Plotting Volcano 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.

```r
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.

```r
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.

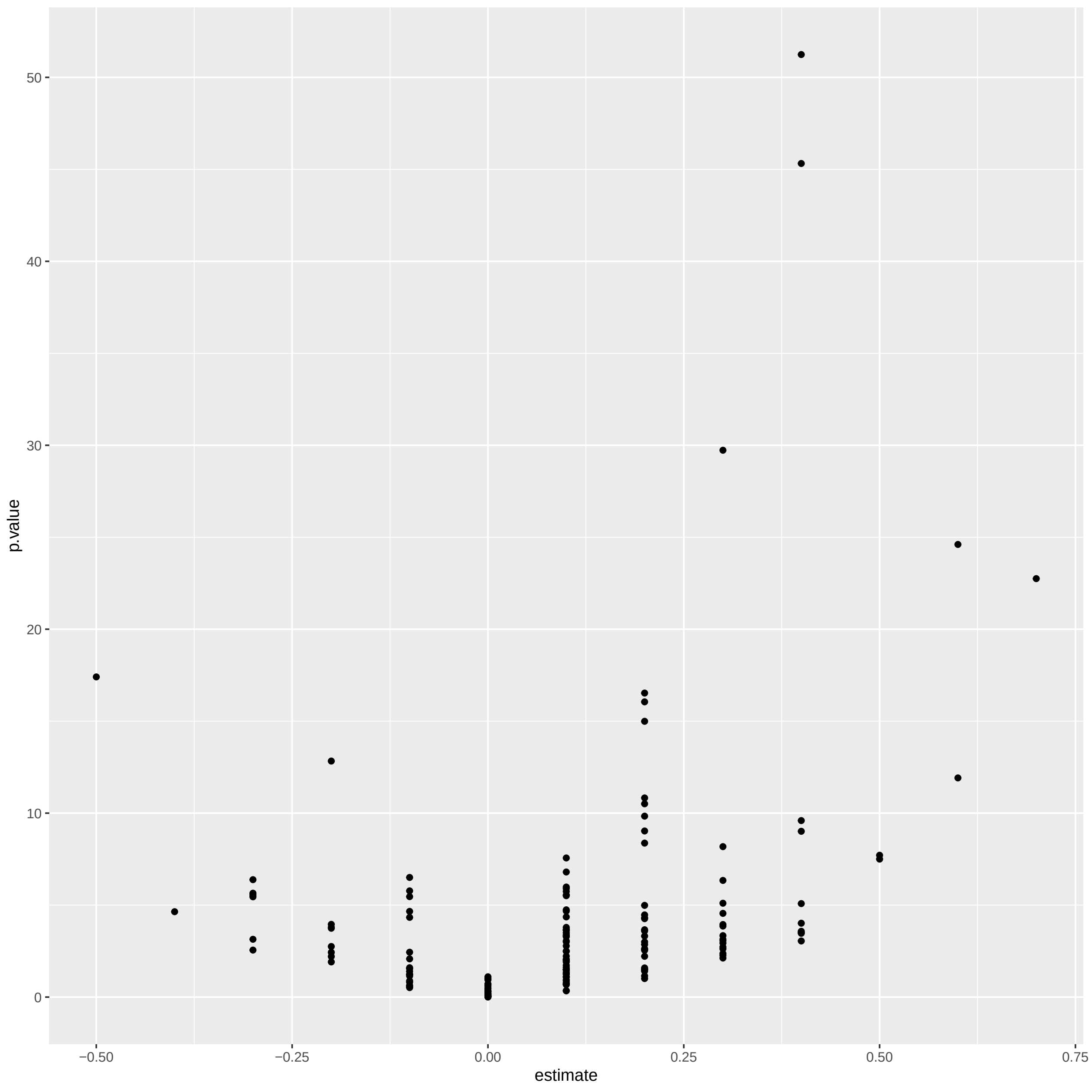
Before plotting, we will first transform `p.value` onto the negative-log scale. This will allow the smallest P-values, which are often of greater interest, to have the largest values when plotted.

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

Plotting

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

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

volcano_plot
```

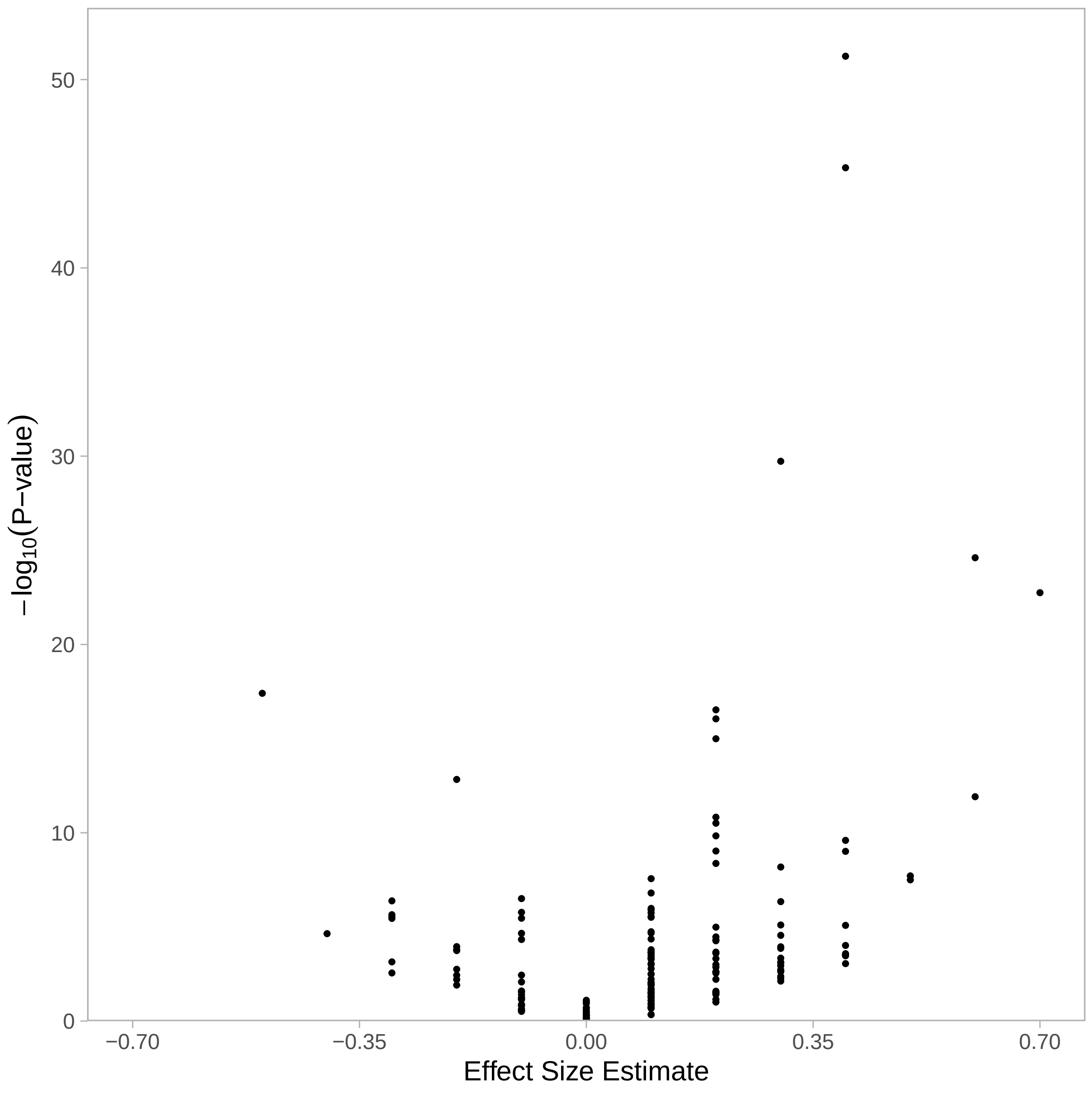


This is a good start, but we want to clean up layout and presentation. We can do this by creating a custom `ggplot2` theme and adjusting scales and layout.

```r
thm <-
  # Good starting theme + set text size
  theme_light(base_size = 18) +
  theme(
    # Remove Grid
    panel.grid.major = 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')
  )
```

```
# 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
```

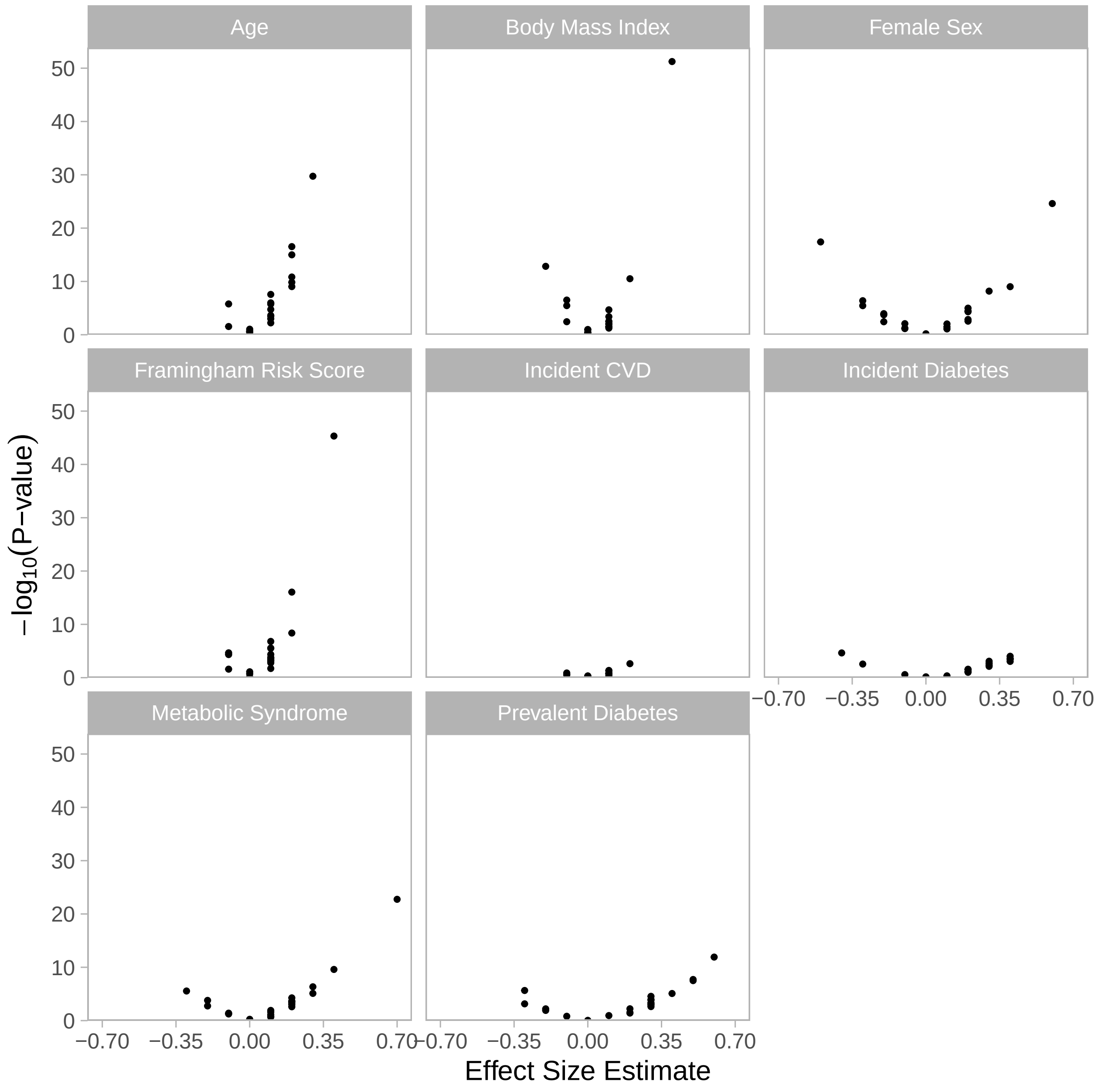
```
volcano_plot <-
  volcano_plot +
  scale_x_continuous(
    'Effect Size Estimate',
    limits =  c(min_lim, max_lim),
    breaks = seq(min_lim, max_lim, length.out = 5)
  ) +
  scale_y_continuous(
    expression(paste(-log[10]('P-value'))),
    # These two lines Set bottom of plot to be 0, while keeping spacing up top
    limits = c(0, NA),
    expand = expand_scale(mult = c(0, 0.05))
  ) +
  thm

volcano_plot
```



Faceting

The above code creates a single volcano 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 volcano plot for each different biological measure of interest in our data. Luckily, `ggplot2` makes this very easy through 'faceting'.

```r
volcano_plot_facet <-
  volcano_plot +
  facet_wrap(~response)

volcano_plot_facet
```



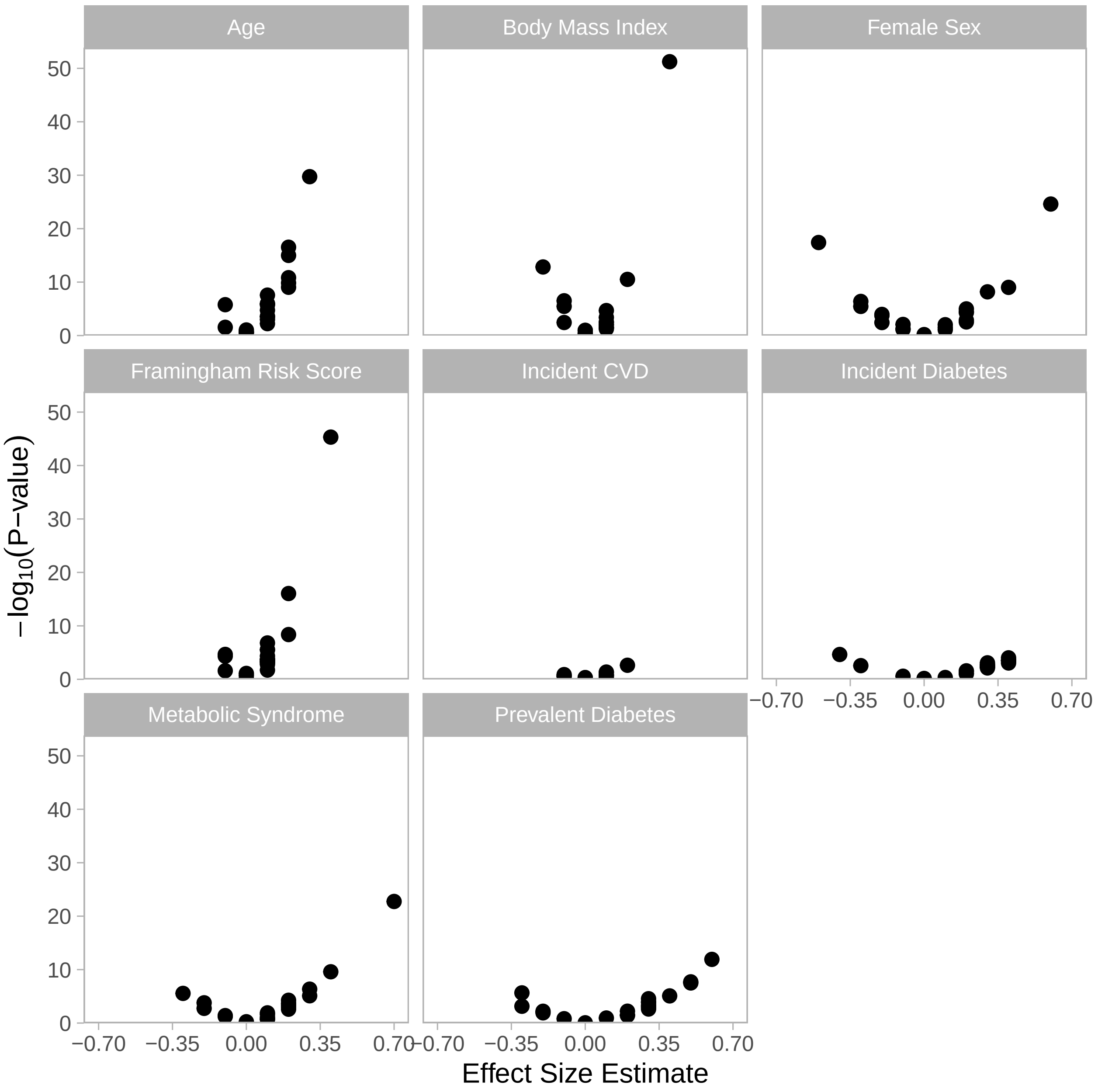
Additional Plot Adjustments

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`.

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

scales_thm_and_facet <-
  list(
    scale_x_continuous(
      'Effect Size Estimate',
      limits =  c(min_lim, max_lim),
      breaks = seq(min_lim, max_lim, length.out = 5)
    ),

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

    thm,

    facet_wrap( ~ response)
  )
```

```
volcano_plot_size <-
  ggplot(plot_data) +
  geom_point(aes(x = estimate, y = p.value), size = 4) +
  scales_thm_and_facet

volcano_plot_size
```



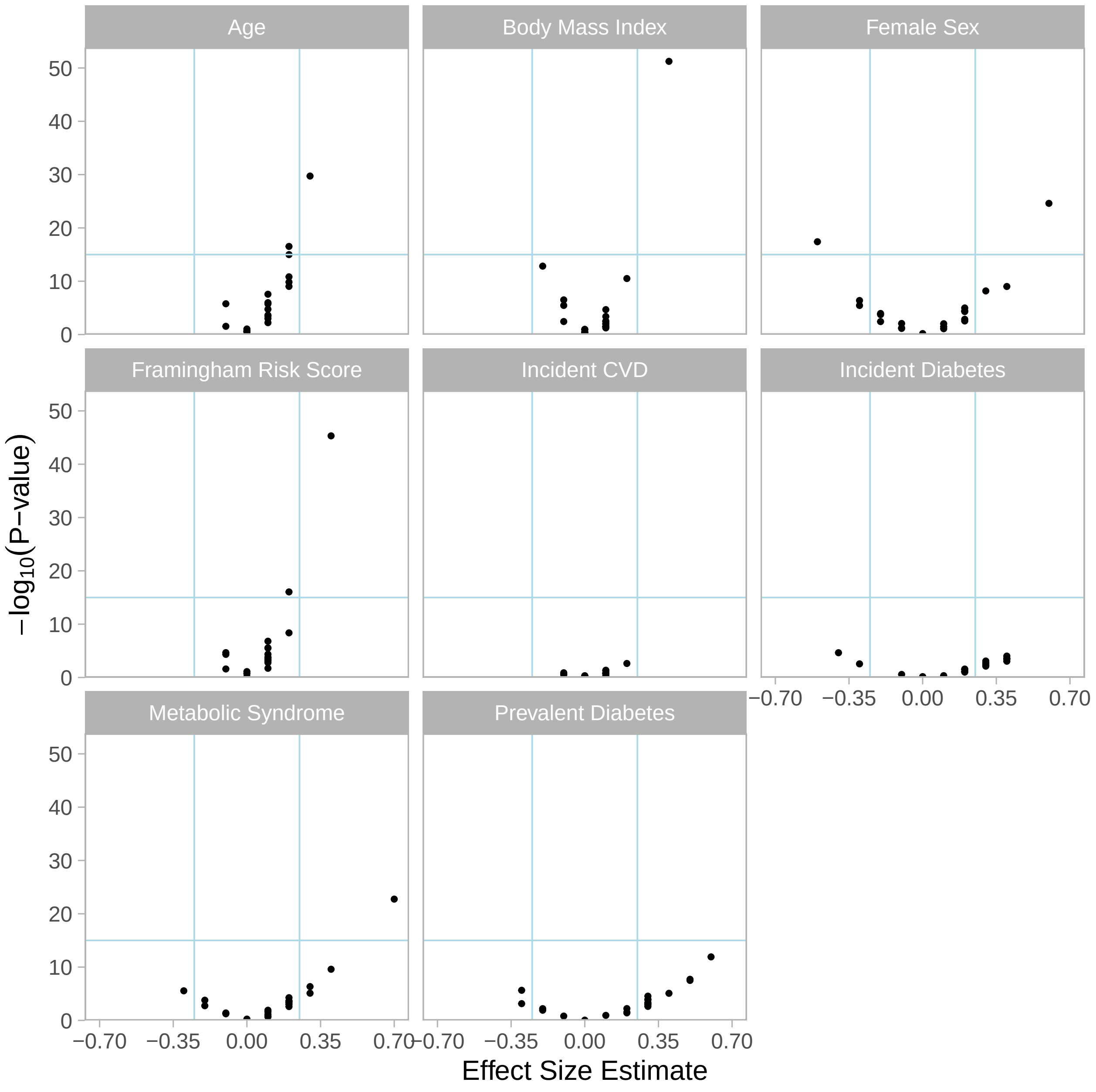
Marking values of interest

Often, we will want to divide a volcano plot into sections based on P-values and effect size estimates. This allows results with both high P-values and extreme effect size estimates to be easily identified. This can be done with lines and/or color.

Lines

We can add lines marking the P-value and effect size estimate thresholds such that all metabolites are considered 'of interest' if they have values more extreme than the thresholds. In this case, we set the P-value threshold to 15 and the effect size estimate threshold to ± 0.25.

```r
pv_threshold <- 15
ese_threshold <- 0.25

volcano_plot_line <-
  ggplot(plot_data) +
  geom_point(aes(x = estimate, y = p.value)) +
  # Add P-value threshold line
  geom_hline(yintercept = pv_threshold, colour = 'lightblue') +
  # Add effect size estimate lines
  geom_vline(xintercept = ese_threshold, colour = 'lightblue') +
  geom_vline(xintercept = -1 * ese_threshold, colour = 'lightblue') +
  scales_thm_and_facet

volcano_plot_line
```

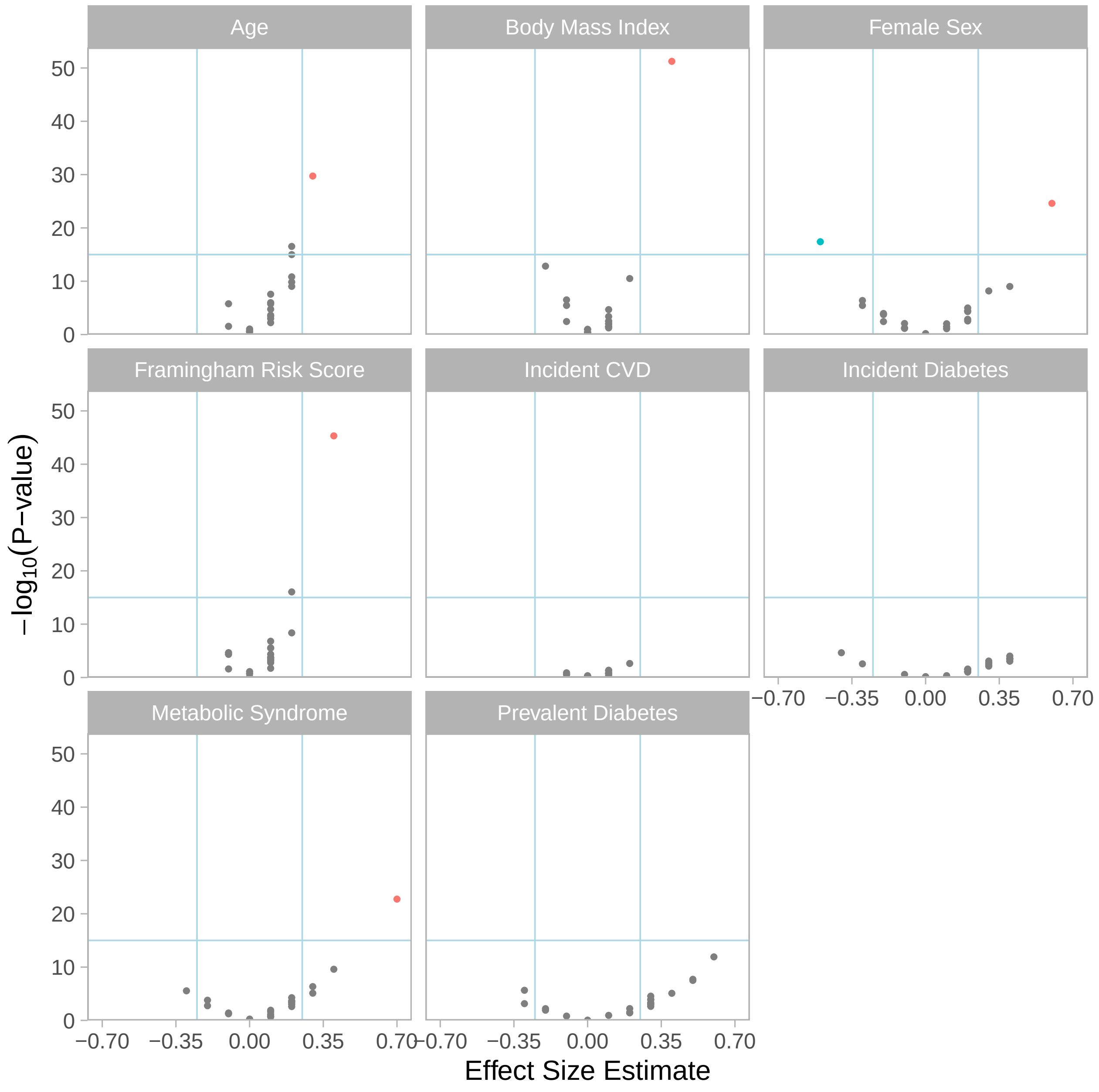


Color

We can also color values that are of interest. In order to do this, we will need to edit the original data and create a column marking values that are more extreme than the P-value and effect size estimate thresholds.

```r
plot_data <-
  plot_data %>%
  mutate(pv_extreme = p.value > pv_threshold,
         es_high = estimate > ese_threshold,
         es_low = estimate < -1 * ese_threshold) %>%
  mutate(
    extreme = case_when(
      pv_extreme & es_high ~ 'high',
      pv_extreme & es_low  ~ 'low',
      TRUE                 ~ NA_character_
    ))
```

```
# Remove color legend
scales_thm_and_facet <-
  c(scales_thm_and_facet,  scale_color_discrete(guide = FALSE))


volcano_plot_line_color <-
  # Use edited data
  ggplot(plot_data) +
  # Color by more extreme values
  geom_point(aes(x = estimate, y = p.value, color = extreme)) +
  geom_hline(yintercept = pv_threshold, colour = 'lightblue') +
  geom_vline(xintercept = ese_threshold, colour = 'lightblue') +
  geom_vline(xintercept = -1 * ese_threshold, colour = 'lightblue') +
  scales_thm_and_facet

volcano_plot_line_color
```



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 volcano 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_label_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)
```

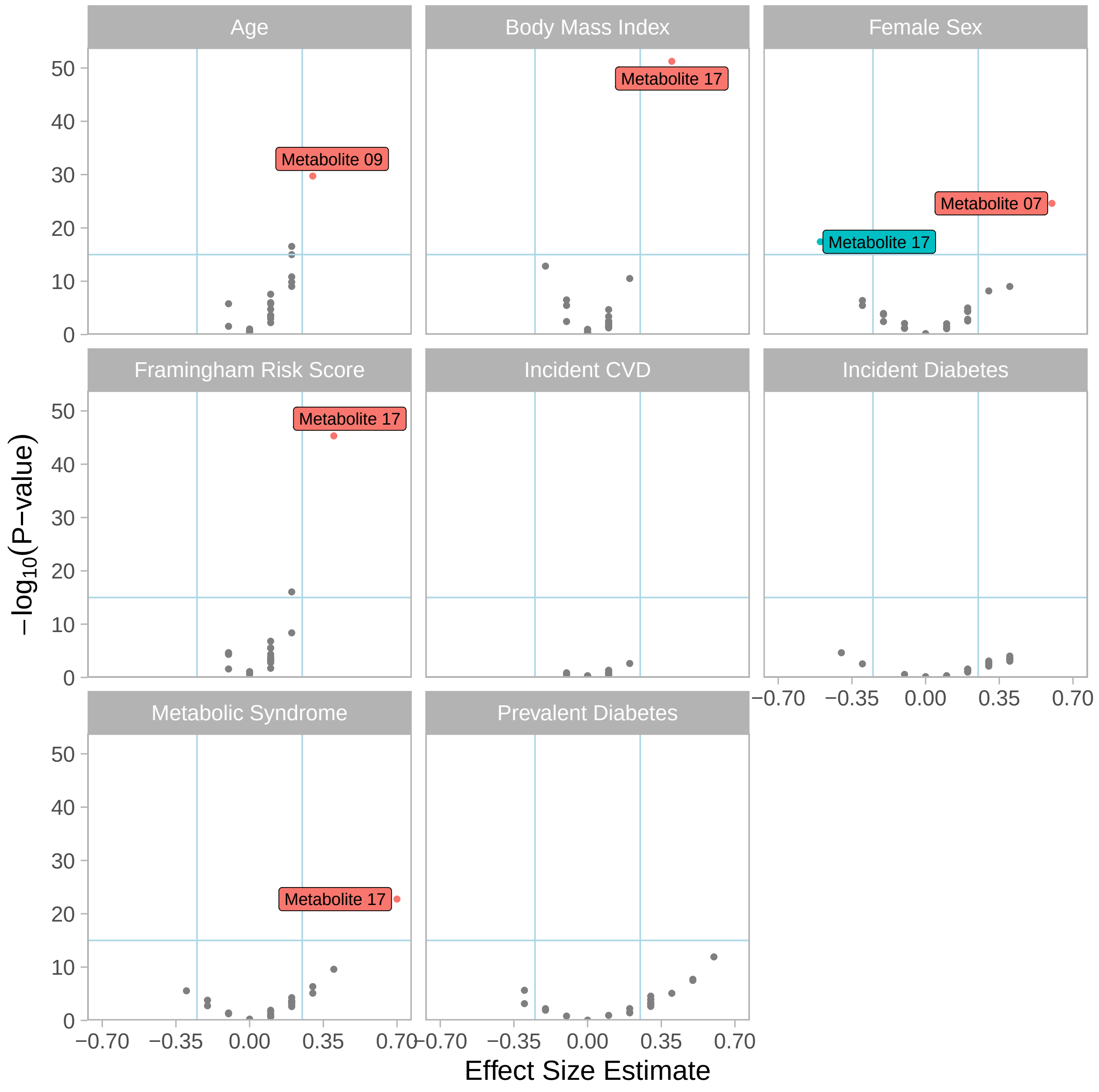
```
## Warning: package 'ggrepel' was built under R version 3.5.2
```

```
plot_data <-
  plot_data %>%
```

```r
  mutate(label = ifelse(is.na(extreme), NA, term))
```

```r
# Remove fill legend
scales_thm_and_facet <-
  c(scales_thm_and_facet, scale_fill_discrete(guide = FALSE))

volcano_plot_line_color_label <-
  # Use edited data
  ggplot(plot_data) +
  geom_point(aes(x = estimate, y = p.value, color = extreme)) +
  geom_hline(yintercept = pv_threshold, colour = 'lightblue') +
  geom_vline(xintercept = ese_threshold, colour = 'lightblue') +
  geom_vline(xintercept = -1 * ese_threshold, colour = 'lightblue') +
  # Adding labels last draws them on top of threshold lines
  geom_label_repel(aes(x = estimate, y = p.value, fill = extreme, label = label)) +
  scales_thm_and_facet

volcano_plot_line_color_label
```

```
## Warning: Removed 162 rows containing missing values (geom_label_repel).
```



Adding statistics

We might want to annotate our plots with summary statistics that make it easier to convey a high level summary of the data. One such measure might be a count of how many values are extreme enough to be of interest. The first step is to create a new dataset that contains the annotation information we wish to add.

```
extreme_count <-
  plot_data %>%
  count(response, wt = !is.na(extreme))

extreme_count
```

```
## # A tibble: 8 x 2
##   response                  n
##   <chr>                 <int>
## 1 Age                        1
## 2 Body Mass Index            1
## 3 Female Sex                 2
## 4 Framingham Risk Score      1
## 5 Incident CVD               0
## 6 Incident Diabetes          0
## 7 Metabolic Syndrome         1
## 8 Prevalent Diabetes         0
```

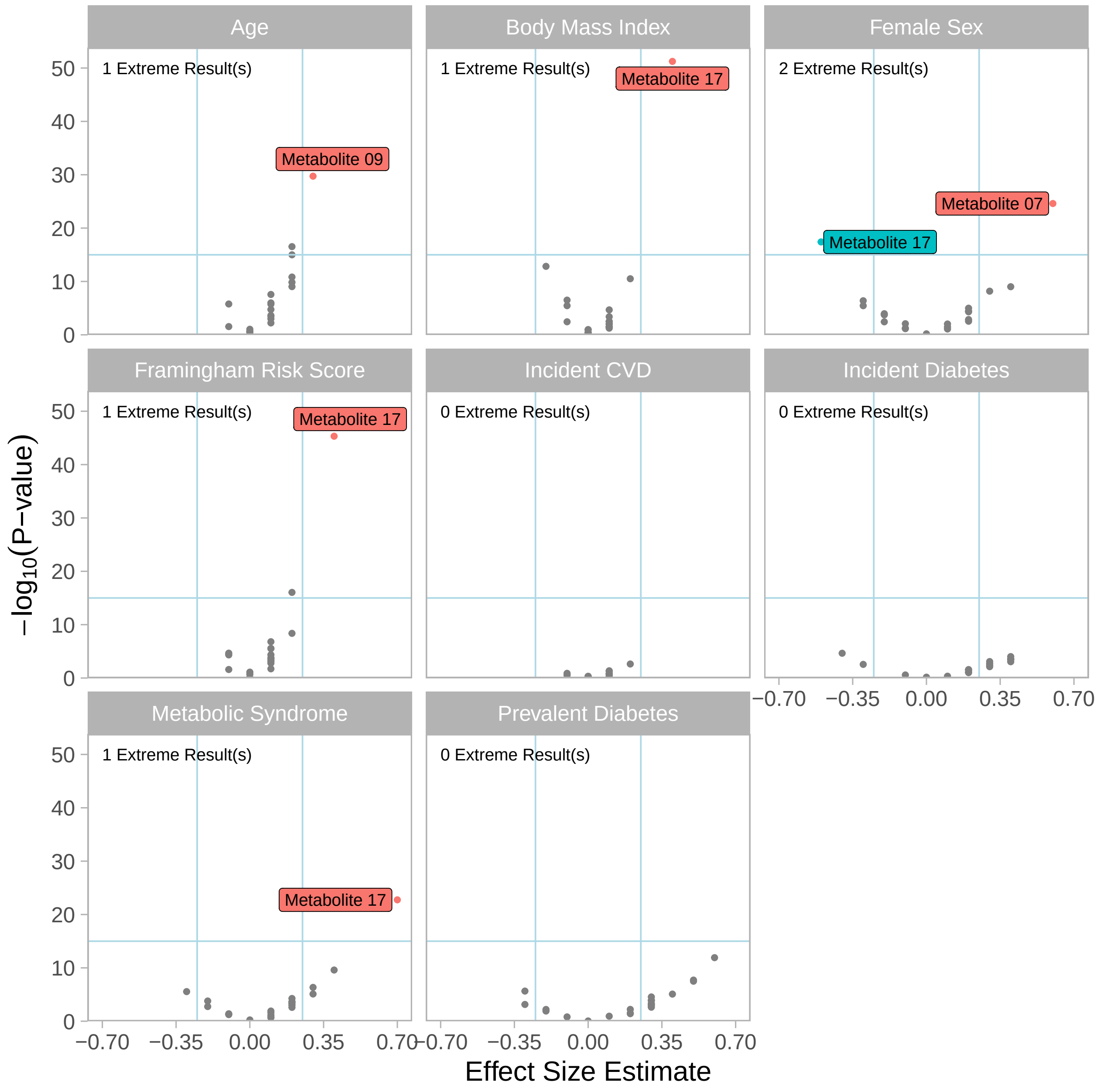
Next we will need to add additional columns based on what we would like the exact wording and location of the annotation to be. In this case, we are placing the annotation in the top left corner with a short message.

```
extreme_count  <-
  extreme_count %>%
  mutate(x = -0.7, y = 50) %>%
  mutate(label = paste(n, 'Extreme Result(s)'))

extreme_count
```

```
## # A tibble: 8 x 5
##   response                  n     x     y label
##   <chr>                 <int> <dbl> <dbl> <chr>
## 1 Age                       1  -0.7    50 1 Extreme Result(s)
## 2 Body Mass Index           1  -0.7    50 1 Extreme Result(s)
## 3 Female Sex                2  -0.7    50 2 Extreme Result(s)
## 4 Framingham Risk Score     1  -0.7    50 1 Extreme Result(s)
## 5 Incident CVD              0  -0.7    50 0 Extreme Result(s)
## 6 Incident Diabetes         0  -0.7    50 0 Extreme Result(s)
## 7 Metabolic Syndrome        1  -0.7    50 1 Extreme Result(s)
## 8 Prevalent Diabetes        0  -0.7    50 0 Extreme Result(s)
```

```
volcano_plot_line_color_label_anno <-
  volcano_plot_line_color_label +
  # Use the extreme_count data.
  # hjust = 0 puts the start of the label at the xy coordinate
  geom_text(aes(x = x, y = y, label = label),
            hjust = 0,
            data = extreme_count)


volcano_plot_line_color_label_anno
```

```
## Warning: Removed 162 rows containing missing values (geom_label_repel).
```


Age
1 Extreme Result(s)
Metabolite 09
Body Mass Index
1 Extreme Result(s)
Metabolite 17
Female Sex
2 Extreme Result(s)
Metabolite 07
Metabolite 17
Framingham Risk Score
1 Extreme Result(s)
Metabolite 17
Incident CVD
0 Extreme Result(s)
Incident Diabetes
0 Extreme Result(s)
Metabolic Syndrome
1 Extreme Result(s)
Metabolite 17
Prevalent Diabetes
0 Extreme Result(s)
$-\log_{10}$(P−value)
Effect Size Estimate
0
10
20
30
40
50
−0.70
−0.35
0.00
0.35
0.70