# Differential Expression Using Limma

2024-03-31

## Differential Expression Using Limma

**Differential expression analysis can be conducted with numerous R packages including DEseq2, EdgeR, and limma-voom. Here, limma-voom is employed. In addition, the impact of repeated measures (biological sample) can be assessed using functions such as duplicateCorrelation or with the Dream package which slots into normal limma workflows. Comparisons between the two methods are made before Dream is employed to calculate differentially expressed genes between treatment groups. Finally, volcano plots and euler plots are made to visualize the results of differential expression and to compare DEGs between treatment groups.**

### Prepare environment

```
rm(list = ls(all.names = TRUE)) # clears global environ.

# Load packages
library(tidyverse) # for data cleaning
library(dplyr)
library(broom)
library(tibble)
library(edgeR)
library(statmod)
library(variancePartition)
library(BiocParallel)
library(ggplot2)
library(EnhancedVolcano)
library(eulerr)
```

### Import data and format

```
load("Selected_Raw_Data.RDATA")

Treatment <- factor(RNAseq$Treatment,
                    levels = c('Control',
                               'Treatment_1',
                               'Treatment_2',
                               'Treatment_3'))

Biological_Sample <- factor(RNAseq$Biological_Sample,
                            levels = seq(1, 10, 1))

# Create DGEList object
# My samples are grouped based on the 'merge' variable
# Set yours to whatever your group designation variable is
```

```
dataset <- DGEList(as.matrix(t(RNAseq[3:ncol(RNAseq)])),
                   group = Treatment)

# Print the dimensions of the new object (Genes x Samples)
dim(dataset)

## [1] 503  40

# It is highly recommended to read up on this function to decide if you force
through origin
# and how you will design the regression model. Briefly, if you will compare
treatment groups to
# each other rather than solely to a universal control, use ~0.
model <- model.matrix(~0 + Treatment)

# Check if any coefficients in model cannot be estimated
# Return value of "NULL" means model has full column rank
nonEstimable(model)

## NULL
```

**If a repeated measure experimental design is employed, you can account for repeated measures in the differential expression. Two major approaches are considered here. The first uses the function duplicateCorrelation while the other uses the Dream package.**

DuplicateCorrelation was originally employed for technical replicates in microarrays, but can be adapted to account for repeated measures. This function incorporates a single random effect (biological replicate) and calculates a single correlation for all genes. This requires the assumption that all genes have the same correlation structure, which is likely inappropriate for such large datasets as RNAseq. The advantage of duplicatCorrelation is vastly shorter compute times compared to Dream.

```
# Run voom and plot mean-variance trend
# voom function automatically converts counts to logCPM
# by extracting library size and calculating normalization factors
dup <- voom(dataset,
            design = model,
            plot = FALSE)

# Measure correlation between repeated samples to correct for baseline
changes in expression between individuals
# block variable depends on where you save your sample ID info for each
sample
corfit <- duplicateCorrelation(dup,
                               model,
                               block = Biological_Sample)

# Rerun voom, adding in blocking of Sample.ID and corfit
dup <- voom(dataset,
            design = model,
```

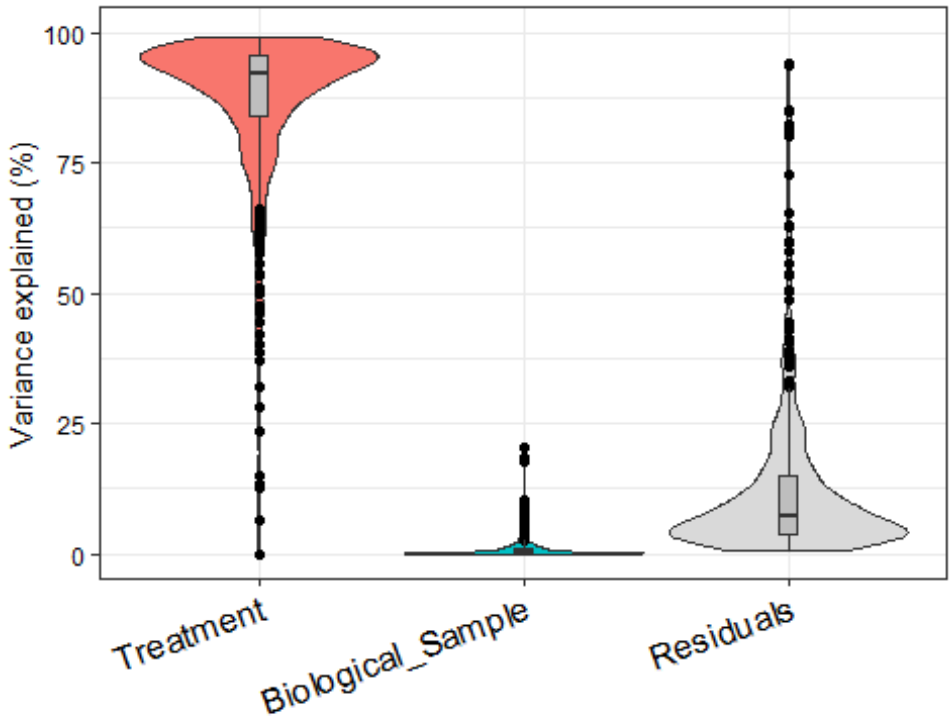
```
            plot = FALSE,
            block = Biological_Sample,
            correlation = corfit$consensus)

# Rerun duplicateCorrelation
corfit <- duplicateCorrelation(dup,
                               model,
                               block = Biological_Sample)

# Fit linear model using weighted least squares for genes
dup_fit <- lmFit(dup,
                 model,
                 block = Biological_Sample,
                 correlation = corfit$consensus)

list <- list(Treatment_1 =
               makeContrasts(TreatmentTreatment_1 - TreatmentControl,
                             levels = model),

             Treatment_2 =
               makeContrasts(TreatmentTreatment_2 - TreatmentControl,
                             levels = model),

             Treatment_3 =
               makeContrasts(TreatmentTreatment_3 - TreatmentControl,
                             levels = model),

             #########

             Treatment_1_vs_Treatment_2 =
               makeContrasts(TreatmentTreatment_1 - TreatmentTreatment_2,
                             levels = model),

             Treatment_1_vs_Treatment_3 =
               makeContrasts(TreatmentTreatment_1 - TreatmentTreatment_3,
                             levels = model),

             ########

             Treatment_2_vs_Treatment_3 =
               makeContrasts(TreatmentTreatment_2 - TreatmentTreatment_3,
                             levels = model))

# Calculate DEGs
dup_DEGs <- lapply(list, function(x){

  tmp <- contrasts.fit(dup_fit, x)
  tmp <- eBayes(tmp)
```

```
  topTable(tmp, sort.by = "P", n = Inf)

})

# Results are the following information:

# logFC: log2 fold change of group1/group0
# AveExpr: Average expression across all samples, in log0 CPM
# t: logFC divided by its standard error
# P.Value: Raw p-value (based on t) from test that logFC differs from 0
# adj.P.Val: Benjamini-Hochberg false discovery rate adjusted p-value
# B: log-odds that gene is DE

dup_DEGs <- lapply(dup_DEGs, function(x){

  tmp <- data.frame(cbind(x[["logFC"]],
                          x[["P.Value"]],
                          x[["adj.P.Val"]],
                          x["AveExpr"]),
                    row.names = rownames(x))

  colnames(tmp) <- c('log2FC', 'P.Value', 'adj.P.Val', 'Expression')
  tmp

})

names(dup_DEGs) <- c('T1_vs_Control',
                     'T2_vs_Control',
                     'T3_vs_Control',
                     'T1_vs_T2',
                     'T1_vs_T3',
                     'T2_vs_T3')
```

**In contrast to duplicateCorrelation, the Dream package can be employed. Dream calculates gene-wise sample correlations, solving the problematic assumption made when using duplicateCorrelation. However, this process takes significantly longer as a variance term must be calculated for each gene.**

```
# Set up parallel processing parameters
param <- SnowParam(4, "SOCK", progressbar = TRUE)

# Will model a linear mixed effect model where treatment group
# information is a fixed effect and sample information is a random
# effect. ~ 0 indicates no intercept (forced through origin)
# which allows for comparisons between all groups. If a universal reference
# group is present for your model (only comparing treatment to control, not
# treatment 1 to treatment 2, etc) remove the 0.
model <- ~ 0 + Treatment + (1 | Biological_Sample)

meta <- RNAseq[1:2]
```

```r
# Create contrast matrix for dream
list <- makeContrastsDream(model,
                           meta,
                           contrasts = c(T1_vs_Control =
"(TreatmentTreatment_1 - TreatmentControl)",
                                         T2_vs_Control =
"(TreatmentTreatment_2 - TreatmentControl)",
                                         T3_vs_Control =
"(TreatmentTreatment_3 - TreatmentControl)",

                                         T1_vs_T2 = "(TreatmentTreatment_1 -
TreatmentTreatment_2)",
                                         T1_vs_T3 = "(TreatmentTreatment_1 -
TreatmentTreatment_3)",

                                         T2_vs_T3 = "(TreatmentTreatment_2 -
TreatmentTreatment_3)"))

# Calculate voom with Dream weights
dream <- voomWithDreamWeights(dataset,
                              model,
                              meta,
                              BPPARAM = param)

## iteration: 1234

# Test impact of each variable on gene variance
form <- ~ (1 | Biological_Sample) + (1 | Treatment)
vp <- fitExtractVarPartModel(dream, form, meta)

Varplot <- plotVarPart(sortCols(vp))

print(Varplot)
```




```r
# Fit linear model using Dream weights
dream_fit <- dream(dream,
                   model,
                   meta,
                   list)

# Empirical Bayes Statistics for Differential Expression
dream_fit <- eBayes(dream_fit)

contrasts <- c('T1_vs_Control',
               'T2_vs_Control',
               'T3_vs_Control',
               'T1_vs_T2',
               'T1_vs_T3',
               'T2_vs_T3')

# Calculate DEGs
dream_DEGs <- lapply(contrasts, function(x){

  topTable(dream_fit, coef = x, sort.by = "P", n = Inf)

})


# Results are the following information:
```

```
# logFC: log0 fold change of group1/group0
# AveExpr: Average expression across all samples, in log0 CPM
# t: logFC divided by its standard error
# P.Value: Raw p-value (based on t) from test that logFC differs from 0
# adj.P.Val: Benjamini-Hochberg false discovery rate adjusted p-value
# B: log-odds that gene is DE

# Isolate key DEG readouts
dream_DEGs <- lapply(dream_DEGs, function(x){

  tmp <- data.frame(cbind(x[["logFC"]],
                          x[["P.Value"]],
                          x[["adj.P.Val"]],
                          x["AveExpr"]),
                    row.names = rownames(x))

  colnames(tmp) <- c('log2FC', 'P.Value', 'adj.P.Val', 'Expression')
  tmp

})

names(dream_DEGs) <- c('T1_vs_Control',
                       'T2_vs_Control',
                       'T3_vs_Control',
                       'T1_vs_T2',
                       'T1_vs_T3',
                       'T2_vs_T3')
```

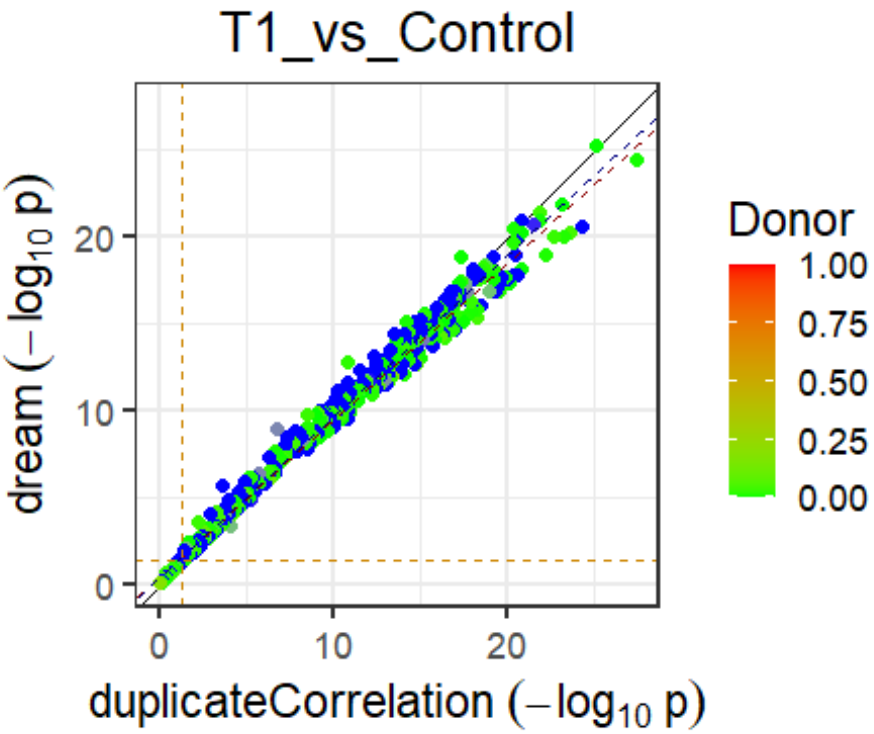
### Lets compare the results of differential expression analysis using duplicateCorrelation versus Dream

Each point is a gene colored by the fraction of expression variation explained by variance across biological replicate. Black line represents a slope of 1. Red dashed line represents best fit line for the 20% highest while blue line represents fit line for 20% lowest expression variance explained by biological replicate variance. Orange dashed lines represent significance cutoffs (p value = 0.05, -log10P = 1.3)

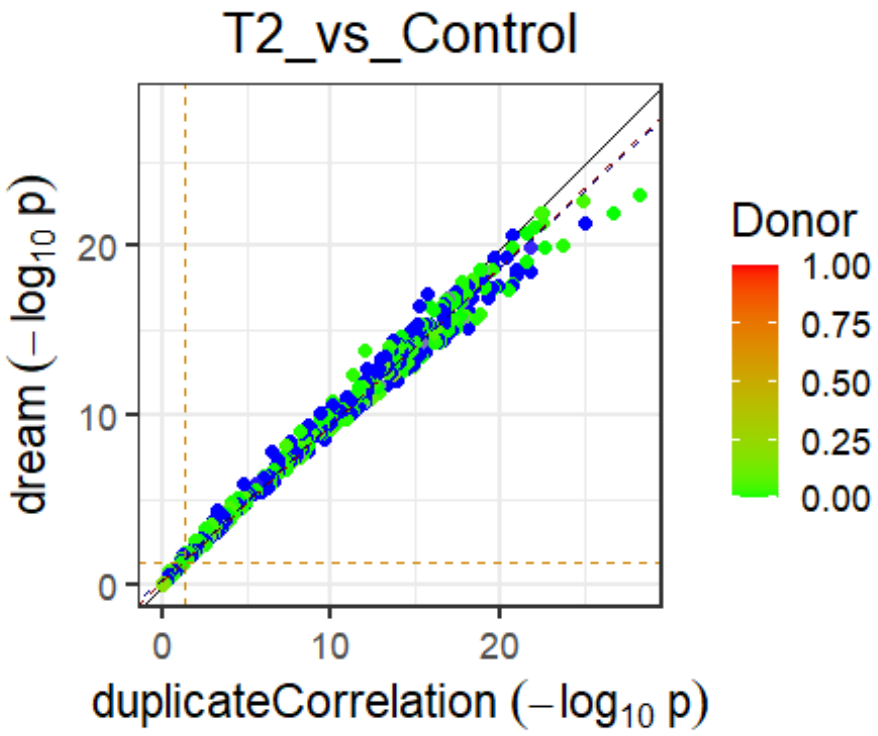
```
lapply(contrasts, function(z){

  tmp <- dup_DEGs[[z]]
  names <- rownames(tmp)
  tmp2 <- dream_DEGs[[z]][names, ]

  plotCompareP(tmp$adj.P.Val,
               tmp2$adj.P.Val,
               vp$Biological_Sample,
               corfit$consensus) +

    ggtitle(as.character(z)) +
```

```
    geom_hline(yintercept = 1.3,
               linetype = "dashed",
               color = "orange3") +
    geom_vline(xintercept = 1.3,
               linetype = "dashed",
               color = "orange3")

})
```

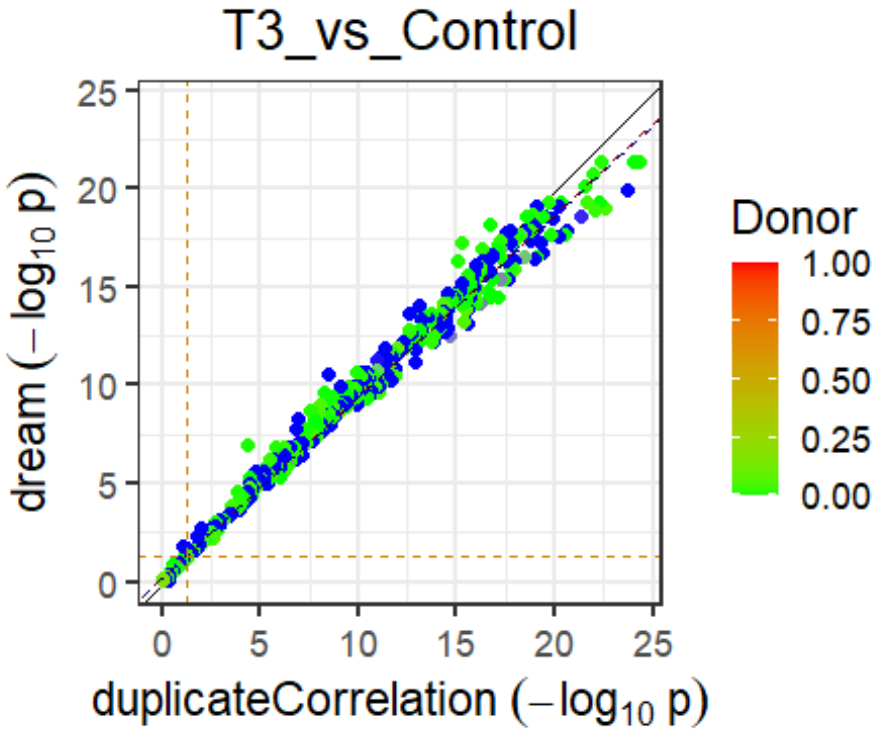
```
## [[1]]
```



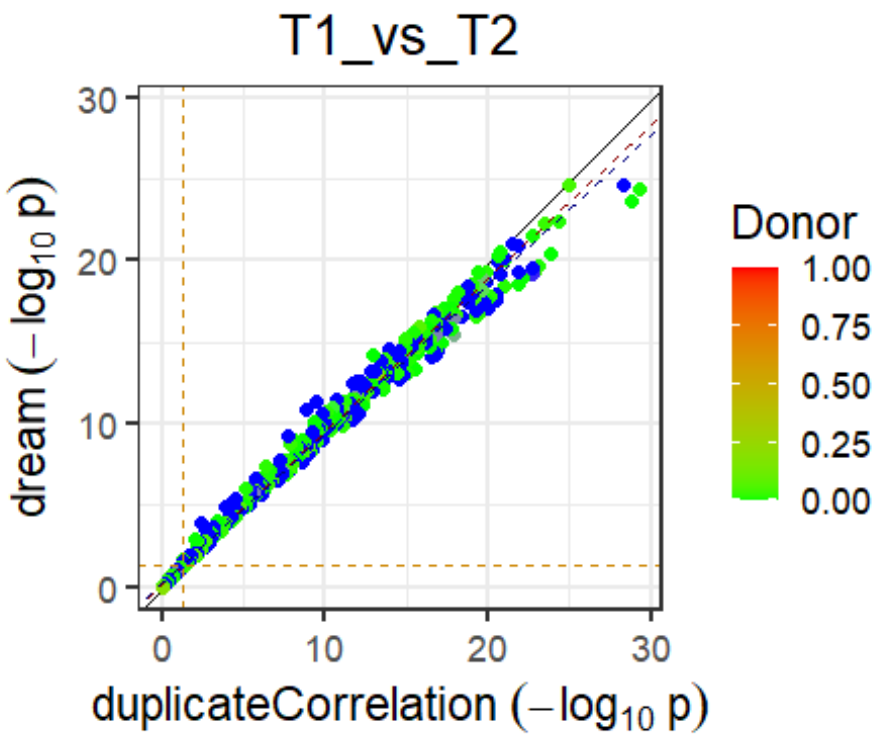

```
##
## [[2]]
```


T2_vs_Control
dream (−log10 p)
duplicateCorrelation (−log10 p)
0
10
20
Donor
1.00
0.75
0.50
0.25
0.00

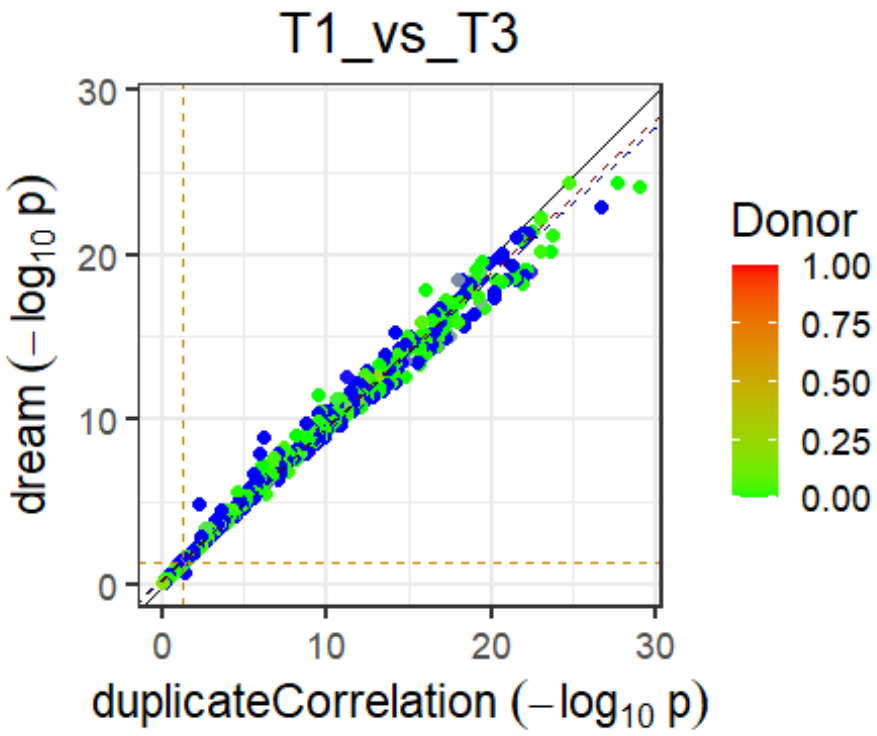
```
##
## [[3]]
```


T3_vs_Control
dream (−log10 p)
duplicateCorrelation (−log10 p)
0
5
10
15
20
25
Donor
1.00
0.75
0.50
0.25
0.00

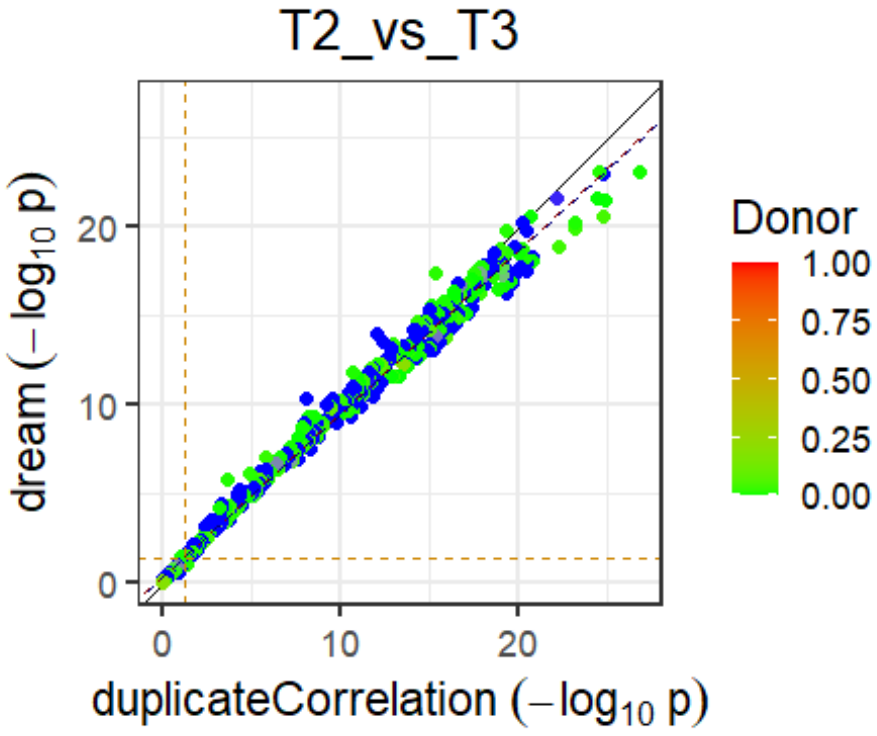
```
##
## [[4]]
```


T1_vs_T2
dream (−log10 p)
duplicateCorrelation (−log10 p)
0
10
20
30
Donor
1.00
0.75
0.50
0.25
0.00

```
##
## [[5]]
```


T1_vs_T3
dream (−log10 p)
duplicateCorrelation (−log10 p)
Donor
1.00
0.75
0.50
0.25
0.00
0
10
20
30

```
##
## [[6]]
```

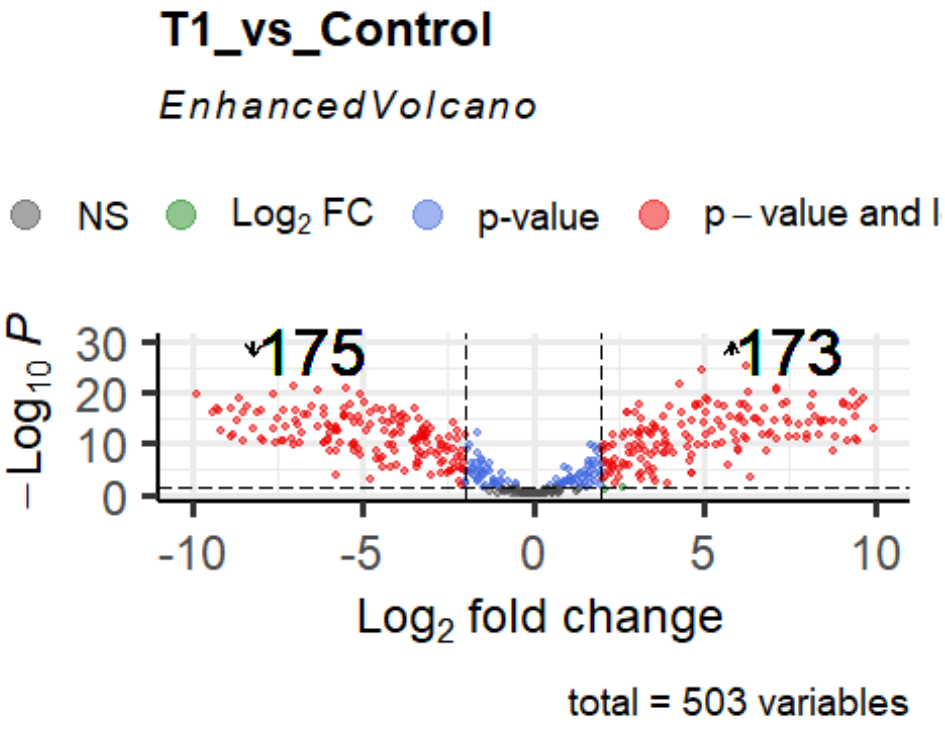

T2_vs_T3
dream (−log10 p)
duplicateCorrelation (−log10 p)
Donor
1.00
0.75
0.50
0.25
0.00
0
10
20

### Visualize the results of differential expression

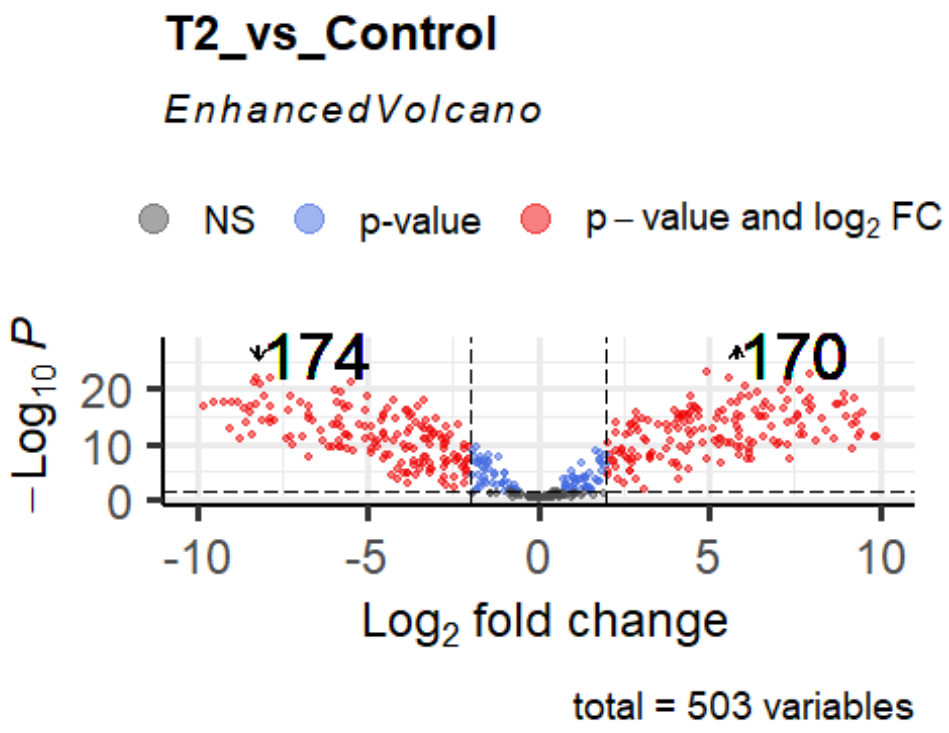
Despite the longer compute time (and the general lack of the effect of biological sample on this synthetic data set), Dream is likely the superior method to account for the impact of biological replicates in repeated measures designs.

```
# Make volcano plots for each comparison
lapply(names(dream_DEGs), function(z){

  # Isolate and count number of upregulated genes
  up <- dream_DEGs[[z]] %>% count(log2FC > 2 &
                                  adj.P.Val < 0.05)

  if(nrow(up) == 2){up <- up[2, 2]}
  else{up <- 0}

  # Isolate and count number of downregulated genes
  down <- dream_DEGs[[z]] %>% count(log2FC < -2 &
                                    adj.P.Val < 0.05)

  if(nrow(down) == 2){down <- down[2, 2]}
  else{down <- 0}

  # Save rownames (ensembl IDs)
  sorting <- rownames(dream_DEGs[[z]])

  # Make volcano plots with hgnc gene labels and comparison as title
  plot <- EnhancedVolcano(dream_DEGs[[z]],
                          lab = NA,
                          # lab = rownames(dream_DEGs[[z]]),
                          x = 'log2FC',
                          y = 'adj.P.Val',
                          title = z,
                          FCcutoff = 2,
                          pCutoff = 0.05,
                          pointSize = 1,
                          xlim = c(-10, 10))

  # Save the y axis max
  max <- max(plot[["plot_env"]][["ylim"]])

  # Add counts of up and downregulated genes
  # Add arrows indicating up/downregulation
  plot <- plot +

    # Downregulated gene number
    ggplot2::geom_text(x = -6.5,
                       y = max* 0.95,
                       label = paste0(down),
```

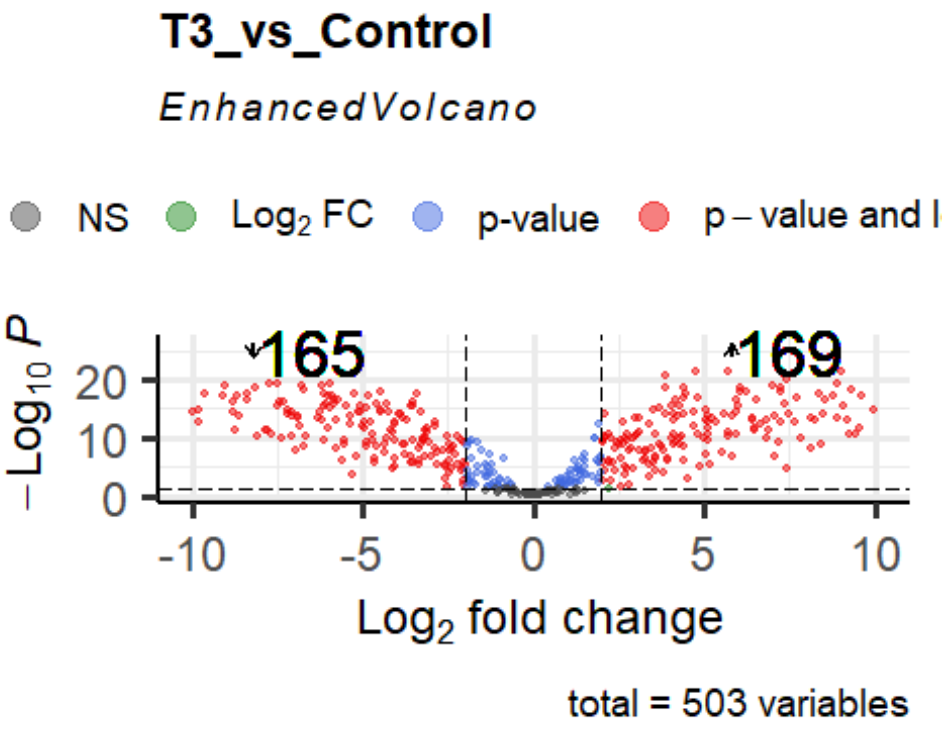
```
                        size = 8.5) +

    # Downregulated gene arrow
    ggplot2::geom_segment(x = -8.2,
                          y = max,
                          xend = -8.2,
                          yend = max * 0.9,
                          arrow = arrow(length = unit(2, "mm")),
                          linewidth = 1) +

    # Upregulated gene number
    ggplot2::geom_text(x = 7.5,
                       y = max * 0.95,
                       label = paste0(up),
                       size = 8.5) +

    # Upregulated gene arrow
    ggplot2::geom_segment(x = 5.8,
                          y = max * 0.9,
                          xend = 5.8,
                          yend = max,
                          arrow = arrow(length = unit(2, "mm")),
                          linewidth = 1)
})

## [[1]]
```



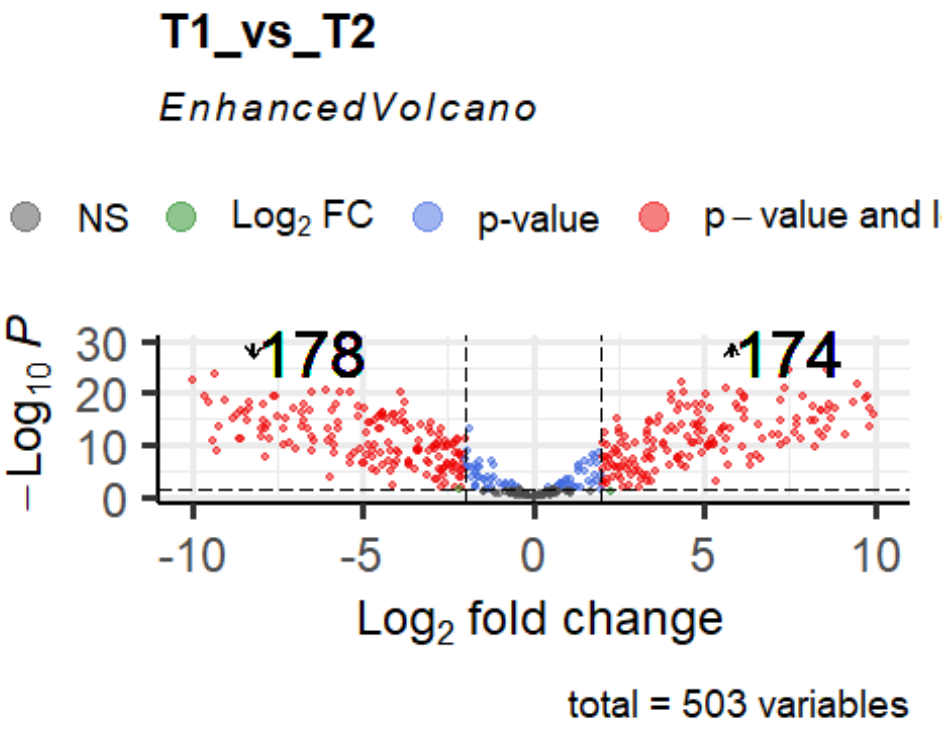
```
##
## [[2]]
```


T2_vs_Control
EnhancedVolcano
NS
p-value
p – value and log2 FC
174
170
−Log10 P
20
10
0
-10
-5
0
5
10
Log2 fold change
total = 503 variables

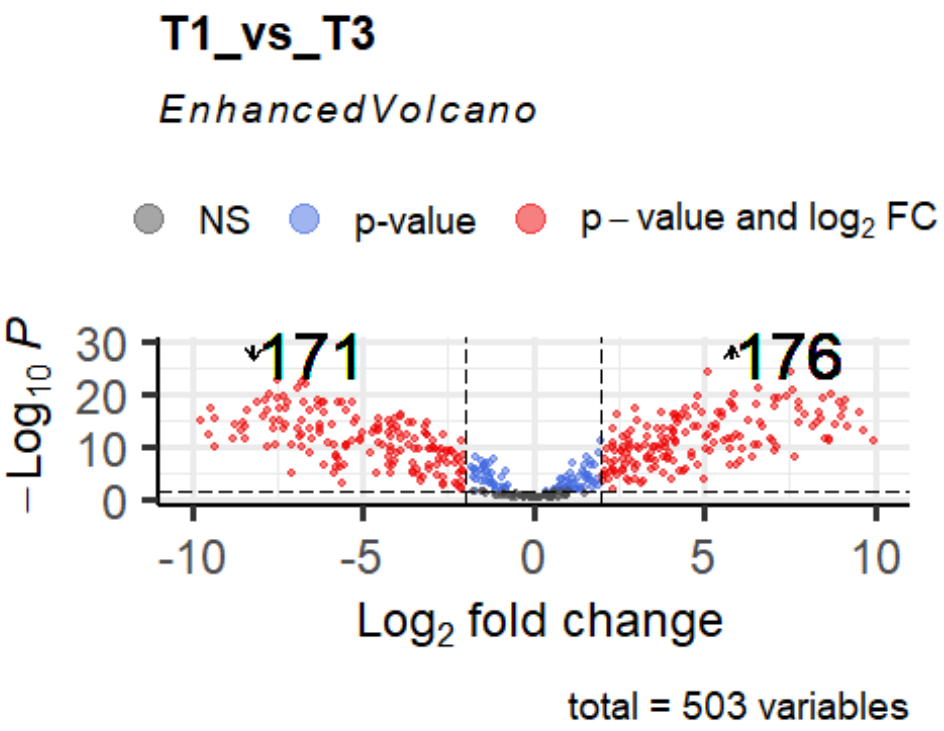
```
##
## [[3]]
```


T3_vs_Control
EnhancedVolcano
NS
Log2 FC
p-value
p – value and l
165
169
−Log10 P
20
10
0
-10
-5
0
5
10
Log2 fold change
total = 503 variables

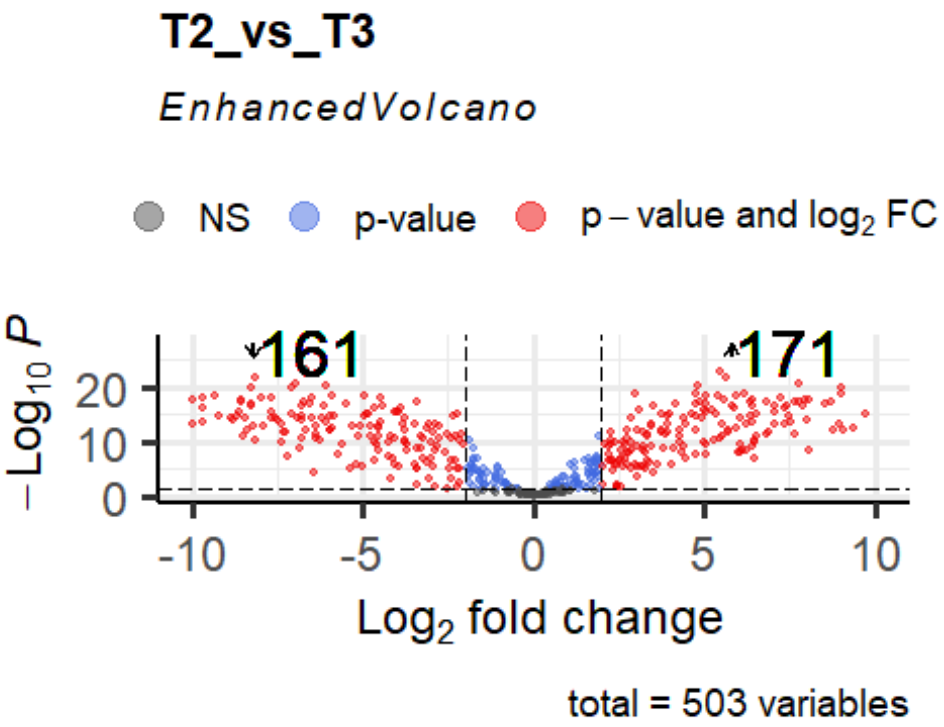
```
##
## [[4]]
```


T1_vs_T2
EnhancedVolcano
NS
Log2 FC
p-value
p – value and l
178
174
−Log10 P
30
20
10
0
-10
-5
0
5
10
Log2 fold change
total = 503 variables

```
##
## [[5]]
```


T1_vs_T3
EnhancedVolcano
NS
p-value
p – value and log2 FC
−Log10 P
30
20
10
0
171
176
-10
-5
0
5
10
Log2 fold change
total = 503 variables

```
##
## [[6]]
```



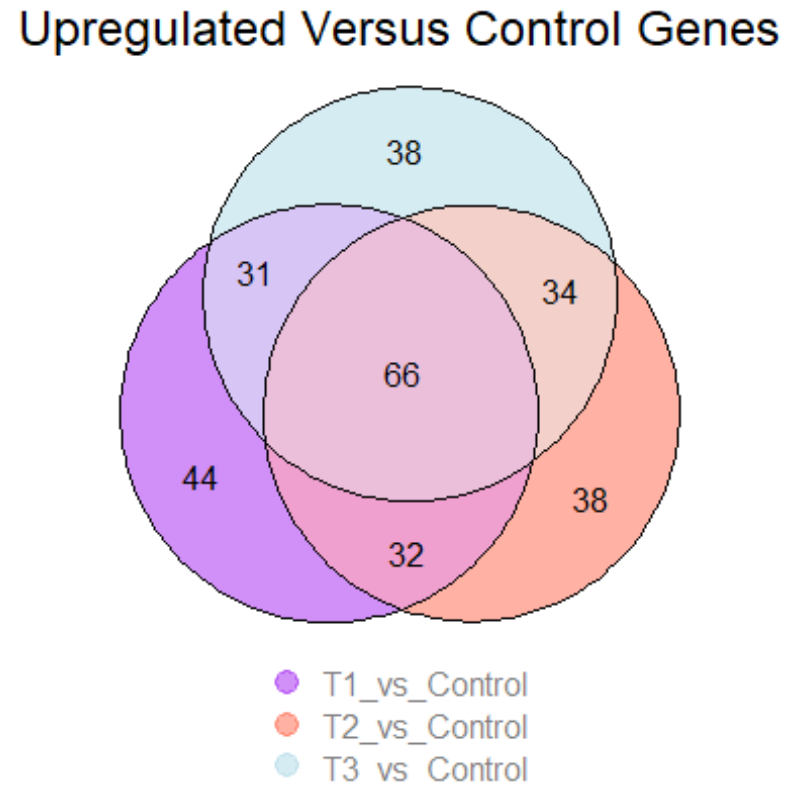

### Another potential method for visualization is the euler plot

Euler plots are similar to venn diagrams, but we have found them easier to interpret in some cases.

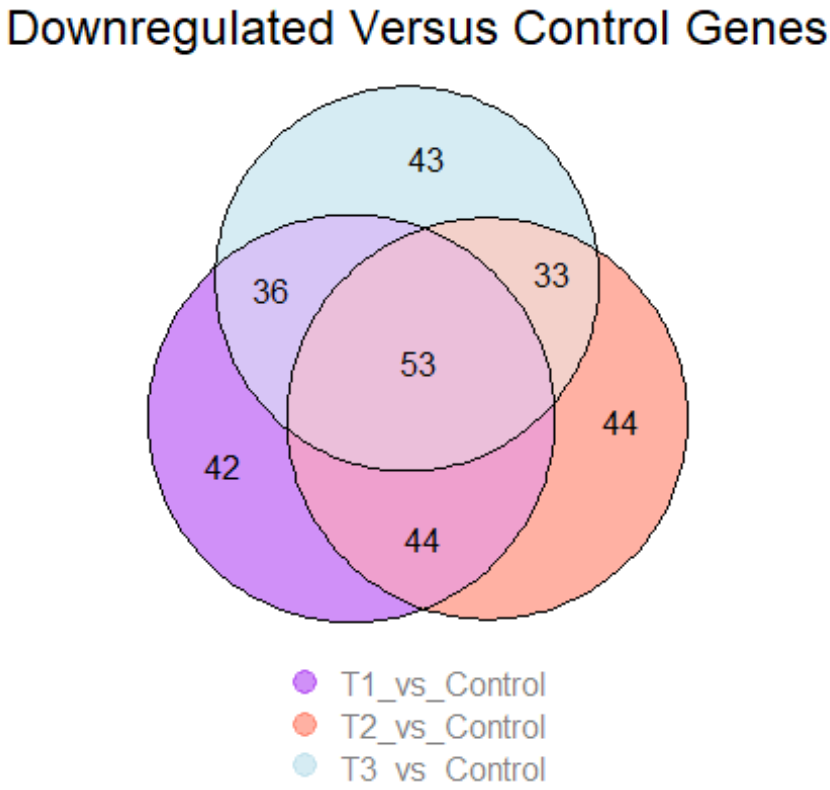
```
list <- names(dream_DEGs)

# Isolate significantly up/downregulated genes for each indicated group
Comparisons <- lapply(list, function(x){

  up <- dream_DEGs[[x]] %>% subset(adj.P.Val < 0.05 & (log2FC >= 2)) %>%
rownames_to_column()
  down <- dream_DEGs[[x]] %>% subset(adj.P.Val < 0.05 & (log2FC <= -2)) %>%
rownames_to_column()

  tmp <- list(up, down)
  names(tmp) <- c('Up', 'Down')
  tmp

})

names(Comparisons) <- list

list <- list(as.list(names(dream_DEGs[1:3])),
             as.list(names(dream_DEGs[4:6])))
```

```r
names(list) <- c('Control', 'Treatment')

lapply(names(list), function(x){

  # Will isolate upregulated and downregulated genes for separate plots
  list_direction <- list('Up', 'Down')

  # Create euler plots for up or downregulated genes
  lapply(list_direction, function(y){

    # Create list containing unique gene names from each group
    # This list has only up or downregulated genes at a given time
    eul_list <- list(unique(Comparisons[[list[[x]][[1]]]][[y]]$rowname),
                     unique(Comparisons[[list[[x]][[2]]]][[y]]$rowname),
                     unique(Comparisons[[list[[x]][[3]]]][[y]]$rowname))

    # Set names of list to match source group
    names(eul_list) <- c(list[[x]])

    # Set colors for groups
    col <- c('purple',
             'tomato',
             'lightblue')

    # Fit the euler plot
    fit <- euler(eul_list)

    # Add fills, legend, text size, axis limits
    fit <- plot(fit,
                fills = list(fill = col, alpha = 0.5),
                legend = list(side = 'bottom', col = col, text.col =
'black'),
                quantities = list(cex = 1),
                main = paste0(y, 'regulated Versus ', x, ' Genes'),
                xlim = c(-15, 15),
                ylim = c(-15, 15))
    })
})
```

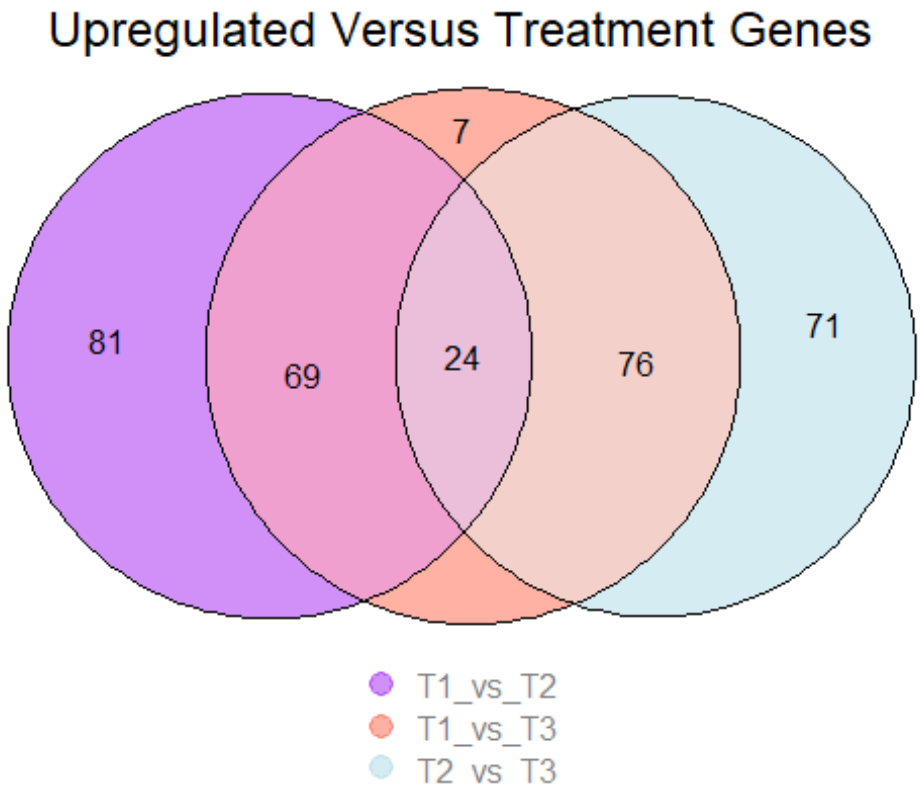
```
## [[1]]
## [[1]][[1]]
```


Upregulated Versus Control Genes
38
31
34
66
44
38
32
T1_vs_Control
T2_vs_Control
T3 vs Control

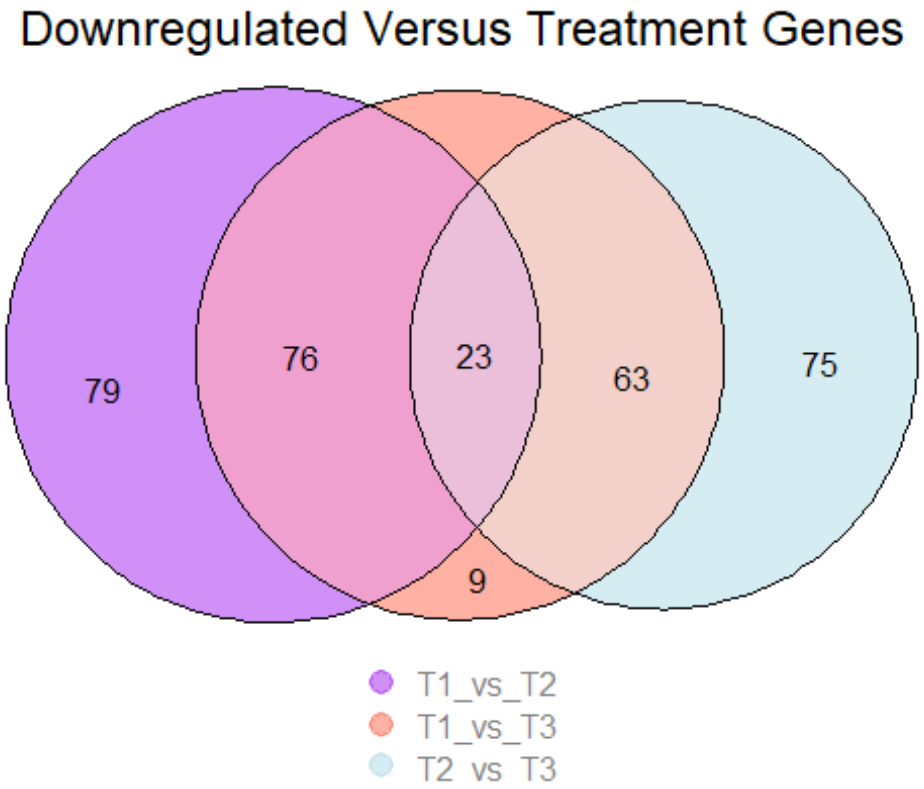
```
##
## [[1]][[2]]
```


Downregulated Versus Control Genes
43
36
33
53
44
42
44
T1_vs_Control
T2_vs_Control
T3 vs Control

```
##
##
## [[2]]
## [[2]][[1]]
```


Upregulated Versus Treatment Genes
81
69
24
7
76
71
T1_vs_T2
T1_vs_T3
T2_vs_T3

```
##
## [[2]][[2]]
```




**Results of differential expression can then be exported**

```
# Export all DEG data
# lapply(names(dream_DEGs), function(z){
#
#   write.csv(dream_DEGs[[z]], file = paste0(z, "", ".csv"))
#
# })
```