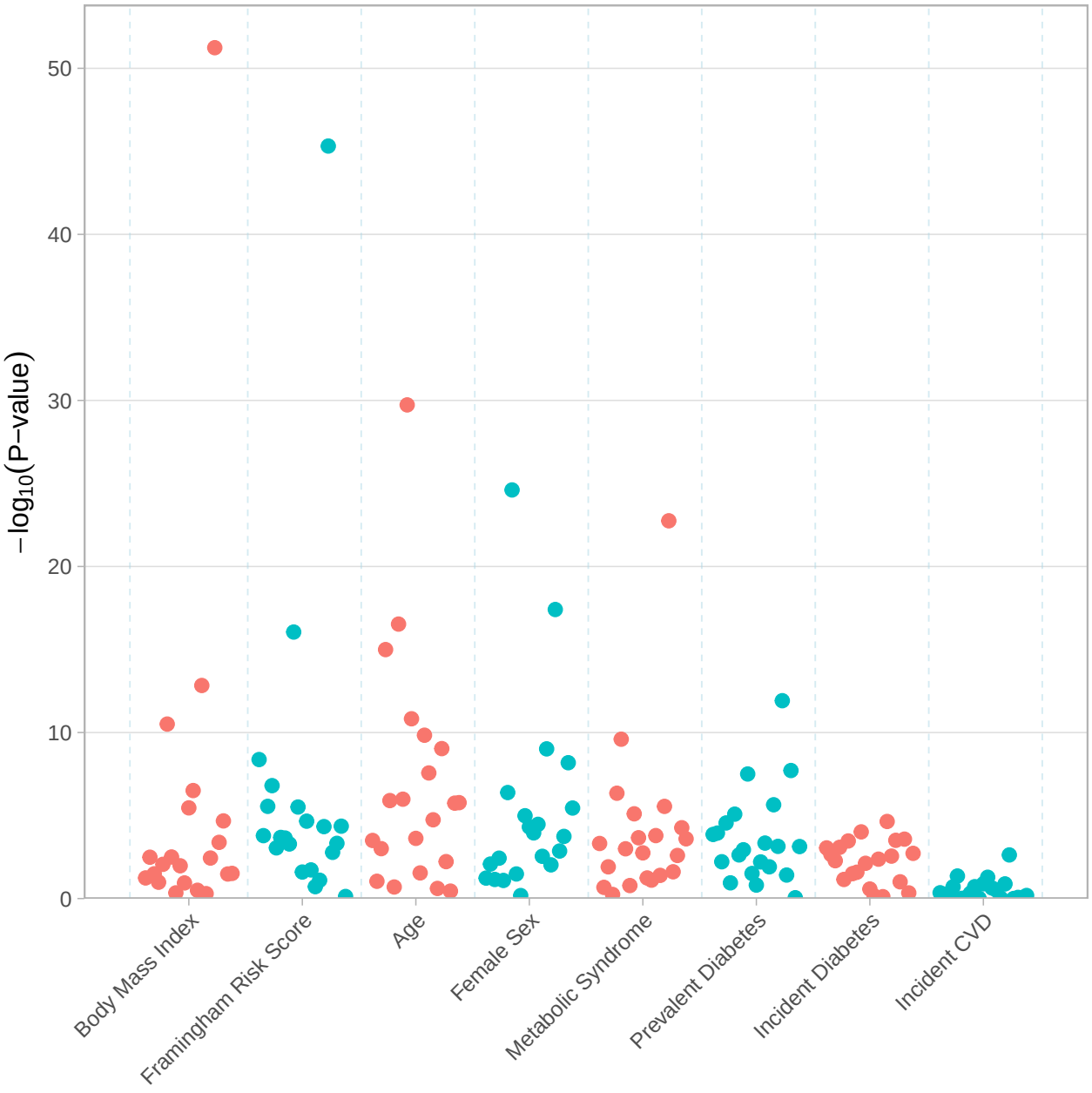
Additional Plot Adjustments

Point Size

Depending on the number of values being plotted at once, we may want to adjust the plotted points to be larger or smaller. This can be done through the `size` argument to `geom_point`.

```r
# Assigning these scales to variables saves typing and allows focus on what is
# changing between plots

scales_thm_and_vlines <-
  list(
    scale_x_continuous(
      breaks = model_set_spacing$center,
      labels = model_set_spacing$response
    ),

    scale_y_continuous(
      expression(paste(-log[10]('P-value'))),
      limits = c(0, NA),
      expand = expand_scale(mult = c(0, 0.05))
    ),

    scale_color_discrete(guide = FALSE),

    geom_vline(
      xintercept = -buffer/2,
      colour = 'lightblue',
      alpha = 0.5,
      linetype = 2
    ) ,

    geom_vline(
      aes(xintercept = c(center + median(1:n_terms) + buffer/2)),
      colour = 'lightblue',
      alpha = 0.5,
      linetype = 2
    ),

    thm
  )
```

```r
manhattan_plot_size <-
  ggplot(plot_data) +
  geom_point(aes(x = x, y = p.value, color = color_group), size = 4) +
  scales_thm_and_vlines

manhattan_plot_size
```



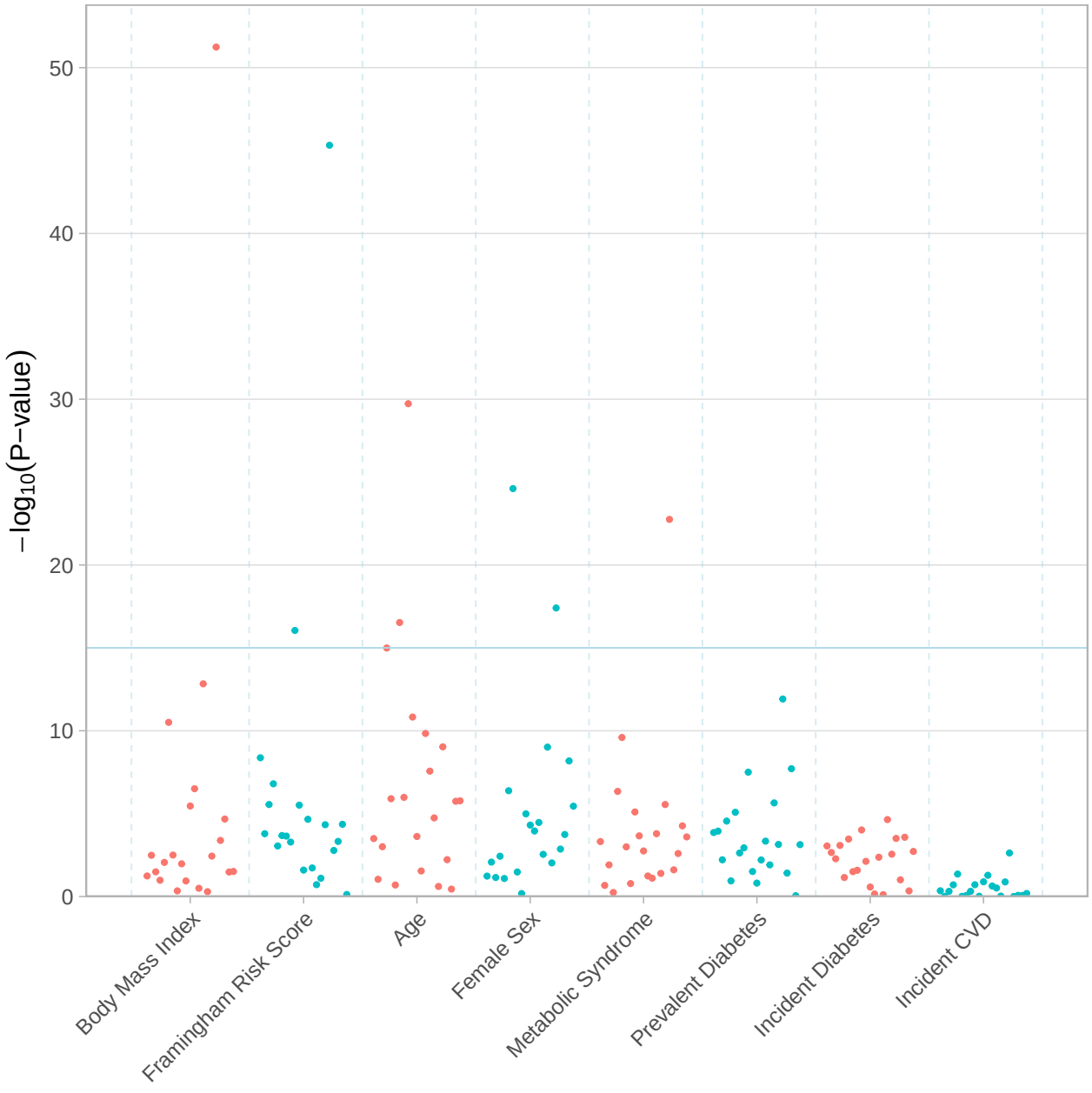
Marking values of interest

Lines

We can add lines marking the P-value threshold such that all metabolites are considered 'of interest' if they have values more extreme than the threshold. In this case, we set the P-value threshold to 15.

```
pv_threshold <- 15

manhattan_plot_line <-
  ggplot(plot_data) +
  geom_point(aes(x = x, y = p.value, color = color_group)) +
  # Add P-value threshold line
```

```
  geom_hline(
    yintercept = pv_threshold,
    colour = 'lightblue'
  ) +
  scales_thm_and_vlines


manhattan_plot_line
```



Annotating values of interest

We may want to label the points that are of interest. While labeling every point in the plot would create unpleasant clutter, labeling the most extreme points is an easy way to increase the information conveyed

by a Manhattan plot without making it unreadable. To do so, we will need to create a label column that contains the desired label text and is `NA` for points that should not be labeled. To plot the labels, we will be using `geom_text_repel` from the `ggrepel` package. Thus function automatically places labels such that they do not overlap, making it a valuable function when values of interest are positioned close together.

```
library(ggrepel)
```

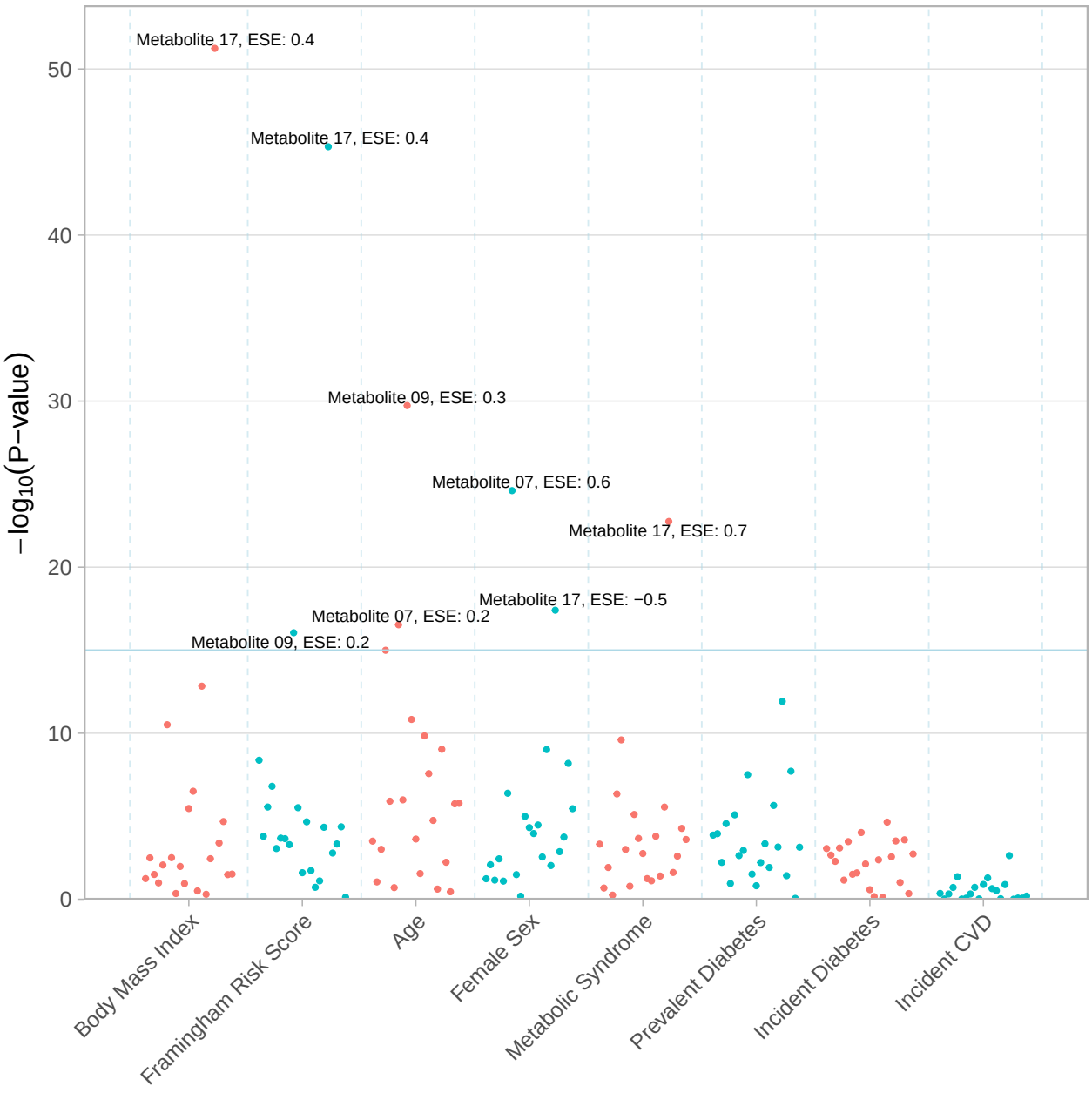
```
## Warning: package 'ggrepel' was built under R version 3.5.2
```

```
# Identify all results with P-value > 15
plot_data <-
  plot_data %>%
  mutate(extreme = p.value > pv_threshold)
```

```
# Create label consisting of the metabolite ID and the effect size estimate
plot_data <-
  plot_data %>%
  mutate(label = ifelse(!extreme, NA, paste0(term, ', ESE: ', estimate)))
```

```
manhattan_plot_line_label <-
  # Use the new data
  ggplot(plot_data) +
  geom_point(aes(x = x, y = p.value, color = color_group)) +
  geom_hline(
    yintercept = pv_threshold,
    colour = 'lightblue'
  ) +
  # Adding labels last draws them on top of threshold lines
  geom_text_repel(aes(x = x, y = p.value, label = label), box.padding = 0.1) +
  scales_thm_and_vlines

manhattan_plot_line_label
```

```
## Warning: Removed 160 rows containing missing values (geom_text_repel).
```


Metabolite 17, ESE: 0.4
Metabolite 17, ESE: 0.4
Metabolite 09, ESE: 0.3
Metabolite 07, ESE: 0.6
Metabolite 17, ESE: 0.7
Metabolite 17, ESE: −0.5
Metabolite 07, ESE: 0.2
Metabolite 09, ESE: 0.2
50
40
30
20
10
0
−log10(P−value)
Body Mass Index
Framingham Risk Score
Age
Female Sex
Metabolic Syndrome
Prevalent Diabetes
Incident Diabetes
Incident CVD