Day 7 : Volcano Plots in R

Ambu Vijayan

Bioinformatician

BioLit, Thiruvananthapuram

A volcano plot is a type of scatter plot represents differential expression of features (genes for example): on the x-axis we typically find the fold change and on the y-axis the p-value..

R packages and data

Load readr

```
library(readr)
```

Import data

```
data <- read_tsv("https://raw.githubusercontent.com/sdgamboa/misc_datasets/master/L0_vs_L20.tsv")
```

```
dim(data)
```

A simple volcano plot

```
plot1 <- ggplot(data, aes(logFC, -log(FDR, 10))) + geom_point(size = 2/5) +
xlab(expression("log"[2]*"FC")) + ylab(expression("-log"[10]*"FDR"))
```

```
plot1
```

The dispersion of the points (representing genes) in the plot is similar to, you guessed, a volcano (that's why they're called volcano plots). Since the FDR values were transformed to their -log10, the higher the position of a point, the more significant its value is (y axis). Points with positive fold change values (to the right) are up-regulated and points with negative fold change values (to the left) are down-regulated (x axis).

Adding color to differentially expressed genes (DEGs)

Differentially expressed genes (DEGs) are usually considered as those with an absolute fold change greater or equal to 2 and a FDR value of 0.05 or less.

So, we can make our volcano plot a bit more informative if we add some color to the DEGs in the plot.

To do so, we'll add an additional column, named 'Expression', indicating whether the expression of a gene is up-regulated, down-regulated, or unchanged:

```
data <- mutate(data, Expression = case_when(logFC >= log(2) & FDR <= 0.05 ~ "Up-regulated", logFC <= -log(2) & FDR <= 0.05 ~ "Down-regulated", TRUE ~ "Unchanged"))
```

```
head(data)
```

Creating a new variable Expression based on the values of logFC and FDR. If logFC is greater than or equal to log(2) and FDR is less than or equal to 0.05, it's labeled as "Up-regulated". If logFC is less than or equal to -log(2) and FDR is less than or equal to 0.05, it's labeled as "Down-regulated". Otherwise, it's labeled as "Unchanged".

mutate(data, ...): This function takes a data frame (data in this case) and modifies it by adding or modifying variables.

Expression = case_when(...): This creates a new variable called Expression and assigns values based on conditions specified in case_when().

logFC >= log(2) & FDR <= 0.05 ~ "Up-regulated": If the condition logFC >= log(2) & FDR <= 0.05 is true, then the corresponding value for Expression is set to "Up-regulated".

logFC <= -log(2) & FDR <= 0.05 ~ "Down-regulated": If the condition logFC <= -log(2) & FDR <= 0.05 is true, then the corresponding value for Expression is set to "Down-regulated".

TRUE ~ "Unchanged": If none of the above conditions are true, then the default value for Expression is set to "Unchanged"

We can now map the column ´Expression' to the color aesthetic of geom_point() and color the points according to their expression classification:

```
plot2 <- ggplot(data, aes(logFC, -log(FDR, 10))) + geom_point(aes(color = Expression), size = 2/5) + xlab(expression("log"[2]*"FC")) + ylab(expression("-log"[10]*"FDR")) + scale_color_manual(values = c("dodgerblue3", "gray50", "firebrick3")) + guides(colour = guide_legend(override.aes = list(size=1.5)))
```

```
plot2
```

If we want to know how many genes are up- or down-regulated, or unchanged, we can use dplyr's count() function.

```
count(data, Expression)
```

Since we already know that the genes towards the right are up-regulated and the genes towards the left are down-regulated, it would be more informative if we colored the points according to their significance level instead. Let's create another column, named 'Significance', and classify the genes according to significance thresholds (0.05, 0.01, and 0.001):

```
data <- mutate(data, Significance = case_when( abs(logFC) >= log(2) & FDR <= 0.05 & FDR > 0.01 ~ "FDR 0.05", abs(logFC) >= log(2) & FDR <= 0.01 & FDR > 0.001 ~ "FDR 0.01", abs(logFC) >= log(2) & FDR <= 0.001 ~ "FDR 0.001", TRUE ~ "Unchanged"))
```

```
head(data)
```

Again, we can use the color aesthetic to map the color of the points to their corresponding significance thresholds:

```
plot3 <- ggplot(data, aes(logFC, -log(FDR,10))) +   geom_point(aes(color = Significance), size = 2/5) +   xlab(expression("log"[2]*"FC")) +   ylab(expression("-log"[10]*"FDR")) +   scale_color_viridis_d() +   guides(colour = guide_legend(override.aes = list(size=1.5)))
```

```
p3
```

ggplot(data, aes(logFC, -log(FDR, 10))): This sets up the basic ggplot object with the data and aesthetic mappings. It specifies that logFC should be plotted on the x-axis and -log(FDR, 10) on the y-axis.

geom_point(aes(color = Significance), size = 2/5): This adds a layer of points to the plot. The color of each point is determined by the "Significance" variable. The size parameter sets the size of the points.

xlab(expression("log"[2]*"FC")): This sets the x-axis label as "log2FC" using mathematical notation.

ylab(expression("-log"[10]*"FDR")): This sets the y-axis label as "-log10FDR" using mathematical notation.

scale_color_viridis_d(): This sets the color scale for the points using the Viridis color palette. The "_d" in "viridis_d" stands for "continuous" scale.

guides(colour = guide_legend(override.aes = list(size=1.5))): This adjusts the size of the legend key (color legend) to 1.5 times the default size.

And we can count how many genes are up- or down-regulated according to the different significance thresholds with count():

```
count(data, Expression, Significance)
```

Adding labels to selected genes

```
top <- 10
```

Filter and arrange for Up-regulated genes

```
top_upregulated <- head(within(data[data$Expression == 'Up-regulated', ], {rank <- order(FDR, -abs(logFC))}),top)
```

Filter and arrange for Down-regulated genes

```
top_downregulated <- head(within(data[data$Expression == 'Down-regulated', ],
{rank <- order(FDR, -abs(logFC))}),top)
```

Combine the results

```
top_genes <- rbind(top_upregulated, top_downregulated)
```

```
top_genes
```

```
install.packages("ggrepel")
```

```
library(ggrepel)
```

ggrepel package, an extension of ggplot2 that provides functions for avoiding overplotting of text labels

```
plot3 <-  plot3 + geom_label_repel(data = top_genes, mapping = aes(logFC, -log(FDR,10), label = Genes), size = 2)
```

```
plot3
```

How to avoid overlaps?

```
plot3 <- ggplot(data, aes(logFC, -log(FDR, 10))) + geom_point(aes(color = Significance), size = 2/5) + xlab(expression("log"[2]*"FC")) + ylab(expression("-log"[10]*"FDR")) + scale_color_viridis_d() + guides(colour = guide_legend(override.aes = list(size = 1.5))) + geom_label_repel(data = top_genes, mapping = aes(logFC, -log(FDR, 10), label = Genes), size = 2, max.overlaps = 200)
```

So, only the top 10 up- and down-regulated genes are labeled, avoiding overcrowding.