Appendix A9: Linear modelling of time-series gene expression data

Table 7.2 and Figure 2

GSE51387:

```
expr_data <- read.csv("Mus_SARS_GSE51387.csv", header = T, stringsAsFactors = T,
na.strings = c("","NA"))
attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X.9", "WT_MA15.D4")
setnames(expr_data, "X.10", "WT_MA15.D4")
setnames(expr_data, "X.11", "WT_MA15.D4")

setnames(expr_data, "X.12", "WT_MA15.D7")
setnames(expr_data, "X.13", "WT_MA15.D7")


setnames(expr_data, "X.14", "WT_Mock.D4")
setnames(expr_data, "X.15", "WT_Mock.D4")

setnames(expr_data, "X.16", "WT_Mock.D7")
setnames(expr_data, "X.17", "WT_Mock.D7")


expr_data1 <- expr_data[20:32638, c(1, 2, 13:21)]

write.csv(expr_data1, file = "GSE51387_dataset.csv")


# Read the dataset
data <- read.csv("GSE51387_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
```

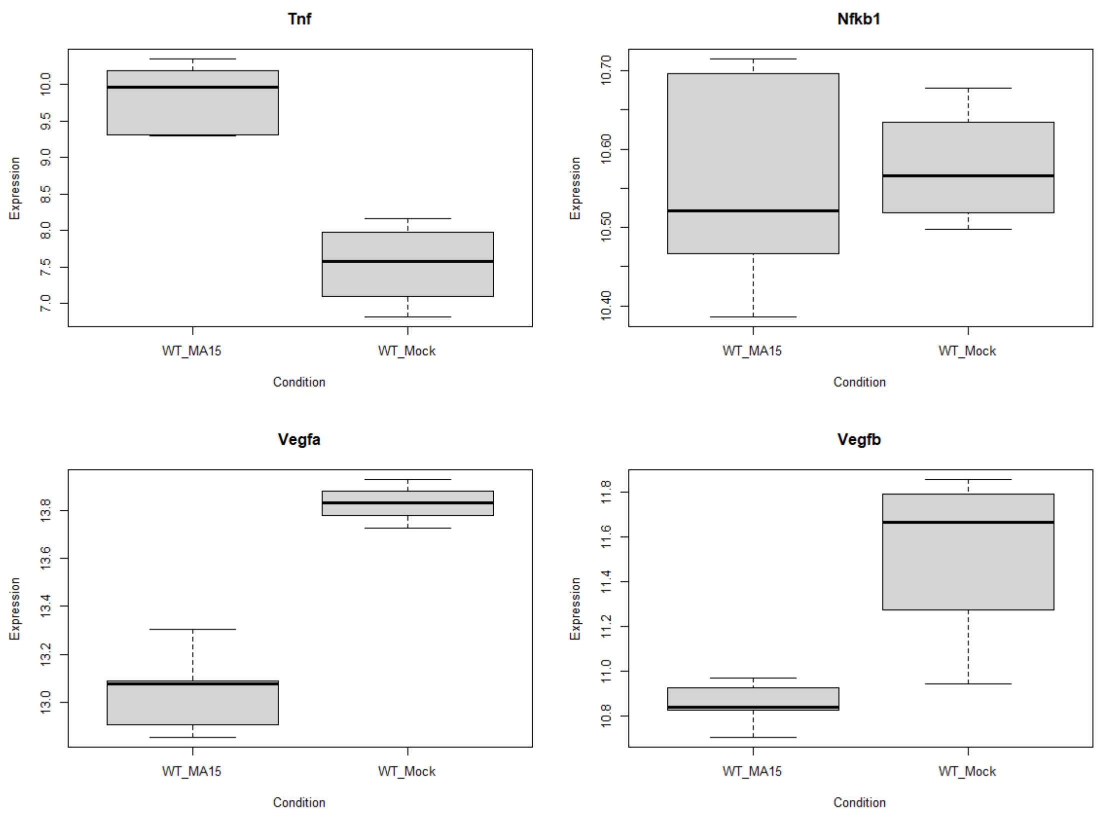
```r
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")

probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)
```

```r
combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()
```

```
# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

first_table1<- df1
first_table1$dataset <- "GSE51387"

first_table1
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.28	8.81	6.44	0.666	0.0004	0.0004	GSE51387
-0.0197	10.6	-0.244	-5.04	0.8140	0.8140	GSE51387
-0.78	13.4	-8.06	2.02	8.66e-05	8.66e-05	GSE51387
-0.679	11.2	-3.67	-2.22	0.0079	0.0079	GSE51387

```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



```
final_summary <- final %>%
  dplyr::group_by(GENE_SYMBOL, Type, Measurement) %>%
  dplyr::summarize(MeanExpression = mean(Expression), .groups = 'drop')

elegant_plot <- ggplot(final_summary, aes(x = Measurement, y = MeanExpression, fill =
Type)) +
  geom_boxplot(position = position_dodge(width = 0.7)) +
  facet_wrap(~GENE_SYMBOL, scales = 'free_y') +
  theme_minimal() +
  theme(
    text = element_text(size = 12),
    plot.title = element_text(hjust = 0.5),
    legend.position = "bottom",
    strip.text.x = element_text(size = 14, face = "bold"),
    axis.text.x = element_text(angle = 45, hjust = 1)
  ) +
  labs(
    title = "Gene Expression under Different Conditions",
    x = "Time Point",
    y = "Mean Expression Level",
    fill = "Type"
  ) +
  scale_fill_brewer(palette = "Pastel1")
```

Change in Time:

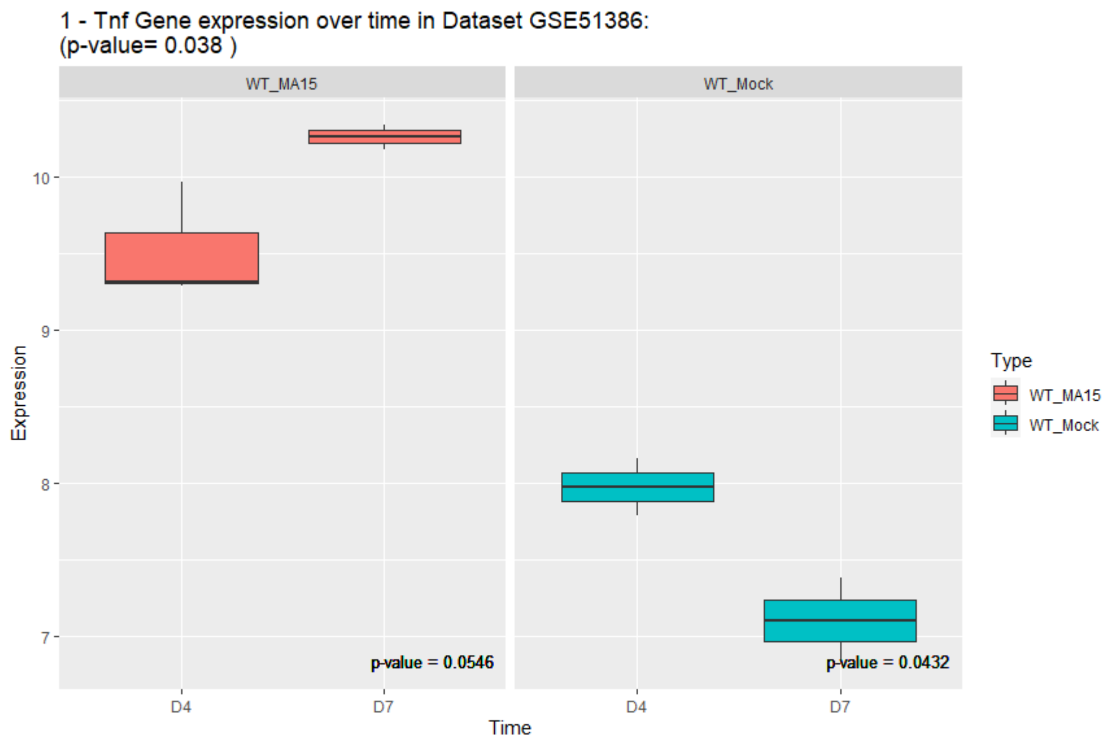
```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
```

```
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA15 vs Mock group
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D4.vs.D7_WT_MA15", "D4.vs.D7_WT_Mock"),
number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
```

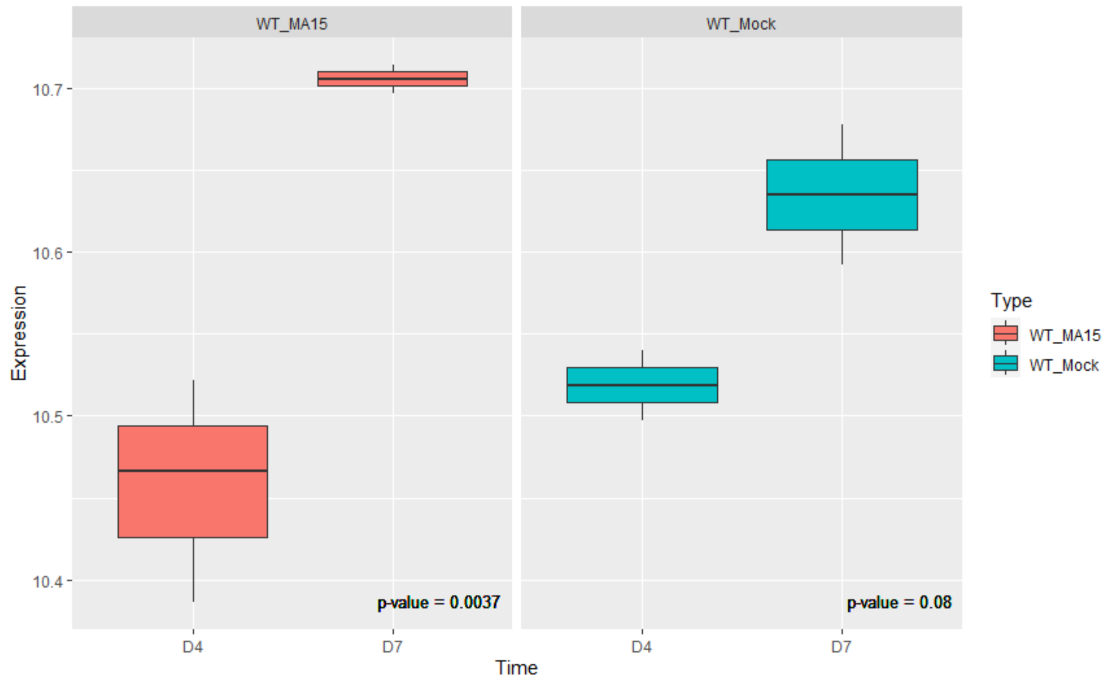
```r
  df2 <- results[[i]]


  pvalues <- c(df2$P.Value)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  df$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Define the column names for p-values
  pvalue_cols <- c("P.Value", "adj.P.Val", "P.Value_D4.vs.D7_WT_MA15",
"P.Value_D4.vs.D7_WT_Mock")

  # Check if all p-value columns exist in the dataframe
  if (all(pvalue_cols %in% names(df2))) {
    # Store the p-values in a separate data frame
    pvalues <- df2[, pvalue_cols]

    # Remove the p-values from the original data frame
    df2 <- df2[, setdiff(names(df2), names(pvalues))]

    # Round the numbers to 3 decimal points
    df2 <- round(df2, 4)

    # Add the p-values back to the data frame
    df2 <- cbind(df2, pvalues)

    # Store the updated data frame in the list
    results[[i]] <- df2
  }
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D4.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$P.Value_D4.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

first_table2 <- df2
first_table2$dataset <- "GSE51387"
first_table2$GENE_SYMBOL <- rownames(first_table2)

first_table2
```

D4.vs.D7_WT_MA15	D4.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D4.vs.D7_WT_MA15	P.Value_D4.vs.D7_WT_Mock	dataset	GENE_SYMBOL
-0.741	0.875	8.81	6.75	0.0380	0.0380	0.0546	0.0432	GSE51387	Tnf
-0.248	-0.116	10.6	15.5	0.0072	0.0072	0.0037	0.0800	GSE51387	Nfkb1
0.134	-0.0061	13.4	0.439	0.6673	0.6673	0.3921	0.9704	GSE51387	Vegfa
-0.156	0.39	11.2	1.3	0.3520	0.3520	0.5459	0.2005	GSE51387	Vegfb

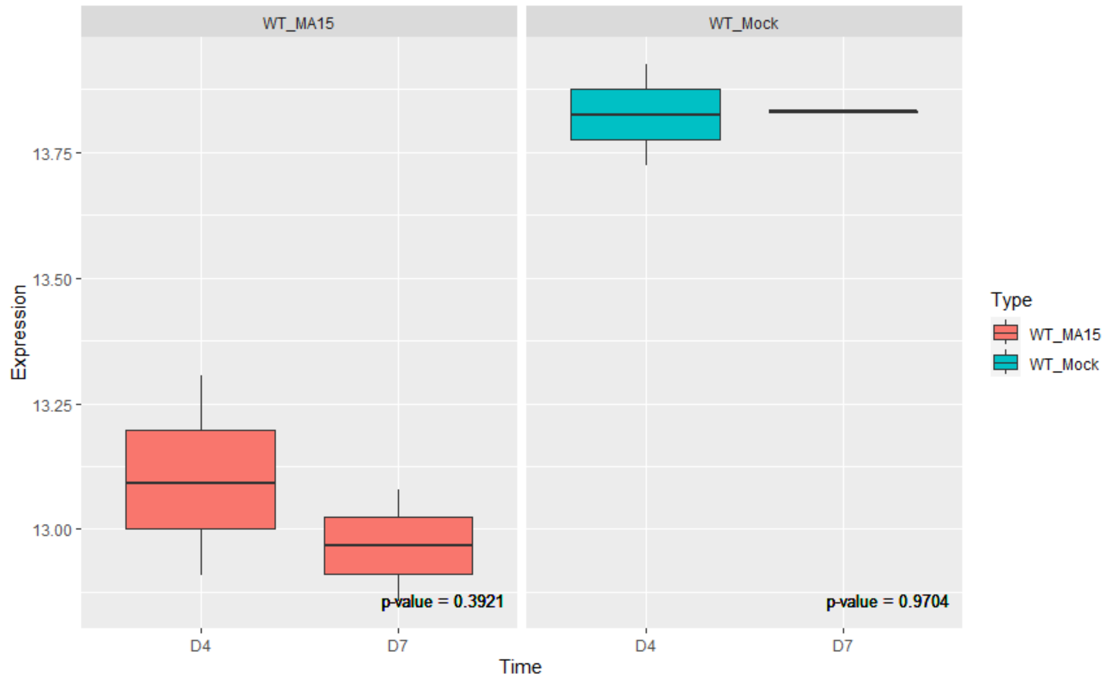
```r
# Convert row names to a column in df2
df2$GENE_SYMBOL <- rownames(df2)

# Ensure GENE_SYMBOL is of type character in both data frames
df2$GENE_SYMBOL <- as.character(df2$GENE_SYMBOL)
final$GENE_SYMBOL <- as.character(final$GENE_SYMBOL)

# Define an offset for the annotation
offset <- 1  # Adjust this based on your data

# Initialize a list to store plots
plots <- list()

# Initialize a counter for the genes
counter <- 1

# Loop over the genes
for (gene in unique(final$GENE_SYMBOL)) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Extract p-values for each type
  p_value_MA15 <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_MA15), 4)
  p_value_Mock <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_Mock), 4)

    overall_p_value <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]$adj.P.Val), 4)
  # Create a boxplot for the current gene
  p <- ggplot(final_gene, aes(x = Measurement, y = Expression, fill = Type)) +
    geom_boxplot() +
    facet_wrap(~Type) +
    labs(title = paste(counter, "-", gene, "Gene expression over time in Dataset
GSE51386:\n(p-value=", overall_p_value, ")")) +
    xlab("Time") +
    ylab("Expression") +
    # Annotate WT_MA15 facet with its p-value
    geom_text(data = subset(final_gene, Type == "WT_MA15"),
              aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_MA15)),
              color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE) +
    # Annotate WT_Mock facet with its p-value
    geom_text(data = subset(final_gene, Type == "WT_Mock"),
              aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_Mock)),
              color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE)

  # Print the plot
  print(p)

  # Increment the counter
  counter <- counter + 1

}
```

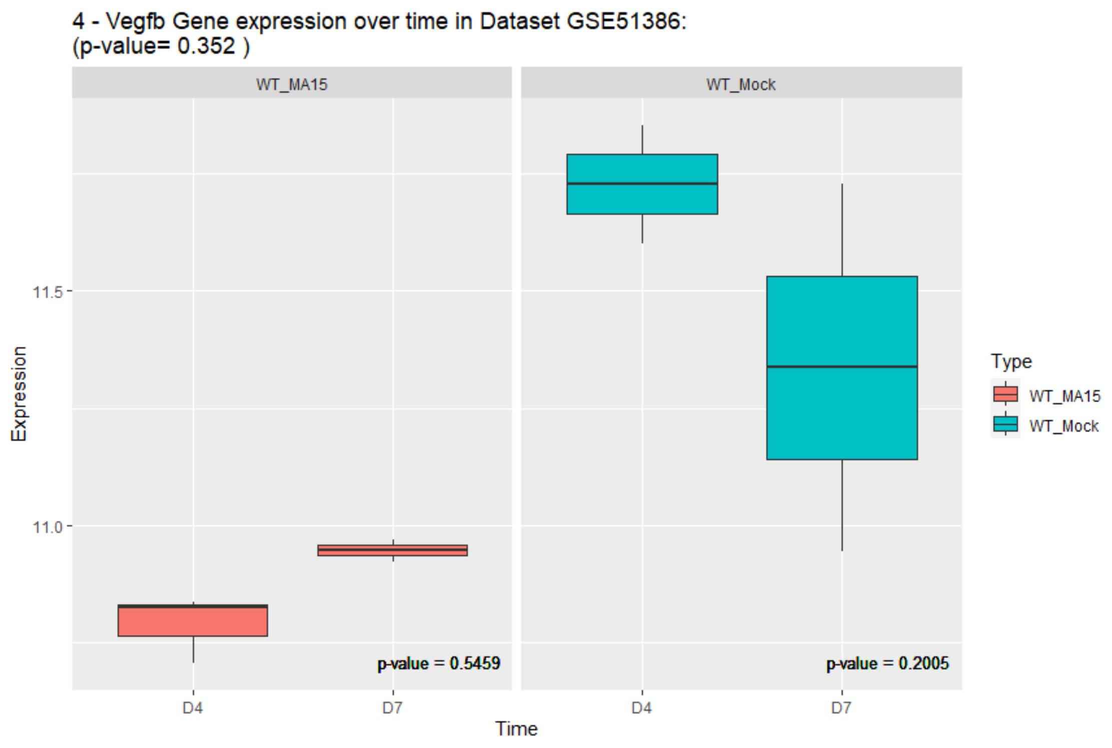


2 - Nfkb1 Gene expression over time in Dataset GSE51386:
(p-value= 0.0072)



3 - Vegfa Gene expression over time in Dataset GSE51386:
(p-value= 0.6673)





```
#================================================================
============================
```

GSE51386:

```r
expr_data <- read.csv("Mus_SARS_GSE51386.csv", header = T, stringsAsFactors = T,
na.strings = c("","NA"))
attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X.5", "WT_MA15.D7")
setnames(expr_data, "X.6", "WT_MA15.D7")
setnames(expr_data, "X.37", "WT_MA15.D7")

setnames(expr_data, "X.7", "WT_Mock.D4")
setnames(expr_data, "X.8", "WT_Mock.D4")
setnames(expr_data, "X.9", "WT_Mock.D4")
setnames(expr_data, "X.10", "WT_Mock.D4")

setnames(expr_data, "X.11", "WT_Mock.D7")
setnames(expr_data, "X.12", "WT_Mock.D7")
setnames(expr_data, "X.13", "WT_Mock.D7")
setnames(expr_data, "X.14", "WT_Mock.D7")


setnames(expr_data, "X.33", "WT_MA15.D4")
setnames(expr_data, "X.34", "WT_MA15.D4")
setnames(expr_data, "X.35", "WT_MA15.D4")
setnames(expr_data, "X.36", "WT_MA15.D4")
```

```r
expr_data1 <- expr_data[16:31002, c(1, 2, 9:18, 37:41)]

write.csv(expr_data1, file = "GSE51386_dataset.csv")

# Read the dataset
data <- read.csv("GSE51386_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))
```

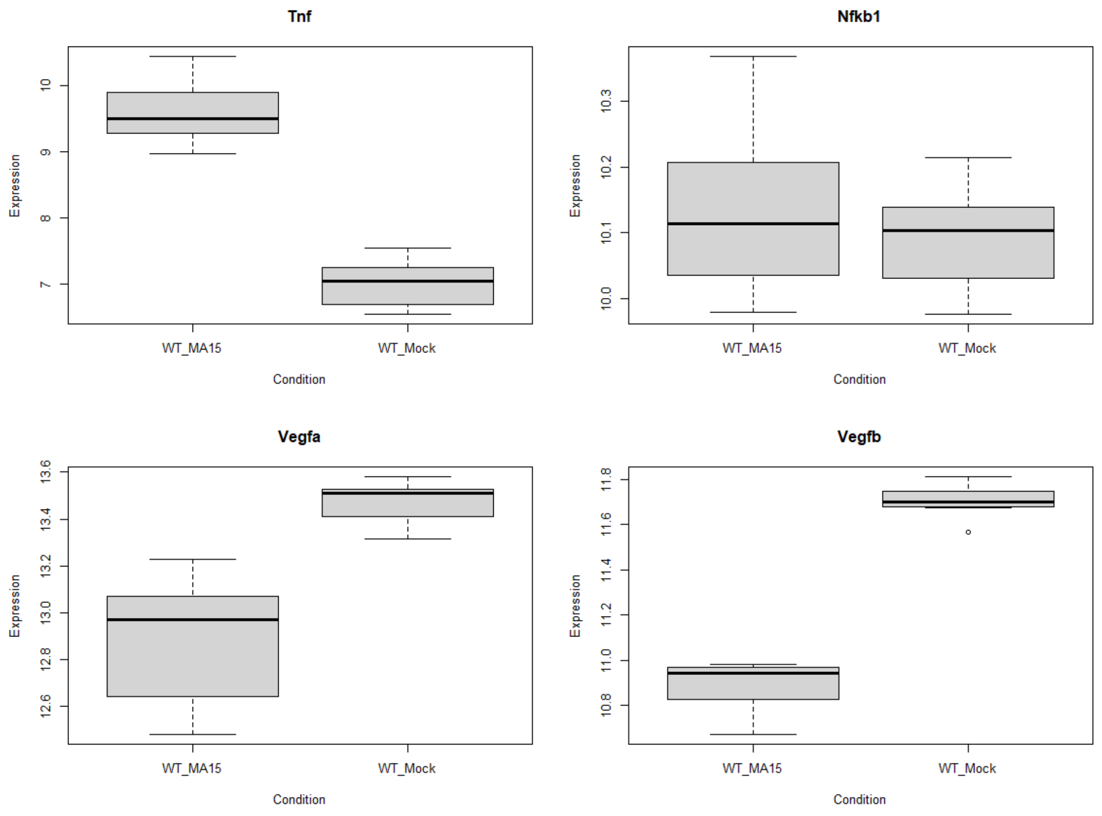
```
# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")

probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))
```

```r
combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
library(limma)

final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))
```

```r
first_table3 <- df1
first_table3$dataset <- "GSE51386"

first_table3
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.6	8.23	11.5	9.22	3.54e-08	3.54e-08	GSE51386
0.0428	10.1	0.737	-5.07	0.4742	0.4742	GSE51386
-0.6	13.2	-5.6	1.65	8.67e-05	8.67e-05	GSE51386
-0.818	11.3	-16.8	13.8	3.33e-10	3.33e-10	GSE51386

```r
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



Change in Time:

```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D4.vs.D7_WT_MA15", "D4.vs.D7_WT_Mock"),
number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
```

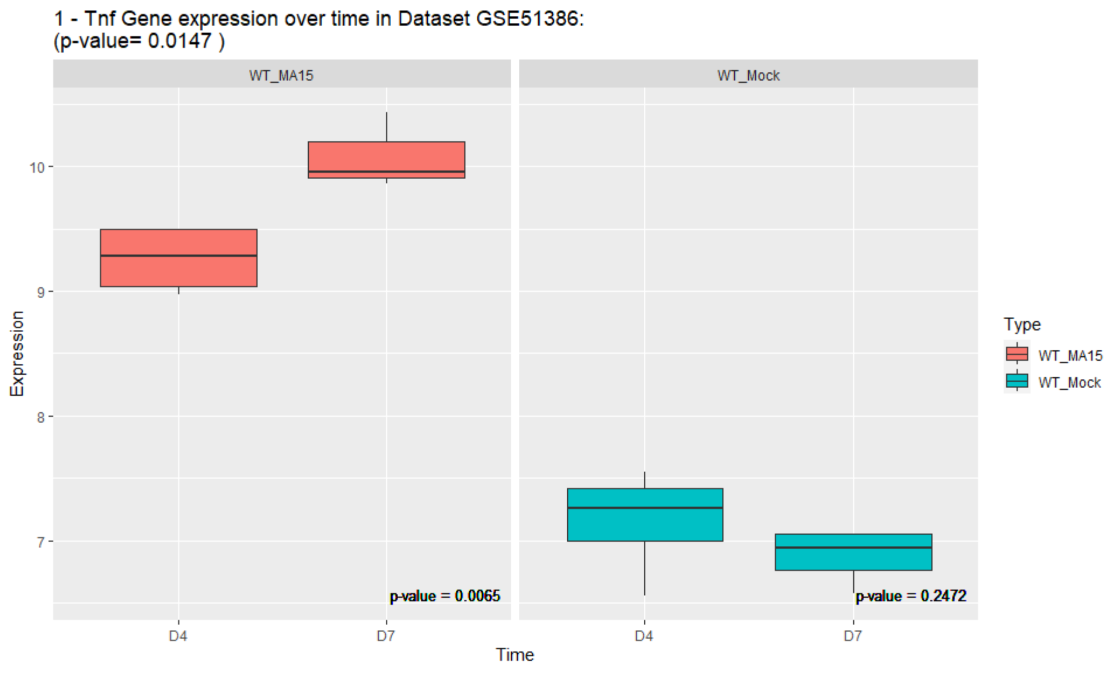
```r
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]


  pvalues <- c(df2$P.Value)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  df$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Define the column names for p-values
  pvalue_cols <- c("P.Value", "adj.P.Val", "P.Value_D4.vs.D7_WT_MA15",
"P.Value_D4.vs.D7_WT_Mock")

  # Check if all p-value columns exist in the dataframe
  if (all(pvalue_cols %in% names(df2))) {
    # Store the p-values in a separate data frame
    pvalues <- df2[, pvalue_cols]

    # Remove the p-values from the original data frame
    df2 <- df2[, setdiff(names(df2), names(pvalues))]

    # Round the numbers to 3 decimal points
    df2 <- round(df2, 4)

    # Add the p-values back to the data frame
    df2 <- cbind(df2, pvalues)

    # Store the updated data frame in the list
    results[[i]] <- df2
  }
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))
```

```
df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D4.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$P.Value_D4.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

first_table4 <- df2
first_table4$dataset <- "GSE51386"

first_table4
```

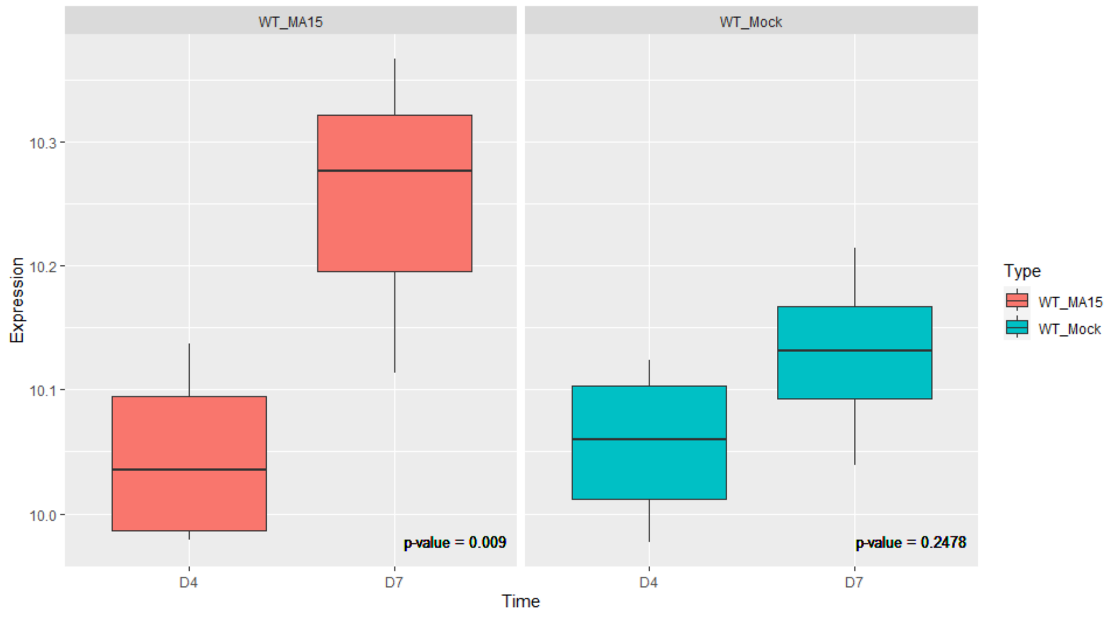
D4.vs.D7_WT_MA15	D4.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D4.vs.D7_WT_MA15	P.Value_D4.vs.D7_WT_Mock	dataset
-0.827	0.28	8.23	6.35	0.0147	0.0147	0.0065	0.2472	GSE51386
-0.206	-0.0736	10.1	5.75	0.0195	0.0195	0.0090	0.2478	GSE51386
0.221	-0.0053	13.2	0.969	0.4097	0.4097	0.1916	0.9718	GSE51386
-0.156	0.0818	11.3	5.1	0.0271	0.0271	0.0179	0.1445	GSE51386

```
# Convert row names to a column in df2
df2$GENE_SYMBOL <- rownames(df2)

# Ensure GENE_SYMBOL is of type character in both data frames
df2$GENE_SYMBOL <- as.character(df2$GENE_SYMBOL)
final$GENE_SYMBOL <- as.character(final$GENE_SYMBOL)

# Define an offset for the annotation
offset <- 1  # Adjust this based on your data

# Initialize a list to store plots
plots <- list()

# Initialize a counter for the genes
counter <- 1

# Loop over the genes
for (gene in unique(final$GENE_SYMBOL)) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Extract p-values for each type
  p_value_MA15 <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_MA15), 4)
  p_value_Mock <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_Mock), 4)

    overall_p_value <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]$adj.P.Val), 4)
```

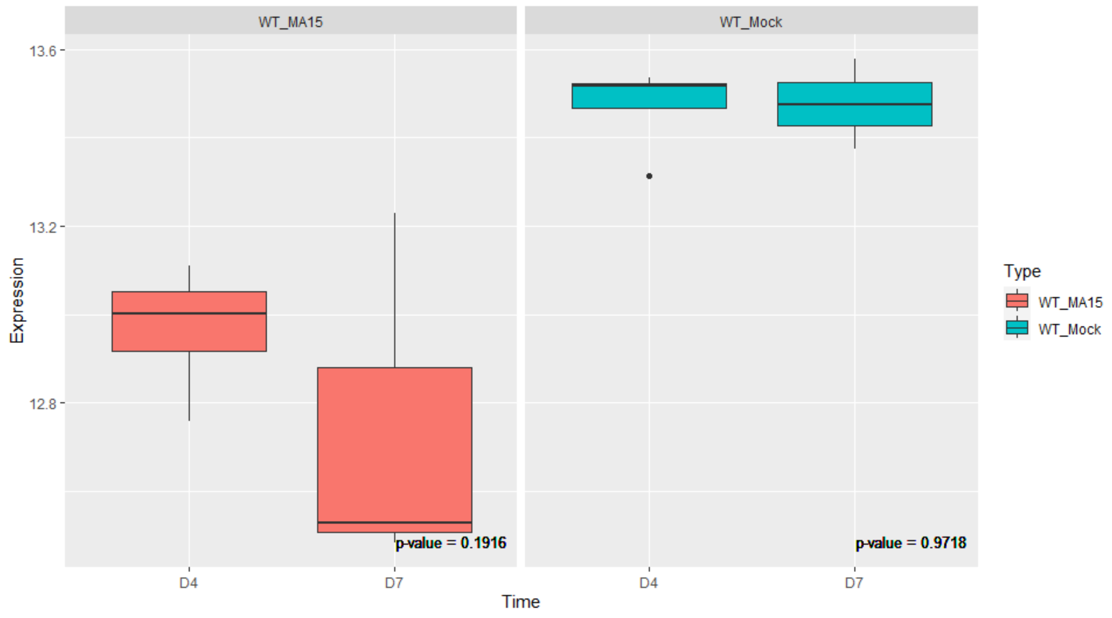
```
 # Create a boxplot for the current gene
 p <- ggplot(final_gene, aes(x = Measurement, y = Expression, fill = Type)) +
   geom_boxplot() +
   facet_wrap(~Type) +
   labs(title = paste(counter, "-", gene, "Gene expression over time in Dataset
GSE51386:\n(p-value=", overall_p_value, ")")) +
   xlab("Time") +
   ylab("Expression") +
   # Annotate WT_MA15 facet with its p-value
   geom_text(data = subset(final_gene, Type == "WT_MA15"),
         aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_MA15)),
         color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE) +
   # Annotate WT_Mock facet with its p-value
   geom_text(data = subset(final_gene, Type == "WT_Mock"),
         aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_Mock)),
         color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE)

 # Print the plot
 print(p)

 # Increment the counter
 counter <- counter + 1

}
```

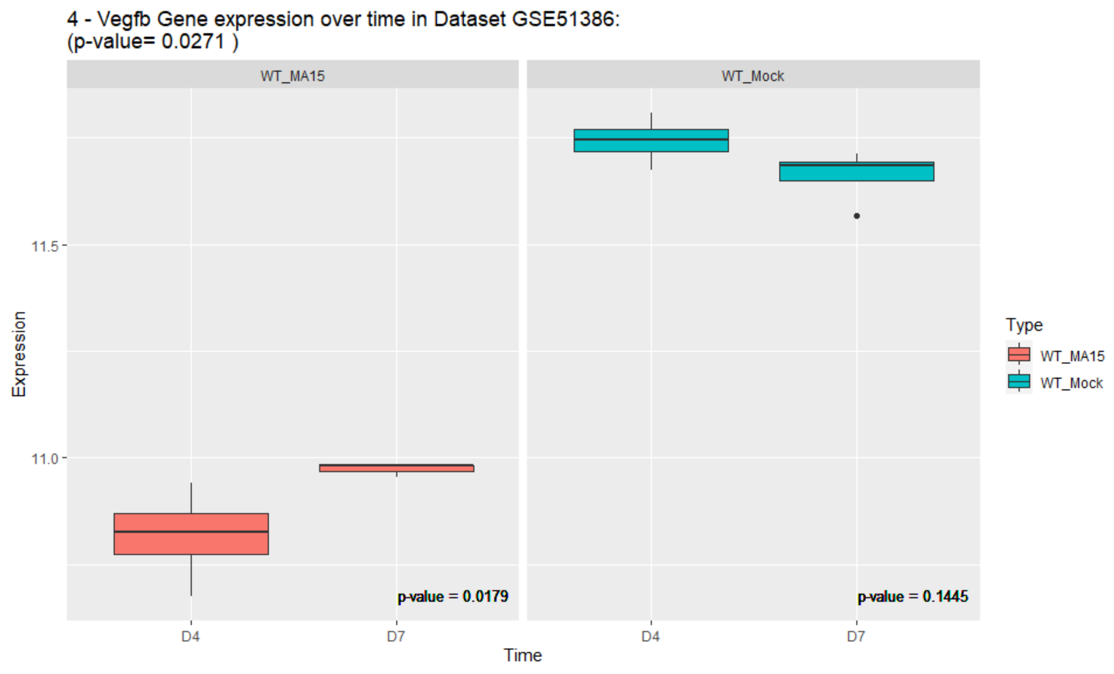


2 - Nfkb1 Gene expression over time in Dataset GSE51386:
(p-value= 0.0195)



3 - Vegfa Gene expression over time in Dataset GSE51386:
(p-value= 0.4097)





#===
===========================

GSE40827:

```
expr_data <- read.csv("Mus_SARS_GSE40827.csv", header = T, stringsAsFactors = T,
na.strings = c("","NA"))
attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X", "WT_MA15.D4")
setnames(expr_data, "X.1", "WT_MA15.D4")

setnames(expr_data, "X.2", "WT_MA15.D7")
setnames(expr_data, "X.3", "WT_MA15.D7")
setnames(expr_data, "X.4", "WT_MA15.D7")

setnames(expr_data, "X.5", "WT_Mock.D4")
setnames(expr_data, "X.6", "WT_Mock.D4")

setnames(expr_data, "X.7", "WT_Mock.D7")
setnames(expr_data, "X.8", "WT_Mock.D7")
setnames(expr_data, "X.9", "WT_Mock.D7")



expr_data1 <- expr_data[21:41194, c(1, 2, 4:13)]

write.csv(expr_data1, file = "GSE40827_dataset.csv")

# Read the dataset
data <- read.csv("GSE40827_dataset.csv", stringsAsFactors = FALSE)
```

```r
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")
```

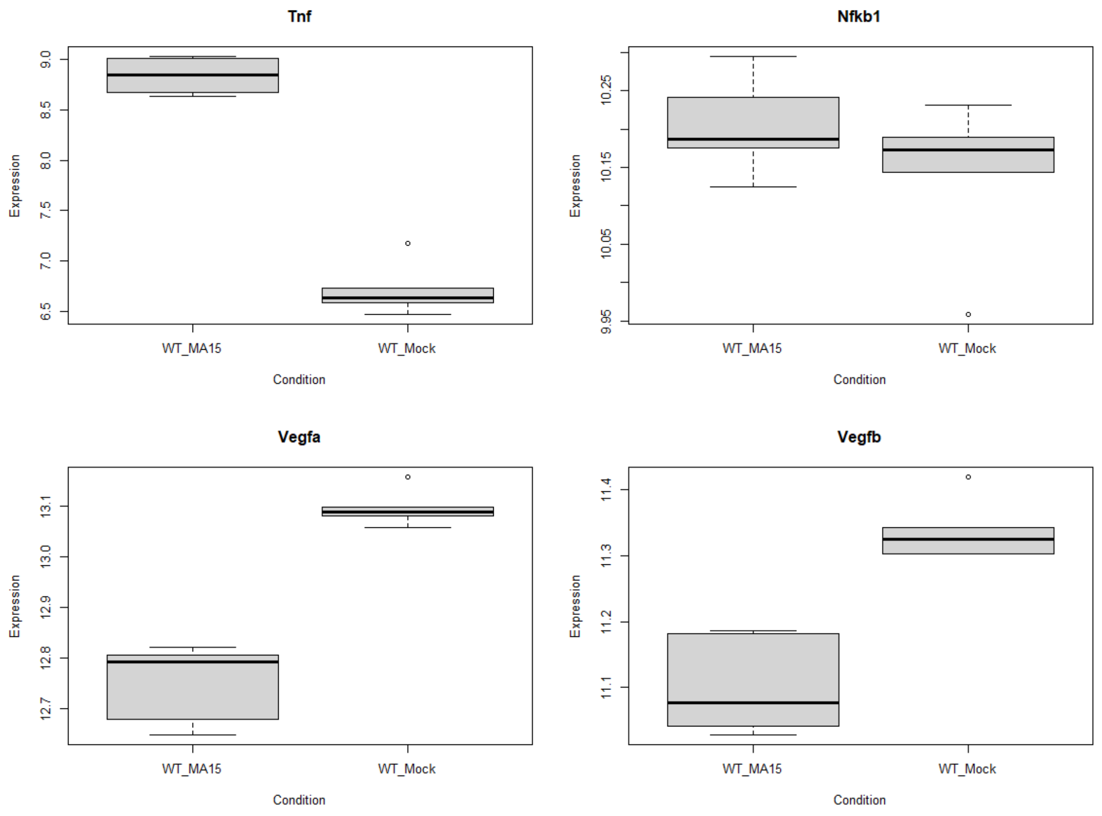
```r
probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
```

```r
combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
library(limma)

final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}
```

```r
for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

first_table5 <- df1
first_table5$dataset <- "GSE40827"

first_table5
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.12	7.78	14.5	6.98	5.05e-07	5.05e-07	GSE40827
0.0652	10.2	1.18	-4.78	0.2735	0.2735	GSE40827
-0.347	12.9	-8.82	3.28	2.15e-05	2.15e-05	GSE40827
-0.235	11.2	-5.84	0.49	0.0004	0.0004	GSE40827

```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



Change in Time:

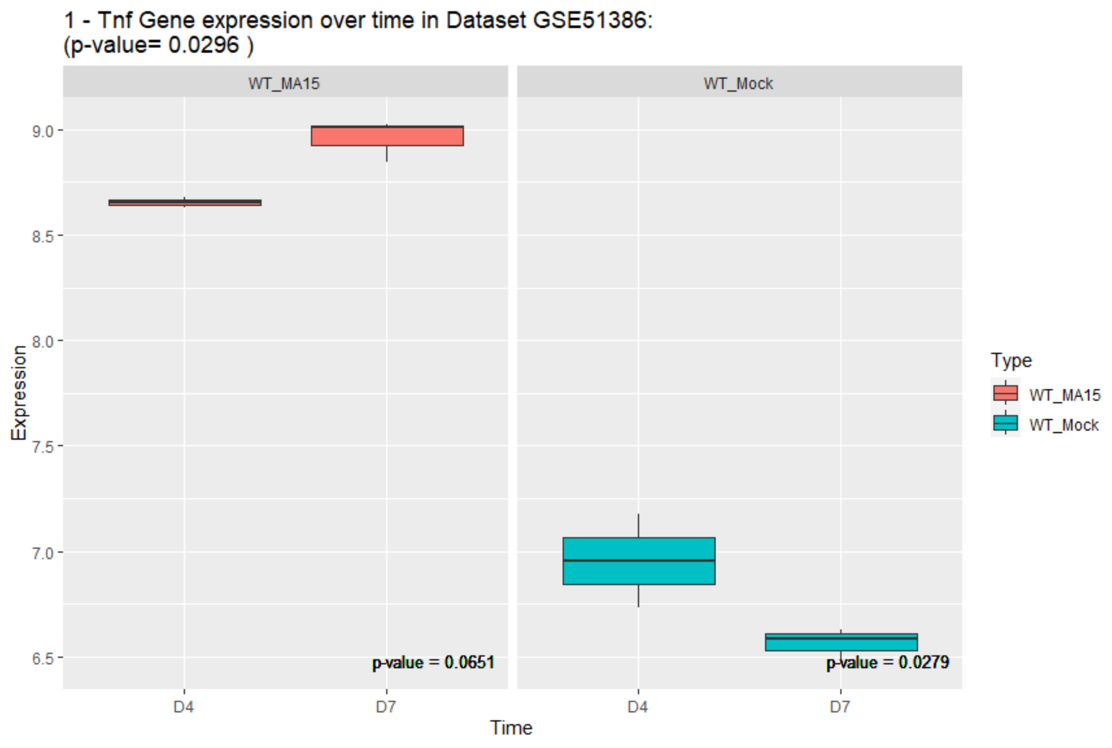
```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
```

```
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D4.vs.D7_WT_MA15", "D4.vs.D7_WT_Mock"),
number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
```

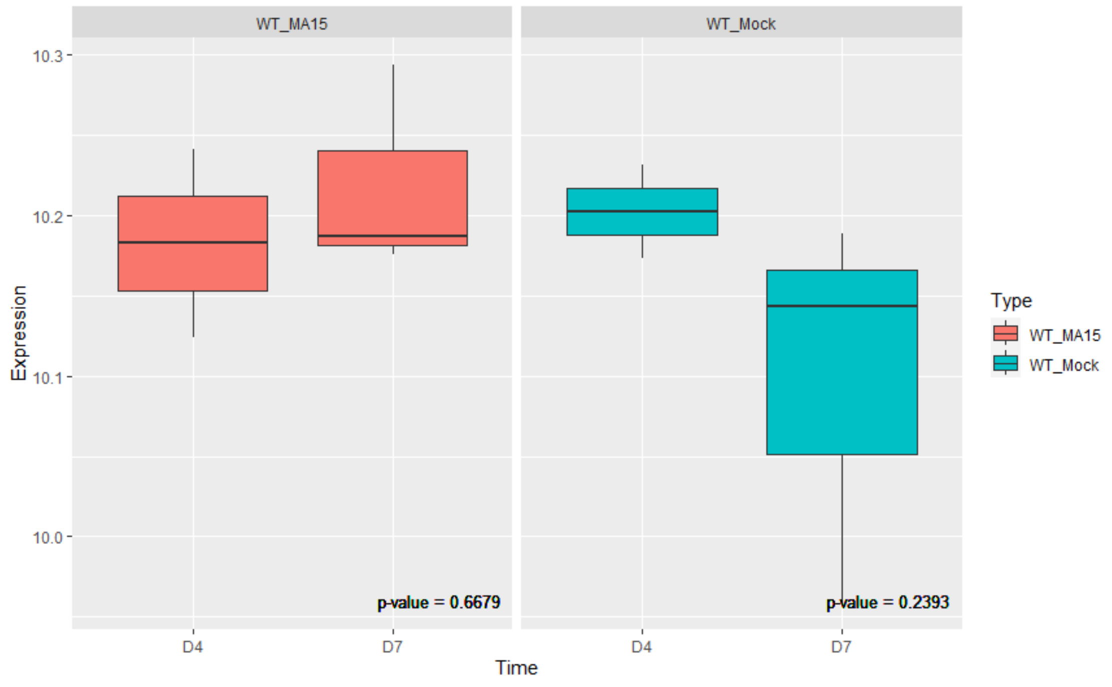
```r
  df2 <- results[[i]]


  pvalues <- c(df2$P.Value)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  df$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Define the column names for p-values
  pvalue_cols <- c("P.Value", "adj.P.Val", "P.Value_D4.vs.D7_WT_MA15",
"P.Value_D4.vs.D7_WT_Mock")

  # Check if all p-value columns exist in the dataframe
  if (all(pvalue_cols %in% names(df2))) {
    # Store the p-values in a separate data frame
    pvalues <- df2[, pvalue_cols]

    # Remove the p-values from the original data frame
    df2 <- df2[, setdiff(names(df2), names(pvalues))]

    # Round the numbers to 3 decimal points
    df2 <- round(df2, 4)

    # Add the p-values back to the data frame
    df2 <- cbind(df2, pvalues)

    # Store the updated data frame in the list
    results[[i]] <- df2
  }
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D4.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$P.Value_D4.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

first_table6 <- df2
first_table6$dataset <- "GSE40827"
first_table6
```

D4.vs.D7_WT_MA15	D4.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D4.vs.D7_WT_MA15	P.Value_D4.vs.D7_WT_Mock	dataset
-0.305	0.391	7.78	6.7	0.0296	0.0296	0.0651	0.0279	GSE40827
-0.0363	0.105	10.2	0.955	0.4365	0.4365	0.6679	0.2393	GSE40827
0.108	0.044	12.9	3.31	0.1075	0.1075	0.0546	0.3685	GSE40827
-0.114	-0.0409	11.2	3.51	0.0981	0.0981	0.0471	0.4039	GSE40827

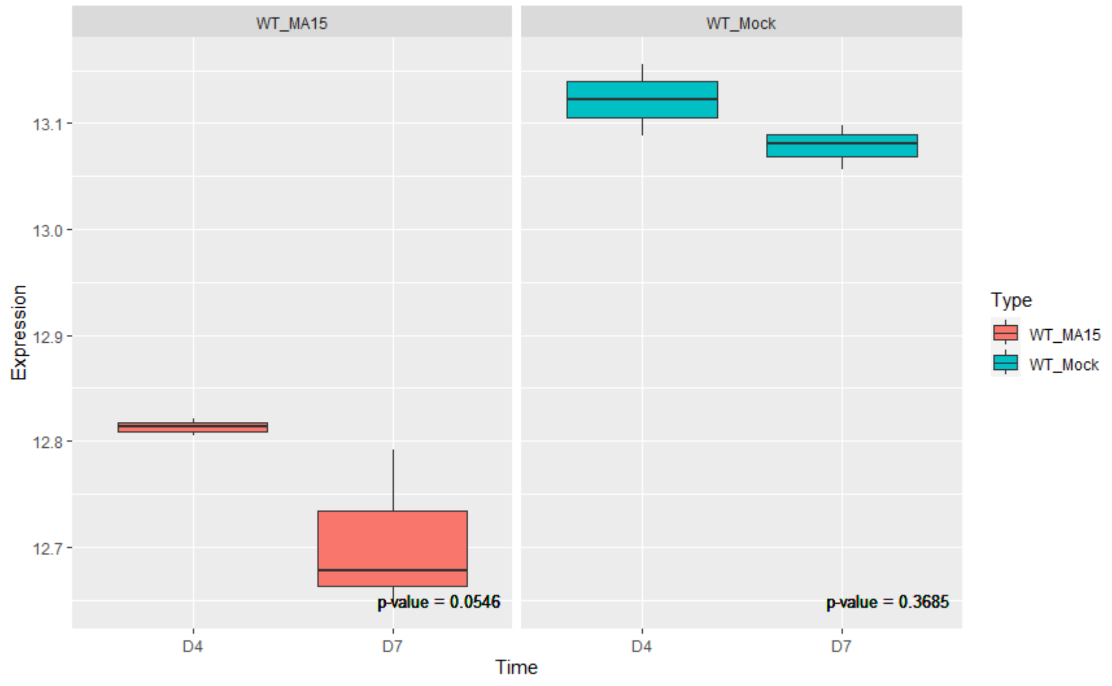
```
# Convert row names to a column in df2
df2$GENE_SYMBOL <- rownames(df2)

# Ensure GENE_SYMBOL is of type character in both data frames
df2$GENE_SYMBOL <- as.character(df2$GENE_SYMBOL)
final$GENE_SYMBOL <- as.character(final$GENE_SYMBOL)

# Define an offset for the annotation
offset <- 1  # Adjust this based on your data

# Initialize a list to store plots
plots <- list()

# Initialize a counter for the genes
counter <- 1

# Loop over the genes
for (gene in unique(final$GENE_SYMBOL)) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Extract p-values for each type
  p_value_MA15 <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_MA15), 4)
  p_value_Mock <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_Mock), 4)

    overall_p_value <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]$adj.P.Val), 4)

  # Create a boxplot for the current gene
  p <- ggplot(final_gene, aes(x = Measurement, y = Expression, fill = Type)) +
    geom_boxplot() +
    facet_wrap(~Type) +
    labs(title = paste(counter, "-", gene, "Gene expression over time in Dataset
GSE51386:\n(p-value=", overall_p_value, ")")) +
    xlab("Time") +
    ylab("Expression") +
    # Annotate WT_MA15 facet with its p-value
    geom_text(data = subset(final_gene, Type == "WT_MA15"),
          aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_MA15)),
          color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE) +
    # Annotate WT_Mock facet with its p-value
    geom_text(data = subset(final_gene, Type == "WT_Mock"),
          aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_Mock)),
          color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE)

  # Print the plot
  print(p)

  # Increment the counter
  counter <- counter + 1

}
```

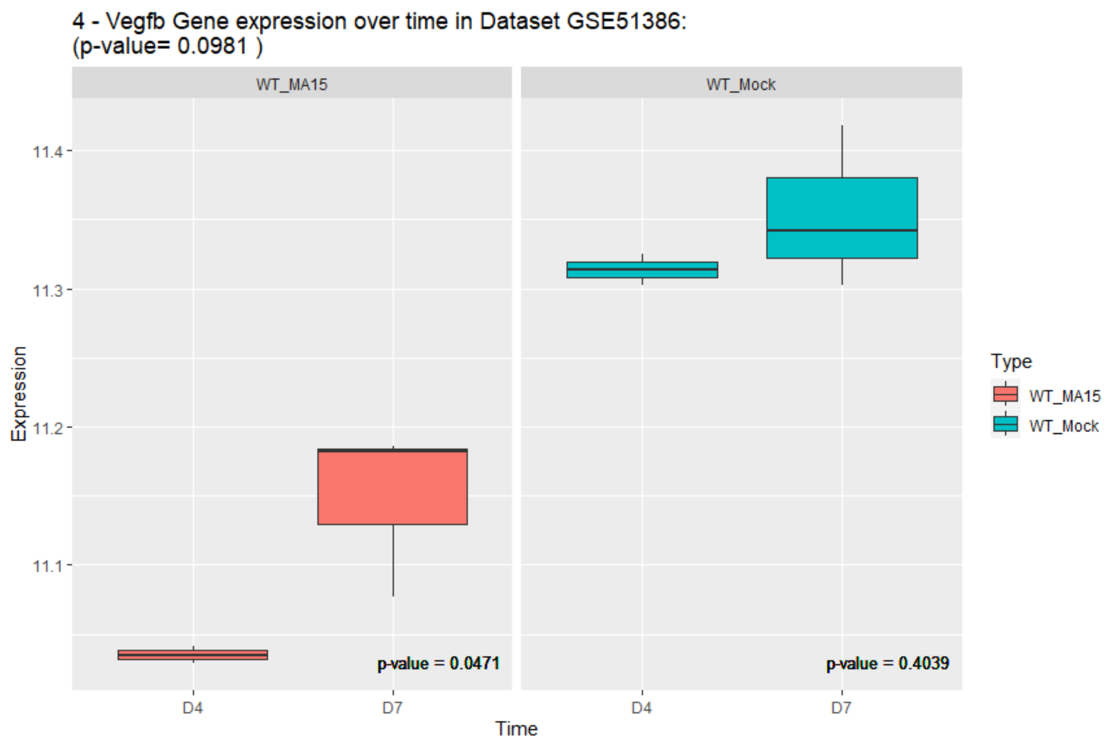


2 - Nfkb1 Gene expression over time in Dataset GSE51386:
(p-value= 0.4365)



3 - Vegfa Gene expression over time in Dataset GSE51386:
(p-value= 0.1075)





```
#=============================================================
==========================
```

GSE40824:

```
expr_data <- read.csv("Mus_SARS_40824.csv", header = T, stringsAsFactors = T,
na.strings = c("","NA"))
attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X", "WT_MA15.D4")
setnames(expr_data, "X.1", "WT_MA15.D4")
setnames(expr_data, "X.2", "WT_MA15.D4")

setnames(expr_data, "X.3", "WT_MA15.D7")
setnames(expr_data, "X.4", "WT_MA15.D7")
setnames(expr_data, "X.5", "WT_MA15.D7")

setnames(expr_data, "X.6", "WT_Mock.D4")
setnames(expr_data, "X.7", "WT_Mock.D4")
setnames(expr_data, "X.8", "WT_Mock.D4")

setnames(expr_data, "X.9", "WT_Mock.D7")
setnames(expr_data, "X.10", "WT_Mock.D7")
setnames(expr_data, "X.11", "WT_Mock.D7")



expr_data1 <- expr_data[22:41194, c(1, 2, 4:15)]
```

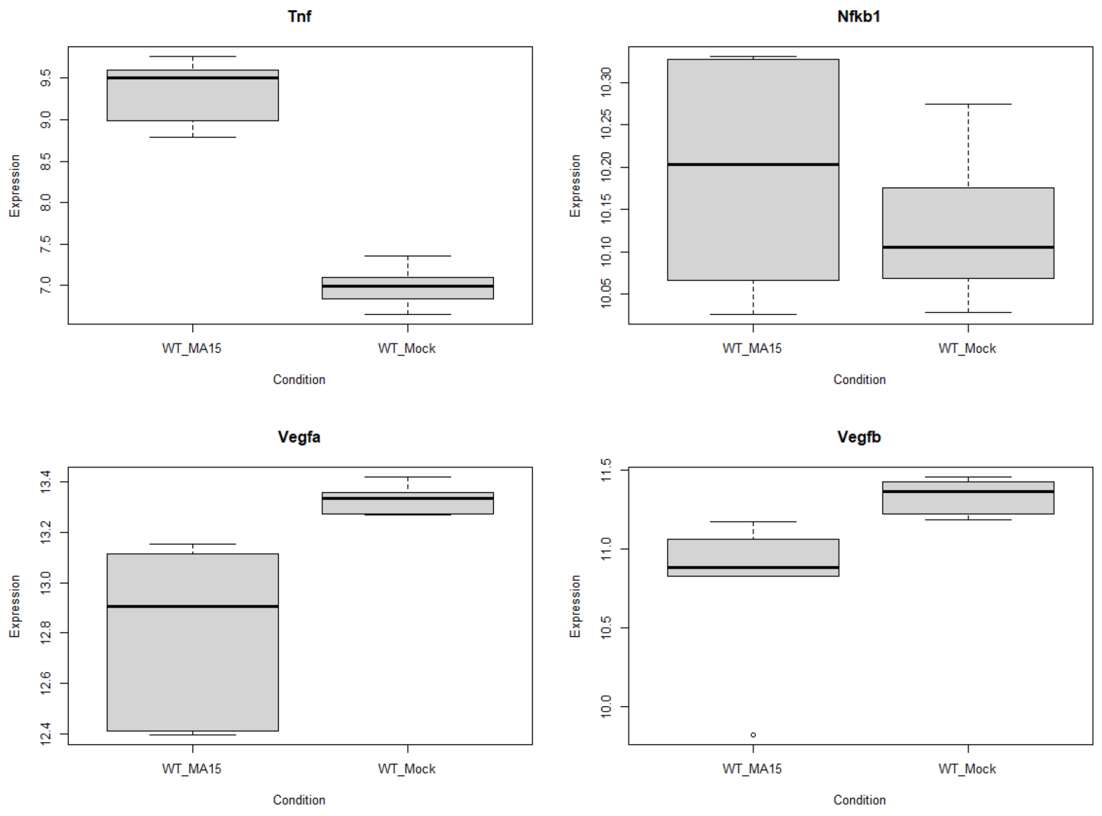
```
write.csv(expr_data1, file = "GSE40824_dataset.csv")

data <- read.csv("GSE40824_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])
```

```
nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")

probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
```

```r
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
library(limma)

final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")
  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

first_table7 <- df1
first_table7$dataset <- "GSE40824"
first_table7
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.37	8.17	12.9	7.98	1.53e-07	1.53e-07	GSE40824
0.0666	10.2	1.04	-4.82	0.3212	0.3212	GSE40824
-0.517	13.1	-3.68	-1.81	0.0043	0.0043	GSE40824
-0.563	11.1	-2.76	-3.11	0.0201	0.0201	GSE40824

```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



Change in Time:

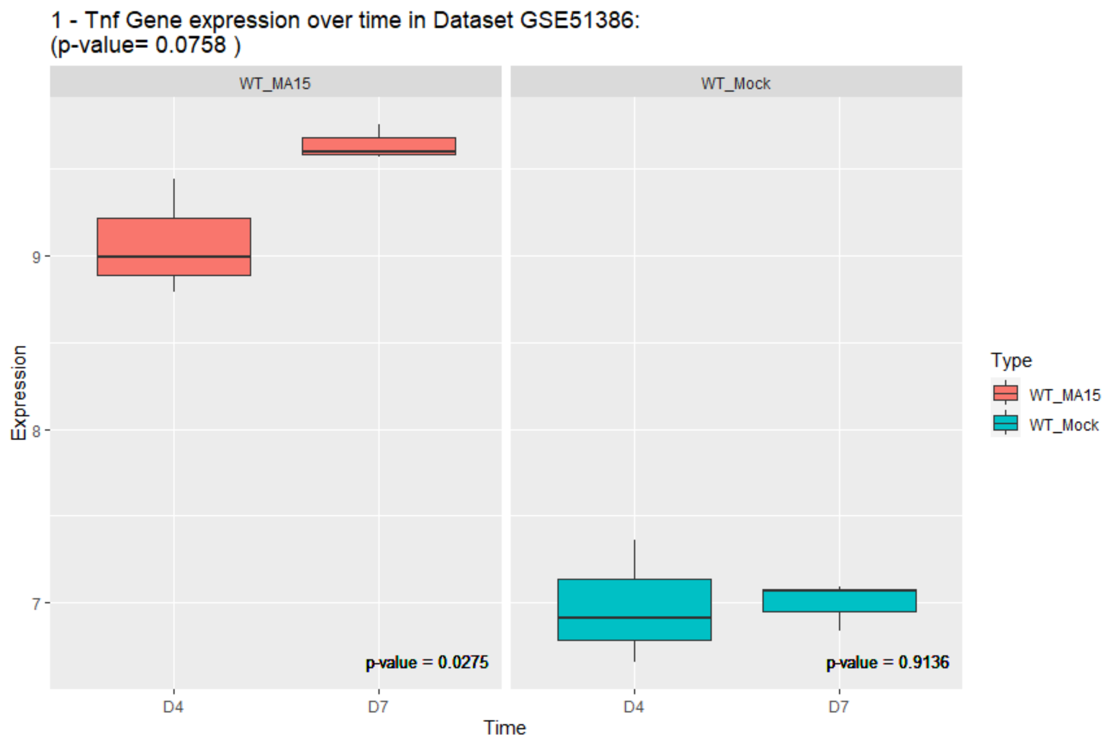
```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
```

```
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D4.vs.D7_WT_MA15", "D4.vs.D7_WT_Mock"),
number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]


  pvalues <- c(df2$P.Value)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  df$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Define the column names for p-values
  pvalue_cols <- c("P.Value", "adj.P.Val", "P.Value_D4.vs.D7_WT_MA15",
"P.Value_D4.vs.D7_WT_Mock")

  # Check if all p-value columns exist in the dataframe
  if (all(pvalue_cols %in% names(df2))) {
    # Store the p-values in a separate data frame
    pvalues <- df2[, pvalue_cols]

    # Remove the p-values from the original data frame
    df2 <- df2[, setdiff(names(df2), names(pvalues))]

    # Round the numbers to 3 decimal points
    df2 <- round(df2, 4)

    # Add the p-values back to the data frame
    df2 <- cbind(df2, pvalues)

    # Store the updated data frame in the list
    results[[i]] <- df2
  }
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D4.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$P.Value_D4.vs.D7_WT_Mock < 0.0001,
```

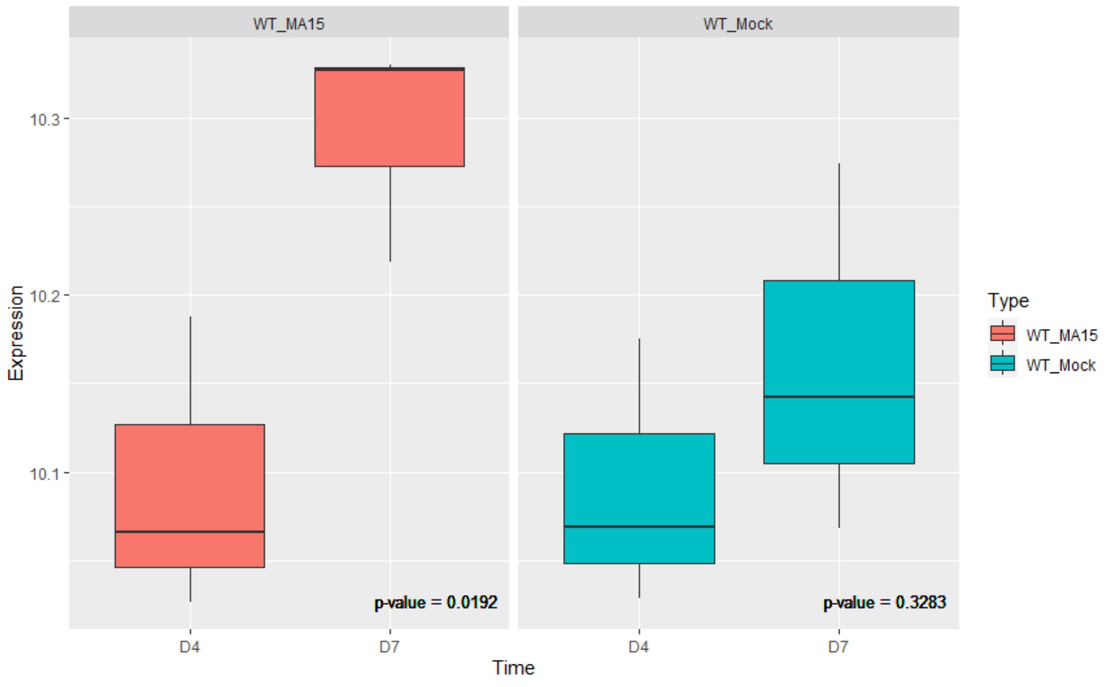
```r
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

first_table8 <- df2
first_table8$dataset <- "GSE40824"
first_table8
```

D4.vs.D7_WT_MA15	D4.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D4.vs.D7_WT_MA15	P.Value_D4.vs.D7_WT_Mock	dataset
-0.568	-0.0237	8.17	3.62	0.0758	0.0758	0.0275	0.9136	GSE40824
-0.199	-0.0707	10.2	4.81	0.0424	0.0424	0.0192	0.3283	GSE40824
0.551	0.0225	13.1	13.4	0.0028	0.0028	0.0008	0.8377	GSE40824
0.179	-0.0976	11.1	0.212	0.8137	0.8137	0.5833	0.7640	GSE40824

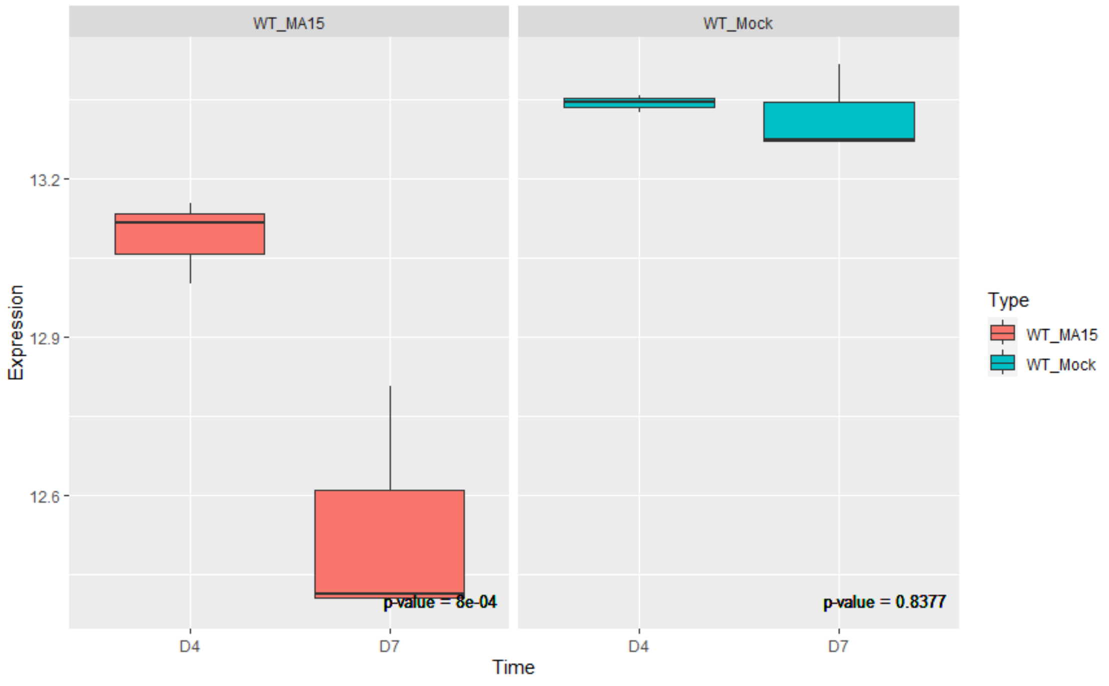
```r
# Convert row names to a column in df2
df2$GENE_SYMBOL <- rownames(df2)

# Ensure GENE_SYMBOL is of type character in both data frames
df2$GENE_SYMBOL <- as.character(df2$GENE_SYMBOL)
final$GENE_SYMBOL <- as.character(final$GENE_SYMBOL)

# Define an offset for the annotation
offset <- 1  # Adjust this based on your data

# Initialize a list to store plots
plots <- list()

# Initialize a counter for the genes
counter <- 1

# Loop over the genes
for (gene in unique(final$GENE_SYMBOL)) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Extract p-values for each type
  p_value_MA15 <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_MA15), 4)
  p_value_Mock <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_Mock), 4)

   overall_p_value <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]$adj.P.Val), 4)

  # Create a boxplot for the current gene
  p <- ggplot(final_gene, aes(x = Measurement, y = Expression, fill = Type)) +
    geom_boxplot() +
    facet_wrap(~Type) +
    labs(title = paste(counter, "-", gene, "Gene expression over time in Dataset
GSE51386:\n(p-value=", overall_p_value, ")")) +
   xlab("Time") +
   ylab("Expression") +
   # Annotate WT_MA15 facet with its p-value
   geom_text(data = subset(final_gene, Type == "WT_MA15"),
           aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_MA15)),
           color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE) +
   # Annotate WT_Mock facet with its p-value
   geom_text(data = subset(final_gene, Type == "WT_Mock"),
           aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_Mock)),
           color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE)

 # Print the plot
 print(p)

 # Increment the counter
 counter <- counter + 1

}
```

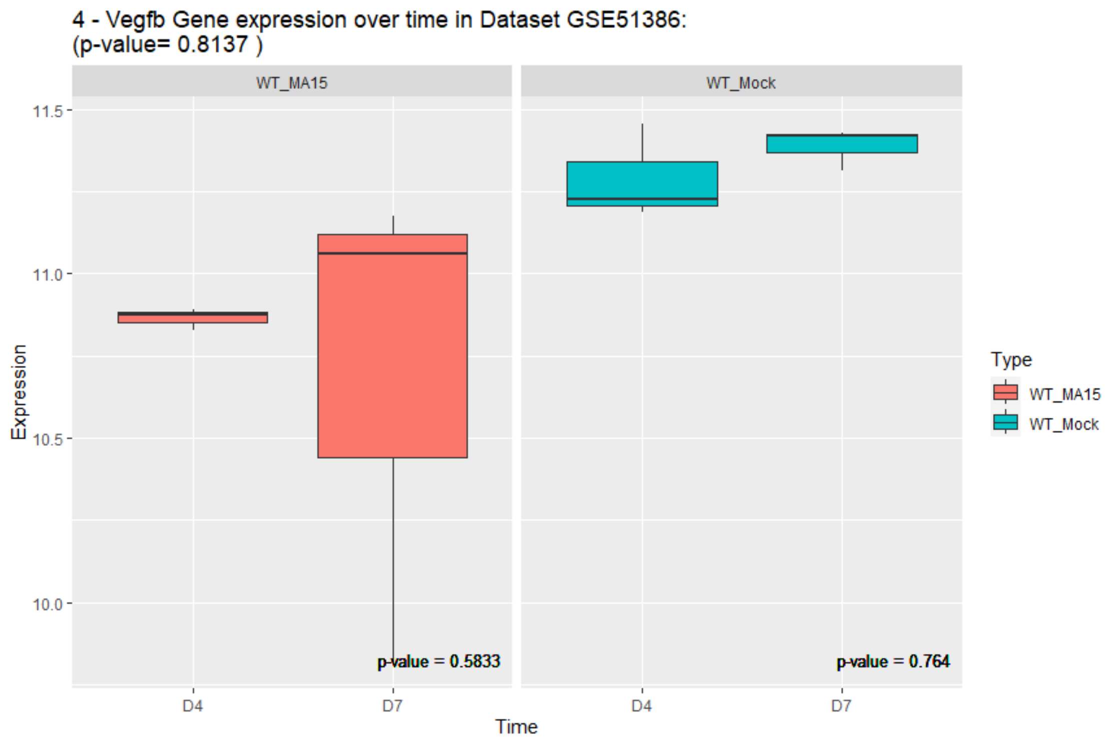


2 - Nfkb1 Gene expression over time in Dataset GSE51386:
(p-value= 0.0424)



3 - Vegfa Gene expression over time in Dataset GSE51386:
(p-value= 0.0028)





```
first_table1 <- rownames_to_column(first_table1, "rowname")
first_table3 <- rownames_to_column(first_table3, "rowname")
first_table5 <- rownames_to_column(first_table5, "rowname")
first_table7 <- rownames_to_column(first_table7, "rowname")

df2 <- merge(first_table1, first_table3, all = TRUE)
df2 <- merge(df2, first_table5, all = TRUE)
df2 <- merge(df2, first_table7, all = TRUE)
df2 <- df2 %>% arrange(dataset)

df2
```

rowname	logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
Nfkb1	0.0666	10.2	1.04	-4.82	0.3212	0.3212	GSE40824
Tnf	2.37	8.17	12.9	7.98	1.53e-07	1.53e-07	GSE40824
Vegfa	-0.517	13.1	-3.68	-1.81	0.0043	0.0043	GSE40824
Vegfb	-0.563	11.1	-2.76	-3.11	0.0201	0.0201	GSE40824
Nfkb1	0.0652	10.2	1.18	-4.78	0.2735	0.2735	GSE40827
Tnf	2.12	7.78	14.5	6.98	5.05e-07	5.05e-07	GSE40827
Vegfa	-0.347	12.9	-8.82	3.28	2.15e-05	2.15e-05	GSE40827
Vegfb	-0.235	11.2	-5.84	0.49	0.0004	0.0004	GSE40827
Nfkb1	0.0428	10.1	0.737	-5.07	0.4742	0.4742	GSE51386
Tnf	2.6	8.23	11.5	9.22	3.54e-08	3.54e-08	GSE51386
Vegfa	-0.6	13.2	-5.6	1.65	8.67e-05	8.67e-05	GSE51386
Vegfb	-0.818	11.3	-16.8	13.8	3.33e-10	3.33e-10	GSE51386

Nfkb1	-0.0197	10.6	-0.244	-5.04	0.8140	0.8140	GSE51387
Tnf	2.28	8.81	6.44	0.666	0.0004	0.0004	GSE51387
Vegfa	-0.78	13.4	-8.06	2.02	8.66e-05	8.66e-05	GSE51387
Vegfb	-0.679	11.2	-3.67	-2.22	0.0079	0.0079	GSE51387

```
first_table2 <- rownames_to_column(first_table2, "rowname")
first_table4 <- rownames_to_column(first_table4, "rowname")
first_table6 <- rownames_to_column(first_table6, "rowname")
first_table8 <- rownames_to_column(first_table8, "rowname")

df3 <- merge(first_table2, first_table4, all = TRUE)
df3 <- merge(df3, first_table6, all = TRUE)
df3 <- merge(df3, first_table8, all = TRUE)
df3 <- df3 %>% arrange(dataset)

df3[,1:10]
```

rowname	D4.vs.D7_WT_MA15	D4.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D4.vs.D7_WT_MA15	P.Value_D4.vs.D7_WT_Mock	dataset
Nfkb1	-0.199	-0.0707	10.2	4.81	0.0424	0.0424	0.0192	0.3283	GSE40824
Tnf	-0.568	-0.0237	8.17	3.62	0.0758	0.0758	0.0275	0.9136	GSE40824
Vegfa	0.551	0.0225	13.1	13.4	0.0028	0.0028	0.0008	0.8377	GSE40824
Vegfb	0.179	-0.0976	11.1	0.212	0.8137	0.8137	0.5833	0.7640	GSE40824
Nfkb1	-0.0363	0.105	10.2	0.955	0.4365	0.4365	0.6679	0.2393	GSE40827
Tnf	-0.305	0.391	7.78	6.7	0.0296	0.0296	0.0651	0.0279	GSE40827
Vegfa	0.108	0.044	12.9	3.31	0.1075	0.1075	0.0546	0.3685	GSE40827

Vegfb	-0.114	-0.0409	11.2	3.51	0.0981	0.0981	0.0471	0.4039	GSE40827
Nfkb1	-0.206	-0.0736	10.1	5.75	0.0195	0.0195	0.0090	0.2478	GSE51386
Tnf	-0.827	0.28	8.23	6.35	0.0147	0.0147	0.0065	0.2472	GSE51386
Vegfa	0.221	-0.0053	13.2	0.969	0.4097	0.4097	0.1916	0.9718	GSE51386
Vegfb	-0.156	0.0818	11.3	5.1	0.0271	0.0271	0.0179	0.1445	GSE51386
Nfkb1	-0.248	-0.116	10.6	15.5	0.0072	0.0072	0.0037	0.0800	GSE51387
Tnf	-0.741	0.875	8.81	6.75	0.0380	0.0380	0.0546	0.0432	GSE51387
Vegfa	0.134	-0.0061	13.4	0.439	0.6673	0.6673	0.3921	0.9704	GSE51387
Vegfb	-0.156	0.39	11.2	1.3	0.3520	0.3520	0.5459	0.2005	GSE51387

#===

Figure 8.3 and Table 8.3

GSE50878:

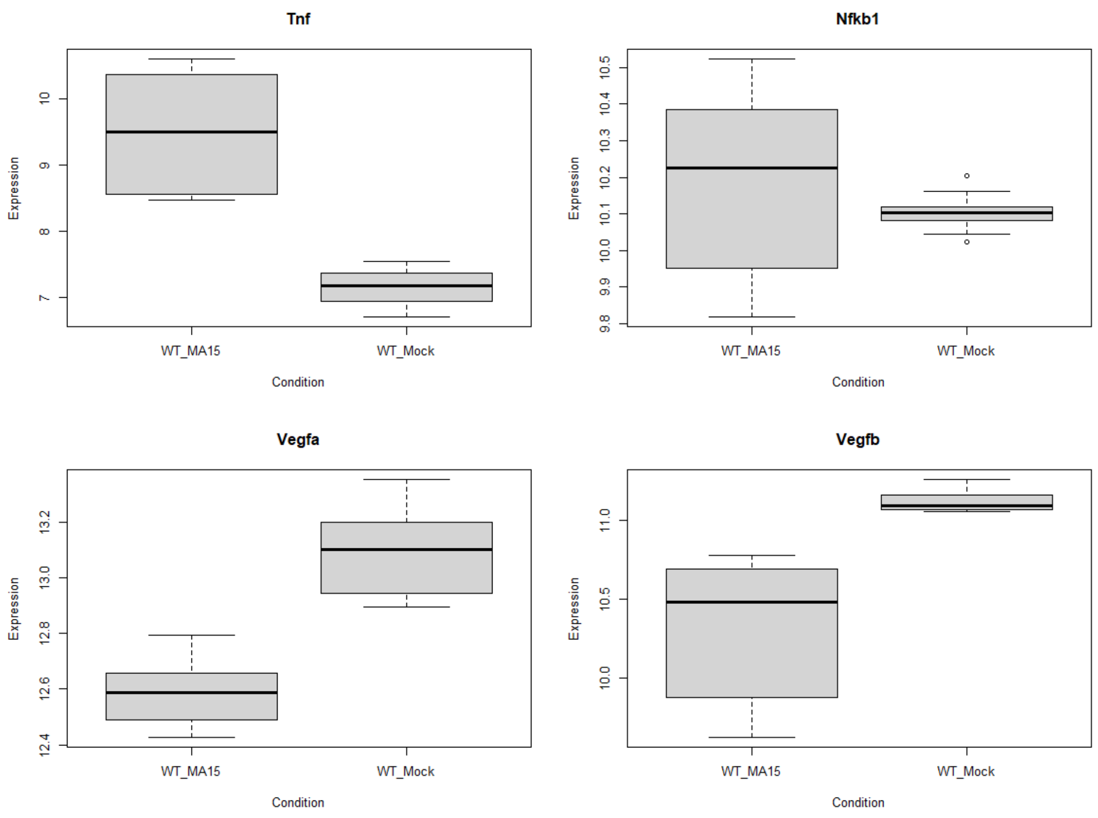
```
expr_data <- read.csv("Mus_SARS_GSE50878.csv", header = T, stringsAsFactors = T,
na.strings = c("","NA"))
attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X.16", "WT_MA15.D2")
setnames(expr_data, "X.17", "WT_MA15.D2")
setnames(expr_data, "X.18", "WT_MA15.D2")

setnames(expr_data, "X.19", "WT_MA15.D4")
setnames(expr_data, "X.20", "WT_MA15.D4")

setnames(expr_data, "X.21", "WT_MA15.D7")
setnames(expr_data, "X.22", "WT_MA15.D7")
setnames(expr_data, "X.23", "WT_MA15.D7")

setnames(expr_data, "X.24", "WT_Mock.D2")
setnames(expr_data, "X.25", "WT_Mock.D2")
setnames(expr_data, "X.26", "WT_Mock.D2")

setnames(expr_data, "X.27", "WT_Mock.D4")
setnames(expr_data, "X.28", "WT_Mock.D4")
setnames(expr_data, "X.29", "WT_Mock.D4")

setnames(expr_data, "X.30", "WT_Mock.D7")
setnames(expr_data, "X.31", "WT_Mock.D7")
setnames(expr_data, "X.32", "WT_Mock.D7")



expr_data1 <- expr_data[20:41193, c(1, 2, 20:36)]

write.csv(expr_data1, file = "GSE50878_dataset.csv")

# Read the dataset
data <- read.csv("GSE50878_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
```

```r
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")

probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
```

```
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
library(limma)

final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]
  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

second_table1 <- df1
second_table1$dataset <- "GSE50878"
second_table1
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.32	8.26	7.3	4.93	2.60e-06	2.60e-06	GSE50878
0.0792	10.1	0.907	-4.77	0.3788	0.3788	GSE50878
-0.514	12.9	-7.53	5.28	1.81e-06	1.81e-06	GSE50878
-0.813	10.7	-5.26	1.49	9.54e-05	9.54e-05	GSE50878

```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



Change in Time:

```r
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D2.vs.D4_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D4,
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D2.vs.D7_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D7,
  D2.vs.D4_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D4,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,
  D2.vs.D7_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D2.vs.D4_WT_MA15", "D4.vs.D7_WT_MA15",
"D2.vs.D7_WT_MA15", "D2.vs.D4_WT_Mock", "D4.vs.D7_WT_Mock",
"D2.vs.D7_WT_Mock"), number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D2.vs.D4_WT_MA15 <- fit$p.value[, "D2.vs.D4_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D2.vs.D7_WT_MA15 <- fit$p.value[, "D2.vs.D7_WT_MA15"]
  gene_results$P.Value_D2.vs.D4_WT_Mock <- fit$p.value[, "D2.vs.D4_WT_Mock"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]
  gene_results$P.Value_D2.vs.D7_WT_Mock <- fit$p.value[, "D2.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Combine the original p-values with the additional p-values from the new temporal
contrasts
  # Update this line to include all the relevant p-value columns from your analysis
  pvalues <- c(df2$P.Value, df2$P.Value_D2.vs.D4_WT_MA15,
df2$P.Value_D4.vs.D7_WT_MA15, df2$P.Value_D2.vs.D7_WT_MA15,
df2$P.Value_D2.vs.D4_WT_Mock, df2$P.Value_D4.vs.D7_WT_Mock,
df2$P.Value_D2.vs.D7_WT_Mock)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  # Assuming the original data frame has a column 'adj.P.Val'
  # We replace it with the adjusted p-values calculated above
  df2$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))

df2$P.Value_D2.vs.D4_WT_MA15 <- ifelse(df2$P.Value_D2.vs.D4_WT_MA15 < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D4.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D2.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D2.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D2.vs.D4_WT_Mock <- ifelse(df2$P.Value_D2.vs.D4_WT_Mock < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "f", digits = 4))


df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$P.Value_D4.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

df2$P.Value_D2.vs.D7_WT_Mock <- ifelse(df2$P.Value_D2.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "f", digits = 4))


second_table2 <- df2
second_table2$dataset <- "GSE50878"
second_table2
```

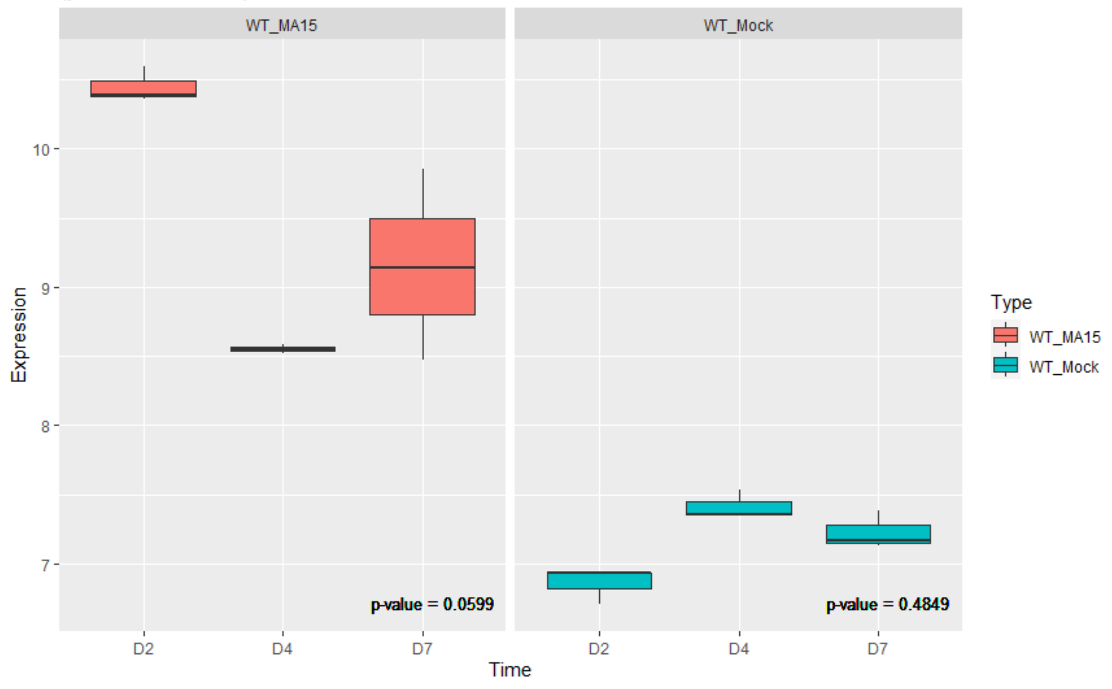
D2.vs.D4_WT_MA15	D4.vs.D7_WT_MA15	D2.vs.D7_WT_MA15	D2.vs.D4_WT_Mock	D4.vs.D7_WT_Mock	D2.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D2.vs.D4_WT_MA15	P.Value_D4.vs.D7_WT_MA15	P.Value_D2.vs.D7_WT_MA15	P.Value_D2.vs.D4_WT_Mock	P.Value_D4.vs.D7_WT_Mock	P.Value_D2.vs.D7_WT_Mock	dataset

1.9	-0.602	1.29	-0.557	0.186	-0.371	8.26	13.5	0.0003	0.0022	3.85e-05	0.0599	0.0004	0.0532	0.4849	0.1770	GSE50878
0.48	-0.138	0.342	-0.0181	-0.0198	-0.038	10.1	5.36	0.0121	0.0845	0.0013	0.2417	0.0057	0.8591	0.8460	0.7107	GSE50878
-0.024	0.0398	0.0158	-0.181	0.0185	-0.162	12.9	0.719	0.5968	1.0000	0.8608	0.7711	0.8968	0.1576	0.8795	0.2006	GSE50878
-0.713	-0.202	-0.915	0.051	-0.122	-0.0712	10.7	26.8	1.29e-05	9.01e-05	2.64e-05	0.0766	8.33e-07	0.5927	0.2138	0.4584	GSE50878

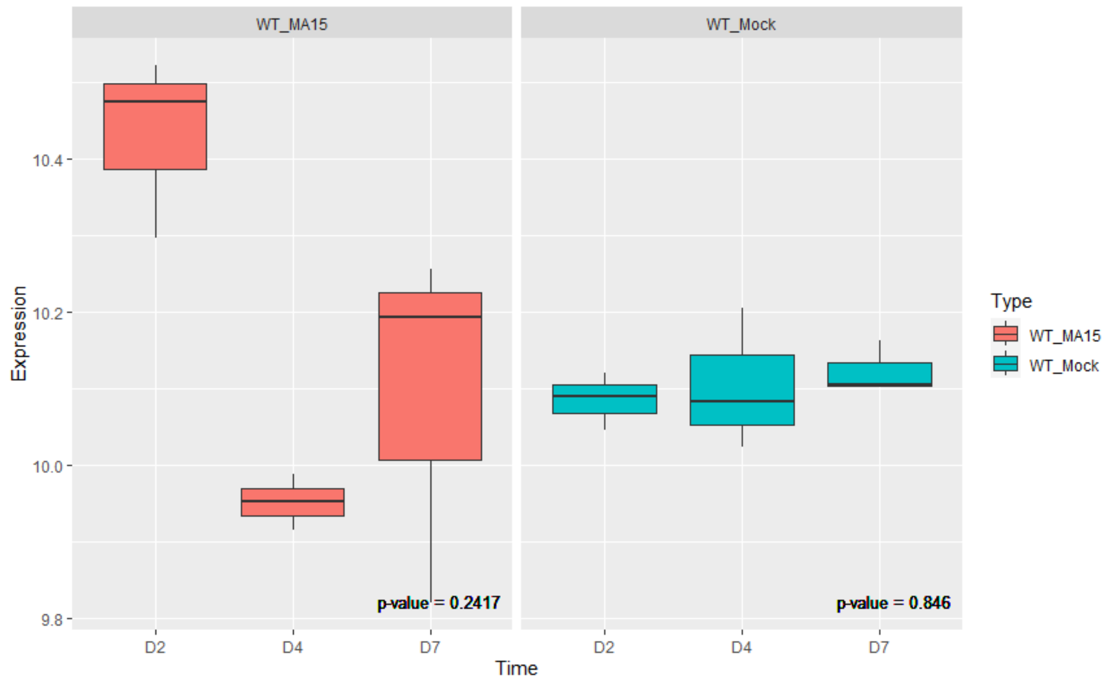
```
# Convert row names to a column in df2
df2$GENE_SYMBOL <- rownames(df2)

# Ensure GENE_SYMBOL is of type character in both data frames
df2$GENE_SYMBOL <- as.character(df2$GENE_SYMBOL)
final$GENE_SYMBOL <- as.character(final$GENE_SYMBOL)

# Define an offset for the annotation
offset <- 1  # Adjust this based on your data

# Initialize a list to store plots
plots <- list()

# Initialize a counter for the genes
counter <- 1

# Loop over the genes
for (gene in unique(final$GENE_SYMBOL)) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]
```

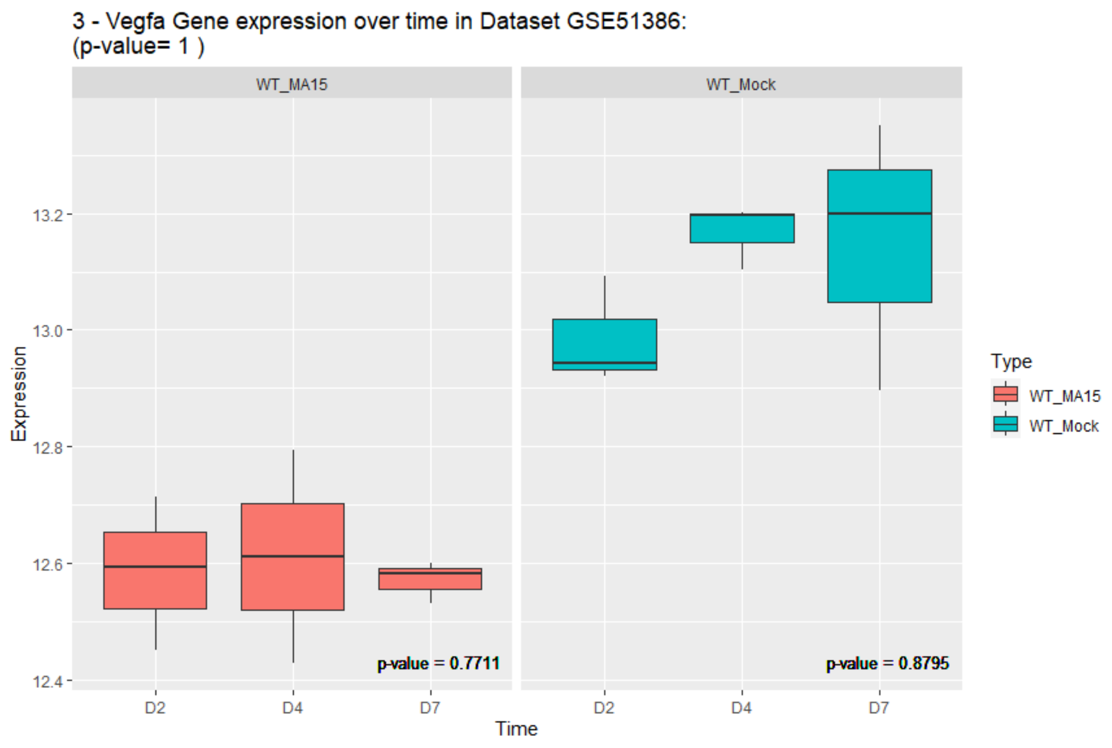
```
 # Extract p-values for each type
 p_value_MA15 <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_MA15), 4)
 p_value_Mock <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]
$P.Value_D4.vs.D7_WT_Mock), 4)

  overall_p_value <- round(as.numeric(df2[df2$GENE_SYMBOL == gene,]$adj.P.Val), 4)

 # Create a boxplot for the current gene
 p <- ggplot(final_gene, aes(x = Measurement, y = Expression, fill = Type)) +
   geom_boxplot() +
   facet_wrap(~Type) +
   labs(title = paste(counter, "-", gene, "Gene expression over time in Dataset
GSE51386:\n(p-value=", overall_p_value, ")")) +
   xlab("Time") +
   ylab("Expression") +
   # Annotate WT_MA15 facet with its p-value
   geom_text(data = subset(final_gene, Type == "WT_MA15"),
             aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_MA15)),
             color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE) +
   # Annotate WT_Mock facet with its p-value
   geom_text(data = subset(final_gene, Type == "WT_Mock"),
             aes(x = Inf, y = -Inf, label = paste("p-value =", p_value_Mock)),
             color = "black", hjust = 1.1, vjust = -1.5, size = 3.5, inherit.aes = FALSE)

 # Print the plot
 print(p)

 # Increment the counter
 counter <- counter + 1

}
```

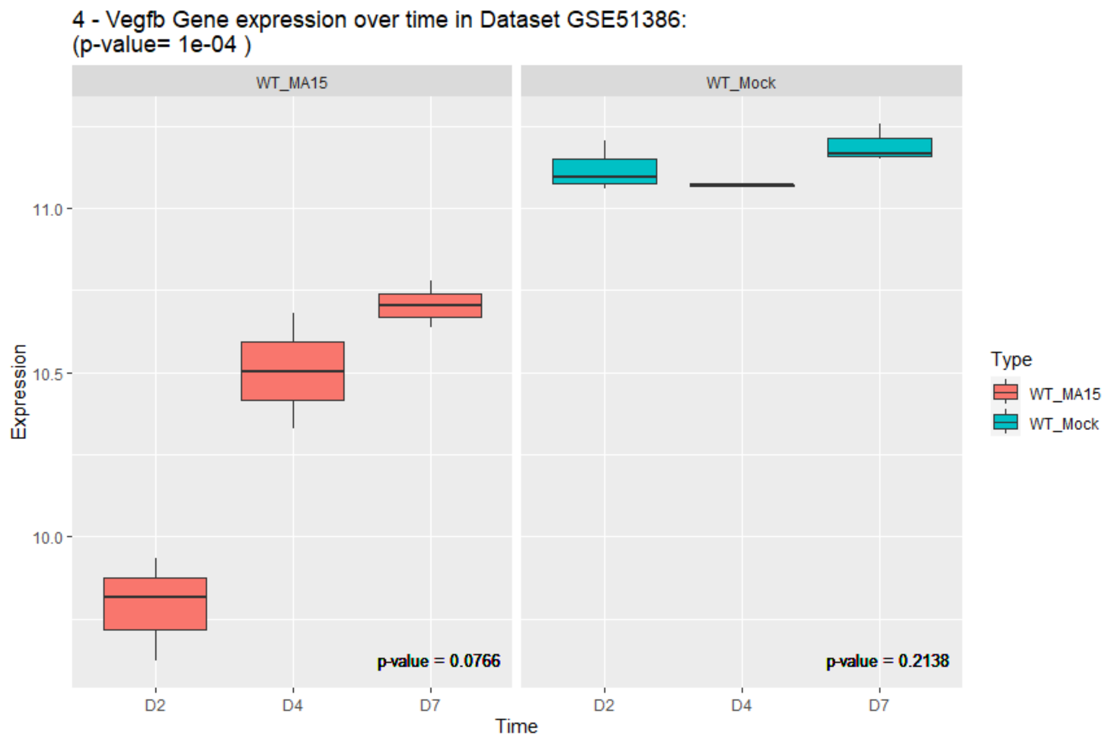
1 - Tnf Gene expression over time in Dataset GSE51386:
(p-value= 0.0022)



2 - Nfkb1 Gene expression over time in Dataset GSE51386:
(p-value= 0.0845)

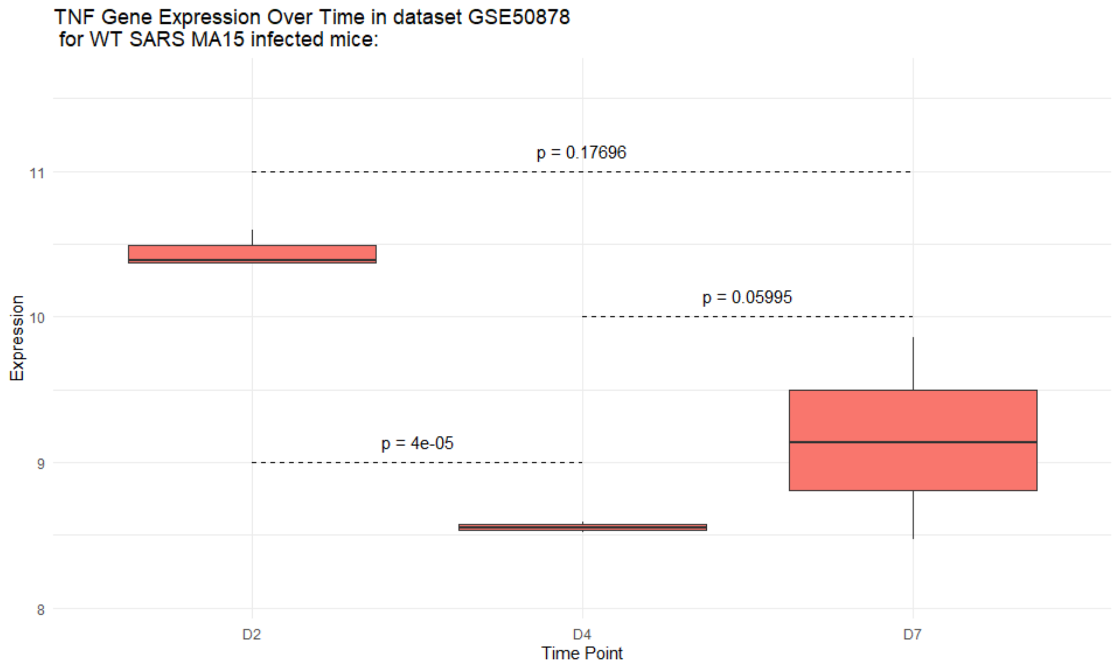




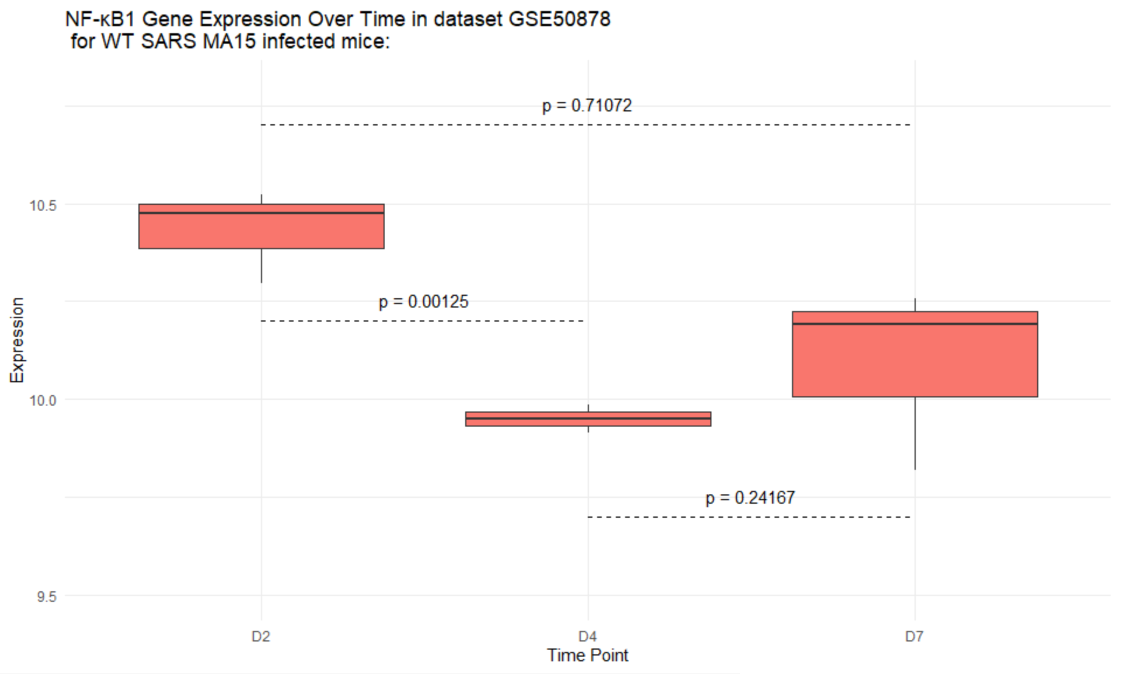


```
tnf_data <- subset(final, GENE_SYMBOL == "Tnf")

tnf_data_WT_MA15 <- tnf_data[tnf_data$Type == "WT_MA15",]

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- results$Tnf$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- results$Tnf$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- results$Tnf$P.Value_D2.vs.D7_WT_Mock

# Create a new data frame for the annotations for WT_MA15 only
```

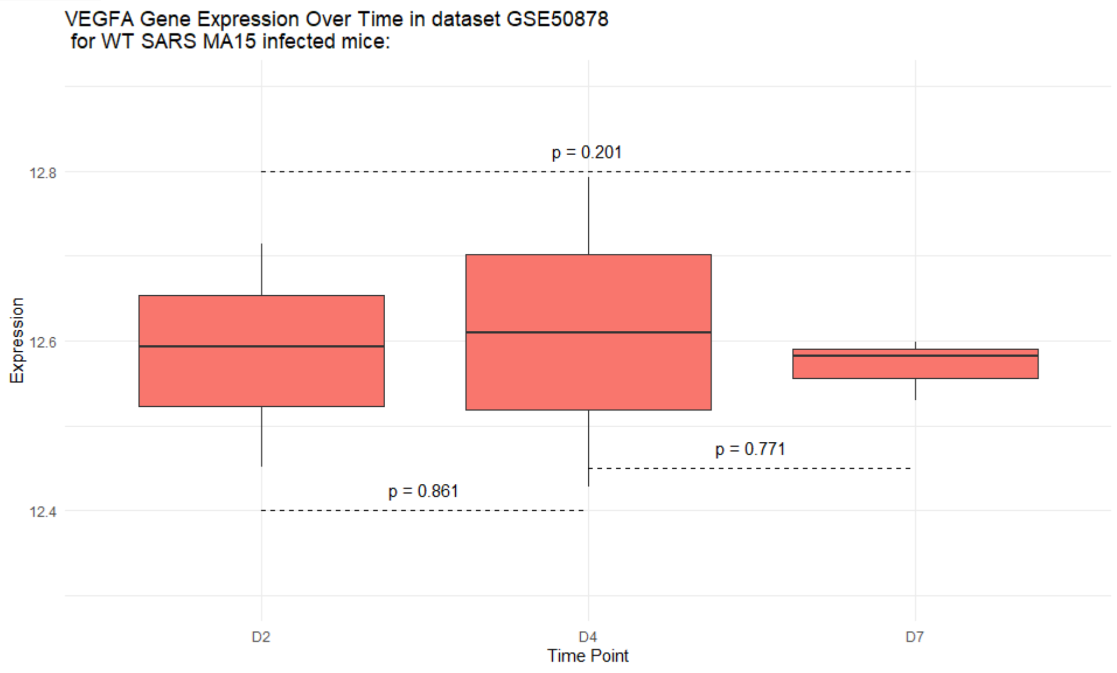
```
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", round(c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2), 5)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "TNF Gene Expression Over Time in dataset GSE50878 \n for WT SARS MA15
infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(8.1, 11.6) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 9, xend = 2, yend = 9), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 10, xend = 3, yend = 10), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11, xend = 3, yend = 11), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(9, 10, 11), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```



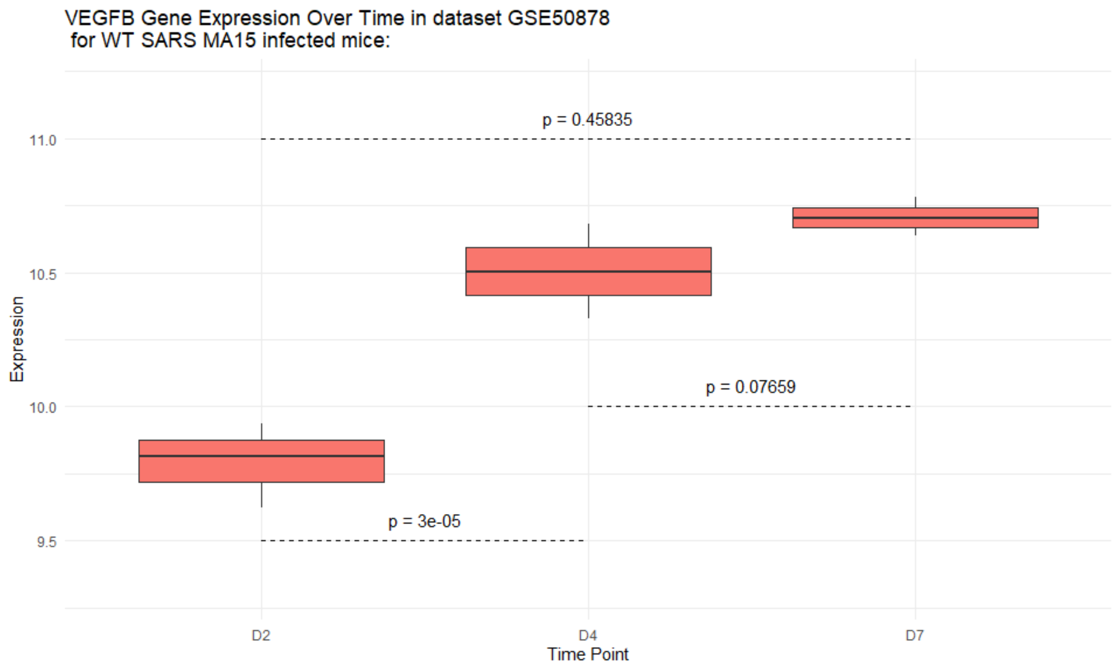
```
nfkb1_data <- subset(final, GENE_SYMBOL == "Nfkb1")

tnf_data_WT_MA15 <- nfkb1_data[nfkb1_data$Type == "WT_MA15",]

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- results$Nfkb1$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- results$Nfkb1$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- results$Nfkb1$P.Value_D2.vs.D7_WT_Mock

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", round(c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2), 5)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "NF-κB1 Gene Expression Over Time in dataset GSE50878 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(9.5, 10.8) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 10.2, xend = 2, yend = 10.2), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 9.7, xend = 3, yend = 9.7), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 10.7, xend = 3, yend = 10.7), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(10.2, 9.7, 10.7), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
vegfa_data <- subset(final, GENE_SYMBOL == "Vegfa")

tnf_data_WT_MA15 <- vegfa_data[vegfa_data$Type == "WT_MA15",]

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- results$Vegfa$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- results$Vegfa$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- results$Vegfa$P.Value_D2.vs.D7_WT_Mock

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", round(c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2), 3)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFA Gene Expression Over Time in dataset GSE50878 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(12.3, 12.9) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 12.4, xend = 2, yend = 12.4), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 12.45, xend = 3, yend = 12.45), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 12.8, xend = 3, yend = 12.8), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(12.4, 12.45, 12.8), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
vegfb_data <- subset(final, GENE_SYMBOL == "Vegfb")

tnf_data_WT_MA15 <- vegfb_data[vegfb_data$Type == "WT_MA15",]

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- results$Vegfb$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- results$Vegfb$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- results$Vegfb$P.Value_D2.vs.D7_WT_Mock

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", round(c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2), 5)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFB Gene Expression Over Time in dataset GSE50878 \n for WT SARS
MA15 infected mice:",
      x = "Time Point",
      y = "Expression") +
  theme_minimal() +
  ylim(9.3, 11.2) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 9.5, xend = 2, yend = 9.5), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 10, xend = 3, yend = 10), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11, xend = 3, yend = 11), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(9.5, 10, 11), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
#==================================================================
===========================
```

GSE68220:

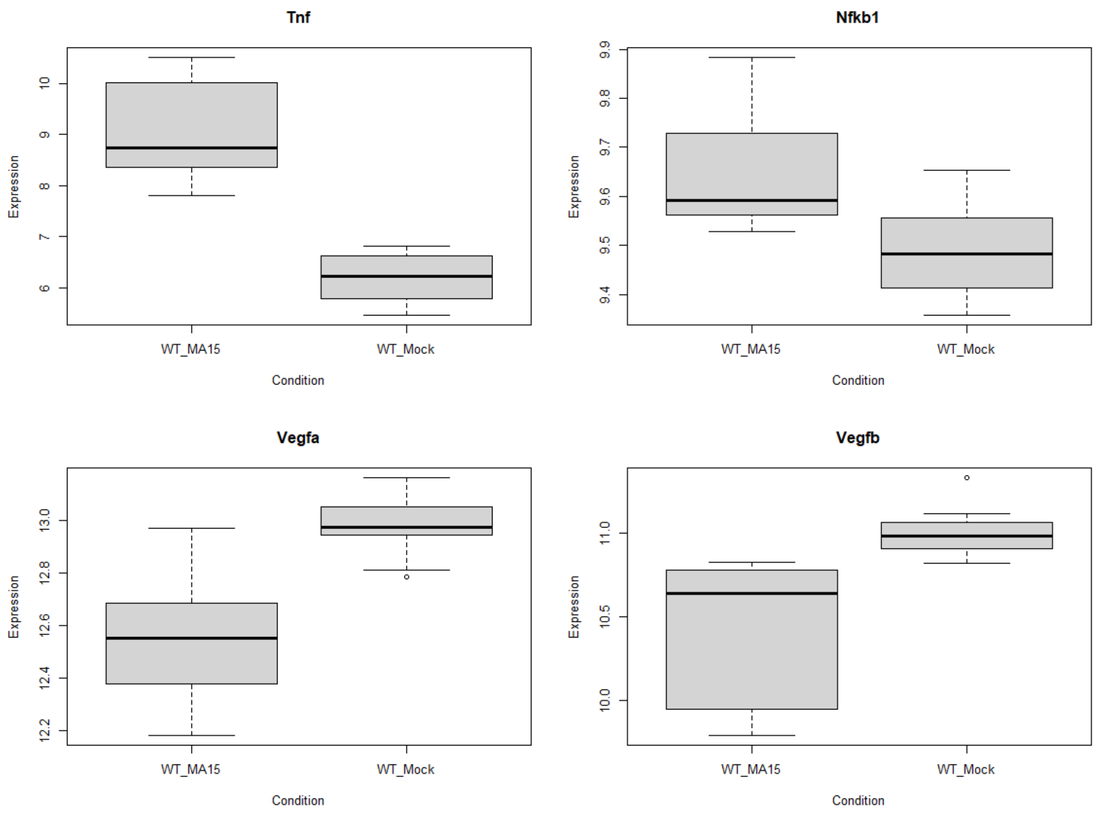
```
expr_data <- read.csv("Mus_SARS_GSE68820_quant_normalised.csv", header = TRUE,
stringsAsFactors = T, na.strings = c("","NA"))

attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "version.1.2", "noname")

setnames(expr_data, "X.19", "WT_Mock.D2")
```

```r
setnames(expr_data, "X.20", "WT_Mock.D2")
setnames(expr_data, "X.21", "WT_Mock.D2")
setnames(expr_data, "X.22", "WT_Mock.D2")
setnames(expr_data, "X.23", "WT_Mock.D2")

setnames(expr_data, "X.24", "WT_MA15.D2")
setnames(expr_data, "X.25", "WT_MA15.D2")
setnames(expr_data, "X.26", "WT_MA15.D2")
setnames(expr_data, "X.27", "WT_MA15.D2")
setnames(expr_data, "X.28", "WT_MA15.D2")

setnames(expr_data, "X.29", "WT_Mock.D4")
setnames(expr_data, "X.30", "WT_Mock.D4")
setnames(expr_data, "X.31", "WT_Mock.D4")
setnames(expr_data, "X.32", "WT_Mock.D4")
setnames(expr_data, "X.33", "WT_Mock.D4")

setnames(expr_data, "X.34", "WT_MA15.D4")
setnames(expr_data, "X.35", "WT_MA15.D4")
setnames(expr_data, "X.36", "WT_MA15.D4")
setnames(expr_data, "X.37", "WT_MA15.D4")

setnames(expr_data, "X.45", "WT_Mock.D7")
setnames(expr_data, "X.46", "WT_Mock.D7")
setnames(expr_data, "X.47", "WT_Mock.D7")
setnames(expr_data, "X.48", "WT_Mock.D7")

setnames(expr_data, "X.49", "WT_MA15.D7")
setnames(expr_data, "X.50", "WT_MA15.D7")
setnames(expr_data, "X.51", "WT_MA15.D7")
setnames(expr_data, "X.52", "WT_MA15.D7")

expr_data1 <- expr_data[15:29663, c(1, 3, 23:41, 49:56)]


write.csv(expr_data1, file = "GSE68220_dataset.csv")

data <- read.csv("GSE68220_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))


nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 8)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")

probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
library(limma)

final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
```

```r
# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

second_table3 <- df1
second_table3$dataset <- "GSE68220"
second_table3
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.89	7.57	9.91	13.2	3.88e-10	3.88e-10	GSE68220
0.14	9.56	3.76	-0.724	0.0009	0.0009	GSE68220
-0.428	12.8	-5.69	3.85	6.27e-06	6.27e-06	GSE68220
-0.582	10.7	-4.7	1.46	8.14e-05	8.14e-05	GSE68220

```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



Change in Time:

```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have 
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)
```

```

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D2.vs.D4_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D4,
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D2.vs.D7_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D7,
  D2.vs.D4_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D4,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,
  D2.vs.D7_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D2.vs.D4_WT_MA15", "D4.vs.D7_WT_MA15",
"D2.vs.D7_WT_MA15", "D2.vs.D4_WT_Mock", "D4.vs.D7_WT_Mock",
"D2.vs.D7_WT_Mock"), number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D2.vs.D4_WT_MA15 <- fit$p.value[, "D2.vs.D4_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D2.vs.D7_WT_MA15 <- fit$p.value[, "D2.vs.D7_WT_MA15"]
  gene_results$P.Value_D2.vs.D4_WT_Mock <- fit$p.value[, "D2.vs.D4_WT_Mock"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]
  gene_results$P.Value_D2.vs.D7_WT_Mock <- fit$p.value[, "D2.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Combine the original p-values with the additional p-values from the new temporal
contrasts
  # Update this line to include all the relevant p-value columns from your analysis
  pvalues <- c(df2$P.Value, df2$P.Value_D2.vs.D4_WT_MA15,
df2$P.Value_D4.vs.D7_WT_MA15, df2$P.Value_D2.vs.D7_WT_MA15,
df2$P.Value_D2.vs.D4_WT_Mock, df2$P.Value_D4.vs.D7_WT_Mock,
df2$P.Value_D2.vs.D7_WT_Mock)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")
  # Add the adjusted p-values to the data frame
  # Assuming the original data frame has a column 'adj.P.Val'
  # We replace it with the adjusted p-values calculated above
  df2$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))

df2$P.Value_D2.vs.D4_WT_MA15 <- ifelse(df2$P.Value_D2.vs.D4_WT_MA15 < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D4.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D2.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D2.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "f", digits = 4))


df2$P.Value_D2.vs.D4_WT_Mock <- ifelse(df2$P.Value_D2.vs.D4_WT_Mock < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$P.Value_D4.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

df2$P.Value_D2.vs.D7_WT_Mock <- ifelse(df2$P.Value_D2.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "f", digits = 4))

second_table4 <- df2
second_table4$dataset <- "GSE68220"
second_table4
```

D2.vs.D4_WT_MA15	D4.vs.D7_WT_MA15	D2.vs.D7_WT_MA15	D2.vs.D4_WT_Mock	D4.vs.D7_WT_Mock	D2.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D2.vs.D4_WT_MA15	P.Value_D4.vs.D7_WT_MA15	P.Value_D2.vs.D7_WT_MA15	P.Value_D2.vs.D4_WT_Mock	P.Value_D4.vs.D7_WT_Mock	P.Value_D2.vs.D7_WT_Mock	dataset
1.89	-0.126	1.77	0.0453	-0.416	-0.37	7.57	15.28	4.22e-06	2.95e-05	9.58e-07	0.6717	2.59e-06	0.8641	0.1488	0.1961	GSE68220
0.113	-0.00293	0.11	-0.0854	0.14	0.0545	9.56	2.91	0.0461	0.3227	0.0603	0.9613	0.0668	0.1250	0.0223	0.3474	GSE68220
0.311	0.236	0.547	-0.0153	0.182	0.167	12.8	21.4	3.63e-07	2.54e-06	7.39e-05	0.0020	2.39e-08	0.8002	0.0090	0.0155	GSE68220
-0.803	-0.104	-0.907	0.0156	-0.199	-0.184	10.7	60.1	3.40e-11	2.38e-10	6.88e-11	0.1537	7.03e-12	0.8067	0.0071	0.0120	GSE68220

```
second_table1 <- rownames_to_column(second_table1, "rowname")
second_table3 <- rownames_to_column(second_table3, "rowname")


df2 <- merge(second_table1, second_table3, all = TRUE)

df2 <- df2 %>% arrange(dataset)

df2
```

rowname	logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
Nfkb1	0.0792	10.1	0.907	-4.77	0.3788	0.3788	GSE50878
Tnf	2.32	8.26	7.3	4.93	2.60e-06	2.60e-06	GSE50878
Vegfa	-0.514	12.9	-7.53	5.28	1.81e-06	1.81e-06	GSE50878
Vegfb	-0.813	10.7	-5.26	1.49	9.54e-05	9.54e-05	GSE50878
Nfkb1	0.14	9.56	3.76	-0.724	0.0009	0.0009	GSE68220
Tnf	2.89	7.57	9.91	13.2	3.88e-10	3.88e-10	GSE68220
Vegfa	-0.428	12.8	-5.69	3.85	6.27e-06	6.27e-06	GSE68220
Vegfb	-0.582	10.7	-4.7	1.46	8.14e-05	8.14e-05	GSE68220

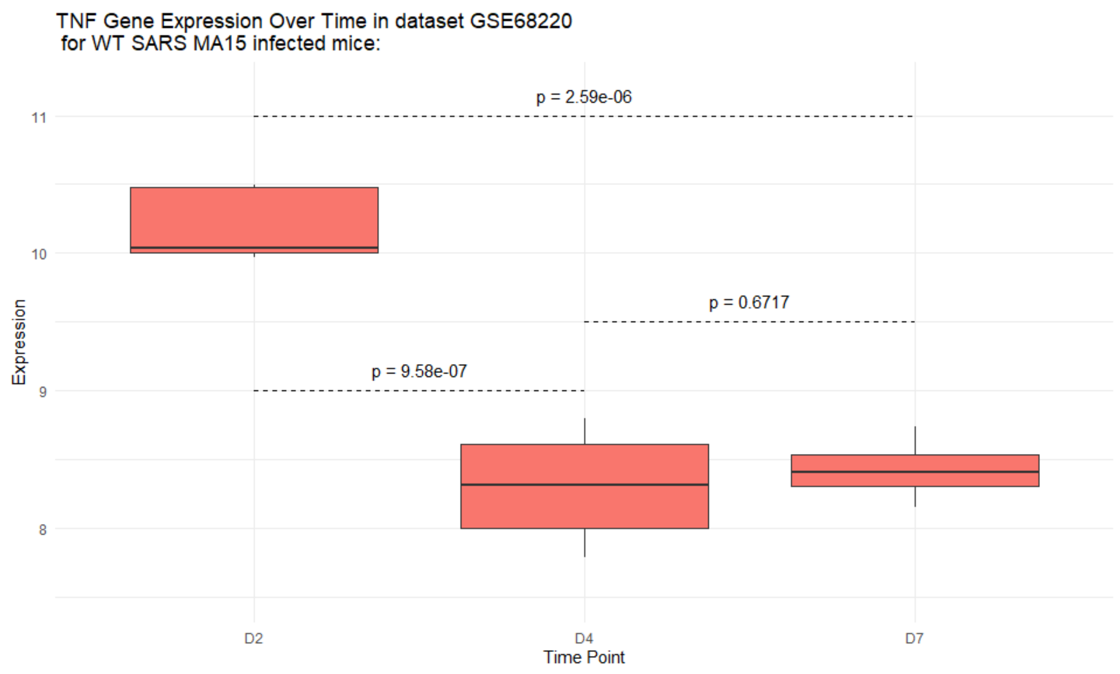
```
second_table2 <- rownames_to_column(second_table2, "rowname")
second_table4 <- rownames_to_column(second_table4, "rowname")


df3 <- merge(second_table2, second_table4, all = TRUE)

df3 <- df3 %>% arrange(dataset)

df3
```

rowname	D2.vs.D4_WT_MA15	D4.vs.D7_WT_MA15	D2.vs.D7_WT_MA15	D2.vs.D4_WT_Mock	D4.vs.D7_WT_Mock	D2.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D2.vs.D4_WT_MA15	P.Value_D4.vs.D7_WT_MA15	P.Value_D2.vs.D7_WT_MA15	P.Value_D2.vs.D4_WT_Mock	P.Value_D4.vs.D7_WT_Mock	P.Value_D2.vs.D7_WT_Mock	dataset
Nfkb1	0.48	-0.138	0.342	-0.0181	-0.0198	-0.038	10.1	5.36	0.0121	0.0845	0.0013	0.2417	0.0057	0.8591	0.8460	0.7107	GSE50878
Tnf	1.9	-0.602	1.29	-0.557	0.186	-0.371	8.26	13.5	0.0003	0.0022	3.85e-05	0.0599	0.0004	0.0532	0.4849	0.1770	GSE50878

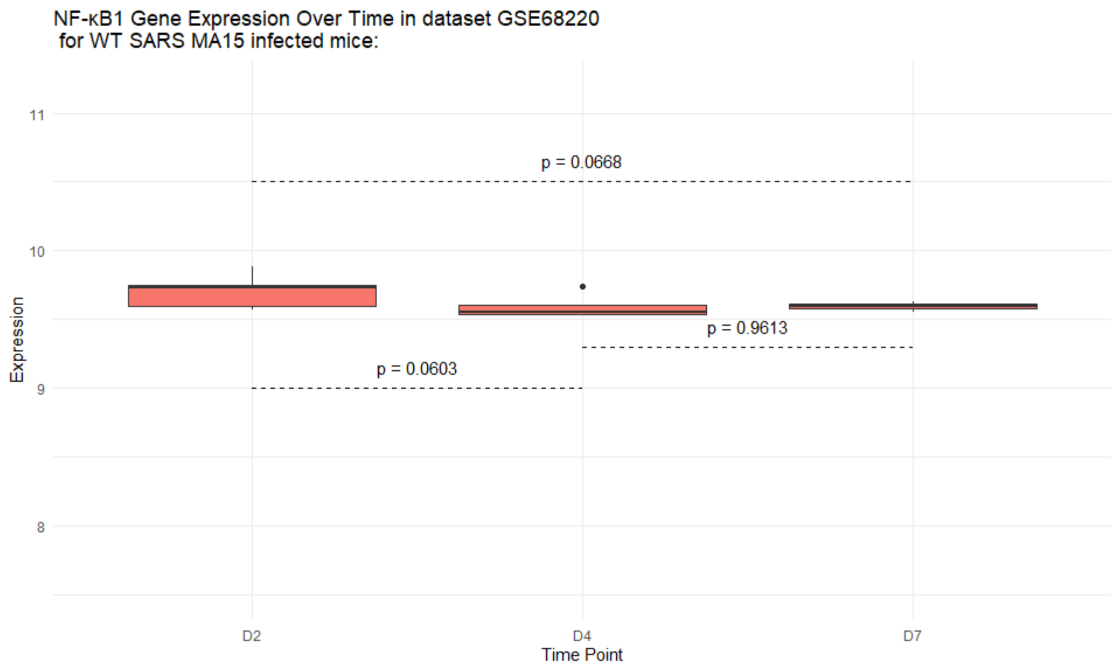
Vegfa	-0.024	0.0398	0.0158	-0.181	0.0185	-0.162	12.9	0.719	0.5968	1.0000	0.8608	0.7711	0.8968	0.1576	0.8795	0.2006	GSE50878
Vegfb	-0.713	-0.202	-0.915	0.051	-0.122	-0.0712	10.7	26.8	1.29e-05	9.01e-05	2.64e-05	0.0766	8.33e-07	0.5927	0.2138	0.4584	GSE50878
Nfkb1	0.113	-0.00293	0.11	-0.0854	0.14	0.0545	9.56	2.91	0.04461	0.3227	0.0603	0.9613	0.0668	0.1250	0.0223	0.3474	GSE68220
Tnf	1.89	-0.126	1.77	0.0453	-0.416	-0.37	7.57	15.8	4.22e-06	2.95e-05	9.58e-07	0.6717	2.59e-06	0.8641	0.1488	0.1961	GSE68220
Vegfa	0.311	0.236	0.547	-0.0153	0.182	0.167	12.8	21.4	3.63e-07	2.54e-06	7.39e-05	0.0020	2.39e-08	0.8002	0.0090	0.0155	GSE68220
Vegfb	-0.803	-0.104	-0.907	0.0156	-0.199	-0.184	10.7	60.1	3.40e-11	2.38e-10	6.88e-11	0.1537	7.03e-12	0.8067	0.0071	0.0120	GSE68220

```
tnf_data <- subset(final, GENE_SYMBOL == "Tnf")

tnf_data_WT_MA15 <- tnf_data[tnf_data$Type == "WT_MA15",]
```

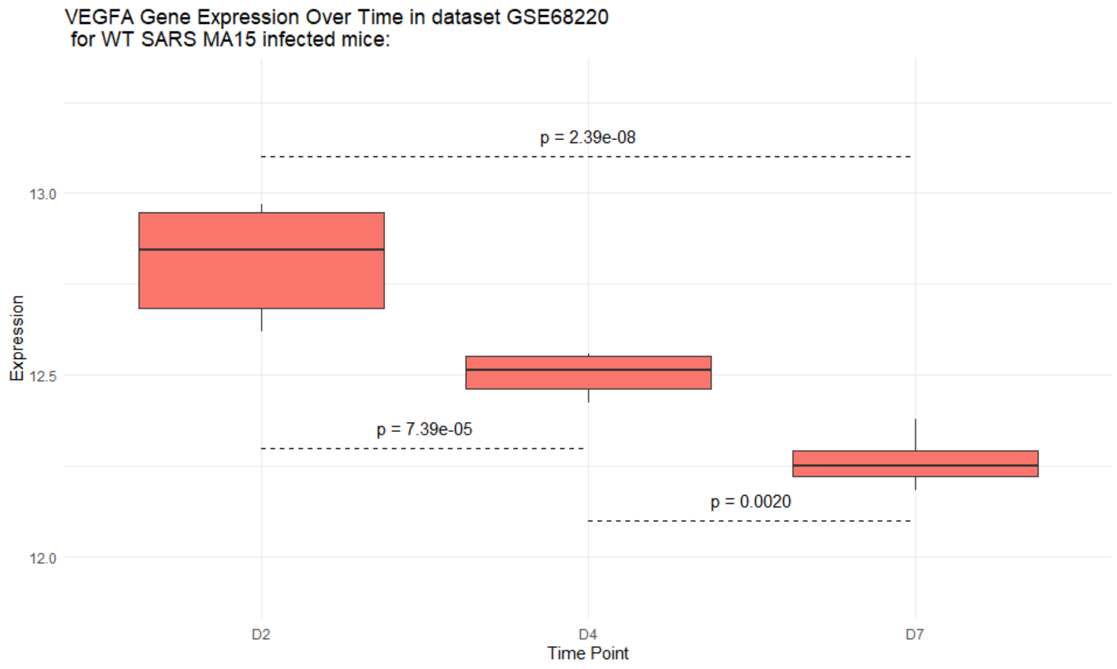
```r
df3_data <- subset(df3, dataset == "GSE68220")

res <- subset(df3_data, rowname == "Tnf")

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)


# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "TNF Gene Expression Over Time in dataset GSE68220 \n for WT SARS MA15
infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(7.5, 11.2) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 9, xend = 2, yend = 9), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 9.5, xend = 3, yend = 9.5), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11, xend = 3, yend = 11), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(9, 9.5, 11), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
nfkb1_data <- subset(final, GENE_SYMBOL == "Nfkb1")

tnf_data_WT_MA15 <- nfkb1_data[nfkb1_data$Type == "WT_MA15",]

df3_data <- subset(df3, dataset == "GSE68220")

res <- subset(df3_data, rowname == "Nfkb1")

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)


# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "NF-κB1 Gene Expression Over Time in dataset GSE68220 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(7.5, 11.2) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 9, xend = 2, yend = 9), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 9.3, xend = 3, yend = 9.3), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 10.5, xend = 3, yend = 10.5), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(9, 9.3, 10.5), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```

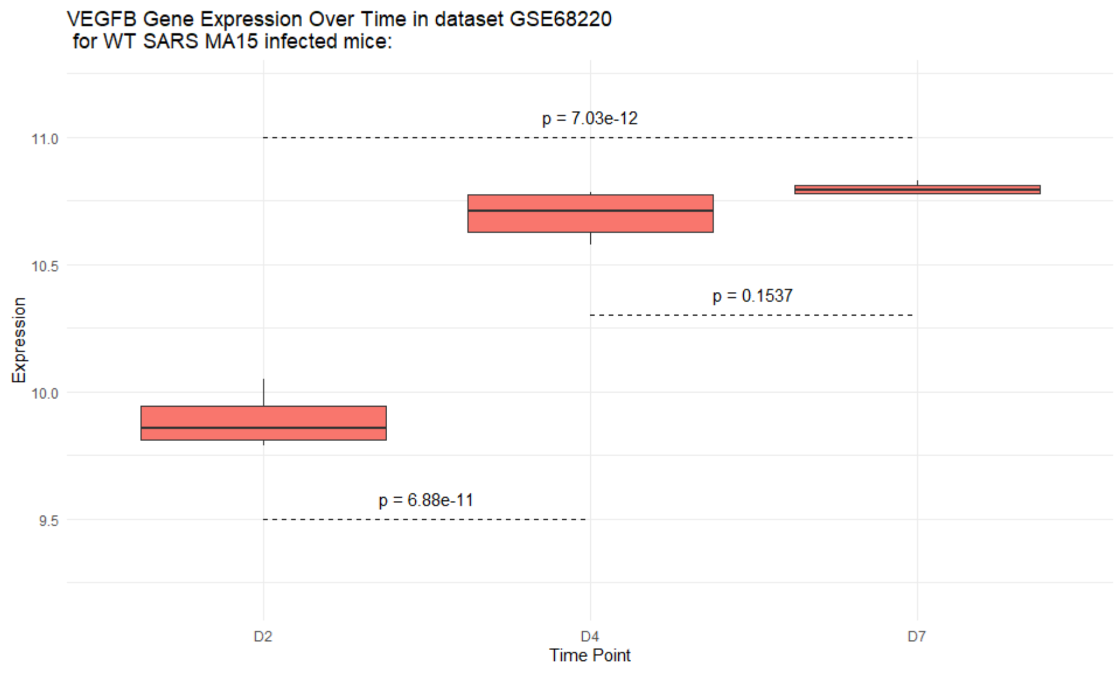


```
Vegfa_data <- subset(final, GENE_SYMBOL == "Vegfa")

tnf_data_WT_MA15 <- Vegfa_data[Vegfa_data$Type == "WT_MA15",]

df3_data <- subset(df3, dataset == "GSE68220")

res <- subset(df3_data, rowname == "Vegfa")

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)


# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFA Gene Expression Over Time in dataset GSE68220 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(11.9, 13.3) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 12.3, xend = 2, yend = 12.3), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 12.1, xend = 3, yend = 12.1), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 13.1, xend = 3, yend = 13.1), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(12.3, 12.1, 13.1), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
Vegfb_data <- subset(final, GENE_SYMBOL == "Vegfb")
```

```r
tnf_data_WT_MA15 <- Vegfb_data[Vegfb_data$Type == "WT_MA15",]

df3_data <- subset(df3, dataset == "GSE68220")

res <- subset(df3_data, rowname == "Vegfb")

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D4", "D7", "D7"),
  Comparison = c("D2", "D4", "D2"),
  Label = paste("p =", c(pvalues1_WT_MA15_D4_D2, pvalues1_WT_MA15_D7_D4,
pvalues1_WT_MA15_D7_D2)),
  Y = c(8, 9, 10)  # Different y-coordinates for each p-value
)


# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFB Gene Expression Over Time in dataset GSE68220 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(9.2, 11.2) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 9.5, xend = 2, yend = 9.5), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 10.3, xend = 3, yend = 10.3), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11, xend = 3, yend = 11), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.5, 2.5, 2), y = c(9.5, 10.3, 11), label =
annotations_WT_MA15$Label, vjust = -1)

print(p)
```



#==
=========================

GSE40840:

```
expr_data <- read.csv("Mus_SARS_GSE40840.csv", header = TRUE, stringsAsFactors = T,
na.strings = c("","NA"))

attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X", "WT_MA15.D4")
setnames(expr_data, "X.1", "WT_MA15.D4")

setnames(expr_data, "X.2", "WT_MA15.D7")
setnames(expr_data, "X.3", "WT_MA15.D7")
setnames(expr_data, "X.4", "WT_MA15.D7")

setnames(expr_data, "X.5", "WT_Mock.D4")
setnames(expr_data, "X.6", "WT_Mock.D4")

setnames(expr_data, "X.7", "WT_Mock.D7")
setnames(expr_data, "X.8", "WT_Mock.D7")
setnames(expr_data, "X.9", "WT_Mock.D7")

expr_data1 <- expr_data[20:41193, c(1, 2, 4:13)]


write.csv(expr_data1, file = "GSE40840_dataset.csv")

data <- read.csv("GSE40840_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")
```

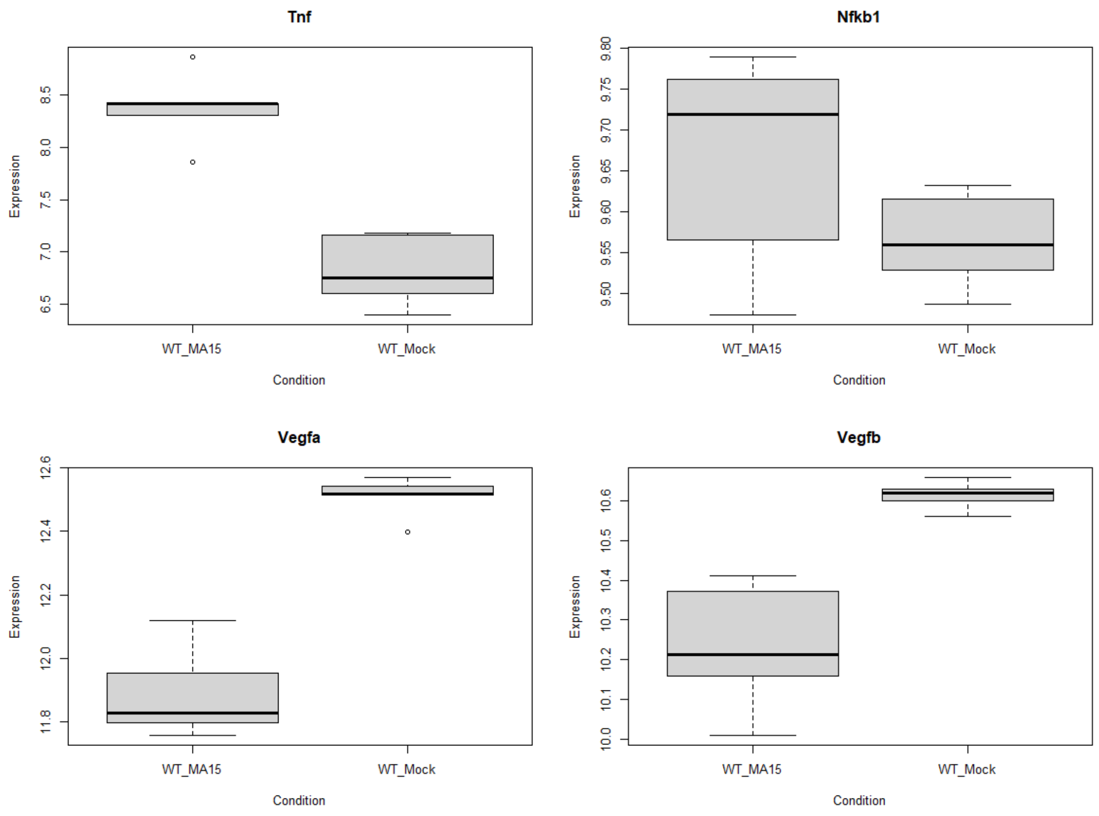
```
probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
```

```
combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

first_table1<- df1
first_table1$dataset <- "GSE40840"
first_table1
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
1.55	7.6	7.01	1.68	0.0001	0.0001	GSE40840
0.0972	9.61	1.46	-4.53	0.1814	0.1814	GSE40840
-0.619	12.2	-8.61	3.11	2.56e-05	2.56e-05	GSE40840

-0.38	10.4	-5.09	-0.343	0.0009	0.0009	GSE40840

```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



```
final_summary <- final %>%
  dplyr::group_by(GENE_SYMBOL, Type, Measurement) %>%
  dplyr::summarize(MeanExpression = mean(Expression), .groups = 'drop')

# Create the plot
elegant_plot <- ggplot(final_summary, aes(x = Measurement, y = MeanExpression, fill =
Type)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.7)) +
  facet_wrap(~GENE_SYMBOL, scales = 'free_y') +
  theme_minimal() +
  theme(
    text = element_text(size = 12),
    plot.title = element_text(hjust = 0.5),
    legend.position = "bottom",
    strip.text.x = element_text(size = 14, face = "bold"),
    axis.text.x = element_text(angle = 45, hjust = 1)
  ) +
  labs(
    title = "Gene Expression under Different Conditions",
    x = "Time Point",
    y = "Mean Expression Level",
    fill = "Type"
  ) +
  scale_fill_brewer(palette = "Pastel1")
```

Change in Time:

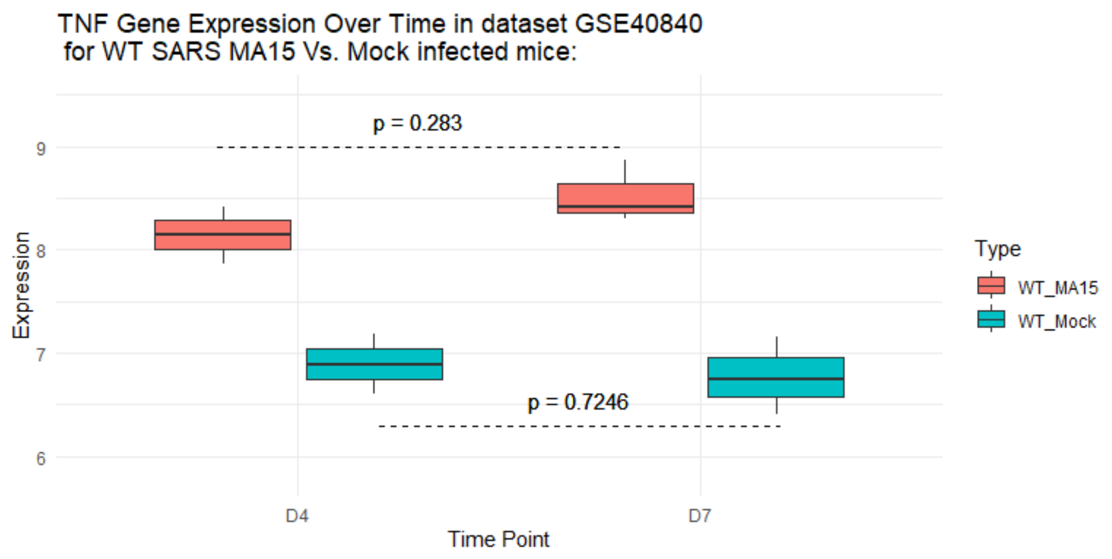
```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D4.vs.D7_WT_MA15", "D4.vs.D7_WT_Mock"),
```

```r
number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]


  pvalues <- c(df2$P.Value)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  df$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Define the column names for p-values
  pvalue_cols <- c("P.Value", "adj.P.Val", "P.Value_D4.vs.D7_WT_MA15",
"P.Value_D4.vs.D7_WT_Mock")

  # Check if all p-value columns exist in the dataframe
  if (all(pvalue_cols %in% names(df2))) {
    # Store the p-values in a separate data frame
    pvalues <- df2[, pvalue_cols]

    # Remove the p-values from the original data frame
    df2 <- df2[, setdiff(names(df2), names(pvalues))]

    # Round the numbers to 3 decimal points
    df2 <- round(df2, 4)

    # Add the p-values back to the data frame
    df2 <- cbind(df2, pvalues)
    # Store the updated data frame in the list
    results[[i]] <- df2
  }
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$P.Value_D4.vs.D7_WT_MA15 < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$P.Value_D4.vs.D7_WT_Mock < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

first_table2 <- df2
first_table2$dataset <- "GSE40840"
first_table2
```

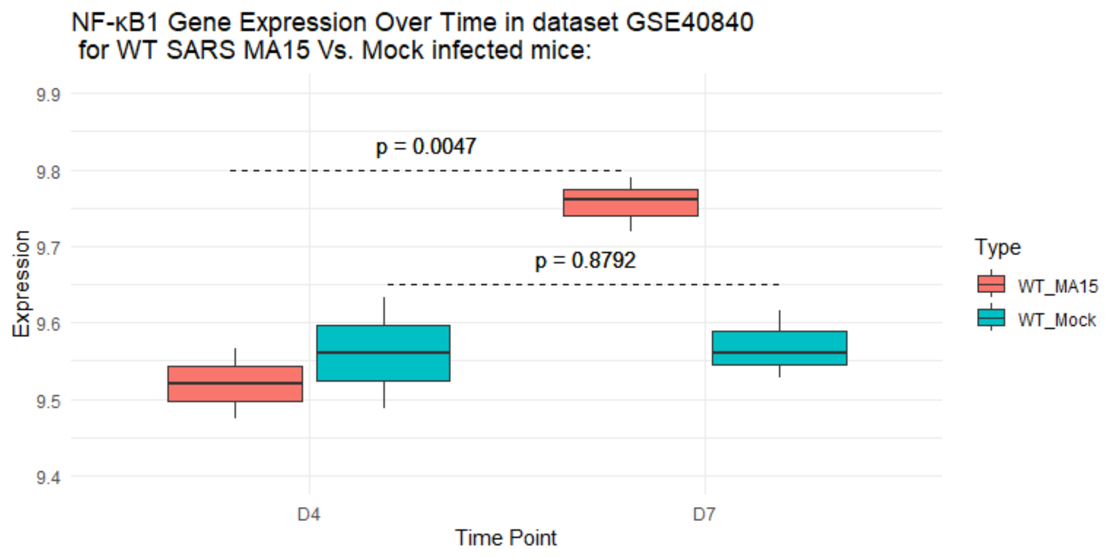
D4.vs.D7_WT_MA15	D4.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D4.vs.D7_WT_MA15	P.Value_D4.vs.D7_WT_Mock	dataset
-0.389	0.122	7.6	0.763	0.5067	0.5067	0.2830	0.7246	GSE40840
-0.236	-0.0086	9.61	9.55	0.0137	0.0137	0.0047	0.8792	GSE40840
0.242	0.0584	12.2	7.73	0.0219	0.0219	0.0087	0.3924	GSE40840
-0.203	-0.007	10.4	2.37	0.1741	0.1741	0.0723	0.9427	GSE40840

```r
first_table2$GENE_SYMBOL <- row.names(first_table2)
tnf_data_WT_MA15 <- subset(final, GENE_SYMBOL == "Tnf")

res <- subset(first_table2, GENE_SYMBOL == "Tnf")


format_pvalue <- function(pvalue) {
  if (is.na(pvalue)) {
    return(NA)
  } else if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 5))
  }
}
```

```r
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_Mock_D7_D4 <- res$P.Value_D4.vs.D7_WT_Mock

pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_Mock_D7_D4 <- as.numeric(pvalues1_WT_Mock_D7_D4)

annotations_WT_MA15 <- data.frame(
  Type = rep(c("WT_MA15", "WT_Mock"), each = 2),
  Measurement = c("D7", "D7", "D7", "D7"),
  Comparison = c("D4", "D4", "D4", "D4"),
  Label = paste("p =", sapply(c(pvalues1_WT_MA15_D7_D4,
pvalues1_WT_Mock_D7_D4), format_pvalue)),
  Y = c(8, 9, 10, 11)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "TNF Gene Expression Over Time in dataset GSE40840 \n for WT SARS MA15
Vs. Mock infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(5.8, 9.5)


p <- p + geom_segment(aes(x = 0.8, y = 9, xend = 1.8, yend = 9), linetype = "dashed",
color = "black") + geom_segment(aes(x = 1.2, y = 6.3, xend = 2.2, yend = 6.3), linetype =
"dashed", color = "black")



# Add lines
p <- p + annotate("text", x = c(1.3, 1.7), y = c(9, 6.3), label =
annotations_WT_MA15$Label[1:2], vjust = -1)

# Add p-values for the second group
p <- p + annotate("text", x = c(1.3, 1.7), y = c(9, 6.3), label =
annotations_WT_MA15$Label[3:4], vjust = -1)

print(p)
```

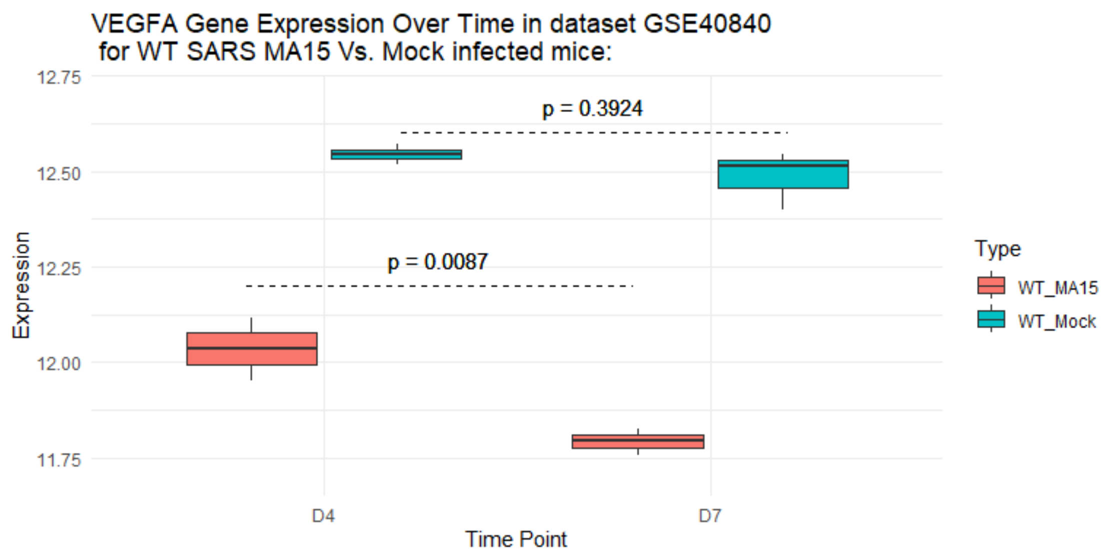


```
tnf_data_WT_MA15 <- subset(final, GENE_SYMBOL == "Nfkb1")

res <- subset(first_table2, GENE_SYMBOL == "Nfkb1")


format_pvalue <- function(pvalue) {
  if (is.na(pvalue)) {
    return(NA)
  } else if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 5))
  }
}


pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_Mock_D7_D4 <- res$P.Value_D4.vs.D7_WT_Mock

pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_Mock_D7_D4 <- as.numeric(pvalues1_WT_Mock_D7_D4)

annotations_WT_MA15 <- data.frame(
  Type = rep(c("WT_MA15", "WT_Mock"), each = 2),
  Measurement = c("D7", "D7", "D7", "D7"),
  Comparison = c("D4", "D4", "D4", "D4"),
  Label = paste("p =", sapply(c(pvalues1_WT_MA15_D7_D4,
pvalues1_WT_Mock_D7_D4), format_pvalue)),
  Y = c(8, 9, 10, 11)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "NF-κB1 Gene Expression Over Time in dataset GSE40840 \n for WT SARS
MA15 Vs. Mock infected mice:",
    x = "Time Point",
    y = "Expression") +
  theme_minimal() +
  ylim(9.4, 9.9)


p <- p + geom_segment(aes(x = 0.8, y = 9.8, xend = 1.8, yend = 9.8), linetype = "dashed",
color = "black") + geom_segment(aes(x = 1.2, y = 9.65, xend = 2.2, yend = 9.65), linetype =
"dashed", color = "black")



# Add lines
p <- p + annotate("text", x = c(1.3, 1.7), y = c(9.8, 9.65), label =
annotations_WT_MA15$Label[1:2], vjust = -1)

# Add p-values for the second group
p <- p + annotate("text", x = c(1.3, 1.7), y = c(9.8, 9.65), label =
annotations_WT_MA15$Label[3:4], vjust = -1)

print(p)
```

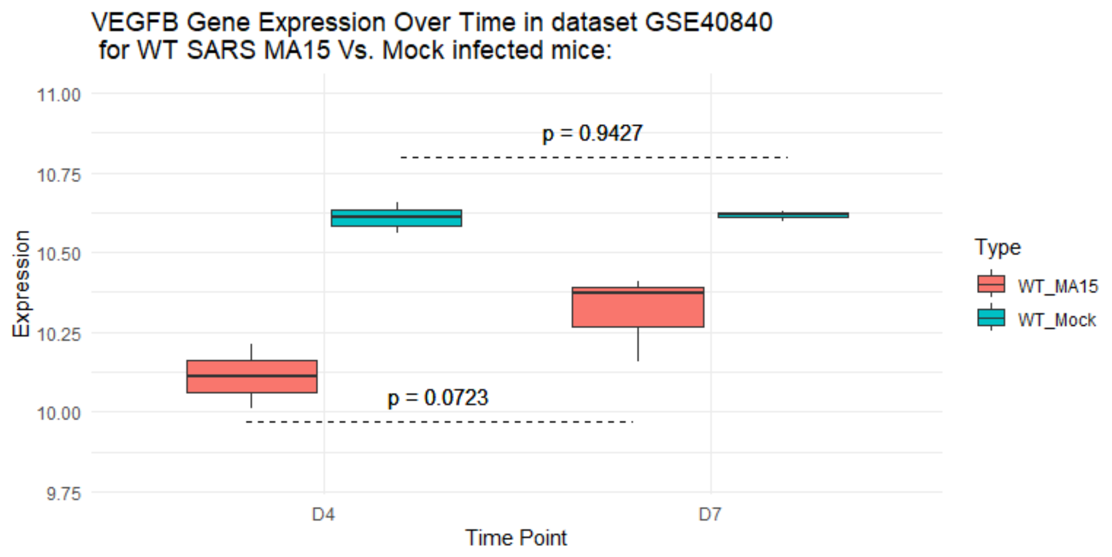


```
tnf_data_WT_MA15 <- subset(final, GENE_SYMBOL == "Vegfa")

res <- subset(first_table2, GENE_SYMBOL == "Vegfa")


format_pvalue <- function(pvalue) {
  if (is.na(pvalue)) {
    return(NA)
  } else if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 5))
  }
}
```

```
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_Mock_D7_D4 <- res$P.Value_D4.vs.D7_WT_Mock

pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_Mock_D7_D4 <- as.numeric(pvalues1_WT_Mock_D7_D4)

annotations_WT_MA15 <- data.frame(
  Type = rep(c("WT_MA15", "WT_Mock"), each = 2),
  Measurement = c("D7", "D7", "D7", "D7"),
  Comparison = c("D4", "D4", "D4", "D4"),
  Label = paste("p =", sapply(c(pvalues1_WT_MA15_D7_D4,
pvalues1_WT_Mock_D7_D4), format_pvalue)),
  Y = c(8, 9, 10, 11)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFA Gene Expression Over Time in dataset GSE40840 \n for WT SARS
MA15 Vs. Mock infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(11.7, 12.7)


p <- p + geom_segment(aes(x = 0.8, y = 12.2, xend = 1.8, yend = 12.2), linetype =
"dashed", color = "black") + geom_segment(aes(x = 1.2, y = 12.6, xend = 2.2, yend = 12.6),
linetype = "dashed", color = "black")



# Add lines
p <- p + annotate("text", x = c(1.3, 1.7), y = c(12.2, 12.6), label =
annotations_WT_MA15$Label[1:2], vjust = -1)

# Add p-values for the second group
p <- p + annotate("text", x = c(1.3, 1.7), y = c(12.2, 12.6), label =
annotations_WT_MA15$Label[3:4], vjust = -1)

print(p)
```



```
tnf_data_WT_MA15 <- subset(final, GENE_SYMBOL == "Vegfb")

res <- subset(first_table2, GENE_SYMBOL == "Vegfb")


format_pvalue <- function(pvalue) {
  if (is.na(pvalue)) {
    return(NA)
  } else if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 5))
  }
}


pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_Mock_D7_D4 <- res$P.Value_D4.vs.D7_WT_Mock

pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_Mock_D7_D4 <- as.numeric(pvalues1_WT_Mock_D7_D4)

annotations_WT_MA15 <- data.frame(
  Type = rep(c("WT_MA15", "WT_Mock"), each = 2),
  Measurement = c("D7", "D7", "D7", "D7"),
  Comparison = c("D4", "D4", "D4", "D4"),
  Label = paste("p =", sapply(c(pvalues1_WT_MA15_D7_D4,
pvalues1_WT_Mock_D7_D4), format_pvalue)),
  Y = c(8, 9, 10, 11)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFB Gene Expression Over Time in dataset GSE40840 \n for WT SARS
MA15 Vs. Mock infected mice:",
    x = "Time Point",
    y = "Expression") +
theme_minimal() +
ylim(9.8, 11)


p <- p + geom_segment(aes(x = 0.8, y = 9.97, xend = 1.8, yend = 9.97), linetype =
"dashed", color = "black") + geom_segment(aes(x = 1.2, y = 10.8, xend = 2.2, yend = 10.8),
linetype = "dashed", color = "black")



# Add lines
p <- p + annotate("text", x = c(1.3, 1.7), y = c(9.97, 10.8), label =
annotations_WT_MA15$Label[1:2], vjust = -1)

# Add p-values for the second group
p <- p + annotate("text", x = c(1.3, 1.7), y = c(9.97, 10.8), label =
annotations_WT_MA15$Label[3:4], vjust = -1)

print(p)
```



```
#===========================================================
=========================
```

Figure 8.4, Table 8.5, Table 8.6

GSE49262:

```
expr_data <- read.csv("Mus_SARS_GSE49262.csv", header = TRUE, stringsAsFactors = T,
na.strings = c("","NA"))

attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")
```

```r
setnames(expr_data, "X.12", "WT_MA15.D1")
setnames(expr_data, "X.13", "WT_MA15.D1")
setnames(expr_data, "X.14", "WT_MA15.D1")

setnames(expr_data, "X.15", "WT_MA15.D2")
setnames(expr_data, "X.16", "WT_MA15.D2")
setnames(expr_data, "X.17", "WT_MA15.D2")

setnames(expr_data, "X.18", "WT_MA15.D4")
setnames(expr_data, "X.19", "WT_MA15.D4")
setnames(expr_data, "X.20", "WT_MA15.D4")

setnames(expr_data, "X.21", "WT_MA15.D7")
setnames(expr_data, "X.22", "WT_MA15.D7")
setnames(expr_data, "X.23", "WT_MA15.D7")

setnames(expr_data, "X.24", "WT_Mock.D1")
setnames(expr_data, "X.25", "WT_Mock.D1")
setnames(expr_data, "X.26", "WT_Mock.D1")
setnames(expr_data, "X.27", "WT_Mock.D2")
setnames(expr_data, "X.28", "WT_Mock.D2")
setnames(expr_data, "X.29", "WT_Mock.D2")

setnames(expr_data, "X.30", "WT_Mock.D4")
setnames(expr_data, "X.31", "WT_Mock.D4")
setnames(expr_data, "X.32", "WT_Mock.D4")

setnames(expr_data, "X.33", "WT_Mock.D7")
setnames(expr_data, "X.34", "WT_Mock.D7")


expr_data1 <- expr_data[15:41188, c(1, 2, 16:38)]


write.csv(expr_data1, file = "GSE49262_dataset.csv")

data <- read.csv("GSE49262_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement", 
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")

probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
```

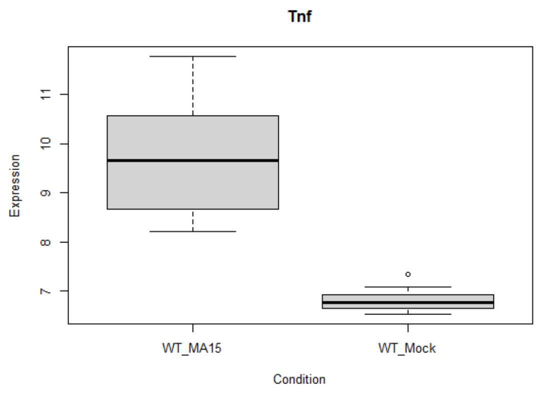
```
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
library(limma)
```

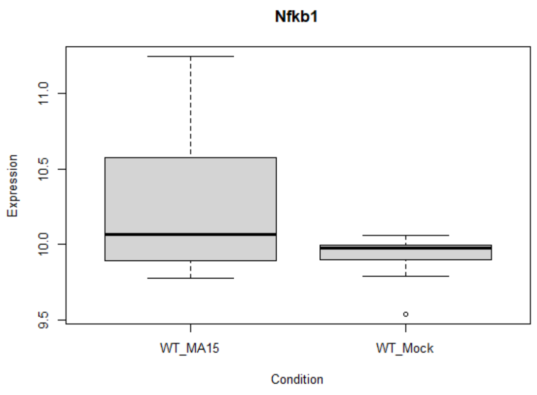
```r
final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]
  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

third_table1 <- df1
third_table1$dataset <- "GSE49262"
third_table1
```

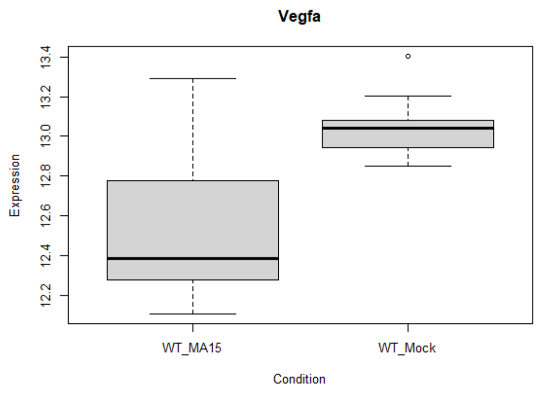
logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.85	8.31	8.13	8.36	6.31e-08	6.31e-08	GSE49262
0.336	10.1	2.19	-3.73	0.0396	0.0396	GSE49262
-0.513	12.8	-3.98	-0.412	0.0007	0.0007	GSE49262
-1.01	10.5	-7.31	6.72	3.38e-07	3.38e-07	GSE49262

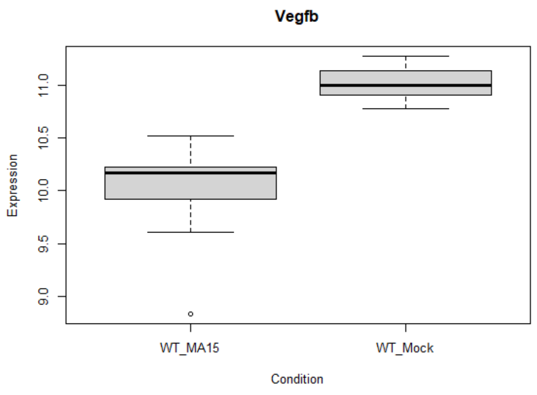
```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
```

```
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```









Change in Time:

```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)
```

```r
 # Fit a linear model
 fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
 # For the MA_10_4 group
 D1.vs.D2_WT_MA15 = InteractionWT_MA15.D1 - InteractionWT_MA15.D2,
 D2.vs.D4_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D4,
 D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
 D2.vs.D7_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D7,
 D1.vs.D4_WT_MA15 = InteractionWT_MA15.D1 - InteractionWT_MA15.D4,
 D1.vs.D7_WT_MA15 = InteractionWT_MA15.D1 - InteractionWT_MA15.D7,

 D1.vs.D2_WT_Mock = InteractionWT_Mock.D1 - InteractionWT_Mock.D2,
 D2.vs.D4_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D4,
 D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,
 D2.vs.D7_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D7,
 D1.vs.D4_WT_Mock = InteractionWT_Mock.D1 - InteractionWT_Mock.D4,
 D1.vs.D7_WT_Mock = InteractionWT_Mock.D1 - InteractionWT_Mock.D7,


 levels = colnames(design)
)

 # Apply eBayes
 fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

 # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D1.vs.D2_WT_MA15", "D2.vs.D4_WT_MA15",
"D4.vs.D7_WT_MA15", "D2.vs.D7_WT_MA15", "D1.vs.D4_WT_MA15",
"D1.vs.D7_WT_MA15", "D1.vs.D2_WT_Mock", "D2.vs.D4_WT_Mock",
"D4.vs.D7_WT_Mock", "D2.vs.D7_WT_Mock", "D1.vs.D4_WT_Mock",
"D1.vs.D7_WT_Mock"), number = Inf)


 # Adjust P-values using Benjamini-Hochberg method
 gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

 # Store the p-values for each comparison in the results
 # The column names should match the ones used in topTable
 gene_results$P.Value_D1.vs.D2_WT_MA15 <- fit$p.value[, "D1.vs.D2_WT_MA15"]
 gene_results$P.Value_D2.vs.D4_WT_MA15 <- fit$p.value[, "D2.vs.D4_WT_MA15"]
 gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
 gene_results$P.Value_D2.vs.D7_WT_MA15 <- fit$p.value[, "D2.vs.D7_WT_MA15"]
 gene_results$P.Value_D1.vs.D4_WT_MA15 <- fit$p.value[, "D1.vs.D4_WT_MA15"]
 gene_results$P.Value_D1.vs.D7_WT_MA15 <- fit$p.value[, "D1.vs.D7_WT_MA15"]

 gene_results$P.Value_D1.vs.D2_WT_Mock <- fit$p.value[, "D1.vs.D2_WT_Mock"]
 gene_results$P.Value_D2.vs.D4_WT_Mock <- fit$p.value[, "D2.vs.D4_WT_Mock"]
 gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]
 gene_results$P.Value_D2.vs.D7_WT_Mock <- fit$p.value[, "D2.vs.D7_WT_Mock"]
 gene_results$P.Value_D1.vs.D4_WT_Mock <- fit$p.value[, "D1.vs.D4_WT_Mock"]
 gene_results$P.Value_D1.vs.D7_WT_Mock <- fit$p.value[, "D1.vs.D7_WT_Mock"]
```

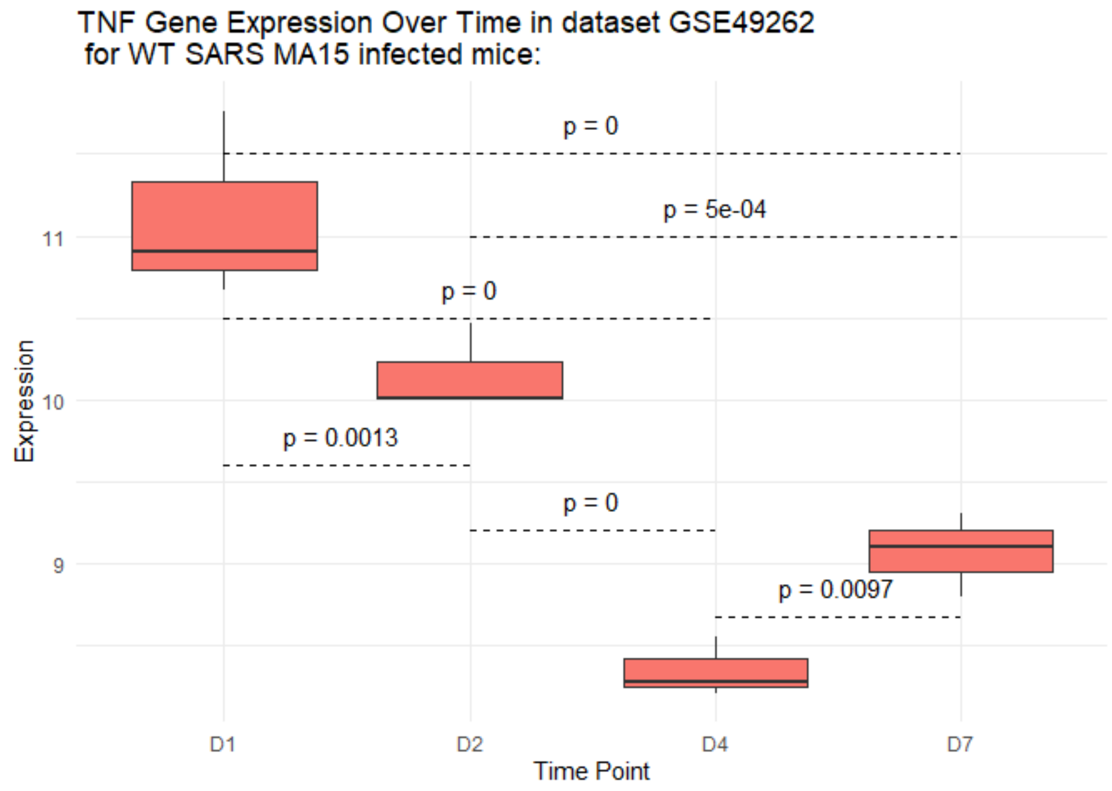
```
  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Combine the original p-values with the additional p-values from the new temporal
contrasts
  # Update this line to include all the relevant p-value columns from your analysis
  pvalues <- c(df2$P.Value, df2$P.Value_D1.vs.D2_WT_MA15,
df2$P.Value_D2.vs.D4_WT_MA15, df2$P.Value_D4.vs.D7_WT_MA15,
df2$P.Value_D2.vs.D7_WT_MA15, df2$P.Value_D1.vs.D4_WT_MA15,
df2$P.Value_D1.vs.D7_WT_MA15, df2$P.Value_D1.vs.D2_WT_Mock,
df2$P.Value_D2.vs.D4_WT_Mock, df2$P.Value_D4.vs.D7_WT_Mock,
df2$P.Value_D2.vs.D7_WT_Mock, df2$P.Value_D1.vs.D4_WT_Mock,
df2$P.Value_D1.vs.D7_WT_Mock)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  # Assuming the original data frame has a column 'adj.P.Val'
  # We replace it with the adjusted p-values calculated above
  df2$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))


df2$P.Value_D1.vs.D2_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D2_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D2_WT_MA15, format = "f", digits = 4))

df2$P.Value_D2.vs.D4_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))
```

```
df2$P.Value_D2.vs.D7_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D1.vs.D4_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D4_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D4_WT_MA15, format = "f", digits = 4))

df2$P.Value_D1.vs.D7_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D1.vs.D2_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D2_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D2_WT_Mock, format = "f", digits = 4))

df2$P.Value_D2.vs.D4_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

df2$P.Value_D2.vs.D7_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "f", digits = 4))

df2$P.Value_D1.vs.D4_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D4_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D4_WT_Mock, format = "f", digits = 4))

df2$P.Value_D1.vs.D7_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D7_WT_Mock, format = "f", digits = 4))

third_table2 <- df2
third_table2$dataset <- "GSE49262"
third_table2
```

D1.vs.D2_WT_MA15	D2.vs.D4_WT_MA15	D4.vs.D7_WT_MA15	D2.vs.D7_WT_MA15	D1.vs.D4_WT_MA15	D1.vs.D7_WT_MA15	D1.vs.D2_WT_Mock	D2.vs.D4_WT_Mock	D4.vs.D7_WT_Mock	D2.vs.D7_WT_Mock	D1.vs.D4_WT_Mock	D1.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D1.vs.D2_WT_MA15	P.Value_D2.vs.D4_WT_MA15	P.Value_D4.vs.D7_WT_MA15	P.Value_D2.vs.D7_WT_MA15	P.Value_D1.vs.D4_WT_MA15	P.Value_D1.vs.D7_WT_MA15	P.Value_D1.vs.D2_WT_Mock	P.Value_D2.vs.D4_WT_Mock	P.Value_D4.vs.D7_WT_Mock	P.Value_D2.vs.D7_WT_Mock	P.Value_D1.vs.D4_WT_Mock	P.Value_D1.vs.D7_WT_Mock	dataset
0.957	1.81	-0.721	1.09	2.77	2.04	0.27	-0.235	0.31	0.755	0.351	0.345	8.31	25.3	4.87e-07	6.33e-06	0.0013	0.0000	0.0097	0.0005	0.0000	0.0000	0.2851	0.3501	0.2723	0.7855	0.8874	0.2239	GSE49262
0.755	0.353	0.00381	0.357	1.11	1.11	-0.0108	0.115	-0.051	0.0639	0.104	0.0531	10.1	18.8	3.38e-06	4.39e-05	0.0000	0.0107	0.9753	0.0100	0.0000	0.0000	0.9302	0.3581	0.7120	0.6439	0.4039	0.7005	GSE49262
0.829	0.0317	0.00303	0.347	0.86	0.864	0.122	-0.209	-0.147	-0.357	-0.0878	-0.235	12.8	22.1	1.20e-06	1.56e-05	2.95e-07	7.44e-01	9.75e-01	7.20e-01	1.83e-07	1.75e-07	2.20e-01	4.36e-02	1.86e-01	4.32e-03	3.70e-01	4.29e-02	GSE49262
-0.199	-0.45	0.088	-0.362	-0.649	-0.561	-0.262	0.0752	-0.0188	0.0564	-0.186	-0.205	10.5	1.78	0.1699	1.0000	4.26e-01	8.44e-02	7.23e-01	1.58e-01	1.76e-02	3.59e-02	3.00e-01	7.62e-01	9.46e-01	8.39e-01	4.56e-01	4.63e-01	GSE49262

```r
third_table2$GENE_SYMBOL <- rownames(third_table2)
final2 <- final[final$GENE_SYMBOL == "Tnf",]

tnf_data_WT_MA15 <- final2[final2$Type == "WT_MA15",]
```

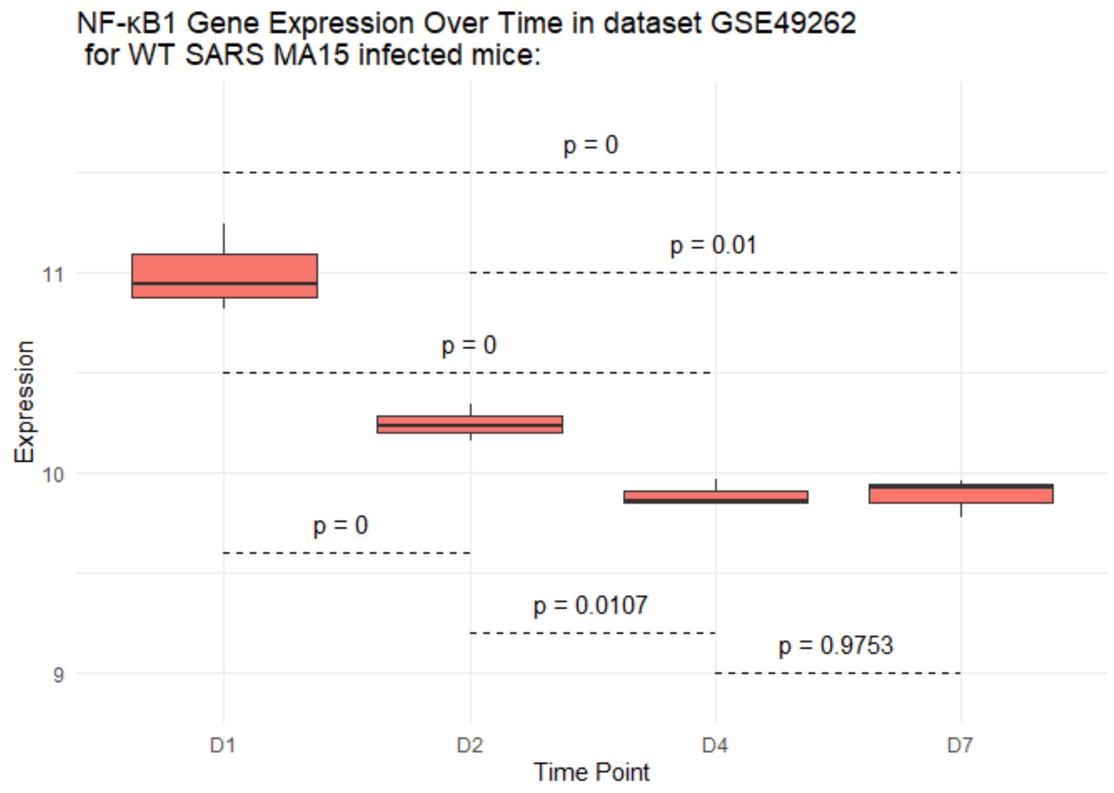
```r
res <- third_table2[third_table2$GENE_SYMBOL == "Tnf",]

format_pvalue <- function(pvalue) {
  if (is.na(pvalue) || !is.numeric(pvalue)) {
    return(NA)
  } else if (pvalue < 0.00001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 15))
  }
}

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D2_D1 <- res$P.Value_D1.vs.D2_WT_MA15
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D4_D1 <- res$P.Value_D1.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D1 <- res$P.Value_D1.vs.D7_WT_MA15

pvalues1_WT_MA15_D2_D1 <- as.numeric(pvalues1_WT_MA15_D2_D1)
pvalues1_WT_MA15_D4_D2 <- as.numeric(pvalues1_WT_MA15_D4_D2)
pvalues1_WT_MA15_D4_D1 <- as.numeric(pvalues1_WT_MA15_D4_D1)
pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_MA15_D7_D2 <- as.numeric(pvalues1_WT_MA15_D7_D2)
pvalues1_WT_MA15_D7_D1 <- as.numeric(pvalues1_WT_MA15_D7_D1)

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D2", "D4", "D7", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D4", "D2", "D1", "D7"),
  Label = paste("p =", c(pvalues1_WT_MA15_D2_D1, pvalues1_WT_MA15_D4_D2,
pvalues1_WT_MA15_D4_D1, pvalues1_WT_MA15_D7_D4, pvalues1_WT_MA15_D7_D2,
pvalues1_WT_MA15_D7_D1)),
  Y = c(8, 9, 10, 11, 12, 13)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "TNF Gene Expression Over Time in dataset GSE49262 \n for WT SARS MA15
infected mice:",
      x = "Time Point",
      y = "Expression") +
  theme_minimal() +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 9.6, xend = 2, yend = 9.6), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 9.2, xend = 3, yend = 9.2), linetype = "dashed", color =
"black") +
    geom_segment(aes(x = 1, y = 10.5, xend = 3, yend = 10.5), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 3, y = 8.67, xend = 4, yend = 8.67), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 2, y = 11, xend = 4, yend = 11), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 1, y = 11.5, xend = 4, yend = 11.5), linetype = "dashed", color =
"black")


# Add p-values
p <- p + annotate("text", x = c(1.48, 2.5, 2, 3.5, 3, 2.5), y = c(9.6, 9.2, 10.5, 8.67, 11, 11.5),
label = annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
final2 <- final[final$GENE_SYMBOL == "Nfkb1",]

tnf_data_WT_MA15 <- final2[final2$Type == "WT_MA15",]

res <- third_table2[third_table2$GENE_SYMBOL == "Nfkb1",]

format_pvalue <- function(pvalue) {
  if (is.na(pvalue) || !is.numeric(pvalue)) {
    return(NA)
  } else if (pvalue < 0.00001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 15))
  }
```

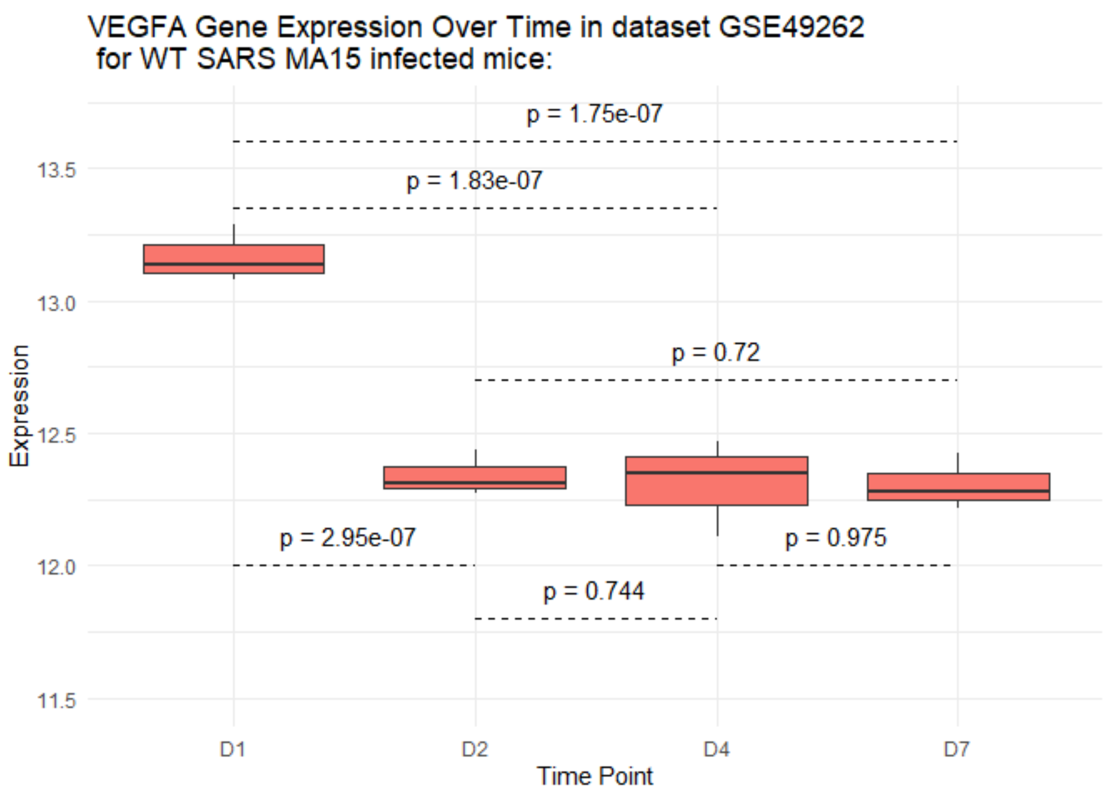
```r
}

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D2_D1 <- res$P.Value_D1.vs.D2_WT_MA15
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D4_D1 <- res$P.Value_D1.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D1 <- res$P.Value_D1.vs.D7_WT_MA15

pvalues1_WT_MA15_D2_D1 <- as.numeric(pvalues1_WT_MA15_D2_D1)
pvalues1_WT_MA15_D4_D2 <- as.numeric(pvalues1_WT_MA15_D4_D2)
pvalues1_WT_MA15_D4_D1 <- as.numeric(pvalues1_WT_MA15_D4_D1)
pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_MA15_D7_D2 <- as.numeric(pvalues1_WT_MA15_D7_D2)
pvalues1_WT_MA15_D7_D1 <- as.numeric(pvalues1_WT_MA15_D7_D1)

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D2", "D4", "D7", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D4", "D2", "D1", "D7"),
  Label = paste("p =", c(pvalues1_WT_MA15_D2_D1, pvalues1_WT_MA15_D4_D2,
pvalues1_WT_MA15_D4_D1, pvalues1_WT_MA15_D7_D4, pvalues1_WT_MA15_D7_D2,
pvalues1_WT_MA15_D7_D1)),
  Y = c(8, 9, 10, 11, 12, 13)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "NF-κB1 Gene Expression Over Time in dataset GSE49262 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(8.9, 11.8) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 9.6, xend = 2, yend = 9.6), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 9.2, xend = 3, yend = 9.2), linetype = "dashed", color =
"black") +
    geom_segment(aes(x = 1, y = 10.5, xend = 3, yend = 10.5), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 3, y = 9, xend = 4, yend = 9), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 2, y = 11, xend = 4, yend = 11), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 1, y = 11.5, xend = 4, yend = 11.5), linetype = "dashed", color =
"black")
```

```
# Add p-values
p <- p + annotate("text", x = c(1.48, 2.5, 2, 3.5, 3, 2.5), y = c(9.6, 9.2, 10.5, 9, 11, 11.5),
label = annotations_WT_MA15$Label, vjust = -1)

print(p)
```



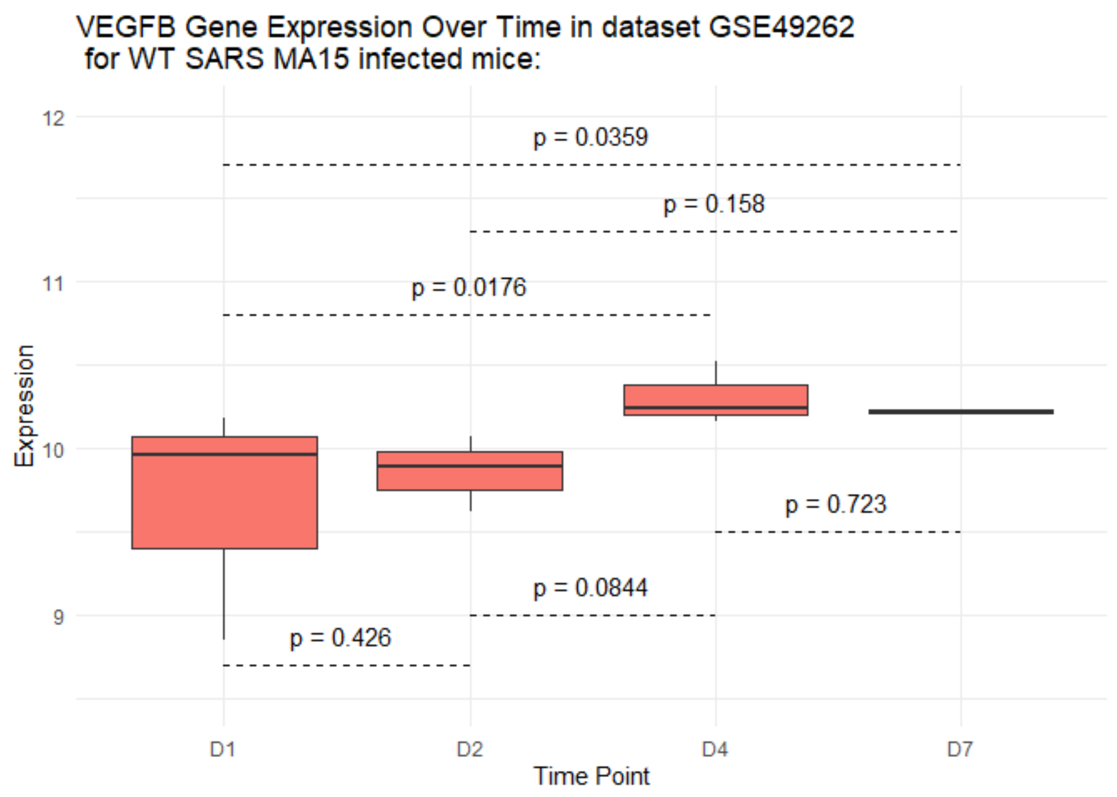
```
final2 <- final[final$GENE_SYMBOL == "Vegfa",]

tnf_data_WT_MA15 <- final2[final2$Type == "WT_MA15",]

res <- third_table2[third_table2$GENE_SYMBOL == "Vegfa",]

format_pvalue <- function(pvalue) {
  if (is.na(pvalue) || !is.numeric(pvalue)) {
    return(NA)
  } else if (pvalue < 0.00001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 15))
  }
}

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D2_D1 <- res$P.Value_D1.vs.D2_WT_MA15
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D4_D1 <- res$P.Value_D1.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D1 <- res$P.Value_D1.vs.D7_WT_MA15
```

```r
pvalues1_WT_MA15_D2_D1 <- as.numeric(pvalues1_WT_MA15_D2_D1)
pvalues1_WT_MA15_D4_D2 <- as.numeric(pvalues1_WT_MA15_D4_D2)
pvalues1_WT_MA15_D4_D1 <- as.numeric(pvalues1_WT_MA15_D4_D1)
pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_MA15_D7_D2 <- as.numeric(pvalues1_WT_MA15_D7_D2)
pvalues1_WT_MA15_D7_D1 <- as.numeric(pvalues1_WT_MA15_D7_D1)

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D2", "D4", "D7", "D7", "D7", "D7"), 
  Comparison = c("D1", "D2", "D4", "D2", "D1", "D7"),
  Label = paste("p =", c(pvalues1_WT_MA15_D2_D1, pvalues1_WT_MA15_D4_D2,
pvalues1_WT_MA15_D4_D1, pvalues1_WT_MA15_D7_D4, pvalues1_WT_MA15_D7_D2,
pvalues1_WT_MA15_D7_D1)),
  Y = c(8, 9, 10, 11, 12, 13)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFA Gene Expression Over Time in dataset GSE49262 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(11.5, 13.7) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 12, xend = 2, yend = 12), linetype = "dashed", color
= "black") +
  geom_segment(aes(x = 2, y = 11.8, xend = 3, yend = 11.8), linetype = "dashed", color =
"black") +
    geom_segment(aes(x = 1, y = 13.35, xend = 3, yend = 13.35), linetype = "dashed", color
= "black")  +
    geom_segment(aes(x = 3, y = 12, xend = 4, yend = 12), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 2, y = 12.7, xend = 4, yend = 12.7), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 1, y = 13.6, xend = 4, yend = 13.6), linetype = "dashed", color =
"black")


# Add p-values
p <- p + annotate("text", x = c(1.48, 2.5, 2, 3.5, 3, 2.5), y = c(12, 11.8, 13.35, 12, 12.7, 13.6),
label = annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
final2 <- final[final$GENE_SYMBOL == "Vegfb",]

tnf_data_WT_MA15 <- final2[final2$Type == "WT_MA15",]

res <- third_table2[third_table2$GENE_SYMBOL == "Vegfb",]

format_pvalue <- function(pvalue) {
  if (is.na(pvalue) || !is.numeric(pvalue)) {
    return(NA)
  } else if (pvalue < 0.00001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 15))
  }
}

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D2_D1 <- res$P.Value_D1.vs.D2_WT_MA15
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D4_D1 <- res$P.Value_D1.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D1 <- res$P.Value_D1.vs.D7_WT_MA15

pvalues1_WT_MA15_D2_D1 <- as.numeric(pvalues1_WT_MA15_D2_D1)
pvalues1_WT_MA15_D4_D2 <- as.numeric(pvalues1_WT_MA15_D4_D2)
pvalues1_WT_MA15_D4_D1 <- as.numeric(pvalues1_WT_MA15_D4_D1)
pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_MA15_D7_D2 <- as.numeric(pvalues1_WT_MA15_D7_D2)
pvalues1_WT_MA15_D7_D1 <- as.numeric(pvalues1_WT_MA15_D7_D1)
```

```
# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = "WT_MA15",
  Measurement = c("D2", "D4", "D7", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D4", "D2", "D1", "D7"),
  Label = paste("p =", c(pvalues1_WT_MA15_D2_D1, pvalues1_WT_MA15_D4_D2,
pvalues1_WT_MA15_D4_D1, pvalues1_WT_MA15_D7_D4, pvalues1_WT_MA15_D7_D2,
pvalues1_WT_MA15_D7_D1)),
  Y = c(8, 9, 10, 11, 12, 13)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFB Gene Expression Over Time in dataset GSE49262 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(8.5, 12) +
  theme(legend.position = "none")

# Add lines
p <- p + geom_segment(aes(x = 1, y = 8.7, xend = 2, yend = 8.7), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 9, xend = 3, yend = 9), linetype = "dashed", color = "black")
+
    geom_segment(aes(x = 1, y = 10.8, xend = 3, yend = 10.8), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 3, y = 9.5, xend = 4, yend = 9.5), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 2, y = 11.3, xend = 4, yend = 11.3), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 1, y = 11.7, xend = 4, yend = 11.7), linetype = "dashed", color =
"black")


# Add p-values
p <- p + annotate("text", x = c(1.48, 2.5, 2, 3.5, 3, 2.5), y = c(8.7, 9, 10.8, 9.5, 11.3, 11.7),
label = annotations_WT_MA15$Label, vjust = -1)

print(p)
```



#==

GSE49263:

```
expr_data <- read.csv("Mus_SARS_GSE49263.csv", header = TRUE, stringsAsFactors = T,
na.strings = c("","NA"))

attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X", "WT_MA15.D1")
setnames(expr_data, "X.1", "WT_MA15.D1")
setnames(expr_data, "X.2", "WT_MA15.D1")
setnames(expr_data, "X.3", "WT_MA15.D1")
setnames(expr_data, "X.4", "WT_MA15.D2")
setnames(expr_data, "X.5", "WT_MA15.D2")
setnames(expr_data, "X.6", "WT_MA15.D2")
setnames(expr_data, "X.7", "WT_MA15.D2")
setnames(expr_data, "X.8", "WT_MA15.D4")
setnames(expr_data, "X.9", "WT_MA15.D4")
setnames(expr_data, "X.10", "WT_MA15.D4")
setnames(expr_data, "X.11", "WT_MA15.D4")
setnames(expr_data, "X.12", "WT_MA15.D7")
setnames(expr_data, "X.13", "WT_MA15.D7")
setnames(expr_data, "X.14", "WT_MA15.D7")

setnames(expr_data, "X.15", "WT_Mock.D1")
```

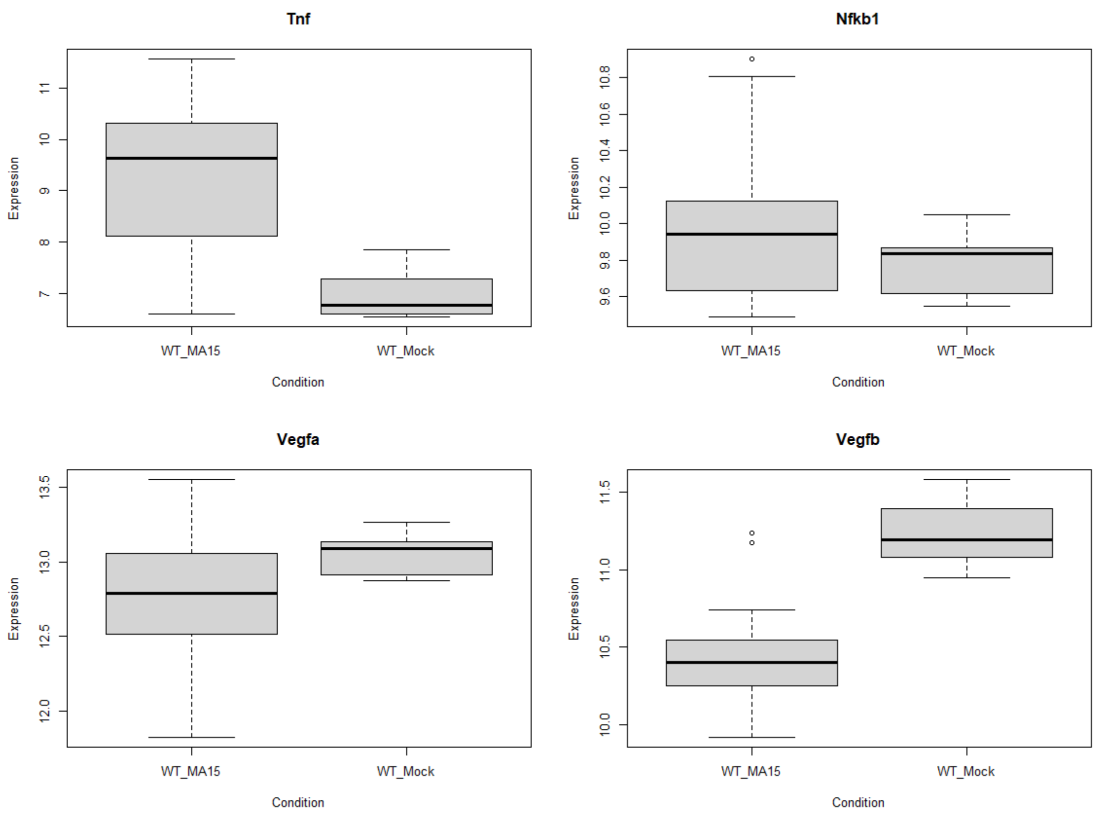
```r
setnames(expr_data, "X.16", "WT_Mock.D1")
setnames(expr_data, "X.17", "WT_Mock.D2")
setnames(expr_data, "X.18", "WT_Mock.D2")
setnames(expr_data, "X.19", "WT_Mock.D2")
setnames(expr_data, "X.20", "WT_Mock.D4")
setnames(expr_data, "X.21", "WT_Mock.D4")
setnames(expr_data, "X.22", "WT_Mock.D4")
setnames(expr_data, "X.23", "WT_Mock.D7")
setnames(expr_data, "X.24", "WT_MA15.D7")
setnames(expr_data, "X.25", "WT_MA15.D7")

expr_data1 <- expr_data[11:41184, c(1, 2, 4:29)]


write.csv(expr_data1, file = "GSE49263_dataset.csv")

data <- read.csv("GSE49263_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)
```

```
combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")

probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
```

```r
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
library(limma)

final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    WT_MA15.vs.WT_Mock = TypeWT_MA15 - TypeWT_Mock,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = "WT_MA15.vs.WT_Mock", number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)

  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)
  # Store the updated data frame in the list
  results[[i]] <- df
}
df1 <- do.call("rbind", results)

df1$P.Value <- ifelse(df1$P.Value < 0.0001, formatC(df1$P.Value, format = "e", digits = 2),
formatC(df1$P.Value, format = "f", digits = 4))

df1$adj.P.Val <- ifelse(df1$adj.P.Val < 0.0001, formatC(df1$adj.P.Val, format = "e", digits =
2), formatC(df1$adj.P.Val, format = "f", digits = 4))

third_table3 <- df1
third_table3$dataset <- "GSE49263"
third_table3
```

logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
2.36	8.54	4.75	1.5	7.93e-05	7.93e-05	GSE49263
0.226	9.94	1.38	-4.59	0.1814	0.1814	GSE49263
-0.28	12.9	-1.68	-4.34	0.1052	0.1052	GSE49263
-0.798	10.7	-6.04	4.55	3.07e-06	3.07e-06	GSE49263

```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab =
"Expression")
}
```



Change in Time:

```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have
this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D1.vs.D2_WT_MA15 = InteractionWT_MA15.D1 - InteractionWT_MA15.D2,
  D2.vs.D4_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D4,
  D4.vs.D7_WT_MA15 = InteractionWT_MA15.D4 - InteractionWT_MA15.D7,
  D2.vs.D7_WT_MA15 = InteractionWT_MA15.D2 - InteractionWT_MA15.D7,
  D1.vs.D4_WT_MA15 = InteractionWT_MA15.D1 - InteractionWT_MA15.D4,
  D1.vs.D7_WT_MA15 = InteractionWT_MA15.D1 - InteractionWT_MA15.D7,

  D1.vs.D2_WT_Mock = InteractionWT_Mock.D1 - InteractionWT_Mock.D2,
  D2.vs.D4_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D4,
  D4.vs.D7_WT_Mock = InteractionWT_Mock.D4 - InteractionWT_Mock.D7,
  D2.vs.D7_WT_Mock = InteractionWT_Mock.D2 - InteractionWT_Mock.D7,
  D1.vs.D4_WT_Mock = InteractionWT_Mock.D1 - InteractionWT_Mock.D4,
  D1.vs.D7_WT_Mock = InteractionWT_Mock.D1 - InteractionWT_Mock.D7,


  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D1.vs.D2_WT_MA15", "D2.vs.D4_WT_MA15",
"D4.vs.D7_WT_MA15", "D2.vs.D7_WT_MA15", "D1.vs.D4_WT_MA15",
"D1.vs.D7_WT_MA15", "D1.vs.D2_WT_Mock", "D2.vs.D4_WT_Mock",
"D4.vs.D7_WT_Mock", "D2.vs.D7_WT_Mock", "D1.vs.D4_WT_Mock",
"D1.vs.D7_WT_Mock"), number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  # The column names should match the ones used in topTable
  gene_results$P.Value_D1.vs.D2_WT_MA15 <- fit$p.value[, "D1.vs.D2_WT_MA15"]
  gene_results$P.Value_D2.vs.D4_WT_MA15 <- fit$p.value[, "D2.vs.D4_WT_MA15"]
  gene_results$P.Value_D4.vs.D7_WT_MA15 <- fit$p.value[, "D4.vs.D7_WT_MA15"]
  gene_results$P.Value_D2.vs.D7_WT_MA15 <- fit$p.value[, "D2.vs.D7_WT_MA15"]
  gene_results$P.Value_D1.vs.D4_WT_MA15 <- fit$p.value[, "D1.vs.D4_WT_MA15"]
  gene_results$P.Value_D1.vs.D7_WT_MA15 <- fit$p.value[, "D1.vs.D7_WT_MA15"]

  gene_results$P.Value_D1.vs.D2_WT_Mock <- fit$p.value[, "D1.vs.D2_WT_Mock"]
  gene_results$P.Value_D2.vs.D4_WT_Mock <- fit$p.value[, "D2.vs.D4_WT_Mock"]
  gene_results$P.Value_D4.vs.D7_WT_Mock <- fit$p.value[, "D4.vs.D7_WT_Mock"]
  gene_results$P.Value_D2.vs.D7_WT_Mock <- fit$p.value[, "D2.vs.D7_WT_Mock"]
  gene_results$P.Value_D1.vs.D4_WT_Mock <- fit$p.value[, "D1.vs.D4_WT_Mock"]
  gene_results$P.Value_D1.vs.D7_WT_Mock <- fit$p.value[, "D1.vs.D7_WT_Mock"]

  # Store the results in the list
  results[[gene]] <- gene_results
}
```

```
for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df2 <- results[[i]]

  # Combine the original p-values with the additional p-values from the new temporal
contrasts
  # Update this line to include all the relevant p-value columns from your analysis
  pvalues <- c(df2$P.Value, df2$P.Value_D1.vs.D2_WT_MA15,
df2$P.Value_D2.vs.D4_WT_MA15, df2$P.Value_D4.vs.D7_WT_MA15,
df2$P.Value_D2.vs.D7_WT_MA15, df2$P.Value_D1.vs.D4_WT_MA15,
df2$P.Value_D1.vs.D7_WT_MA15, df2$P.Value_D1.vs.D2_WT_Mock,
df2$P.Value_D2.vs.D4_WT_Mock, df2$P.Value_D4.vs.D7_WT_Mock,
df2$P.Value_D2.vs.D7_WT_Mock, df2$P.Value_D1.vs.D4_WT_Mock,
df2$P.Value_D1.vs.D7_WT_Mock)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  # Assuming the original data frame has a column 'adj.P.Val'
  # We replace it with the adjusted p-values calculated above
  df2$adj.P.Val <- p_adjusted[1:nrow(df2)]

  # Store the updated data frame in the list
  results[[i]] <- df2
}
df2 <- do.call("rbind", results)

df2$P.Value <- ifelse(df2$P.Value < 0.0001, formatC(df2$P.Value, format = "e", digits = 2),
formatC(df2$P.Value, format = "f", digits = 4))

df2$adj.P.Val <- ifelse(df2$adj.P.Val < 0.0001, formatC(df2$adj.P.Val, format = "e", digits =
2), formatC(df2$adj.P.Val, format = "f", digits = 4))


df2$P.Value_D1.vs.D2_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D2_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D2_WT_MA15, format = "f", digits = 4))

df2$P.Value_D2.vs.D4_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_MA15, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D2.vs.D7_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_MA15, format = "f", digits = 4))
```

```r
df2$P.Value_D1.vs.D4_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D4_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D4_WT_MA15, format = "f", digits = 4))

df2$P.Value_D1.vs.D7_WT_MA15 <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D7_WT_MA15, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D7_WT_MA15, format = "f", digits = 4))

df2$P.Value_D1.vs.D2_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D2_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D2_WT_Mock, format = "f", digits = 4))

df2$P.Value_D2.vs.D4_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D4_WT_Mock, format = "f", digits = 4))

df2$P.Value_D4.vs.D7_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D4.vs.D7_WT_Mock, format = "f", digits = 4))

df2$P.Value_D2.vs.D7_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D2.vs.D7_WT_Mock, format = "f", digits = 4))

df2$P.Value_D1.vs.D4_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D4_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D4_WT_Mock, format = "f", digits = 4))

df2$P.Value_D1.vs.D7_WT_Mock <- ifelse(df2$adj.P.Val < 0.0001,
formatC(df2$P.Value_D1.vs.D7_WT_Mock, format = "e", digits = 2),
formatC(df2$P.Value_D1.vs.D7_WT_Mock, format = "f", digits = 4))

third_table4 <- df2
third_table4$dataset <- "GSE49263"
third_table4
```

D1.vs.D2_WT_MA15	D2.vs.D4_WT_MA15	D4.vs.D7_WT_MA15	D2.vs.D7_WT_MA15	D1.vs.D4_WT_MA15	D1.vs.D7_WT_MA15	D1.vs.D2_WT_Mock	D2.vs.D4_WT_Mock	D4.vs.D7_WT_Mock	D2.vs.D7_WT_Mock	D1.vs.D4_WT_Mock	D1.vs.D7_WT_Mock	AveExpr	adj.P.Val	P.Value_D1.vs.D2_WT_MA15	P.Value_D2.vs.D4_WT_MA15	P.Value_D4.vs.D7_WT_MA15	P.Value_D2.vs.D7_WT_MA15	P.Value_D1.vs.D4_WT_MA15	P.Value_D1.vs.D7_WT_MA15	P.Value_D1.vs.D2_WT_Mock	P.Value_D2.vs.D4_WT_Mock	P.Value_D4.vs.D7_WT_Mock	P.Value_D2.vs.D7_WT_Mock	P.Value_D1.vs.D4_WT_Mock	P.Value_D1.vs.D7_WT_Mock	dataset

0.742	2.08	-0.301	1.78	2.82	2.52	0.658	-0.663	0.694	0.0315	-0.00448	0.69	8.54	6.97	0.0006	0.0077	1.87e-01	1.16e-03	5.64e-01	2.71e-03	5.70e-05	1.10e-04	3.58e-01	3.02e-01	4.41e-01	9.72e-01	9.95e-01	4.70e-01	GSE49263
0.764	0.441	-0.253	0.188	1.21	0.952	-0.13	0.0268	0.0465	0.0733	-0.103	-0.0569	9.94	19.5	5.60e-07	7.28e-06	0.0000	0.0016	0.0372	0.1131	0.0000	0.0000	0.4070	0.8475	0.8131	0.7099	0.5085	0.7852	GSE49263
0.498	0.337	0.114	0.451	0.835	0.948	-0.0987	-0.11	0.268	0.159	-0.208	0.0599	12.9	4.74	0.0046	0.0601	2.59e-02	1.17e-01	5.66e-01	3.24e-02	7.14e-04	1.21e-04	7.14e-01	6.49e-01	4.33e-01	6.41e-01	4.41e-01	8.68e-01	GSE49263
0.204	-0.297	-0.309	-0.607	-0.0938	-0.403	-0.0163	0.386	-0.0643	0.321	0.369	0.305	10.7	2.69	0.0484	0.6297	2.98e-01	1.35e-01	1.03e-01	3.44e-03	6.27e-01	3.82e-02	9.48e-01	9.58e-02	8.38e-01	3.14e-01	1.49e-01	3.66e-01	GSE49263

```r
third_table1 <- rownames_to_column(third_table1, "rowname")
third_table3 <- rownames_to_column(third_table3, "rowname")
```

```r
df2 <- merge(third_table1, third_table3, all = TRUE)
```

```r
df2 <- df2 %>% arrange(dataset)
```

```r
df2
```

rowname	logFC	AveExpr	t	B	P.Value	adj.P.Val	dataset
Nfkb1	0.336	10.1	2.19	-3.73	0.0396	0.0396	GSE49262
Tnf	2.85	8.31	8.13	8.36	6.31e-08	6.31e-08	GSE49262
Vegfa	-0.513	12.8	-3.98	-0.412	0.0007	0.0007	GSE49262
Vegfb	-1.01	10.5	-7.31	6.72	3.38e-07	3.38e-07	GSE49262
Nfkb1	0.226	9.94	1.38	-4.59	0.1814	0.1814	GSE49263
Tnf	2.36	8.54	4.75	1.5	7.93e-05	7.93e-05	GSE49263

Vegfa	-0.28	12.9	-1.68	-4.34	0.1052	0.1052	GSE49263
Vegfb	-0.798	10.7	-6.04	4.55	3.07e-06	3.07e-06	GSE49263

```
third_table2 <- rownames_to_column(third_table2, "rowname")
third_table4 <- rownames_to_column(third_table4, "rowname")


df3 <- merge(third_table2, third_table4, all = TRUE)

df3 <- df3 %>% arrange(dataset)

df3
```

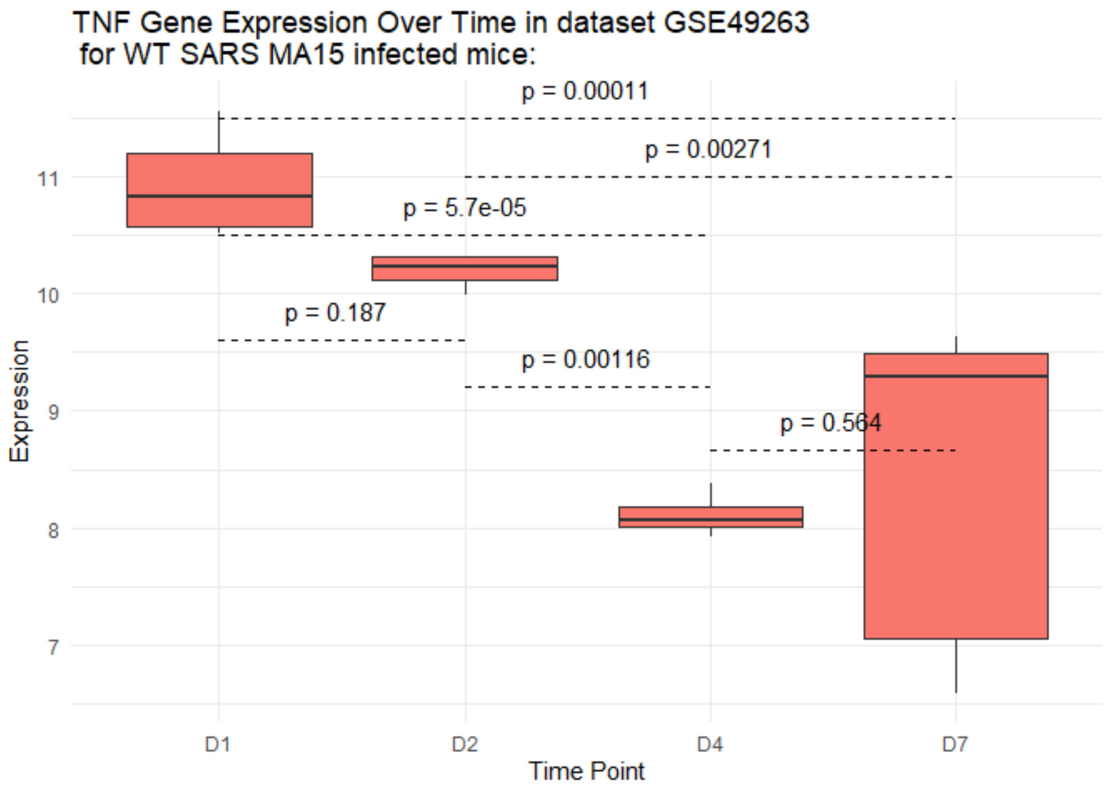
rowname	D1.vs.D2_WT_MA15	D2.vs.D4_WT_MA15	D4.vs.D7_WT_MA15	D2.vs.D7_WT_MA15	D1.vs.D4_WT_MA15	D1.vs.D7_WT_MA15	D1.vs.D2_WT_Mock	D2.vs.D4_WT_Mock	D4.vs.D7_WT_Mock	D2.vs.D7_WT_Mock	D1.vs.D4_WT_Mock	D1.vs.D7_WT_Mock	AveExpr	F	P.Value	adj.P.Val	P.Value_D1.vs.D2_WT_MA15	P.Value_D2.vs.D4_WT_MA15	P.Value_D4.vs.D7_WT_MA15	P.Value_D2.vs.D7_WT_MA15	P.Value_D1.vs.D4_WT_MA15	P.Value_D1.vs.D7_WT_MA15	P.Value_D1.vs.D2_WT_Mock	P.Value_D2.vs.D4_WT_Mock	P.Value_D4.vs.D7_WT_Mock	P.Value_D2.vs.D7_WT_Mock	P.Value_D1.vs.D4_WT_Mock	P.Value_D1.vs.D7_WT_Mock	dataset	GENE_SYMBOL
Nfkb1	0.755	0.353	0.00381	0.357	1.11	1.11	-0.0108	0.115	-0.051	0.0639	0.104	0.0531	10.1	18.8	3.38e-06	4.39e-05	0.0000	0.0107	0.9753	0.0100	0.0000	0.0000	0.9302	0.3581	0.7120	0.6439	0.4039	0.7005	GSE49262	Nfkb1
Tnf	0.957	1.81	-0.721	1.09	2.77	2.04	0.27	-0.235	0.31	0.0755	0.0351	0.345	8.31	25.3	4.87e-07	6.33e-06	0.0013	0.0000	0.0097	0.0005	0.0000	0.0000	0.2851	0.3501	0.2723	0.7855	0.8874	0.2239	GSE49262	Tnf

Vegfa	0.829	0.0317	0.00303	0.0347	0.86	0.864	0.122	-0.209	-0.147	-0.357	-0.0878	-0.235	12.8	22.1	1.20e-06	1.56e-05	2.95e-07	7.44e-01	9.75e-01	7.20e-01	1.83e-07	1.75e-07	2.20e-01	4.36e-02	1.86e-01	4.32e-03	3.70e-01	4.29e-02	GSE49262	Vegfa
Vegfb	-0.199	-0.45	0.088	-0.362	-0.649	-0.561	-0.262	0.0752	-0.0188	0.0564	-0.186	-0.205	10.5	1.78	0.1699	1.0000	4.26e-01	8.44e-02	7.23e-01	1.58e-01	1.76e-02	3.59e-02	3.00e-01	7.62e-01	9.46e-01	8.39e-01	4.56e-01	4.63e-01	GSE49262	Vegfb
Nfkb1	0.764	0.441	-0.253	0.188	1.21	0.952	-0.13	0.0268	0.0465	0.0733	-0.103	-0.0569	9.94	19.5	5.60e-07	7.28e-06	0.0000	0.0016	0.0372	0.1131	0.0000	0.0000	0.4070	0.8475	0.8131	0.7099	0.5085	0.7852	GSE49263	
Tnf	0.742	2.08	-0.301	1.78	2.82	2.52	0.658	-0.663	0.694	0.0315	-0.00448	0.69	8.54	6.97	0.0006	0.0077	1.87e-01	1.16e-03	5.64e-01	2.71e-03	5.70e-05	1.10e-04	3.58e-01	3.02e-01	4.41e-01	9.72e-01	9.95e-01	4.70e-01	GSE49263	
Vegfa	0.498	0.337	0.114	0.451	0.835	0.948	-0.0987	-0.11	0.268	0.159	-0.208	0.0599	12.9	4.74	0.0046	0.0601	2.59e-02	1.17e-01	5.66e-01	3.24e-02	7.14e-04	1.21e-04	7.14e-01	6.49e-01	4.33e-01	6.41e-01	4.41e-01	8.68e-01	GSE49263	
Vegfb	0.204	-0.297	-0.309	-0.607	-0.0938	-0.403	-0.0163	0.386	-0.0643	0.321	0.369	0.305	10.7	2.69	0.0484	0.6297	2.98e-01	1.35e-01	1.03e-01	3.44e-03	6.27e-01	3.82e-02	9.48e-01	9.58e-02	8.38e-01	3.14e-01	1.49e-01	3.66e-01	GSE49263	

```
final2 <- final[final$GENE_SYMBOL == "Tnf",]

tnf_data_WT_MA15 <- final2[final2$Type == "WT_MA15",]

res <- third_table4[third_table4$rowname == "Tnf",]
```

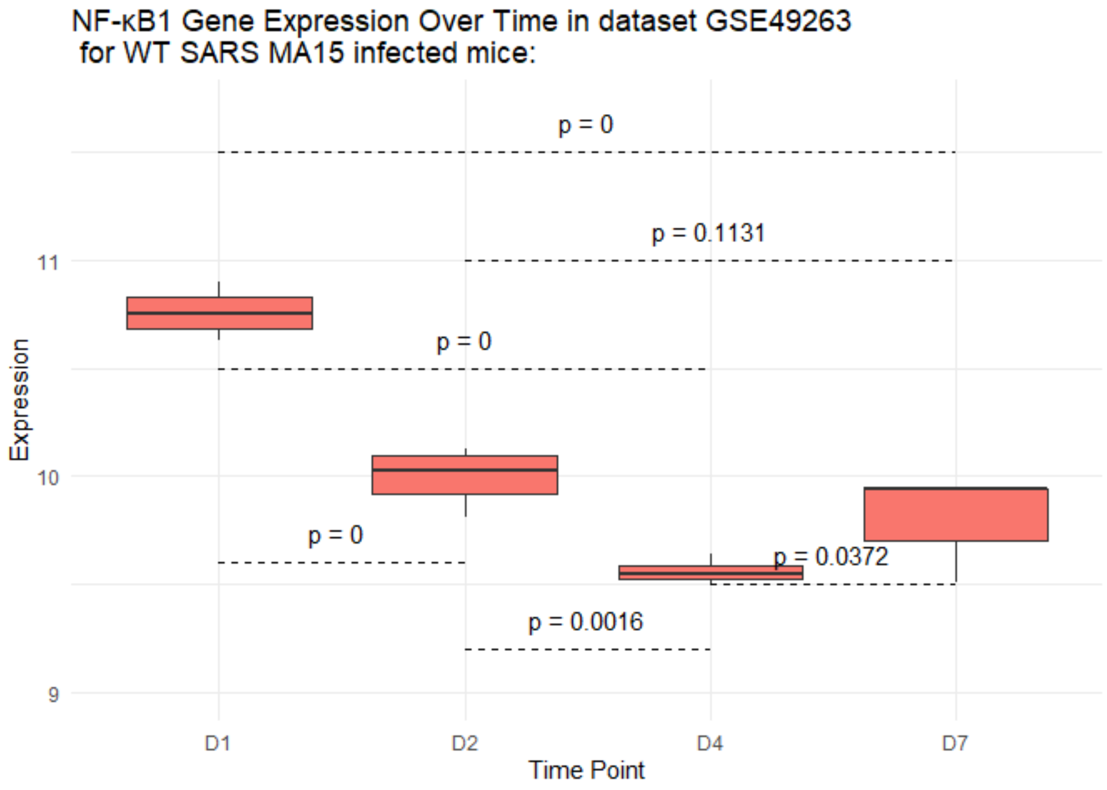
```
format_pvalue <- function(pvalue) {
  if (is.na(pvalue) || !is.numeric(pvalue)) {
    return(NA)
  } else if (pvalue < 0.00001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 15))
  }
}

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D2_D1 <- res$P.Value_D1.vs.D2_WT_MA15
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D4_D1 <- res$P.Value_D1.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D1 <- res$P.Value_D1.vs.D7_WT_MA15

pvalues1_WT_MA15_D2_D1 <- as.numeric(pvalues1_WT_MA15_D2_D1)
pvalues1_WT_MA15_D4_D2 <- as.numeric(pvalues1_WT_MA15_D4_D2)
pvalues1_WT_MA15_D4_D1 <- as.numeric(pvalues1_WT_MA15_D4_D1)
pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_MA15_D7_D2 <- as.numeric(pvalues1_WT_MA15_D7_D2)
pvalues1_WT_MA15_D7_D1 <- as.numeric(pvalues1_WT_MA15_D7_D1)

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = rep("WT_MA15", 6),
  Measurement = c("D2", "D4", "D7", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D4", "D2", "D1", "D1"),
  Label = paste("p =", c(pvalues1_WT_MA15_D2_D1, pvalues1_WT_MA15_D4_D2,
pvalues1_WT_MA15_D4_D1, pvalues1_WT_MA15_D7_D4, pvalues1_WT_MA15_D7_D2,
pvalues1_WT_MA15_D7_D1)),
  Y = c(8, 9, 10, 11, 12, 13)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "TNF Gene Expression Over Time in dataset GSE49263 \n for WT SARS MA15
infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  theme(legend.position = "none")


# Add lines
p <- p + geom_segment(aes(x = 1, y = 9.6, xend = 2, yend = 9.6), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 9.2, xend = 3, yend = 9.2), linetype = "dashed", color =
```

```r
"black") +
    geom_segment(aes(x = 1, y = 10.5, xend = 3, yend = 10.5), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 3, y = 8.67, xend = 4, yend = 8.67), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 2, y = 11, xend = 4, yend = 11), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 1, y = 11.5, xend = 4, yend = 11.5), linetype = "dashed", color =
"black")


# Add p-values
p <- p + annotate("text", x = c(1.48, 2.5, 2, 3.5, 3, 2.5), y = c(9.6, 9.2, 10.5, 8.67, 11, 11.5),
label = annotations_WT_MA15$Label, vjust = -1)

print(p)
```



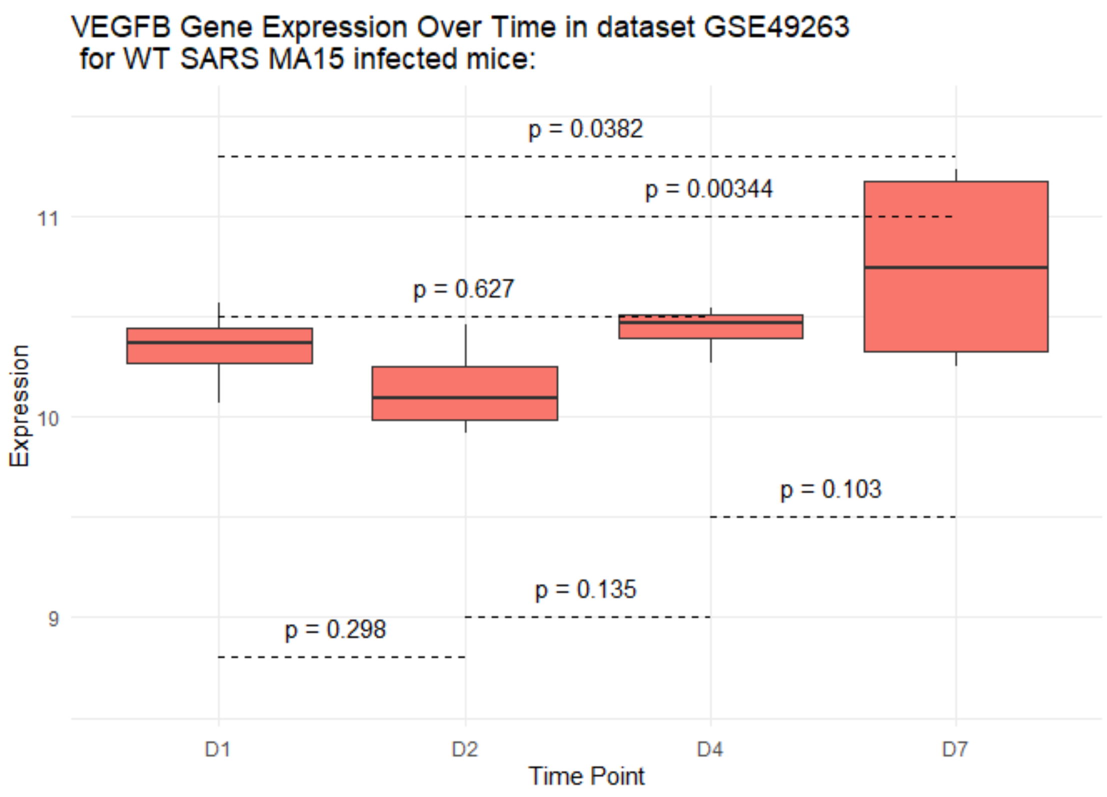
```r
final2 <- final[final$GENE_SYMBOL == "Nfkb1",]

tnf_data_WT_MA15 <- final2[final2$Type == "WT_MA15",]

res <- third_table4[third_table4$rowname == "Nfkb1",]

format_pvalue <- function(pvalue) {
  if (is.na(pvalue) || !is.numeric(pvalue)) {
    return(NA)
  } else if (pvalue < 0.1) {
    return(format(pvalue, scientific = TRUE, digits = 15))
  } else {
    return(round(pvalue, 15))
  }
```

```
}

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D2_D1 <- res$P.Value_D1.vs.D2_WT_MA15
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D4_D1 <- res$P.Value_D1.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D1 <- res$P.Value_D1.vs.D7_WT_MA15

pvalues1_WT_MA15_D2_D1 <- as.numeric(pvalues1_WT_MA15_D2_D1)
pvalues1_WT_MA15_D4_D2 <- as.numeric(pvalues1_WT_MA15_D4_D2)
pvalues1_WT_MA15_D4_D1 <- as.numeric(pvalues1_WT_MA15_D4_D1)
pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_MA15_D7_D2 <- as.numeric(pvalues1_WT_MA15_D7_D2)
pvalues1_WT_MA15_D7_D1 <- as.numeric(pvalues1_WT_MA15_D7_D1)

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
 Type = rep("WT_MA15", 6),
 Measurement = c("D2", "D4", "D7", "D7", "D7", "D7"),
 Comparison = c("D1", "D2", "D4", "D2", "D1", "D1"),
 Label = paste("p =", c(pvalues1_WT_MA15_D2_D1, pvalues1_WT_MA15_D4_D2,
pvalues1_WT_MA15_D4_D1, pvalues1_WT_MA15_D7_D4, pvalues1_WT_MA15_D7_D2,
pvalues1_WT_MA15_D7_D1)),
 Y = c(8, 9, 10, 11, 12, 13)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
 geom_boxplot() +
 labs(title = "NF-κB1 Gene Expression Over Time in dataset GSE49263 \n for WT SARS
MA15 infected mice:",
     x = "Time Point",
     y = "Expression") +
 theme_minimal() +
 ylim(9, 11.7) +
 theme(legend.position = "none")


# Add lines
p <- p + geom_segment(aes(x = 1, y = 9.6, xend = 2, yend = 9.6), linetype = "dashed",
color = "black") +
 geom_segment(aes(x = 2, y = 9.2, xend = 3, yend = 9.2), linetype = "dashed", color =
"black") +
   geom_segment(aes(x = 1, y = 10.5, xend = 3, yend = 10.5), linetype = "dashed", color =
"black")  +
   geom_segment(aes(x = 3, y = 9.5, xend = 4, yend = 9.5), linetype = "dashed", color =
"black")  +
   geom_segment(aes(x = 2, y = 11, xend = 4, yend = 11), linetype = "dashed", color =
"black")  +
   geom_segment(aes(x = 1, y = 11.5, xend = 4, yend = 11.5), linetype = "dashed", color =
"black")

# Add p-values
p <- p + annotate("text", x = c(1.48, 2.5, 2, 3.5, 3, 2.5), y = c(9.6, 9.2, 10.5, 9.5, 11, 11.5),
label = annotations_WT_MA15$Label, vjust = -1)

print(p)
```



```
final2 <- final[final$GENE_SYMBOL == "Vegfb",]

tnf_data_WT_MA15 <- final2[final2$Type == "WT_MA15",]

res <- third_table4[third_table4$rowname == "Vegfb",]

format_pvalue <- function(pvalue) {
  if (is.na(pvalue) || !is.numeric(pvalue)) {
    return(NA)
  } else if (pvalue < 0.1) {
    return(format(pvalue, scientific = TRUE, digits = 15))
  } else {
    return(round(pvalue, 15))
  }
}

# Extract the p-values for WT_MA15 only
pvalues1_WT_MA15_D2_D1 <- res$P.Value_D1.vs.D2_WT_MA15
pvalues1_WT_MA15_D4_D2 <- res$P.Value_D2.vs.D4_WT_MA15
pvalues1_WT_MA15_D4_D1 <- res$P.Value_D1.vs.D4_WT_MA15
pvalues1_WT_MA15_D7_D4 <- res$P.Value_D4.vs.D7_WT_MA15
pvalues1_WT_MA15_D7_D2 <- res$P.Value_D2.vs.D7_WT_MA15
```

```r
pvalues1_WT_MA15_D7_D1 <- res$P.Value_D1.vs.D7_WT_MA15

pvalues1_WT_MA15_D2_D1 <- as.numeric(pvalues1_WT_MA15_D2_D1)
pvalues1_WT_MA15_D4_D2 <- as.numeric(pvalues1_WT_MA15_D4_D2)
pvalues1_WT_MA15_D4_D1 <- as.numeric(pvalues1_WT_MA15_D4_D1)
pvalues1_WT_MA15_D7_D4 <- as.numeric(pvalues1_WT_MA15_D7_D4)
pvalues1_WT_MA15_D7_D2 <- as.numeric(pvalues1_WT_MA15_D7_D2)
pvalues1_WT_MA15_D7_D1 <- as.numeric(pvalues1_WT_MA15_D7_D1)

# Create a new data frame for the annotations for WT_MA15 only
annotations_WT_MA15 <- data.frame(
  Type = rep("WT_MA15", 6),
  Measurement = c("D2", "D4", "D7", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D4", "D2", "D1", "D1"),
  Label = paste("p =", c(pvalues1_WT_MA15_D2_D1, pvalues1_WT_MA15_D4_D2,
pvalues1_WT_MA15_D4_D1, pvalues1_WT_MA15_D7_D4, pvalues1_WT_MA15_D7_D2,
pvalues1_WT_MA15_D7_D1)),
  Y = c(8, 9, 10, 11, 12, 13)
)

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "VEGFB Gene Expression Over Time in dataset GSE49263 \n for WT SARS
MA15 infected mice:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(8.6, 11.5) +
  theme(legend.position = "none")


# Add lines
p <- p + geom_segment(aes(x = 1, y = 8.8, xend = 2, yend = 8.8), linetype = "dashed",
color = "black") +
  geom_segment(aes(x = 2, y = 9, xend = 3, yend = 9), linetype = "dashed", color = "black")
+
    geom_segment(aes(x = 1, y = 10.5, xend = 3, yend = 10.5), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 3, y = 9.5, xend = 4, yend = 9.5), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 2, y = 11, xend = 4, yend = 11), linetype = "dashed", color =
"black")  +
    geom_segment(aes(x = 1, y = 11.3, xend = 4, yend = 11.3), linetype = "dashed", color =
"black")


# Add p-values
p <- p + annotate("text", x = c(1.48, 2.5, 2, 3.5, 3, 2.5), y = c(8.8, 9, 10.5, 9.5, 11, 11.3),
label = annotations_WT_MA15$Label, vjust = -1)

print(p)
```



#==

Table 8.7, Figure 5

GSE50000

```
expr_data <- read.csv("Mus_SARS_GSE50000.csv", header = TRUE, stringsAsFactors = T,
na.strings = c("","NA"))

attach(expr_data)
setnames(expr_data, "qlucore", "GENE_SYMBOL")
setnames(expr_data, "gedata", "noname")

setnames(expr_data, "X", "MA_10_4.D1")
setnames(expr_data, "X.1", "MA_10_4.D1")
setnames(expr_data, "X.2", "MA_10_4.D1")
setnames(expr_data, "X.3", "MA_10_4.D1")

setnames(expr_data, "X.4", "MA_10_4.D2")
setnames(expr_data, "X.5", "MA_10_4.D2")
setnames(expr_data, "X.6", "MA_10_4.D2")
setnames(expr_data, "X.7", "MA_10_4.D2")

setnames(expr_data, "X.8", "MA_10_4.D4")
setnames(expr_data, "X.9", "MA_10_4.D4")
setnames(expr_data, "X.10", "MA_10_4.D4")
setnames(expr_data, "X.11", "MA_10_4.D4")
```

```r
setnames(expr_data, "X.12", "MA_10_4.D7")
setnames(expr_data, "X.13", "MA_10_4.D7")
setnames(expr_data, "X.14", "MA_10_4.D7")
setnames(expr_data, "X.15", "MA_10_4.D7")

setnames(expr_data, "X.16", "MA_10_5.D1")
setnames(expr_data, "X.17", "MA_10_5.D1")
setnames(expr_data, "X.18", "MA_10_5.D1")
setnames(expr_data, "X.19", "MA_10_5.D1")

setnames(expr_data, "X.20", "MA_10_5.D2")
setnames(expr_data, "X.21", "MA_10_5.D2")
setnames(expr_data, "X.22", "MA_10_5.D2")
setnames(expr_data, "X.23", "MA_10_5.D2")

setnames(expr_data, "X.24", "MA_10_5.D4")
setnames(expr_data, "X.25", "MA_10_5.D4")
setnames(expr_data, "X.26", "MA_10_5.D4")
setnames(expr_data, "X.27", "MA_10_5.D4")
setnames(expr_data, "X.28", "MA_10_5.D4")

setnames(expr_data, "X.29", "MA_10_5.D7")
setnames(expr_data, "X.30", "MA_10_5.D7")
setnames(expr_data, "X.31", "MA_10_5.D7")

setnames(expr_data, "X.70", "Mock.D1")
setnames(expr_data, "X.71", "Mock.D1")
setnames(expr_data, "X.72", "Mock.D1")
setnames(expr_data, "X.73", "Mock.D1")

setnames(expr_data, "X.74", "Mock.D2")
setnames(expr_data, "X.75", "Mock.D2")
setnames(expr_data, "X.76", "Mock.D2")
setnames(expr_data, "X.77", "Mock.D2")

setnames(expr_data, "X.78", "Mock.D4")
setnames(expr_data, "X.79", "Mock.D4")
setnames(expr_data, "X.80", "Mock.D4")
setnames(expr_data, "X.81", "Mock.D4")

setnames(expr_data, "X.82", "Mock.D7")
setnames(expr_data, "X.83", "Mock.D7")
setnames(expr_data, "X.84", "Mock.D7")
setnames(expr_data, "X.85", "Mock.D7")

expr_data1 <- expr_data[31:41204, c(1, 2, 4:35, 74:89)]

write.csv(expr_data1, file = "GSE50000_dataset.csv")


# Read the dataset
data <- read.csv("GSE50000_dataset.csv", stringsAsFactors = FALSE)
```

```
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)

tnf <- long_data_filtered[long_data_filtered == "Tnf", c(1, 3:5)]
nfkb_data <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))


probe1 <- as.data.frame(subset(nfkb_data, noname == "A_51_P283759"))
probe2 <- as.data.frame(subset(nfkb_data, noname == "A_52_P32733"))
probe3 <- as.data.frame(subset(nfkb_data, noname == "A_52_P569539"))
probe4 <- as.data.frame(subset(nfkb_data, noname == "A_52_P582969"))

nfkb_data$Expression <- as.numeric(as.character(nfkb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

nfkb_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfa_data <- subset(long_data_filtered, GENE_SYMBOL == "Vegfa")
```

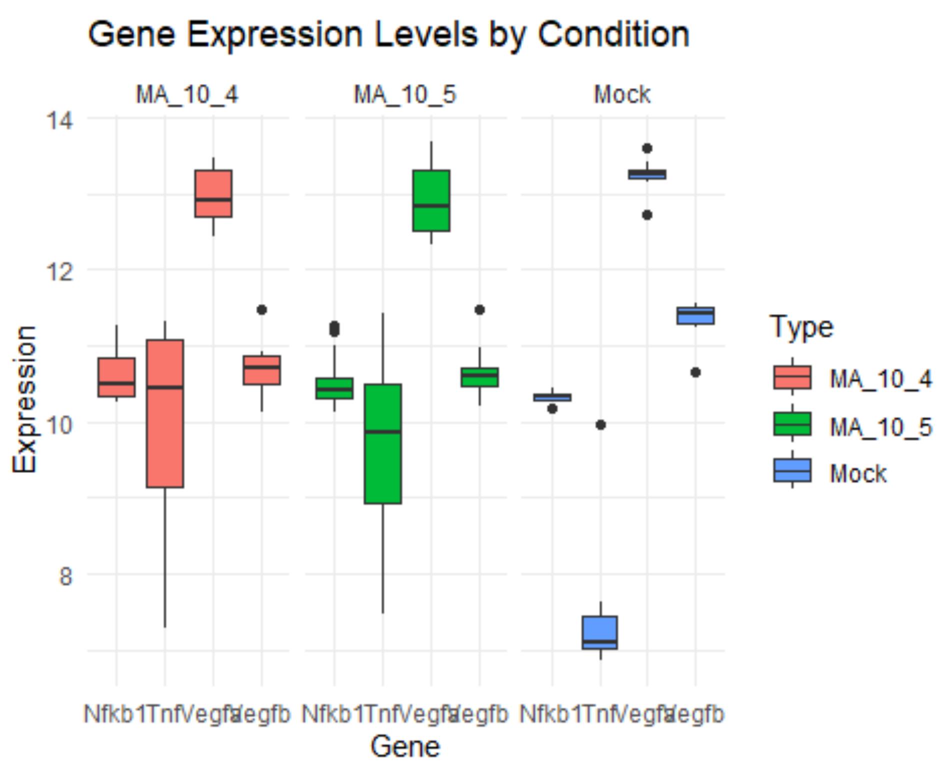
```r
probe1 <- as.data.frame(subset(vegfa_data, noname == "A_51_P482552"))
probe2 <- as.data.frame(subset(vegfa_data, noname == "A_52_P229471"))
probe3 <- as.data.frame(subset(vegfa_data, noname == "A_52_P249424"))
probe4 <- as.data.frame(subset(vegfa_data, noname == "A_52_P638895"))

vegfa_data$Expression <- as.numeric(as.character(vegfa_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))

combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4")])

vegfa_data <- combined_dataset[,c(1, 3, 4, 9)]
vegfb_data <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(vegfb_data, noname == "A_51_P458168"))
probe2 <- as.data.frame(subset(vegfb_data, noname == "A_52_P436628"))

vegfb_data$Expression <- as.numeric(as.character(vegfb_data$Expression))

combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")
combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
```

```
combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

vegfb_data <- combined_dataset[,c(1, 3, 4, 7)]
final <- rbind(tnf, nfkb_data, vegfa_data, vegfb_data)

genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Create a design matrix
  design <- model.matrix(~0 + Type, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

  # Contrast matrix
  contrast_matrix <- makeContrasts(
    MA15_10_4.vs.Mock = TypeMA_10_4 - TypeMock,
    MA15_10_5.vs.Mock = TypeMA_10_5 - TypeMock,
    MA15_10_5.vs.MA15_10_4 = TypeMA_10_5 - TypeMA_10_4,
    levels = colnames(design)
  )

  # eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results
  gene_results <- topTable(fit, coef = c("MA15_10_4.vs.Mock", "MA15_10_5.vs.Mock",
"MA15_10_5.vs.MA15_10_4"), number = Inf)
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  # Store the p-values for each comparison in the results
  gene_results$P.Value_MA15_10_4.vs.Mock <- fit$p.value[, "MA15_10_4.vs.Mock"]
```

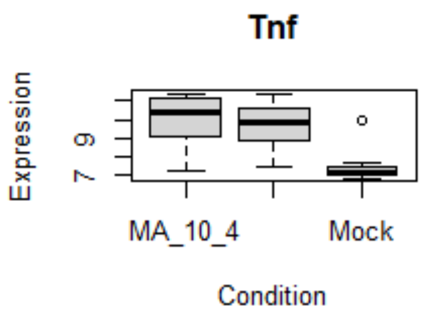
```r
  gene_results$P.Value_MA15_10_5.vs.Mock <- fit$p.value[, "MA15_10_5.vs.Mock"]
  gene_results$P.Value_MA15_10_5.vs.MA15_10_4 <- fit$p.value[,
"MA15_10_5.vs.MA15_10_4"]


    gene_results$logFC_MA15_10_4.vs.Mock <-
log2(mean(final_gene$Expression[final_gene$Type == "MA_10_4"]) /
mean(final_gene$Expression[final_gene$Type == "Mock"]))
  gene_results$logFC_MA15_10_5.vs.Mock <-
log2(mean(final_gene$Expression[final_gene$Type == "MA_10_5"]) /
mean(final_gene$Expression[final_gene$Type == "Mock"]))
  gene_results$logFC_MA15_10_5.vs.MA15_10_4 <-
log2(mean(final_gene$Expression[final_gene$Type == "MA_10_5"]) /
mean(final_gene$Expression[final_gene$Type == "MA_10_4"]))

  # Store the results in the list
  results[[gene]] <- gene_results
}



for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values
  pvalues <- c(df$P.Value, df$P.Value_MA15_10_4.vs.Mock,
df$P.Value_MA15_10_5.vs.Mock, df$P.Value_MA15_10_5.vs.MA15_10_4)

  # Adjust the p-values
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Store the p-values in a separate data frame
  pvalues <- df[, c("P.Value", "adj.P.Val", "P.Value_MA15_10_4.vs.Mock",
"P.Value_MA15_10_5.vs.Mock", "P.Value_MA15_10_5.vs.MA15_10_4")]

  # Remove the p-values from the original data frame
  df <- df[, setdiff(names(df), names(pvalues))]

  # Round the numbers to 3 decimal points
  df <- round(df, 4)
  # Add the p-values back to the data frame
  df <- cbind(df, pvalues)

  # Store the updated data frame in the list
  results[[i]] <- df
}



for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Rearrange the columns
  df <- df[, c("MA15_10_4.vs.Mock", "logFC_MA15_10_4.vs.Mock",
"P.Value_MA15_10_4.vs.Mock",
              "MA15_10_5.vs.Mock", "logFC_MA15_10_5.vs.Mock",
"P.Value_MA15_10_5.vs.Mock",
              "MA15_10_5.vs.MA15_10_4", "logFC_MA15_10_5.vs.MA15_10_4",
"P.Value_MA15_10_5.vs.MA15_10_4", "AveExpr", "F",
              "P.Value", "adj.P.Val")]

  # Store the updated data frame in the list
  results[[i]] <- df
}
df <- do.call("rbind", results)

df$P.Value <- ifelse(df$P.Value < 0.0001, formatC(df$P.Value, format = "e", digits = 2),
formatC(df$P.Value, format = "f", digits = 4))

df$adj.P.Val <- ifelse(df$adj.P.Val < 0.0001, formatC(df$adj.P.Val, format = "e", digits = 2),
formatC(df$adj.P.Val, format = "f", digits = 4))

df$P.Value_MA15_10_4.vs.Mock <- ifelse(df$P.Value_MA15_10_4.vs.Mock < 0.0001,
formatC(df$P.Value_MA15_10_4.vs.Mock, format = "e", digits = 2),
formatC(df$P.Value_MA15_10_4.vs.Mock, format = "f", digits = 4))

df$P.Value_MA15_10_5.vs.MA15_10_4 <- ifelse(df$P.Value_MA15_10_5.vs.MA15_10_4 <
0.0001, formatC(df$P.Value_MA15_10_5.vs.MA15_10_4, format = "e", digits = 2),
formatC(df$P.Value_MA15_10_5.vs.MA15_10_4, format = "f", digits = 4))


df$P.Value_MA15_10_5.vs.Mock <- ifelse(df$P.Value_MA15_10_5.vs.Mock < 0.0001,
formatC(df$P.Value_MA15_10_5.vs.Mock, format = "e", digits = 2),
formatC(df$P.Value_MA15_10_5.vs.Mock, format = "f", digits = 4))


df
```

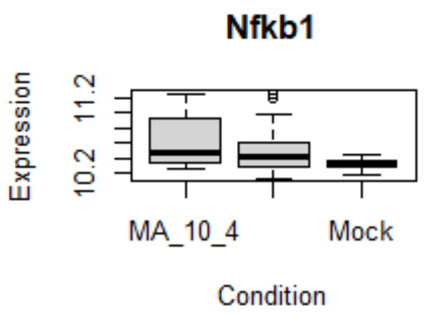
MA15_10_4.vs.Mock	logFC_MA15_10_4.vs.Mock	P.Value_MA15_10_4.vs.Mock	MA15_10_5.vs.Mock	logFC_MA15_10_5.vs.Mock	P.Value_MA15_10_5.vs.Mock	MA15_10_5.vs.MA15_10_4	logFC_MA15_10_5.vs.MA15_10_4	P.Value_MA15_10_5.vs.MA15_10_4	AveExpr	F	P.Value	adj.P.Val
2.71	0.452	2.45e-09	2.38	0.404	5.26e-08	-0.328	-0.0478	0.3736	9.05	32.8	1.64e-09	6.57e-09
0.303	0.0418	0.0052	0.192	0.0266	0.0696	-0.112	-0.0152	0.2854	10.5	4.42	0.0176	0.0705
-0.298	-0.0328	0.0186	-0.362	-0.04	0.0048	-0.0643	-0.0072	0.6007	13	5.02	0.0107	0.0430
-0.701	-0.0917	1.57e-08	-0.745	-0.0978	3.51e-09	-0.0448	-0.0061	0.6621	10.9	33.6	1.16e-09	4.66e-09

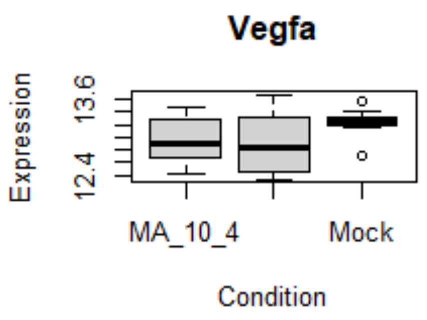
```r
# Calculate the average expression for each gene and condition
gene_data <- final %>%
  dplyr::group_by(GENE_SYMBOL, Type) %>%
  dplyr::summarize(Average_Expression = mean(Expression, na.rm = TRUE)) %>%
  ungroup()

# Rename columns for use in ggplot
names(gene_data) <- c("Gene", "Condition", "Expression")

ggplot(final, aes(x = GENE_SYMBOL, y = Expression, fill = Type)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Gene Expression Levels by Condition", x = "Gene", y = "Expression") +
  facet_wrap(~Type)
```

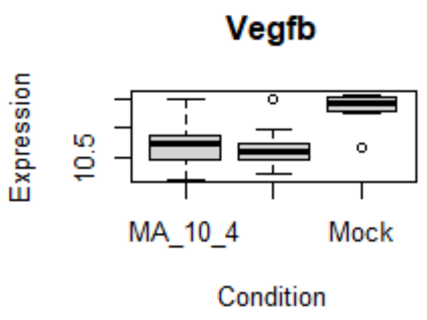


```
genes <- unique(final$GENE_SYMBOL)

# Number of rows and columns for the plot layout
nrow = ceiling(sqrt(length(genes)))
ncol = ceiling(length(genes) / nrow)

# Set up the plot layout
par(mfrow = c(nrow, ncol))

# Create a boxplot for each gene
for (gene in genes) {
  gene_data <- subset(final, GENE_SYMBOL == gene)
  boxplot(Expression ~ Type, data = gene_data, main = gene, xlab = "Condition", ylab = 
"Expression")
}
```









Change in Time:

```
genes <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")

# Create an empty list to store the results
results <- list()

# Loop over the genes
for (gene in genes) {
  # Subset the data for the current gene
  final_gene <- final[final$GENE_SYMBOL == gene, ]

  # Convert necessary columns to appropriate types
  final_gene$Type <- as.factor(final_gene$Type)
  final_gene$Measurement <- as.factor(final_gene$Measurement)  # Assuming you have this column
  final_gene$Expression <- as.numeric(as.character(final_gene$Expression))

  # Define the interaction between type and time point
  final_gene$Interaction <- interaction(final_gene$Type, final_gene$Measurement)

  # Create a design matrix
  design <- model.matrix(~0 + Interaction, data = final_gene)

  # Fit a linear model
  fit <- lmFit(final_gene$Expression, design)

contrast_matrix <- makeContrasts(
  # For the MA_10_4 group
  D1.vs.D2_MA_10_4 = InteractionMA_10_4.D1 - InteractionMA_10_4.D2,
  D2.vs.D4_MA_10_4 = InteractionMA_10_4.D2 - InteractionMA_10_4.D4,
  D4.vs.D7_MA_10_4 = InteractionMA_10_4.D4 - InteractionMA_10_4.D7,
  D1.vs.D4_MA_10_4 = InteractionMA_10_4.D1 - InteractionMA_10_4.D4,
  D2.vs.D7_MA_10_4 = InteractionMA_10_4.D2 - InteractionMA_10_4.D7,
  D1.vs.D7_MA_10_4 = InteractionMA_10_4.D1 - InteractionMA_10_4.D7,

  D1.vs.D2_MA_10_5 = InteractionMA_10_5.D1 - InteractionMA_10_5.D2,
  D2.vs.D4_MA_10_5 = InteractionMA_10_5.D2 - InteractionMA_10_5.D4,
  D4.vs.D7_MA_10_5 = InteractionMA_10_5.D4 - InteractionMA_10_5.D7,
  D1.vs.D4_MA_10_5 = InteractionMA_10_5.D1 - InteractionMA_10_5.D4,
  D2.vs.D7_MA_10_5 = InteractionMA_10_5.D2 - InteractionMA_10_5.D7,
  D1.vs.D7_MA_10_5 = InteractionMA_10_5.D1 - InteractionMA_10_5.D7,

  D1.vs.D2_Mock = InteractionMock.D1 - InteractionMock.D2,
  D2.vs.D4_Mock = InteractionMock.D2 - InteractionMock.D4,
  D4.vs.D7_Mock = InteractionMock.D4 - InteractionMock.D7,
  D1.vs.D4_Mock = InteractionMock.D1 - InteractionMock.D4,
  D2.vs.D7_Mock = InteractionMock.D2 - InteractionMock.D7,
  D1.vs.D7_Mock = InteractionMock.D1 - InteractionMock.D7,

  levels = colnames(design)
)

  # Apply eBayes
  fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

  # Extract results for the temporal contrasts
gene_results <- topTable(fit, coef = c("D1.vs.D2_MA_10_4", "D2.vs.D4_MA_10_4",
"D4.vs.D7_MA_10_4", "D1.vs.D4_MA_10_4", "D2.vs.D7_MA_10_4", "D1.vs.D7_MA_10_4",

                                       "D1.vs.D2_MA_10_5", "D2.vs.D4_MA_10_5",
"D4.vs.D7_MA_10_5", "D1.vs.D4_MA_10_5", "D2.vs.D7_MA_10_5", "D1.vs.D7_MA_10_5",

                                       "D1.vs.D2_Mock", "D2.vs.D4_Mock", "D4.vs.D7_Mock",
"D1.vs.D4_Mock", "D2.vs.D7_Mock", "D1.vs.D7_Mock"), number = Inf)


  # Adjust P-values using Benjamini-Hochberg method
  gene_results$adj.P.Val <- p.adjust(gene_results$P.Value, method = "BH")

  gene_results$P.Value_D1.vs.D2_MA_10_4 <- fit$p.value[, "D1.vs.D2_MA_10_4"]
  gene_results$P.Value_D2.vs.D4_MA_10_4 <- fit$p.value[, "D2.vs.D4_MA_10_4"]
  gene_results$P.Value_D4.vs.D7_MA_10_4 <- fit$p.value[, "D4.vs.D7_MA_10_4"]
  gene_results$P.Value_D1.vs.D4_MA_10_4 <- fit$p.value[, "D1.vs.D4_MA_10_4"]
  gene_results$P.Value_D2.vs.D7_MA_10_4 <- fit$p.value[, "D2.vs.D7_MA_10_4"]
  gene_results$P.Value_D1.vs.D7_MA_10_4 <- fit$p.value[, "D1.vs.D7_MA_10_4"]

  gene_results$P.Value_D1.vs.D2_MA_10_5 <- fit$p.value[, "D1.vs.D2_MA_10_5"]
  gene_results$P.Value_D2.vs.D4_MA_10_5 <- fit$p.value[, "D2.vs.D4_MA_10_5"]
  gene_results$P.Value_D4.vs.D7_MA_10_5 <- fit$p.value[, "D4.vs.D7_MA_10_5"]
  gene_results$P.Value_D1.vs.D4_MA_10_5 <- fit$p.value[, "D1.vs.D4_MA_10_5"]
  gene_results$P.Value_D2.vs.D7_MA_10_5 <- fit$p.value[, "D2.vs.D7_MA_10_5"]
  gene_results$P.Value_D1.vs.D7_MA_10_5 <- fit$p.value[, "D1.vs.D7_MA_10_5"]
```

```
 gene_results$P.Value_D1.vs.D2_Mock <- fit$p.value[, "D1.vs.D2_Mock"]
 gene_results$P.Value_D2.vs.D4_Mock <- fit$p.value[, "D2.vs.D4_Mock"]
 gene_results$P.Value_D4.vs.D7_Mock <- fit$p.value[, "D4.vs.D7_Mock"]
 gene_results$P.Value_D1.vs.D4_Mock <- fit$p.value[, "D1.vs.D4_Mock"]
 gene_results$P.Value_D2.vs.D7_Mock <- fit$p.value[, "D2.vs.D7_Mock"]
 gene_results$P.Value_D1.vs.D7_Mock <- fit$p.value[, "D1.vs.D7_Mock"]


 gene_results$logFC_D1.vs.D2_MA_10_4 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D2"]))

  gene_results$logFC_D2.vs.D4_MA_10_4 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D2"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D4"]))

  gene_results$logFC_D4.vs.D7_MA_10_4 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D4"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D7"]))

  gene_results$logFC_D1.vs.D4_MA_10_4 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D4"]))

  gene_results$logFC_D2.vs.D7_MA_10_4 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D2"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D7"]))

  gene_results$logFC_D1.vs.D7_MA_10_4 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_4.D7"]))


  gene_results$logFC_D1.vs.D2_MA_10_5 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D2"]))

  gene_results$logFC_D2.vs.D4_MA_10_5 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D2"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D4"]))

  gene_results$logFC_D4.vs.D7_MA_10_5 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D4"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D7"]))

  gene_results$logFC_D1.vs.D4_MA_10_5 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D4"]))

  gene_results$logFC_D2.vs.D7_MA_10_5 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D2"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D7"]))
```

```
    gene_results$logFC_D1.vs.D7_MA_10_5 <-
log2(mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "MA_10_5.D7"]))


   gene_results$logFC_D1.vs.D2_Mock <-
log2(mean(final_gene$Expression[final_gene$Interaction == "Mock.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "Mock.D2"]))

    gene_results$logFC_D2.vs.D4_Mock <-
log2(mean(final_gene$Expression[final_gene$Interaction == "Mock.D2"])/
mean(final_gene$Expression[final_gene$Interaction == "Mock.D4"]))

    gene_results$logFC_D4.vs.D7_Mock <-
log2(mean(final_gene$Expression[final_gene$Interaction == "Mock.D4"])/
mean(final_gene$Expression[final_gene$Interaction == "Mock.D7"]))

    gene_results$logFC_D1.vs.D4_Mock <-
log2(mean(final_gene$Expression[final_gene$Interaction == "Mock.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "Mock.D4"]))

    gene_results$logFC_D2.vs.D7_Mock <-
log2(mean(final_gene$Expression[final_gene$Interaction == "Mock.D2"])/
mean(final_gene$Expression[final_gene$Interaction == "Mock.D7"]))

    gene_results$logFC_D1.vs.D7_Mock <-
log2(mean(final_gene$Expression[final_gene$Interaction == "Mock.D1"])/
mean(final_gene$Expression[final_gene$Interaction == "Mock.D7"]))


  # Store the results in the list
  results[[gene]] <- gene_results
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Combine the original p-values with the additional p-values from the new temporal
contrasts
  # Update this line to include all the relevant p-value columns from your analysis
  pvalues <- c(df$P.Value, df$P.Value_D1.vs.D2_MA_10_4, df$P.Value_D2.vs.D4_MA_10_4,
               df$P.Value_D4.vs.D7_MA_10_4, df$P.Value_D1.vs.D4_MA_10_4,
df$P.Value_D2.vs.D7_MA_10_4, df$P.Value_D1.vs.D7_MA_10_4,

               df$P.Value_D1.vs.D2_MA_10_5, df$P.Value_D2.vs.D4_MA_10_5,
               df$P.Value_D4.vs.D7_MA_10_5, df$P.Value_D1.vs.D4_MA_10_5,
df$P.Value_D2.vs.D7_MA_10_5, df$P.Value_D1.vs.D7_MA_10_5,

               df$P.Value_D1.vs.D2_Mock, df$P.Value_D2.vs.D4_Mock,
               df$P.Value_D4.vs.D7_Mock, df$P.Value_D1.vs.D4_Mock,
df$P.Value_D2.vs.D7_Mock, df$P.Value_D1.vs.D7_Mock)

  # Adjust the p-values using the Bonferroni method
  p_adjusted <- p.adjust(pvalues, method = "bonferroni")

  # Add the adjusted p-values to the data frame
  # Assuming the original data frame has a column 'adj.P.Val'
  # We replace it with the adjusted p-values calculated above
  df$adj.P.Val <- p_adjusted[1:nrow(df)]

  # Store the updated data frame in the list
  results[[i]] <- df
}


for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]

  # Define the column names for p-values
  pvalue_cols <- c("P.Value", "adj.P.Val", "P.Value_D1.vs.D2_MA_10_4",
"P.Value_D2.vs.D4_MA_10_4", "P.Value_D4.vs.D7_MA_10_4",
"P.Value_D1.vs.D4_MA_10_4", "P.Value_D2.vs.D7_MA_10_4",
"P.Value_D1.vs.D7_MA_10_4",
                   "P.Value_D1.vs.D2_MA_10_5", "P.Value_D2.vs.D4_MA_10_5",
"P.Value_D4.vs.D7_MA_10_5", "P.Value_D1.vs.D4_MA_10_5",
"P.Value_D2.vs.D7_MA_10_5", "P.Value_D1.vs.D7_MA_10_5",
                   "P.Value_D1.vs.D2_Mock", "P.Value_D2.vs.D4_Mock",
"P.Value_D4.vs.D7_Mock", "P.Value_D1.vs.D4_Mock", "P.Value_D2.vs.D7_Mock",
"P.Value_D1.vs.D7_Mock")

  # Check if all p-value columns exist in the dataframe
  if (all(pvalue_cols %in% names(df))) {
    # Store the p-values in a separate data frame
    pvalues <- df[, pvalue_cols]

    # Remove the p-values from the original data frame
    df <- df[, setdiff(names(df), names(pvalues))]

    # Round the numbers to 3 decimal points
    df <- round(df, 4)

    # Add the p-values back to the data frame
    df <- cbind(df, pvalues)

    # Store the updated data frame in the list
    results[[i]] <- df
  }
}

for (i in seq_along(results)) {
  # Extract the data frame for the current gene
  df <- results[[i]]
```

```r

  # Rearrange the columns
  df <- df[, c("D1.vs.D2_MA_10_4", "logFC_D1.vs.D2_MA_10_4",
"P.Value_D1.vs.D2_MA_10_4",
             "D2.vs.D4_MA_10_4", "logFC_D2.vs.D4_MA_10_4",
"P.Value_D2.vs.D4_MA_10_4",
             "D4.vs.D7_MA_10_4", "logFC_D4.vs.D7_MA_10_4",
"P.Value_D4.vs.D7_MA_10_4",
             "D1.vs.D4_MA_10_4", "logFC_D1.vs.D4_MA_10_4",
"P.Value_D1.vs.D4_MA_10_4",
             "D2.vs.D7_MA_10_4", "logFC_D2.vs.D7_MA_10_4",
"P.Value_D2.vs.D7_MA_10_4",
             "D1.vs.D7_MA_10_4", "logFC_D1.vs.D7_MA_10_4",
"P.Value_D1.vs.D7_MA_10_4",

             "D1.vs.D2_MA_10_5", "logFC_D1.vs.D2_MA_10_5",
"P.Value_D1.vs.D2_MA_10_5",
             "D2.vs.D4_MA_10_5", "logFC_D2.vs.D4_MA_10_5",
"P.Value_D2.vs.D4_MA_10_5",
             "D4.vs.D7_MA_10_5", "logFC_D4.vs.D7_MA_10_5",
"P.Value_D4.vs.D7_MA_10_5",
             "D1.vs.D4_MA_10_5", "logFC_D1.vs.D4_MA_10_5",
"P.Value_D1.vs.D4_MA_10_5",
             "D2.vs.D7_MA_10_5", "logFC_D2.vs.D7_MA_10_5",
"P.Value_D2.vs.D7_MA_10_5",
             "D1.vs.D7_MA_10_5", "logFC_D1.vs.D7_MA_10_5",
"P.Value_D1.vs.D7_MA_10_5",

             "D1.vs.D2_Mock", "logFC_D1.vs.D2_Mock", "P.Value_D1.vs.D2_Mock",
             "D2.vs.D4_Mock", "logFC_D2.vs.D4_Mock", "P.Value_D2.vs.D4_Mock",
             "D4.vs.D7_Mock", "logFC_D4.vs.D7_Mock", "P.Value_D4.vs.D7_Mock",
             "D1.vs.D4_Mock", "logFC_D1.vs.D4_Mock", "P.Value_D1.vs.D4_Mock",
             "D2.vs.D7_Mock", "logFC_D2.vs.D7_Mock", "P.Value_D2.vs.D7_Mock",
             "D1.vs.D7_Mock", "logFC_D1.vs.D7_Mock", "P.Value_D1.vs.D7_Mock",

             # Add the rest of the columns in the same pattern
             "AveExpr", "F", "P.Value", "adj.P.Val")]

  # Store the updated data frame in the list
  results[[i]] <- df
}
df <- do.call("rbind", results)

df$P.Value <- ifelse(df$P.Value < 0.0001, formatC(df$P.Value, format = "e", digits = 2),
formatC(df$P.Value, format = "f", digits = 4))

df$adj.P.Val <- ifelse(df$adj.P.Val < 0.0001, formatC(df$adj.P.Val, format = "e", digits = 2),
formatC(df$adj.P.Val, format = "f", digits = 4))

df$P.Value_D1.vs.D2_MA_10_4 <- ifelse(df$P.Value_D1.vs.D2_MA_10_4 < 0.0001,
formatC(df$P.Value_D1.vs.D2_MA_10_4, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D2_MA_10_4, format = "f", digits = 4))
```

```
df$P.Value_D2.vs.D4_MA_10_4 <- ifelse(df$P.Value_D2.vs.D4_MA_10_4 < 0.0001,
formatC(df$P.Value_D2.vs.D4_MA_10_4, format = "e", digits = 2),
formatC(df$P.Value_D2.vs.D4_MA_10_4, format = "f", digits = 4))

df$P.Value_D4.vs.D7_MA_10_4 <- ifelse(df$P.Value_D4.vs.D7_MA_10_4 < 0.0001,
formatC(df$P.Value_D4.vs.D7_MA_10_4, format = "e", digits = 2),
formatC(df$P.Value_D4.vs.D7_MA_10_4, format = "f", digits = 4))

df$P.Value_D1.vs.D4_MA_10_4 <- ifelse(df$P.Value_D1.vs.D4_MA_10_4 < 0.0001,
formatC(df$P.Value_D1.vs.D4_MA_10_4, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D4_MA_10_4, format = "f", digits = 4))

df$P.Value_D2.vs.D7_MA_10_4 <- ifelse(df$P.Value_D2.vs.D7_MA_10_4 < 0.0001,
formatC(df$P.Value_D2.vs.D7_MA_10_4, format = "e", digits = 2),
formatC(df$P.Value_D2.vs.D7_MA_10_4, format = "f", digits = 4))

df$P.Value_D1.vs.D7_MA_10_4 <- ifelse(df$P.Value_D1.vs.D7_MA_10_4 < 0.0001,
formatC(df$P.Value_D1.vs.D7_MA_10_4, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D7_MA_10_4, format = "f", digits = 4))



df$P.Value_D1.vs.D2_MA_10_5 <- ifelse(df$P.Value_D1.vs.D2_MA_10_5 < 0.0001,
formatC(df$P.Value_D1.vs.D2_MA_10_5, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D2_MA_10_4, format = "f", digits = 4))

df$P.Value_D2.vs.D4_MA_10_5 <- ifelse(df$P.Value_D2.vs.D4_MA_10_5 < 0.0001,
formatC(df$P.Value_D2.vs.D4_MA_10_5, format = "e", digits = 2),
formatC(df$P.Value_D2.vs.D4_MA_10_5, format = "f", digits = 4))

df$P.Value_D4.vs.D7_MA_10_5 <- ifelse(df$P.Value_D4.vs.D7_MA_10_5 < 0.0001,
formatC(df$P.Value_D4.vs.D7_MA_10_5, format = "e", digits = 2),
formatC(df$P.Value_D4.vs.D7_MA_10_5, format = "f", digits = 4))

df$P.Value_D1.vs.D4_MA_10_5 <- ifelse(df$P.Value_D1.vs.D4_MA_10_5 < 0.0001,
formatC(df$P.Value_D1.vs.D4_MA_10_5, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D4_MA_10_5, format = "f", digits = 4))

df$P.Value_D2.vs.D7_MA_10_5 <- ifelse(df$P.Value_D2.vs.D7_MA_10_5 < 0.0001,
formatC(df$P.Value_D2.vs.D7_MA_10_5, format = "e", digits = 2),
formatC(df$P.Value_D2.vs.D7_MA_10_5, format = "f", digits = 4))

df$P.Value_D1.vs.D7_MA_10_5 <- ifelse(df$P.Value_D1.vs.D7_MA_10_5 < 0.0001,
formatC(df$P.Value_D1.vs.D7_MA_10_5, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D7_MA_10_5, format = "f", digits = 4))



df$P.Value_D1.vs.D2_Mock <- ifelse(df$P.Value_D1.vs.D2_Mock < 0.0001,
formatC(df$P.Value_D1.vs.D2_Mock, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D2_Mock, format = "f", digits = 4))
```

```r
df$P.Value_D2.vs.D4_Mock <- ifelse(df$P.Value_D2.vs.D4_Mock < 0.0001,
formatC(df$P.Value_D2.vs.D4_Mock, format = "e", digits = 2),
formatC(df$P.Value_D2.vs.D4_Mock, format = "f", digits = 4))

df$P.Value_D4.vs.D7_Mock <- ifelse(df$P.Value_D4.vs.D7_Mock < 0.0001,
formatC(df$P.Value_D4.vs.D7_Mock, format = "e", digits = 2),
formatC(df$P.Value_D4.vs.D7_Mock, format = "f", digits = 4))

df$P.Value_D1.vs.D4_Mock <- ifelse(df$P.Value_D1.vs.D4_Mock < 0.0001,
formatC(df$P.Value_D1.vs.D4_Mock, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D4_Mock, format = "f", digits = 4))

df$P.Value_D2.vs.D7_Mock <- ifelse(df$P.Value_D2.vs.D7_Mock < 0.0001,
formatC(df$P.Value_D2.vs.D7_Mock, format = "e", digits = 2),
formatC(df$P.Value_D2.vs.D7_Mock, format = "f", digits = 4))

df$P.Value_D1.vs.D7_Mock <- ifelse(df$P.Value_D1.vs.D7_Mock < 0.0001,
formatC(df$P.Value_D1.vs.D7_Mock, format = "e", digits = 2),
formatC(df$P.Value_D1.vs.D7_Mock, format = "f", digits = 4))
```

df

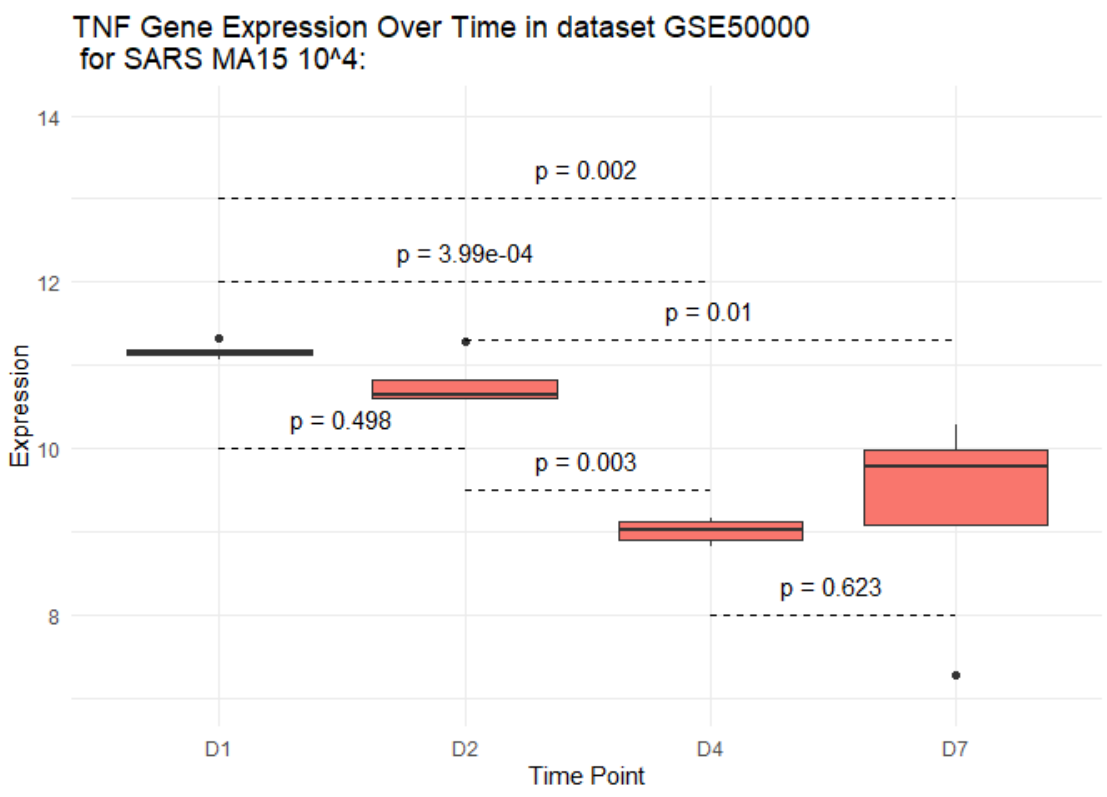
```
  P   P   P   P   P   P   P   P   P   P   P   P
  .   .   .   .   .   .   .   .   .   .   .   .
 I V I V I V I V I V I V I V I V I V I V I V I V
 oa  oa  oa  oa  oa  oa  oa  oa  oa  oa  oa  oa   P   P   P   P   P   P
 gl  gl  gl  gl  gl  gl  gl  gl  gl  gl  gl  gl   .   .   .   .   .   .
 Fu  Fu  Fu  Fu  Fu  Fu  Fu  Fu  Fu  Fu  Fu  Fu  I V I V I V I V I V I V
 Ce  Ce  Ce  Ce  Ce  Ce  Ce  Ce  Ce  Ce  Ce  Ce  oa  oa  oa  oa  oa  oa
 __  __  __  __  __  __  __  __  __  __  __  __  gl  gl  gl  gl  gl  gl
DDD DDD DDD DDD DDD DDD DDD DDD DDD DDD DDD DDD  Fu  Fu  Fu  Fu  Fu  Fu
111 222 444 111 222 111 111 222 444 111 222 111  Ce  Ce  Ce  Ce  Ce  Ce
... ... ... ... ... ... ... ... ... ... ... ...  __  __  __  __  __  __
vvv vvv vvv vvv vvv vvv vvv vvv vvv vvv vvv vvv DDDDDDDDDDDDDDDDDDD
sss sss sss sss sss sss sss sss sss sss sss sss 11122244411122211 1
... ... ... ... ... ... ... ... ... ... ... ... .................
DDD DDD DDD DDD DDD DDD DDD DDD DDD DDD DDD DDD vv vvv vvv vvv vvv vvv v
222 444 777 444 777 777 222 444 777 444 777 777 ss sss sss sss sss sss s  a
___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ..................  d
MMM MMM MMM MMM MMM MMM MMM MMM MMM MMM MMM MMM DDDDDDDDDDDDDDDDDDAj
AAA AAA AAA AAA AAA AAA AAA AAA AAA AAA AAA AAA 22 244 477 744 477 777 7v..
___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___________________eVP
111 111 111 111 111 111 111 111 111 111 111 111 MMMMMMMMMMMMMMMMMMEa
000 000 000 000 000 000 000 000 000 000 000 000 oo ooo ooo ooo ooo ooo oxlV
___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ cc ccc ccc ccc ccc ccc cpua
444 444 444 444 444 444 555 555 555 555 555 555 kk kkk kkk kkk kkk kkk kkd
```

```
00 0 10 0 - - 0 20 0 10 0 10 0 10   00 0 - - 0 20 0 00 0 10 0 00 0- - 00 0 0- - 00 0 00 0 0 90
. . . . . . 00 . . . . . . . . . . . . 0 . . . 00 . . . . . . . . . . . . . . 00 . . . . 00 . . . . . . . . . ..
30 4 72 0 . . 6 13 0 52 0 82 0 21 .  91 0 . . 1 23 0 10 7 42 0 20 6 . . 35 1 3 . . 60 0 93 0 50 60
74 9 86 0 20 2 61 0 11 0 96 0 67 4 65 7 71 8 32 0 82 5 50 2 65 35 0 58 1 02 0 56 1 03 6 55 90
99 8    2 74 3    0  7 9   7 1   5 9 71 3 82 0   6 0 67 7   3 0 83 0 19 9   3 14 4 95 3 64 6 0 07
 8 2    7 43 1    4    9     6     8   9 13 5     1  8 7     8    94 9 2     86 6 58 3 0   2 0 41
            3                      2                            8           9
```

```
00 2 00 0 - - 0 00 5 00 0 00 9 00 3 00 0 - - 0 00 7 00 0 00 1 00 0- - 0- - 0- - 0- - 0- - 037
. . . . . . 00 . . . . . . . . . . . . . . . . 00 . . . . . . . . . . . . . . 00 . 00 . 00 . 00 . 00 . 0.
60 9 20 0 . . 2 81 7 10 0 71 4 50 1 20 0 . . 0 81 9 00 4 60 3 00 9 . . 4 . . 9 . . 4 . . 3 . . 4 . 95
18 4 43 0 00 9 61 4 62 4 70 3 57 4 53 0 10 2 11 2 60 2 28 4 00 900 400 300 500 900 0563
81 e 63 3 81 3 46 e 12 9 94 e 54 e 55 1 82 8     e 89 7 44 e 00 560 800 060 160 860 1 e
 9 -   97 41 0     -   1 2     -   5 -   57 66 9     - 75 2     - 51 408 361 708 679 679 7 -
   0       57         1           1     0                 1           0     75   9     24   64   24   11
   9                   3           2     8                 3           8                              43
00 0 00 0 00 0 00 0 00 0 00 1 00   00 0 00 0 00 2 00 8 10 1 - - 000 0- - 000 000 000 017
. . . . . . . . . . . . . . . . . . . . . . 0 . . . . . . . . . . . . . . . . 00 . . . . 00 . . . . . . . . . . 31.
50 0 00 8 10 4 50 0 10 3 70 3 30 .  50 0 20 0 80 3 70 7 01 6 . . 900 6 . . 700 700 900 9 . 19
66 0 20 4 11 1 86 0 41 1 07 5 13 0 35 0 52 9 49 6 88 7 92 0 00 650 700 250 000 400 720
11 0 73 4 53 7 94 0 26 5 47 e 13 0 19 0   9 1 23 e 18 e   2 e 00 886 640 735 691 540 7 e
 7 3 61 4     0   8 2   1 1   7 -   8 0   6 3     2   4 -   5 -     - 50 274 895 118 16   8   4 5 -
                              0     3                 0     0     0 66           14                  00
                              5                       7     6     8                                  87
00 0 - - 0 - - 0 00 0 - - 0 - - 0 00   - - 0 00 0 - - 0 00 0 00 0 - - 000 0- - 000 000 0- - 021
. . . 00 . 00 . . . . 00 . 00 . . . 0 00 . . . . 00 . . . . . . . . 00 . . . . 00 . . . . . . . . 00 . 0.
60 0 . . 0 . . 2 20 2 . . 0 . . 9 00 . . . 9 00 8 . . 9 00 9 00 9 . . 510 3 . . 400 700 8 . . 7.10
69 0 40 1 20 2 12 5 60 0 00 3 00 0 00 5 20 9 00 7 10 4 10 2 10 672 610 960 440 100 290
81 0 56 8 33 1 48 3 89 0 10 3 40 0 10 0 53 5 00 1 41 3 82 5 01 0   1 321 717 035 460 7 50
 4 9 42 8   0 4   6 8 43 7 52 0 56 0 01 8 24 5 60 1 39 1 86 3 83 6   6 266 968 765 648 5 70
       7     7         4   61       9 95       49                 7         1               82
```

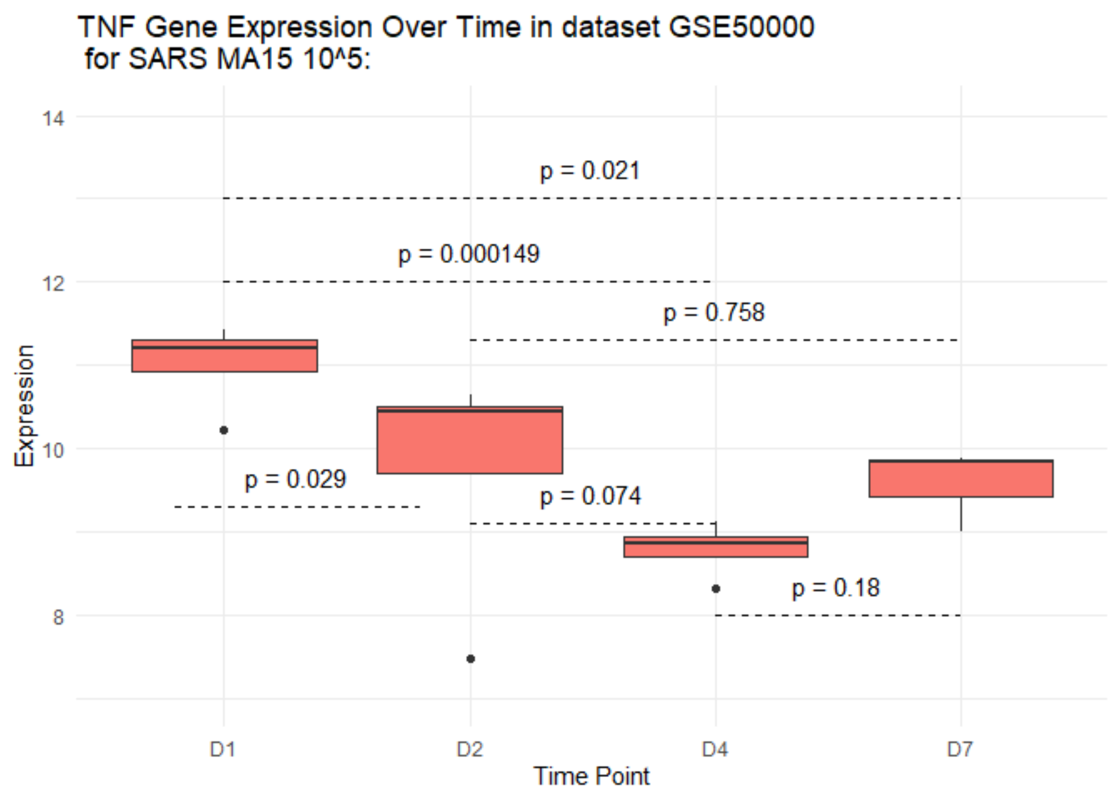
###TNF:

```
format_pvalue <- function(pvalue) {
  if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 3))
  }
}

tnf_data <- subset(final,
                   GENE_SYMBOL %in% c("Tnf"))

tnf_data_MA_10_4 <- tnf_data[tnf_data$Type == "MA_10_4",]


# Extract the p-values for MA_10_4
pvalues1_MA_10_4_D2_D1 <- results$Tnf$P.Value_D1.vs.D2_MA_10_4
pvalues1_MA_10_4_D4_D2 <- results$Tnf$P.Value_D2.vs.D4_MA_10_4
pvalues1_MA_10_4_D4_D1 <- results$Tnf$P.Value_D1.vs.D4_MA_10_4
pvalues1_MA_10_4_D7_D4 <- results$Tnf$P.Value_D4.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D2 <- results$Tnf$P.Value_D2.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D1 <- results$Tnf$P.Value_D1.vs.D7_MA_10_4

# Create a new data frame for the annotations for MA_10_4
annotations_MA_10_4 <- data.frame(
  Type = "MA_10_4",
  Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
  Label = paste("p =", sapply(c(pvalues1_MA_10_4_D2_D1, pvalues1_MA_10_4_D4_D2,
pvalues1_MA_10_4_D4_D1, pvalues1_MA_10_4_D7_D4, pvalues1_MA_10_4_D7_D2,
pvalues1_MA_10_4_D7_D1), format_pvalue)),
 Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)

# Create the line chart for MA_10_4
p_MA_10_4 <- ggplot(data = tnf_data_MA_10_4, aes(x = Measurement, y = Expression, fill
= Type)) +
 geom_boxplot() +
 labs(title = "TNF Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^4:",
      x = "Time Point",
      y = "Expression") +
 theme_minimal() +
 ylim(7, 14) +
 theme(legend.position = "none")

# Add lines for MA_10_4
p_MA_10_4 <- p_MA_10_4 +
 geom_segment(aes(x = 1, y = 10, xend = 2, yend = 10), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 2, y = 9.5, xend = 3, yend = 9.5), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 1, y = 12, xend = 3, yend = 12), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 3, y = 8, xend = 4, yend = 8), linetype = "dashed", color = "black")
+
 geom_segment(aes(x = 2, y = 11.3, xend = 4, yend = 11.3), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 1, y = 13, xend = 4, yend = 13), linetype = "dashed", color =
"black")

# Add p-values for MA_10_4
p_MA_10_4 <- p_MA_10_4 + annotate("text", x = c(1.5, 2.5, 2, 3.5, 3, 2.5), y = c(10, 9.5,
12, 8, 11.3, 13), label = annotations_MA_10_4$Label, vjust = -1)

print(p_MA_10_4)
```



```
# Define a function to format p-values
format_pvalue <- function(pvalue) {
  if (pvalue < 0.001) {
    return(format(pvalue, scientific = FALSE, digits = 3))
  } else {
    return(round(pvalue, 3))
  }
}


tnf_data_MA_10_5 <- tnf_data[tnf_data$Type == "MA_10_5",]

# Extract the p-values for MA_10_4
pvalues1_MA_10_5_D2_D1 <- results$Tnf$P.Value_D1.vs.D2_MA_10_5
pvalues1_MA_10_5_D4_D2 <- results$Tnf$P.Value_D2.vs.D4_MA_10_5
pvalues1_MA_10_5_D4_D1 <- results$Tnf$P.Value_D1.vs.D4_MA_10_5
pvalues1_MA_10_5_D7_D4 <- results$Tnf$P.Value_D4.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D2 <- results$Tnf$P.Value_D2.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D1 <- results$Tnf$P.Value_D1.vs.D7_MA_10_5

# Create a new data frame for the annotations for MA_10_5
annotations_MA_10_5 <- data.frame(
  Type = "MA_10_4",
  Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
  Label = paste("p =", sapply(c(pvalues1_MA_10_5_D2_D1, pvalues1_MA_10_5_D4_D2,
pvalues1_MA_10_5_D4_D1, pvalues1_MA_10_5_D7_D4, pvalues1_MA_10_5_D7_D2,
pvalues1_MA_10_5_D7_D1), format_pvalue)),
  Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)
```

```r
# Create the line chart for MA_10_4
p_MA_10_5 <- ggplot(data = tnf_data_MA_10_5, aes(x = Measurement, y = Expression, fill
= Type)) +
 geom_boxplot() +
 labs(title = "TNF Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^5:",
     x = "Time Point",
     y = "Expression") +
 theme_minimal() +
 ylim(7, 14) +
 theme(legend.position = "none")

# Add lines for MA_10_5
p_MA_10_5 <- p_MA_10_5 +
 geom_segment(aes(x = 0.8, y = 9.3, xend = 1.8, yend = 9.3), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 2, y = 9.1, xend = 3, yend = 9.1), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 1, y = 12, xend = 3, yend = 12), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 3, y = 8, xend = 4, yend = 8), linetype = "dashed", color = "black")
+
 geom_segment(aes(x = 2, y = 11.3, xend = 4, yend = 11.3), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 1, y = 13, xend = 4, yend = 13), linetype = "dashed", color =
"black")

# Add p-values for MA_10_4
p_MA_10_5 <- p_MA_10_5 + annotate("text", x = c(1.3, 2.5, 2, 3.5, 3, 2.5), y = c(9.3, 9.1,
12, 8, 11.3, 13), label = annotations_MA_10_5$Label, vjust = -1)

print(p_MA_10_5)
```



Between different Time points:

Day 1:

```
tnf_data$Measurement <- as.character(tnf_data$Measurement)

# Subset the data to include only observations at time point D1
tnf_data_D1 <- subset(tnf_data,
                     Measurement %in% c("D1"))

tnf_data_D1$group <- interaction(tnf_data_D1$Type, tnf_data_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data_D1)

# Fit the model
fit <- lmFit(tnf_data_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit1_Day1 <- contrasts.fit(fit, cont.matrix)
```

```
# Compute differential expression statistics
fit1_Day1 <- eBayes(fit1_Day1)

# Get the top table
results_D1 <- topTable(fit1_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##     logFC  AveExpr         t   P.Value adj.P.Val         B
## 1 0.15225 9.873725 0.6105722 0.5565853 0.5565853 -4.642057
```

Day 2:

```
tnf_data$Measurement <- as.character(tnf_data$Measurement)

# Subset the data to include only observations at time point D1
tnf_data_D2 <- subset(tnf_data,
                      Measurement %in% c("D2"))

tnf_data_D2$group <- interaction(tnf_data_D2$Type, tnf_data_D2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data_D2)

# Fit the model
fit <- lmFit(tnf_data_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
  levels = design
)

# Fit the contrasts
fit1_Day2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day2 <- eBayes(fit1_Day2)

# Get the top table
results_D2 <- topTable(fit1_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##    logFC  AveExpr        t   P.Value adj.P.Val         B
## 1 1.0335 9.238075 1.616176 0.1405119 0.1405119 -4.420975
```

Day 4:

```
tnf_data$Measurement <- as.character(tnf_data$Measurement)

# Subset the data to include only observations at time point D1
```

```
tnf_data_D4 <- subset(tnf_data,
                      Measurement %in% c("D4"))

tnf_data_D4$group <- interaction(tnf_data_D4$Type, tnf_data_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data_D4)

# Fit the model
fit <- lmFit(tnf_data_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
   D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
   levels = design
)

# Fit the contrasts
fit1_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day4 <- eBayes(fit1_Day4)

# Get the top table
results_D4 <- topTable(fit1_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##     logFC  AveExpr         t   P.Value adj.P.Val        B
## 1 0.21842 8.515908 0.3787094 0.7128199 0.7128199 -4.60766
```

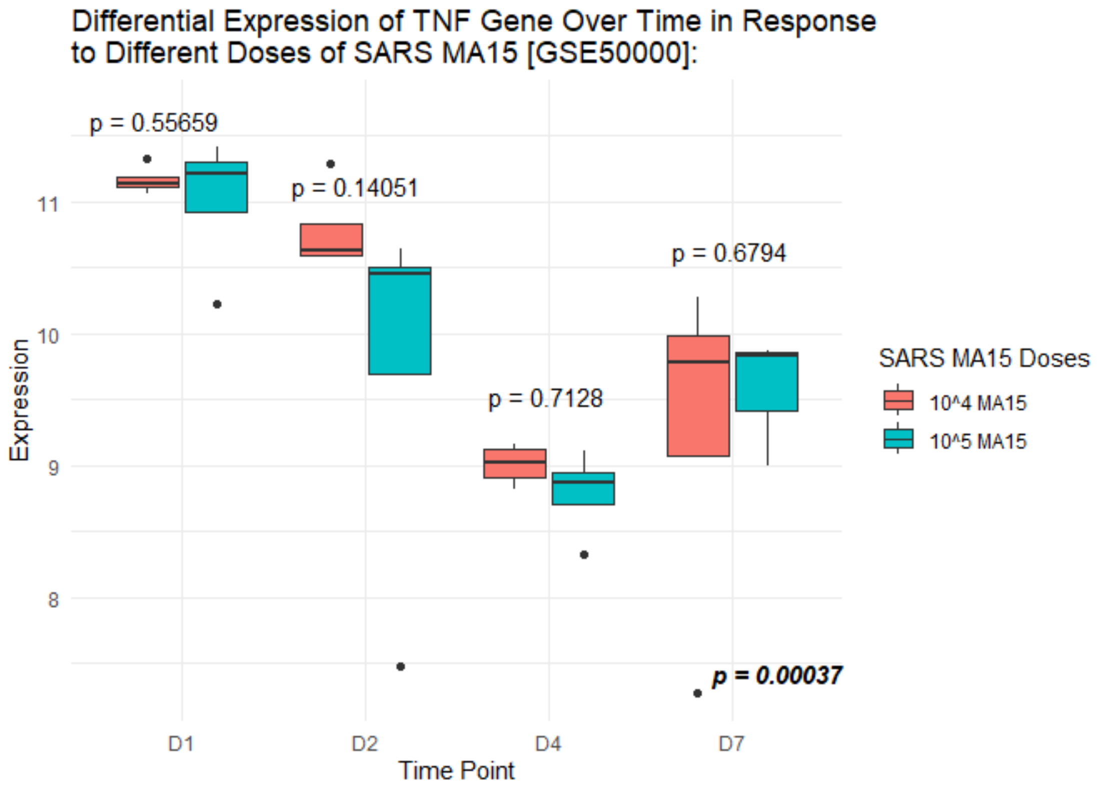
Day 7:

```
tnf_data$Measurement <- as.character(tnf_data$Measurement)

# Subset the data to include only observations at time point D1
tnf_data_D7 <- subset(tnf_data,
                      Measurement %in% c("D7"))

tnf_data_D7$group <- interaction(tnf_data_D7$Type, tnf_data_D7$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data_D7)

# Fit the model
fit <- lmFit(tnf_data_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
   D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
   levels = design
)
```

```
# Fit the contrasts
fit1_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day7 <- eBayes(fit1_Day7)

# Get the top table
results_D7 <- topTable(fit1_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##        logFC  AveExpr          t   P.Value adj.P.Val         B
## 1 -0.2880167 8.568482 -0.4288196 0.6793678 0.6793678 -4.603889
```

Graph 3:

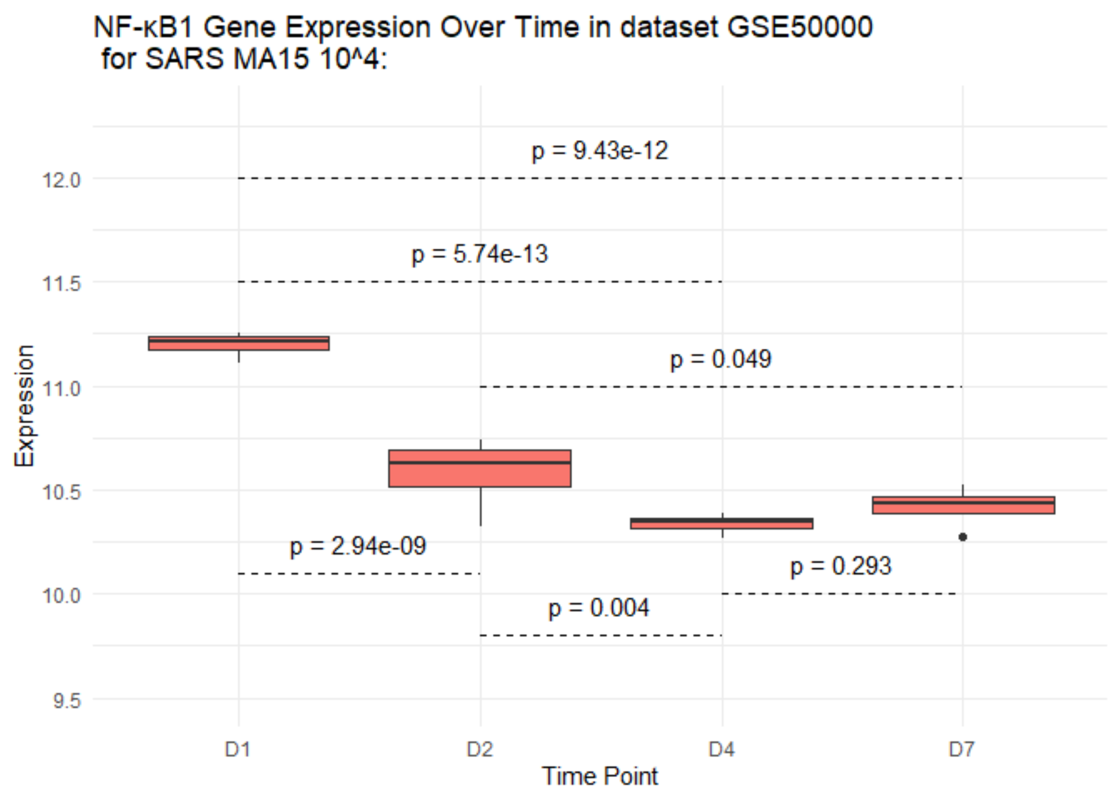
```
tnf_data1 <- subset(tnf_data,
                    Type %in% c("MA_10_4", "MA_10_5"))

tnf_data_WT_MA15 <- tnf_data1

P.Value1 <- results$Tnf$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value


# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "Differential Expression of TNF Gene Over Time in Response \nto Different
Doses of SARS MA15 [GSE50000]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^4 MA15", "10^5 MA15"))
+
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = Inf, y = 7.5, label = paste0("p = ", round(P.Value1, 5)), hjust =
1, vjust = 1, size = 4, fontface = "bold.italic") +
  annotate("text", x = 1.2, y = 11.7, label = paste0("p = ", round(P.Value2, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 2.3, y = 11.2, label = paste0("p = ", round(P.Value3, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 3.3, y = 9.6, label = paste0("p = ", round(P.Value4, 4)), hjust = 1, vjust
= 1) +
  annotate("text", x = 4.3, y = 10.7, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)


print(p)
```



NFKB1:

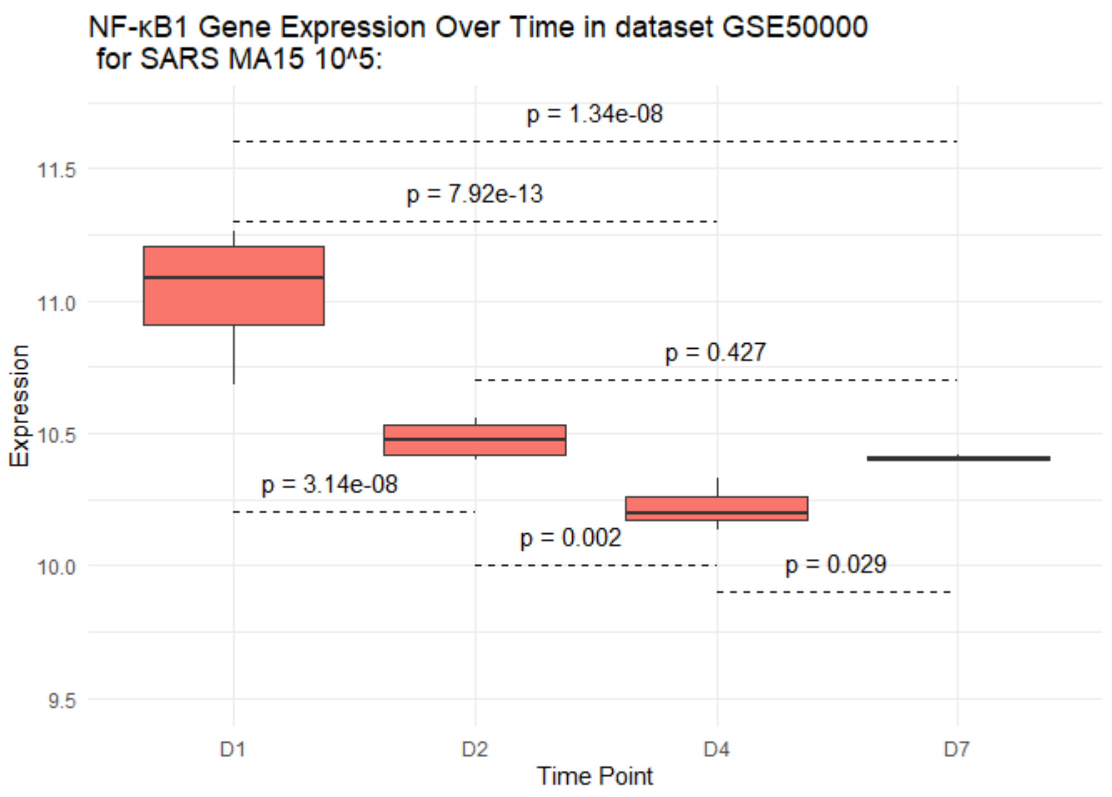
```
# Define a function to format p-values
format_pvalue <- function(pvalue) {
  if (pvalue < 0.0001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 3))
  }
}


nfkb_data <- subset(final,
                    GENE_SYMBOL %in% c("Nfkb1"))

tnf_data_MA_10_4 <- nfkb_data[nfkb_data$Type == "MA_10_4",]


# Extract the p-values for MA_10_4
pvalues1_MA_10_4_D2_D1 <- results$Nfkb1$P.Value_D1.vs.D2_MA_10_4
pvalues1_MA_10_4_D4_D2 <- results$Nfkb1$P.Value_D2.vs.D4_MA_10_4
pvalues1_MA_10_4_D4_D1 <- results$Nfkb1$P.Value_D1.vs.D4_MA_10_4
```

```
pvalues1_MA_10_4_D7_D4 <- results$Nfkb1$P.Value_D4.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D2 <- results$Nfkb1$P.Value_D2.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D1 <- results$Nfkb1$P.Value_D1.vs.D7_MA_10_4


# Create a new data frame for the annotations for MA_10_4
annotations_MA_10_4 <- data.frame(
 Type = "MA_10_4",
 Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
 Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
 Label = paste("p =", sapply(c(pvalues1_MA_10_4_D2_D1, pvalues1_MA_10_4_D4_D2,
pvalues1_MA_10_4_D4_D1, pvalues1_MA_10_4_D7_D4, pvalues1_MA_10_4_D7_D2,
pvalues1_MA_10_4_D7_D1), format_pvalue)),
 Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)

# Create the line chart for MA_10_4
p_MA_10_4 <- ggplot(data = tnf_data_MA_10_4, aes(x = Measurement, y = Expression, fill
= Type)) +
 geom_boxplot() +
 labs(title = "NF-κB1 Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^4:",
      x = "Time Point",
      y = "Expression") +
 theme_minimal() +
 ylim(9.5, 12.3) +
 theme(legend.position = "none")

# Add lines for MA_10_4
p_MA_10_4 <- p_MA_10_4 +
 geom_segment(aes(x = 1, y = 10.1, xend = 2, yend = 10.1), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 2, y = 9.8, xend = 3, yend = 9.8), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 1, y = 11.5, xend = 3, yend = 11.5), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 3, y = 10, xend = 4, yend = 10), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 2, y = 11, xend = 4, yend = 11), linetype = "dashed", color =
"black") +
 geom_segment(aes(x = 1, y = 12, xend = 4, yend = 12), linetype = "dashed", color =
"black")

# Add p-values for MA_10_4
p_MA_10_4 <- p_MA_10_4 + annotate("text", x = c(1.5, 2.5, 2, 3.5, 3, 2.5), y = c(10.1, 9.8,
11.5, 10, 11, 12), label = annotations_MA_10_4$Label, vjust = -1)

print(p_MA_10_4)
```



```
# Define a function to format p-values
format_pvalue <- function(pvalue) {
  if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 3))
  }
}


tnf_data_MA_10_5 <- nfkb_data[nfkb_data$Type == "MA_10_5",]

# Extract the p-values for MA_10_4
pvalues1_MA_10_5_D2_D1 <- results$Nfkb1$P.Value_D1.vs.D2_MA_10_5
pvalues1_MA_10_5_D4_D2 <- results$Nfkb1$P.Value_D2.vs.D4_MA_10_5
pvalues1_MA_10_5_D4_D1 <- results$Nfkb1$P.Value_D1.vs.D4_MA_10_5
pvalues1_MA_10_5_D7_D4 <- results$Nfkb1$P.Value_D4.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D2 <- results$Nfkb1$P.Value_D2.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D1 <- results$Nfkb1$P.Value_D1.vs.D7_MA_10_5

# Create a new data frame for the annotations for MA_10_5
annotations_MA_10_5 <- data.frame(
  Type = "MA_10_4",
  Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
  Label = paste("p =", sapply(c(pvalues1_MA_10_5_D2_D1, pvalues1_MA_10_5_D4_D2,
pvalues1_MA_10_5_D4_D1, pvalues1_MA_10_5_D7_D4, pvalues1_MA_10_5_D7_D2,
pvalues1_MA_10_5_D7_D1), format_pvalue)),
  Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)
```

```r
# Create the line chart for MA_10_4
p_MA_10_5 <- ggplot(data = tnf_data_MA_10_5, aes(x = Measurement, y = Expression, fill
= Type)) +
  geom_boxplot() +
  labs(title = "NF-κB1 Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^5:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(9.5, 11.7) +
  theme(legend.position = "none")

# Add lines for MA_10_5
p_MA_10_5 <- p_MA_10_5 +
  geom_segment(aes(x = 1, y = 10.2, xend = 2, yend = 10.2), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 10, xend = 3, yend = 10), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11.3, xend = 3, yend = 11.3), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 3, y = 9.9, xend = 4, yend = 9.9), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 10.7, xend = 4, yend = 10.7), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11.6, xend = 4, yend = 11.6), linetype = "dashed", color =
"black")

# Add p-values for MA_10_4
p_MA_10_5 <- p_MA_10_5 + annotate("text", x = c(1.4, 2.4, 2, 3.5, 3, 2.5), y = c(10.2, 10,
11.3, 9.9, 10.7, 11.6), label = annotations_MA_10_5$Label, vjust = -1)

print(p_MA_10_5)
```



Between different Time points:

Day 1:

```
nfkb_data$Measurement <- as.character(nfkb_data$Measurement)

# Subset the data to include only observations at time point D1
nfkb_data_D1 <- subset(nfkb_data,
                      Measurement %in% c("D1"))

nfkb_data_D1$group <- interaction(nfkb_data_D1$Type, nfkb_data_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb_data_D1)

# Fit the model
fit <- lmFit(nfkb_data_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
   D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

   levels = design
)

# Fit the contrasts
fit1_Day1 <- contrasts.fit(fit, cont.matrix)
```

```
# Compute differential expression statistics
fit1_Day1 <- eBayes(fit1_Day1)

# Get the top table
results_D1 <- topTable(fit1_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##       logFC  AveExpr        t   P.Value adj.P.Val         B
## 1 0.1679375 10.84079 1.510168 0.1652824 0.1652824 -4.497914
```

Day 2:

```
nfkb_data$Measurement <- as.character(nfkb_data$Measurement)

# Subset the data to include only observations at time point D1
nfkb_data_D2 <- subset(nfkb_data,
                       Measurement %in% c("D2"))

nfkb_data_D2$group <- interaction(nfkb_data_D2$Type, nfkb_data_D2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb_data_D2)

# Fit the model
fit <- lmFit(nfkb_data_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
  levels = design
)

# Fit the contrasts
fit1_Day2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day2 <- eBayes(fit1_Day2)

# Get the top table
results_D2 <- topTable(fit1_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##       logFC  AveExpr        t   P.Value adj.P.Val         B
## 1 0.1050687 10.44949 1.208095 0.2577961 0.2577961 -4.652613
```

Day 4:

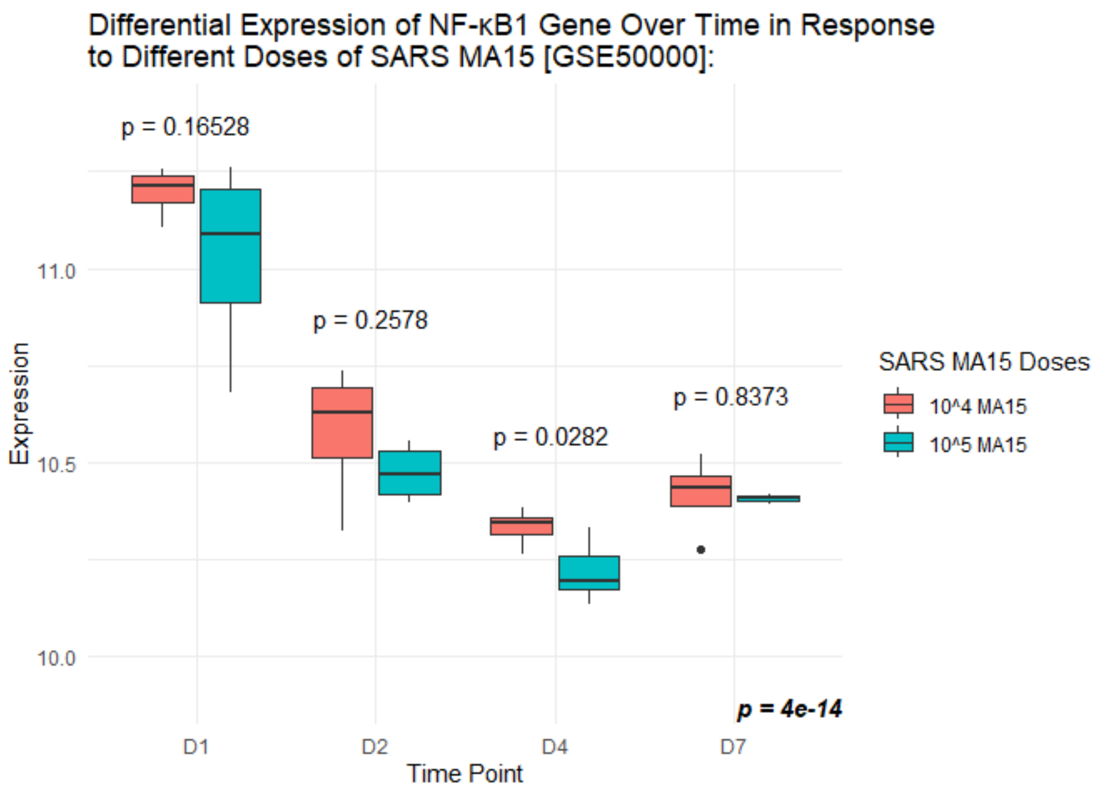
```
nfkb_data$Measurement <- as.character(nfkb_data$Measurement)

# Subset the data to include only observations at time point D1
```

```
nfkb_data_D4 <- subset(nfkb_data,
                       Measurement %in% c("D4"))

nfkb_data_D4$group <- interaction(nfkb_data_D4$Type, nfkb_data_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb_data_D4)

# Fit the model
fit <- lmFit(nfkb_data_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
    D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
    levels = design
)

# Fit the contrasts
fit1_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day4 <- eBayes(fit1_Day4)

# Get the top table
results_D4 <- topTable(fit1_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##       logFC  AveExpr        t    P.Value  adj.P.Val         B
## 1 0.1141913 10.29647 2.563451 0.02820572 0.02820572 -3.378412
```

Day 7:

```
nfkb_data$Measurement <- as.character(nfkb_data$Measurement)

# Subset the data to include only observations at time point D1
nfkb_data_D7 <- subset(nfkb_data,
                       Measurement %in% c("D7"))

nfkb_data_D7$group <- interaction(nfkb_data_D7$Type, nfkb_data_D7$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb_data_D7)

# Fit the model
fit <- lmFit(nfkb_data_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
    D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
    levels = design
)
```

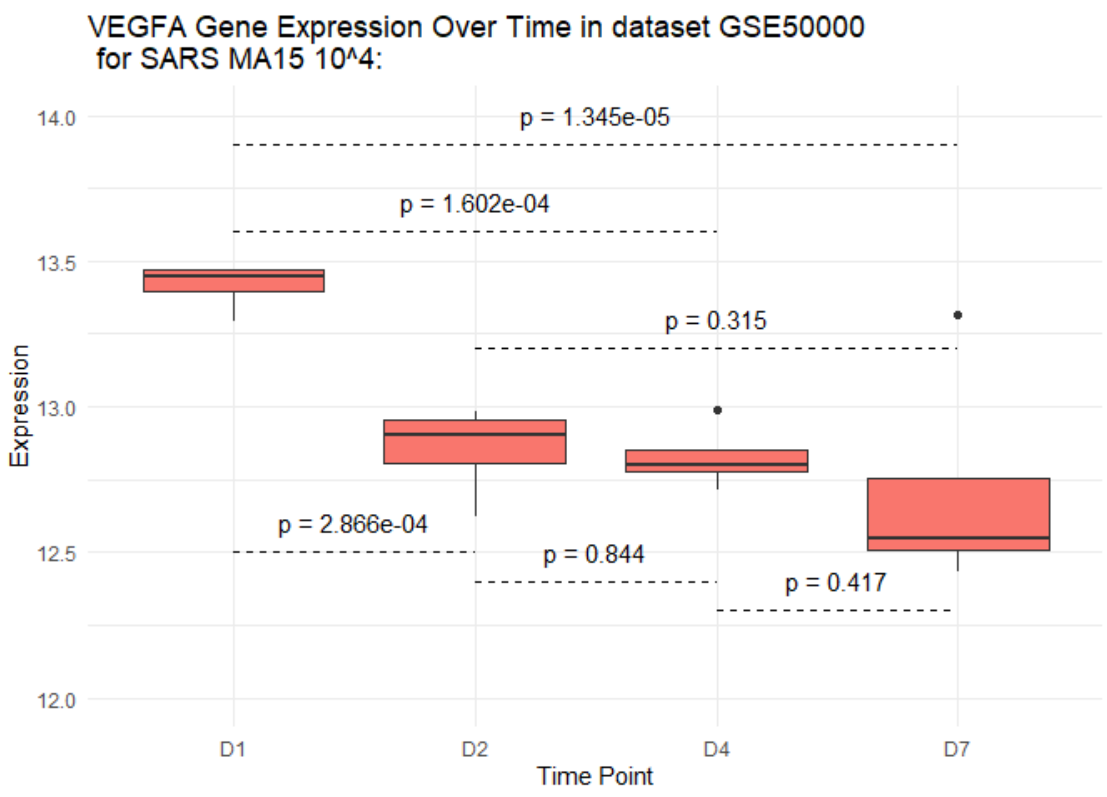
```
# Fit the contrasts
fit1_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day7 <- eBayes(fit1_Day7)

# Get the top table
results_D7 <- topTable(fit1_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##        logFC  AveExpr         t   P.Value adj.P.Val         B
## 1 0.01259792 10.39455 0.2121635 0.8372853 0.8372853 -5.248713
nfkb_data1 <- subset(nfkb_data,
                     Type %in% c("MA_10_4", "MA_10_5"))

tnf_data_WT_MA15 <- nfkb_data1

P.Value1 <- results$Nfkb1$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type)) 
+
  geom_boxplot() +
  labs(title = "Differential Expression of NF-κB1 Gene Over Time in Response \nto Different 
Doses of SARS MA15 [GSE50000]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^4 MA15", "10^5 MA15")) 
+
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = Inf, y = 9.9, label = paste0("p = ", format(round(P.Value1, 15), 
scientific = TRUE)), hjust = 1, vjust = 1, size = 4, fontface = "bold.italic") +

  annotate("text", x = 1.3, y = 11.4, label = paste0("p = ", round(P.Value2, 5)), hjust = 1, 
vjust = 1) +
  annotate("text", x = 2.3, y = 10.9, label = paste0("p = ", round(P.Value3, 5)), hjust = 1, 
vjust = 1) +
  annotate("text", x = 3.3, y = 10.6, label = paste0("p = ", round(P.Value4, 4)), hjust = 1,
vjust = 1) +
  annotate("text", x = 4.3, y = 10.7, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)


print(p)
```



VEGFA:

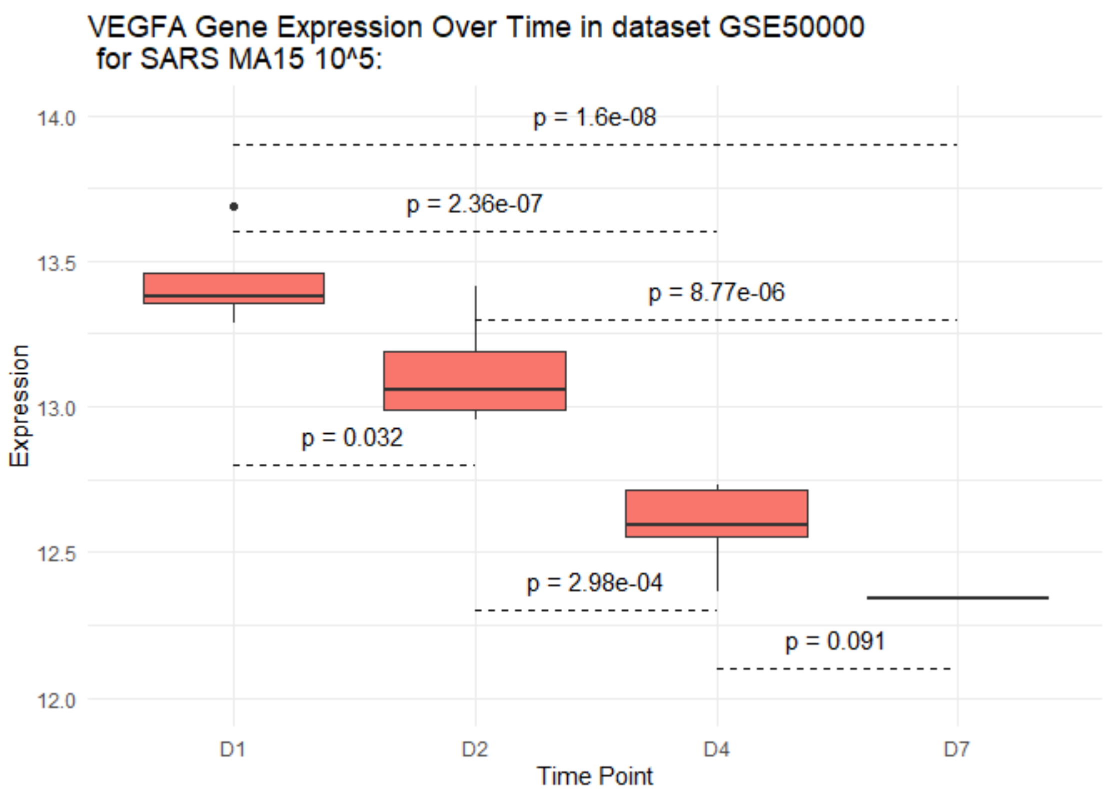
```
# Define a function to format p-values
format_pvalue <- function(pvalue) {
  if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 4))
  } else {
    return(round(pvalue, 3))
  }
}

vegfa_data <- subset(final,
                     GENE_SYMBOL %in% c("Vegfa"))


tnf_data_MA_10_4 <- vegfa_data[vegfa_data$Type == "MA_10_4",]

# Extract the p-values for MA_10_4
pvalues1_MA_10_4_D2_D1 <- results$Vegfa$P.Value_D1.vs.D2_MA_10_4
pvalues1_MA_10_4_D4_D2 <- results$Vegfa$P.Value_D2.vs.D4_MA_10_4
pvalues1_MA_10_4_D4_D1 <- results$Vegfa$P.Value_D1.vs.D4_MA_10_4
pvalues1_MA_10_4_D7_D4 <- results$Vegfa$P.Value_D4.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D2 <- results$Vegfa$P.Value_D2.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D1 <- results$Vegfa$P.Value_D1.vs.D7_MA_10_4
```

```r
# Create a new data frame for the annotations for MA_10_4
annotations_MA_10_4 <- data.frame(
  Type = "MA_10_4",
  Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
  Label = paste("p =", sapply(c(pvalues1_MA_10_4_D2_D1, pvalues1_MA_10_4_D4_D2,
pvalues1_MA_10_4_D4_D1, pvalues1_MA_10_4_D7_D4, pvalues1_MA_10_4_D7_D2,
pvalues1_MA_10_4_D7_D1), format_pvalue)),
  Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)

# Create the line chart for MA_10_4
p_MA_10_4 <- ggplot(data = tnf_data_MA_10_4, aes(x = Measurement, y = Expression, fill
= Type)) +
  geom_boxplot() +
  labs(title = "VEGFA Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^4:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(12, 14) +
  theme(legend.position = "none")

# Add lines for MA_10_4
p_MA_10_4 <- p_MA_10_4 +
  geom_segment(aes(x = 1, y = 12.5, xend = 2, yend = 12.5), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 12.4, xend = 3, yend = 12.4), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 13.6, xend = 3, yend = 13.6), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 3, y = 12.3, xend = 4, yend = 12.3), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 13.2, xend = 4, yend = 13.2), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 13.9, xend = 4, yend = 13.9), linetype = "dashed", color =
"black")

# Add p-values for MA_10_4
p_MA_10_4 <- p_MA_10_4 + annotate("text", x = c(1.5, 2.5, 2, 3.5, 3, 2.5), y = c(12.5, 12.4,
13.6, 12.3, 13.2, 13.9), label = annotations_MA_10_4$Label, vjust = -1)

print(p_MA_10_4)
```



```
# Define a function to format p-values
format_pvalue <- function(pvalue) {
  if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 3))
  }
}


tnf_data_MA_10_5 <- vegfa_data[vegfa_data$Type == "MA_10_5",]

# Extract the p-values for MA_10_5
pvalues1_MA_10_5_D2_D1 <- results$Vegfa$P.Value_D1.vs.D2_MA_10_5
pvalues1_MA_10_5_D4_D2 <- results$Vegfa$P.Value_D2.vs.D4_MA_10_5
pvalues1_MA_10_5_D4_D1 <- results$Vegfa$P.Value_D1.vs.D4_MA_10_5
pvalues1_MA_10_5_D7_D4 <- results$Vegfa$P.Value_D4.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D2 <- results$Vegfa$P.Value_D2.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D1 <- results$Vegfa$P.Value_D1.vs.D7_MA_10_5

# Create a new data frame for the annotations for MA_10_5
annotations_MA_10_5 <- data.frame(
  Type = "MA_10_4",
  Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
  Label = paste("p =", sapply(c(pvalues1_MA_10_5_D2_D1, pvalues1_MA_10_5_D4_D2,
pvalues1_MA_10_5_D4_D1, pvalues1_MA_10_5_D7_D4, pvalues1_MA_10_5_D7_D2,
pvalues1_MA_10_5_D7_D1), format_pvalue)),
  Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)
```

```
# Create the line chart for MA_10_4
p_MA_10_5 <- ggplot(data = tnf_data_MA_10_5, aes(x = Measurement, y = Expression, fill
= Type)) +
  geom_boxplot() +
  labs(title = "VEGFA Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^5:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(12, 14) +
  theme(legend.position = "none")

# Add lines for MA_10_5
p_MA_10_5 <- p_MA_10_5 +
  geom_segment(aes(x = 1, y = 12.8, xend = 2, yend = 12.8), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 12.3, xend = 3, yend = 12.3), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 13.6, xend = 3, yend = 13.6), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 3, y = 12.1, xend = 4, yend = 12.1), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 13.3, xend = 4, yend = 13.3), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 13.9, xend = 4, yend = 13.9), linetype = "dashed", color =
"black")


# Add p-values for MA_10_4
p_MA_10_5 <- p_MA_10_5 + annotate("text", x = c(1.5, 2.5, 2, 3.5, 3, 2.5), y = c(12.8, 12.3,
13.6, 12.1, 13.3, 13.9), label = annotations_MA_10_5$Label, vjust = -1)

print(p_MA_10_5)
```



Between different Time points:

Day 1:

```
vegfa_data$Measurement <- as.character(vegfa_data$Measurement)

# Subset the data to include only observations at time point D1
vegfa_data_D1 <- subset(vegfa_data,
                Measurement %in% c("D1"))

vegfa_data_D1$group <- interaction(vegfa_data_D1$Type, vegfa_data_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfa_data_D1)

# Fit the model
fit <- lmFit(vegfa_data_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit1_Day1 <- contrasts.fit(fit, cont.matrix)
```

```
# Compute differential expression statistics
fit1_Day1 <- eBayes(fit1_Day1)

# Get the top table
results_D1 <- topTable(fit1_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##    logFC  AveExpr          t   P.Value adj.P.Val         B
## 1 -0.018 13.36958 -0.1939669 0.8505091 0.8505091 -4.969258
```

Day 2:

```
vegfa_data$Measurement <- as.character(vegfa_data$Measurement)

# Subset the data to include only observations at time point D1
vegfa_data_D2 <- subset(vegfa_data,
                        Measurement %in% c("D2"))

vegfa_data_D2$group <- interaction(vegfa_data_D2$Type, vegfa_data_D2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfa_data_D2)

# Fit the model
fit <- lmFit(vegfa_data_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
  levels = design
)

# Fit the contrasts
fit1_Day2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day2 <- eBayes(fit1_Day2)

# Get the top table
results_D2 <- topTable(fit1_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##        logFC  AveExpr         t    P.Value  adj.P.Val         B
## 1 -0.2680312 13.08059 -2.451881 0.03664347 0.03664347 -3.553817
```

Day 4:

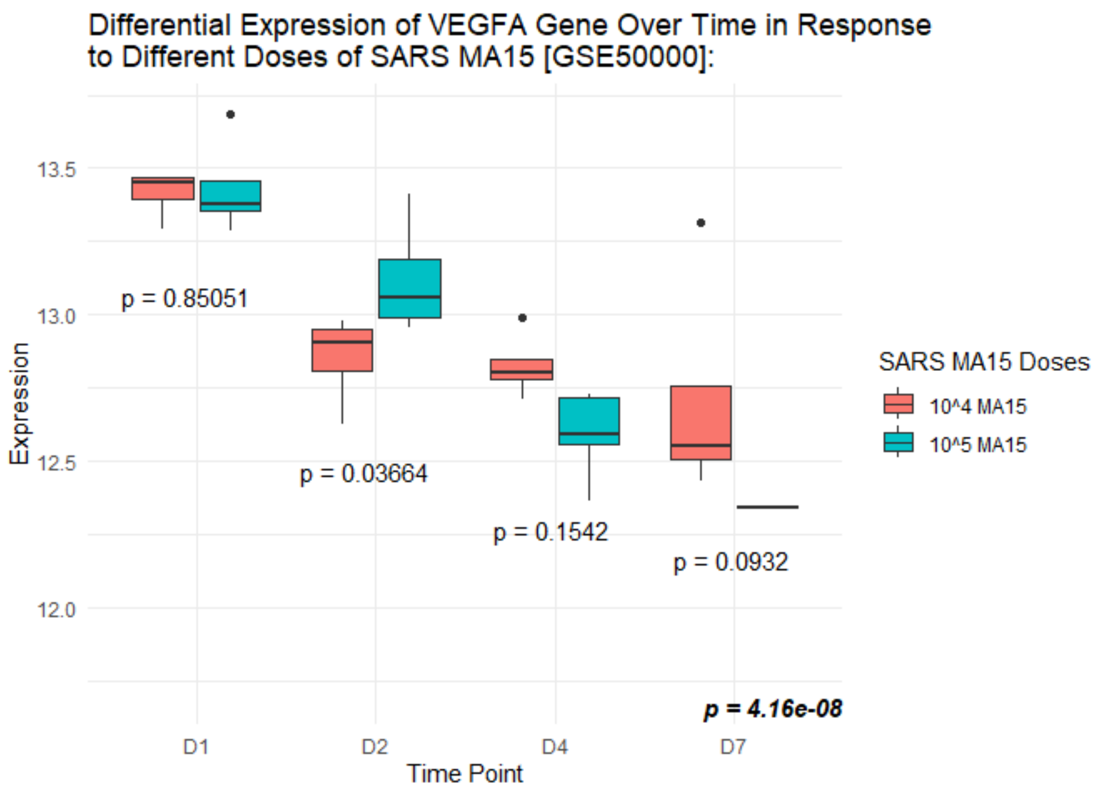
```
vegfa_data$Measurement <- as.character(vegfa_data$Measurement)

# Subset the data to include only observations at time point D1
```

```
vegfa_data_D4 <- subset(vegfa_data,
                        Measurement %in% c("D4"))

vegfa_data_D4$group <- interaction(vegfa_data_D4$Type, vegfa_data_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfa_data_D4)

# Fit the model
fit <- lmFit(vegfa_data_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
  levels = design
)

# Fit the contrasts
fit1_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day4 <- eBayes(fit1_Day4)

# Get the top table
results_D4 <- topTable(fit1_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##     logFC  AveExpr        t   P.Value adj.P.Val         B
## 1 0.23502 12.85305 1.541681 0.1541778 0.1541778 -4.475101
```

Day 7:

```
vegfa_data$Measurement <- as.character(vegfa_data$Measurement)

# Subset the data to include only observations at time point D1
vegfa_data_D7 <- subset(vegfa_data,
                        Measurement %in% c("D7"))

vegfa_data_D7$group <- interaction(vegfa_data_D7$Type, vegfa_data_D7$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfa_data_D7)

# Fit the model
fit <- lmFit(vegfa_data_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
  levels = design
)
```

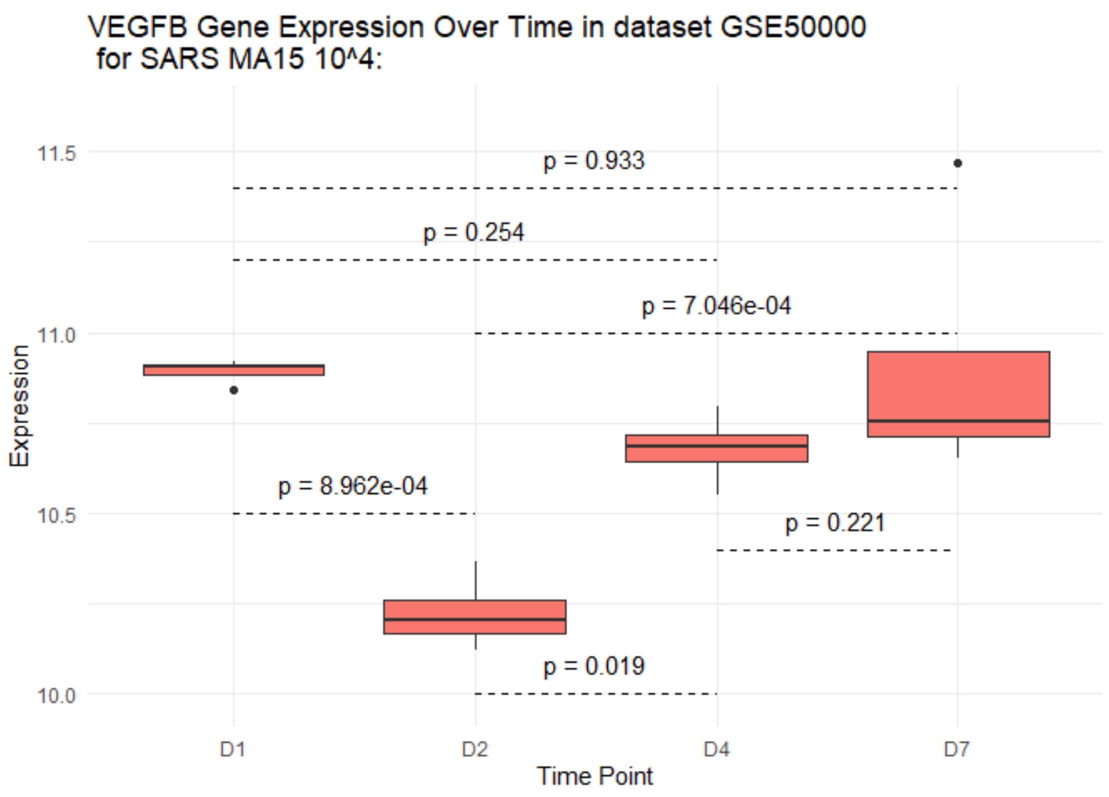
```
# Fit the contrasts
fit1_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day7 <- eBayes(fit1_Day7)

# Get the top table
results_D7 <- topTable(fit1_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##       logFC  AveExpr        t    P.Value  adj.P.Val         B
## 1 0.3707333 12.80863 1.905653 0.09315311 0.09315311 -4.175564
vegfa_data1 <- subset(vegfa_data,
                      Type %in% c("MA_10_4", "MA_10_5"))

tnf_data_WT_MA15 <- vegfa_data1

P.Value1 <- results$Vegfa$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "Differential Expression of VEGFA Gene Over Time in Response \nto Different
Doses of SARS MA15 [GSE50000]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^4 MA15", "10^5 MA15"))
+
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = Inf, y = 11.7, label = paste0("p = ", format(round(P.Value1, 10),
scientific = TRUE)), hjust = 1, vjust = 1, size = 4, fontface = "bold.italic") +

  annotate("text", x = 1.3, y = 13.1, label = paste0("p = ", round(P.Value2, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 2.3, y = 12.5, label = paste0("p = ", round(P.Value3, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 3.3, y = 12.3, label = paste0("p = ", round(P.Value4, 4)), hjust = 1,
vjust = 1) +
  annotate("text", x = 4.3, y = 12.2, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)


print(p)
```



VEGFB:

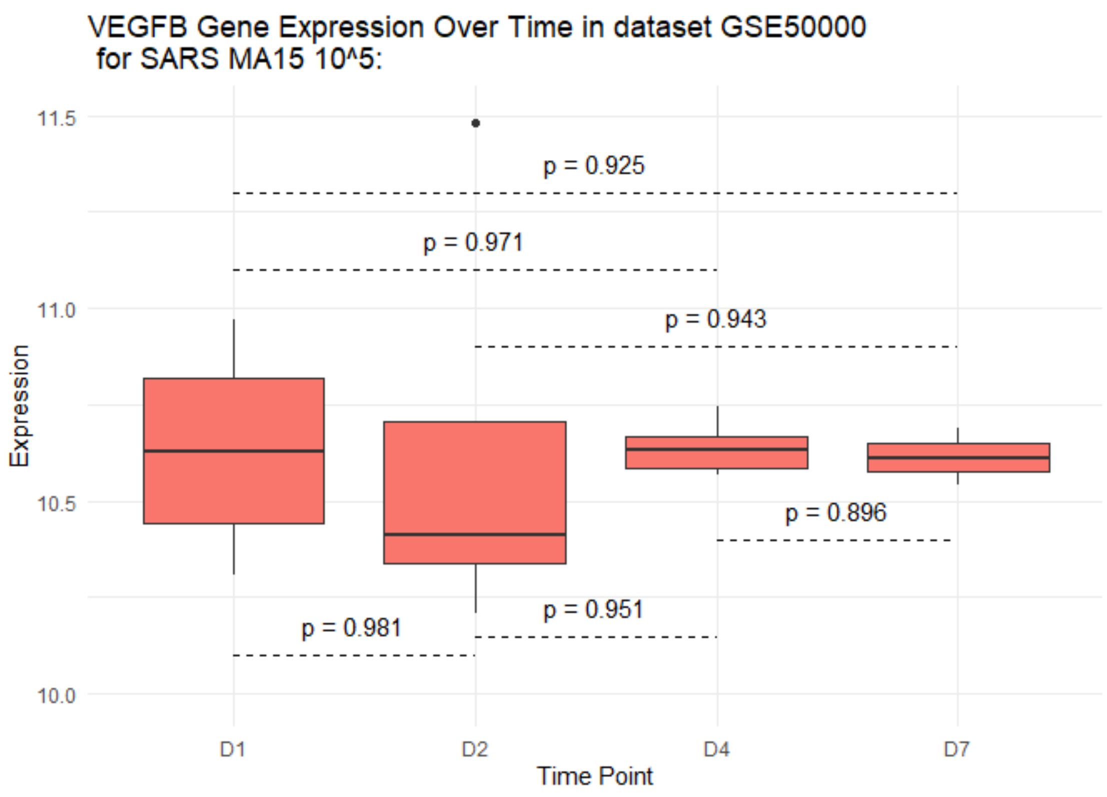
```
# Define a function to format p-values
format_pvalue <- function(pvalue) {
  if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 4))
  } else {
    return(round(pvalue, 3))
  }
}

vegfb_data <- subset(final,
                    GENE_SYMBOL %in% c("Vegfb"))


tnf_data_MA_10_4 <- vegfb_data[vegfb_data$Type == "MA_10_4",]

# Extract the p-values for MA_10_4
pvalues1_MA_10_4_D2_D1 <- results$Vegfb$P.Value_D1.vs.D2_MA_10_4
pvalues1_MA_10_4_D4_D2 <- results$Vegfb$P.Value_D2.vs.D4_MA_10_4
pvalues1_MA_10_4_D4_D1 <- results$Vegfb$P.Value_D1.vs.D4_MA_10_4
pvalues1_MA_10_4_D7_D4 <- results$Vegfb$P.Value_D4.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D2 <- results$Vegfb$P.Value_D2.vs.D7_MA_10_4
pvalues1_MA_10_4_D7_D1 <- results$Vegfb$P.Value_D1.vs.D7_MA_10_4
```

```r
# Create a new data frame for the annotations for MA_10_4
annotations_MA_10_4 <- data.frame(
  Type = "MA_10_4",
  Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
  Label = paste("p =", sapply(c(pvalues1_MA_10_4_D2_D1, pvalues1_MA_10_4_D4_D2,
pvalues1_MA_10_4_D4_D1, pvalues1_MA_10_4_D7_D4, pvalues1_MA_10_4_D7_D2,
pvalues1_MA_10_4_D7_D1), format_pvalue)),
  Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)

# Create the line chart for MA_10_4
p_MA_10_4 <- ggplot(data = tnf_data_MA_10_4, aes(x = Measurement, y = Expression, fill
= Type)) +
  geom_boxplot() +
  labs(title = "VEGFB Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^4:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(9.99, 11.6) +
  theme(legend.position = "none")

# Add lines for MA_10_4
p_MA_10_4 <- p_MA_10_4 +
  geom_segment(aes(x = 1, y = 10.5, xend = 2, yend = 10.5), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 10, xend = 3, yend = 10), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11.2, xend = 3, yend = 11.2), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 3, y = 10.4, xend = 4, yend = 10.4), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 11, xend = 4, yend = 11), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11.4, xend = 4, yend = 11.4), linetype = "dashed", color =
"black")

# Add p-values for MA_10_4
p_MA_10_4 <- p_MA_10_4 + annotate("text", x = c(1.5, 2.5, 2, 3.5, 3, 2.5), y = c(10.5, 10,
11.2, 10.4, 11, 11.4), label = annotations_MA_10_4$Label, vjust = -1)

print(p_MA_10_4)
```



```
# Define a function to format p-values
format_pvalue <- function(pvalue) {
  if (pvalue < 0.001) {
    return(format(pvalue, scientific = TRUE, digits = 3))
  } else {
    return(round(pvalue, 3))
  }
}


tnf_data_MA_10_5 <- vegfb_data[vegfb_data$Type == "MA_10_5",]

# Extract the p-values for MA_10_5
pvalues1_MA_10_5_D2_D1 <- results$Vegfb$P.Value_D1.vs.D2_MA_10_5
pvalues1_MA_10_5_D4_D2 <- results$Vegfb$P.Value_D2.vs.D4_MA_10_5
pvalues1_MA_10_5_D4_D1 <- results$Vegfb$P.Value_D1.vs.D4_MA_10_5
pvalues1_MA_10_5_D7_D4 <- results$Vegfb$P.Value_D4.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D2 <- results$Vegfb$P.Value_D2.vs.D7_MA_10_5
pvalues1_MA_10_5_D7_D1 <- results$Vegfb$P.Value_D1.vs.D7_MA_10_5

# Create a new data frame for the annotations for MA_10_5
annotations_MA_10_5 <- data.frame(
  Type = "MA_10_4",
  Measurement = c("D2", "D4", "D4", "D7", "D7", "D7"),
  Comparison = c("D1", "D2", "D1", "D4", "D2", "D1"),
  Label = paste("p =", sapply(c(pvalues1_MA_10_5_D2_D1, pvalues1_MA_10_5_D4_D2,
pvalues1_MA_10_5_D4_D1, pvalues1_MA_10_5_D7_D4, pvalues1_MA_10_5_D7_D2,
pvalues1_MA_10_5_D7_D1), format_pvalue)),
  Y = c(8, 9, 10, 11, 12, 13)  # Different y-coordinates for each p-value
)
```

```
# Create the line chart for MA_10_4
p_MA_10_5 <- ggplot(data = tnf_data_MA_10_5, aes(x = Measurement, y = Expression, fill
= Type)) +
  geom_boxplot() +
  labs(title = "VEGFB Gene Expression Over Time in dataset GSE50000 \n for SARS MA15
10^5:",
       x = "Time Point",
       y = "Expression") +
  theme_minimal() +
  ylim(9.99, 11.5) +
  theme(legend.position = "none")

# Add lines for MA_10_5
p_MA_10_5 <- p_MA_10_5 +
  geom_segment(aes(x = 1, y = 10.1, xend = 2, yend = 10.1), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 10.15, xend = 3, yend = 10.15), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11.1, xend = 3, yend = 11.1), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 3, y = 10.4, xend = 4, yend = 10.4), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 2, y = 10.9, xend = 4, yend = 10.9), linetype = "dashed", color =
"black") +
  geom_segment(aes(x = 1, y = 11.3, xend = 4, yend = 11.3), linetype = "dashed", color =
"black")


# Add p-values for MA_10_4
p_MA_10_5 <- p_MA_10_5 + annotate("text", x = c(1.5, 2.5, 2, 3.5, 3, 2.5), y = c(10.1,
10.15, 11.1, 10.4, 10.9, 11.3), label = annotations_MA_10_5$Label, vjust = -1)

print(p_MA_10_5)
```



Between different Time points:

Day 1:

```
vegfb_data$Measurement <- as.character(vegfb_data$Measurement)

# Subset the data to include only observations at time point D1
vegfb_data_D1 <- subset(vegfb_data,
                Measurement %in% c("D1"))

vegfb_data_D1$group <- interaction(vegfb_data_D1$Type, vegfb_data_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfb_data_D1)

# Fit the model
fit <- lmFit(vegfb_data_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit1_Day1 <- contrasts.fit(fit, cont.matrix)
```

```
# Compute differential expression statistics
fit1_Day1 <- eBayes(fit1_Day1)

# Get the top table
results_D1 <- topTable(fit1_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##      logFC  AveExpr        t    P.Value  adj.P.Val         B
## 1 0.258875 10.95767 2.023753 0.07367411 0.07367411 -4.044699
```

Day 2:

```
vegfb_data$Measurement <- as.character(vegfb_data$Measurement)

# Subset the data to include only observations at time point D1
vegfb_data_D2 <- subset(vegfb_data,
                        Measurement %in% c("D2"))

vegfb_data_D2$group <- interaction(vegfb_data_D2$Type, vegfb_data_D2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfb_data_D2)

# Fit the model
fit <- lmFit(vegfb_data_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
  levels = design
)

# Fit the contrasts
fit1_Day2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day2 <- eBayes(fit1_Day2)

# Get the top table
results_D2 <- topTable(fit1_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##        logFC  AveExpr         t   P.Value adj.P.Val         B
## 1 -0.4049875 10.76952 -1.653414 0.1326358 0.1326358 -4.391601
```

Day 4:

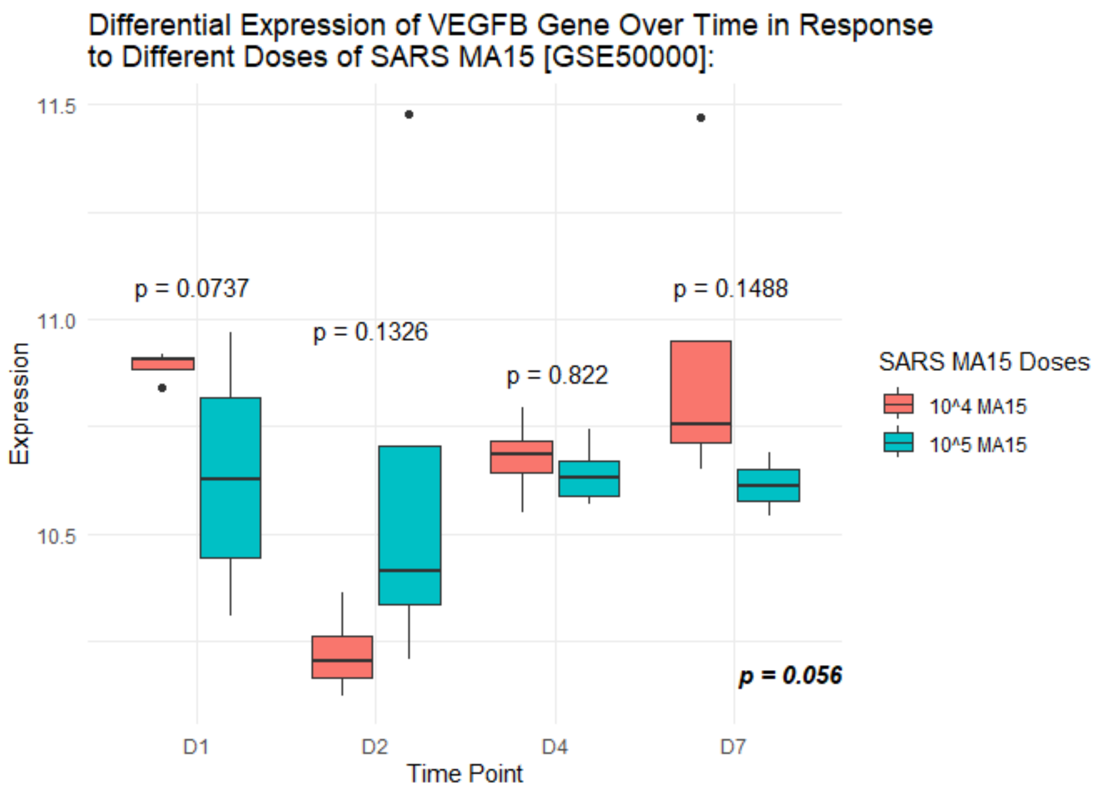
```
vegfb_data$Measurement <- as.character(vegfb_data$Measurement)

# Subset the data to include only observations at time point D1
```

```
vegfb_data_D4 <- subset(vegfb_data,
                        Measurement %in% c("D4"))

vegfb_data_D4$group <- interaction(vegfb_data_D4$Type, vegfb_data_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfb_data_D4)

# Fit the model
fit <- lmFit(vegfb_data_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
  levels = design
)

# Fit the contrasts
fit1_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day4 <- eBayes(fit1_Day4)

# Get the top table
results_D4 <- topTable(fit1_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##       logFC  AveExpr         t   P.Value adj.P.Val        B
## 1 0.0384575 10.85036 0.2309333 0.8220229 0.8220229 -4.74134
```

Day 7:

```
vegfb_data$Measurement <- as.character(vegfb_data$Measurement)

# Subset the data to include only observations at time point D1
vegfb_data_D7 <- subset(vegfb_data,
                        Measurement %in% c("D7"))

vegfb_data_D7$group <- interaction(vegfb_data_D7$Type, vegfb_data_D7$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = vegfb_data_D7)

# Fit the model
fit <- lmFit(vegfb_data_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
  levels = design
)
```

```
# Fit the contrasts
fit1_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit1_Day7 <- eBayes(fit1_Day7)

# Get the top table
results_D7 <- topTable(fit1_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##       logFC AveExpr       t   P.Value adj.P.Val         B
## 1 0.2933458 11.0113 1.59782 0.1487502 0.1487502 -4.437253
vegfb_data1 <- subset(vegfb_data,
                      Type %in% c("MA_10_4", "MA_10_5"))

tnf_data_WT_MA15 <- vegfb_data1

P.Value1 <- results$Vegfb$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "Differential Expression of VEGFB Gene Over Time in Response \nto Different
Doses of SARS MA15 [GSE50000]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^4 MA15", "10^5 MA15"))
+
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = Inf, y = 10.2, label = paste0("p = ", format(round(P.Value1, 3),
scientific = FALSE)), hjust = 1, vjust = 1, size = 4, fontface = "bold.italic") +

  annotate("text", x = 1.3, y = 11.1, label = paste0("p = ", round(P.Value2, 4)), hjust = 1,
vjust = 1) +
  annotate("text", x = 2.3, y = 11, label = paste0("p = ", round(P.Value3, 4)), hjust = 1, vjust
= 1) +
  annotate("text", x = 3.3, y = 10.9, label = paste0("p = ", round(P.Value4, 4)), hjust = 1,
vjust = 1) +
  annotate("text", x = 4.3, y = 11.1, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)


print(p)
```



```
#=====================================================================
==============================
```

Figure 6 Table 8.9, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16

##GSE33266:

```
expr_data2 <- read.csv("Mus_SARS_GSE33266_quant_normalised.csv", header = TRUE,
stringsAsFactors = T, na.strings = c("","NA"))

attach(expr_data2)
setnames(expr_data2, "qlucore", "GENE_SYMBOL")
setnames(expr_data2, "gedata", "noname")

setnames(expr_data2, "X.12", "MA_10_2.D1")
setnames(expr_data2, "X.13", "MA_10_2.D1")
setnames(expr_data2, "X.14", "MA_10_2.D1")
setnames(expr_data2, "X.15", "MA_10_2.D1")
setnames(expr_data2, "X.16", "MA_10_2.D1")

setnames(expr_data2, "X.17", "MA_10_2.D2")
setnames(expr_data2, "X.18", "MA_10_2.D2")
setnames(expr_data2, "X.19", "MA_10_2.D2")
setnames(expr_data2, "X.20", "MA_10_2.D2")
setnames(expr_data2, "X.21", "MA_10_2.D2")

setnames(expr_data2, "X.22", "MA_10_2.D4")
setnames(expr_data2, "X.23", "MA_10_2.D4")
setnames(expr_data2, "X.24", "MA_10_2.D4")
setnames(expr_data2, "X.25", "MA_10_2.D4")
setnames(expr_data2, "X.26", "MA_10_2.D4")

setnames(expr_data2, "X.27", "MA_10_2.D7")
setnames(expr_data2, "X.28", "MA_10_2.D7")
setnames(expr_data2, "X.29", "MA_10_2.D7")
setnames(expr_data2, "X.30", "MA_10_2.D7")
setnames(expr_data2, "X.31", "MA_10_2.D7")

#---------------------------------------------------------

setnames(expr_data2, "X.32", "MA_10_3.D1")
setnames(expr_data2, "X.33", "MA_10_3.D1")
setnames(expr_data2, "X.34", "MA_10_3.D1")
setnames(expr_data2, "X.35", "MA_10_3.D1")
setnames(expr_data2, "X.36", "MA_10_3.D1")

setnames(expr_data2, "X.37", "MA_10_3.D2")
setnames(expr_data2, "X.38", "MA_10_3.D2")
setnames(expr_data2, "X.39", "MA_10_3.D2")
setnames(expr_data2, "X.40", "MA_10_3.D2")
setnames(expr_data2, "X.41", "MA_10_3.D2")

setnames(expr_data2, "X.42", "MA_10_3.D4")
setnames(expr_data2, "X.43", "MA_10_3.D4")
setnames(expr_data2, "X.44", "MA_10_3.D4")
setnames(expr_data2, "X.45", "MA_10_3.D4")
setnames(expr_data2, "X.46", "MA_10_3.D4")

setnames(expr_data2, "X.47", "MA_10_3.D7")
setnames(expr_data2, "X.48", "MA_10_3.D7")
setnames(expr_data2, "X.49", "MA_10_3.D7")
setnames(expr_data2, "X.50", "MA_10_3.D7")
setnames(expr_data2, "X.51", "MA_10_3.D7")

#---------------------------------------------------------

setnames(expr_data2, "X.52", "MA_10_4.D1")
setnames(expr_data2, "X.53", "MA_10_4.D1")
setnames(expr_data2, "X.54", "MA_10_4.D1")
setnames(expr_data2, "X.55", "MA_10_4.D1")
setnames(expr_data2, "X.56", "MA_10_4.D1")

setnames(expr_data2, "X.57", "MA_10_4.D2")
setnames(expr_data2, "X.58", "MA_10_4.D2")
setnames(expr_data2, "X.59", "MA_10_4.D2")
setnames(expr_data2, "X.60", "MA_10_4.D2")
setnames(expr_data2, "X.61", "MA_10_4.D2")

setnames(expr_data2, "X.62", "MA_10_4.D4")
setnames(expr_data2, "X.63", "MA_10_4.D4")
setnames(expr_data2, "X.64", "MA_10_4.D4")
setnames(expr_data2, "X.65", "MA_10_4.D4")
setnames(expr_data2, "X.66", "MA_10_4.D4")

setnames(expr_data2, "X.67", "MA_10_4.D7")
setnames(expr_data2, "X.68", "MA_10_4.D7")
setnames(expr_data2, "X.69", "MA_10_4.D7")
setnames(expr_data2, "X.70", "MA_10_4.D7")
setnames(expr_data2, "X.71", "MA_10_4.D7")

#----------------------------------------------------------

setnames(expr_data2, "X.72", "MA_10_5.D1")
setnames(expr_data2, "X.73", "MA_10_5.D1")
setnames(expr_data2, "X.74", "MA_10_5.D1")
setnames(expr_data2, "X.75", "MA_10_5.D1")
setnames(expr_data2, "X.76", "MA_10_5.D1")

setnames(expr_data2, "X.77", "MA_10_5.D2")
setnames(expr_data2, "X.78", "MA_10_5.D2")
setnames(expr_data2, "X.79", "MA_10_5.D2")
setnames(expr_data2, "X.80", "MA_10_5.D2")
setnames(expr_data2, "X.81", "MA_10_5.D2")

setnames(expr_data2, "X.82", "MA_10_5.D4")
setnames(expr_data2, "X.83", "MA_10_5.D4")
setnames(expr_data2, "X.84", "MA_10_5.D4")
setnames(expr_data2, "X.85", "MA_10_5.D4")
setnames(expr_data2, "X.86", "MA_10_5.D4")

setnames(expr_data2, "X.87", "MA_10_5.D7")
setnames(expr_data2, "X.88", "MA_10_5.D7")
setnames(expr_data2, "X.89", "MA_10_5.D7")
setnames(expr_data2, "X.90", "MA_10_5.D7")
setnames(expr_data2, "X.91", "MA_10_5.D7")


expr_data2_1 <- expr_data2[13:45030, c(1, 2, 16:95)]


write.csv(expr_data2_1, file = "GSE33266_dataset.csv")


# Read the dataset
data <- read.csv("GSE33266_dataset.csv", stringsAsFactors = FALSE)
data_long <- data %>%
  pivot_longer(cols = -c(X, GENE_SYMBOL, noname), names_to = "Type_Measurement",
values_to = "Expression") %>%
  separate(Type_Measurement, into = c("Type", "Measurement"), sep = "\\.")

# Filter for genes of interest
genes_of_interest <- c("Tnf", "Nfkb1", "Vegfa", "Vegfb")
long_data_filtered <- data_long[data_long$GENE_SYMBOL %in% genes_of_interest, ]

long_data_filtered <- long_data_filtered[,2:6]
long_data_filtered <- data.frame(long_data_filtered)
long_data_filtered$GENE_SYMBOL <- as.character(long_data_filtered$GENE_SYMBOL)

# Subset the data to include only observations at time point D1
tnf_data2 <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Tnf"))

probe1 <- as.data.frame(subset(tnf_data2, noname == "2274"))
probe2 <- as.data.frame(subset(tnf_data2, noname == "3129"))
probe3 <- as.data.frame(subset(tnf_data2, noname == "5422"))
probe4 <- as.data.frame(subset(tnf_data2, noname == "7837"))
probe5 <- as.data.frame(subset(tnf_data2, noname == "16548"))
probe6 <- as.data.frame(subset(tnf_data2, noname == "28397"))
probe7 <- as.data.frame(subset(tnf_data2, noname == "28984"))
probe8 <- as.data.frame(subset(tnf_data2, noname == "29213"))
probe9 <- as.data.frame(subset(tnf_data2, noname == "32970"))
probe10 <- as.data.frame(subset(tnf_data2, noname == "35950"))

tnf_data2$Expression <- as.numeric(as.character(tnf_data2$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression, probe5$Expression, probe6$Expression, probe7$Expression,
probe8$Expression, probe9$Expression, probe10$Expression)

setnames(combined_dataset, "Expression", "Expression1")

combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression5" =
`probe5$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression6" =
`probe6$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression7" =
`probe7$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression8" =
`probe8$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression9" =
`probe9$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression10" =
`probe10$Expression`)
```

```
combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))
combined_dataset$Expression5 <-
as.numeric(as.character(combined_dataset$Expression5))
combined_dataset$Expression6 <-
as.numeric(as.character(combined_dataset$Expression6))
combined_dataset$Expression7 <-
as.numeric(as.character(combined_dataset$Expression7))
combined_dataset$Expression8 <-
as.numeric(as.character(combined_dataset$Expression8))
combined_dataset$Expression9 <-
as.numeric(as.character(combined_dataset$Expression9))
combined_dataset$Expression10 <-
as.numeric(as.character(combined_dataset$Expression10))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4", "Expression5", "Expression6", "Expression7",
"Expression8", "Expression9", "Expression10")])

tnf_data2 <- combined_dataset[,c(1, 3, 4, 15)]
nfkb1_data2 <- subset(long_data_filtered,
                    GENE_SYMBOL %in% c("Nfkb1"))

probe1 <- as.data.frame(subset(nfkb1_data2, noname == "796"))
probe2 <- as.data.frame(subset(nfkb1_data2, noname == "1672"))
probe3 <- as.data.frame(subset(nfkb1_data2, noname == "6236"))
probe4 <- as.data.frame(subset(nfkb1_data2, noname == "12456"))
probe5 <- as.data.frame(subset(nfkb1_data2, noname == "20925"))
probe6 <- as.data.frame(subset(nfkb1_data2, noname == "23033"))
probe7 <- as.data.frame(subset(nfkb1_data2, noname == "24617"))
probe8 <- as.data.frame(subset(nfkb1_data2, noname == "24726"))
probe9 <- as.data.frame(subset(nfkb1_data2, noname == "27633"))
probe10 <- as.data.frame(subset(nfkb1_data2, noname == "31997"))
probe11 <- as.data.frame(subset(nfkb1_data2, noname == "33704"))
probe12 <- as.data.frame(subset(nfkb1_data2, noname == "36599"))
probe13 <- as.data.frame(subset(nfkb1_data2, noname == "37814"))

nfkb1_data2$Expression <- as.numeric(as.character(nfkb1_data2$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression, probe5$Expression, probe6$Expression, probe7$Expression,
probe8$Expression, probe9$Expression, probe10$Expression, probe11$Expression,
probe12$Expression, probe13$Expression)
```

```
setnames(combined_dataset, "Expression", "Expression1")

combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression5" =
`probe5$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression6" =
`probe6$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression7" =
`probe7$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression8" =
`probe8$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression9" =
`probe9$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression10" =
`probe10$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression11" =
`probe11$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression12" =
`probe12$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression13" =
`probe13$Expression`)

combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))
combined_dataset$Expression5 <-
as.numeric(as.character(combined_dataset$Expression5))
combined_dataset$Expression6 <-
as.numeric(as.character(combined_dataset$Expression6))
combined_dataset$Expression7 <-
as.numeric(as.character(combined_dataset$Expression7))
combined_dataset$Expression8 <-
as.numeric(as.character(combined_dataset$Expression8))
combined_dataset$Expression9 <-
as.numeric(as.character(combined_dataset$Expression9))
combined_dataset$Expression10 <-
as.numeric(as.character(combined_dataset$Expression10))

combined_dataset$Expression11 <-
as.numeric(as.character(combined_dataset$Expression11))
combined_dataset$Expression12 <-
```

```
as.numeric(as.character(combined_dataset$Expression12))
combined_dataset$Expression13 <-
as.numeric(as.character(combined_dataset$Expression13))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4", "Expression5", "Expression6", "Expression7",
"Expression8", "Expression9", "Expression10", "Expression11", "Expression12",
"Expression13")])

nfkb1_data2 <- combined_dataset[,c(1, 3, 4, 18)]
Vegfa_data2 <- subset(long_data_filtered,
                      GENE_SYMBOL %in% c("Vegfa"))

probe1 <- as.data.frame(subset(Vegfa_data2, noname == "9226"))
probe2 <- as.data.frame(subset(Vegfa_data2, noname == "10795"))
probe3 <- as.data.frame(subset(Vegfa_data2, noname == "13699"))
probe4 <- as.data.frame(subset(Vegfa_data2, noname == "18075"))
probe5 <- as.data.frame(subset(Vegfa_data2, noname == "19030"))
probe6 <- as.data.frame(subset(Vegfa_data2, noname == "25072"))
probe7 <- as.data.frame(subset(Vegfa_data2, noname == "28205"))
probe8 <- as.data.frame(subset(Vegfa_data2, noname == "30470"))
probe9 <- as.data.frame(subset(Vegfa_data2, noname == "30697"))
probe10 <- as.data.frame(subset(Vegfa_data2, noname == "38339"))
probe11 <- as.data.frame(subset(Vegfa_data2, noname == "41073"))
probe12 <- as.data.frame(subset(Vegfa_data2, noname == "41625"))


Vegfa_data2$Expression <- as.numeric(as.character(Vegfa_data2$Expression))

combined_dataset <- cbind(probe1, probe2$Expression, probe3$Expression,
probe4$Expression, probe5$Expression, probe6$Expression, probe7$Expression,
probe8$Expression, probe9$Expression, probe10$Expression, probe11$Expression,
probe12$Expression)

setnames(combined_dataset, "Expression", "Expression1")

combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression3" =
`probe3$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression4" =
`probe4$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression5" =
`probe5$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression6" =
`probe6$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression7" =
`probe7$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression8" =
`probe8$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression9" =
`probe9$Expression`)
```

```
combined_dataset <- dplyr::rename(combined_dataset, "Expression10" =
`probe10$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression11" =
`probe11$Expression`)
combined_dataset <- dplyr::rename(combined_dataset, "Expression12" =
`probe12$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

combined_dataset$Expression3 <-
as.numeric(as.character(combined_dataset$Expression3))
combined_dataset$Expression4 <-
as.numeric(as.character(combined_dataset$Expression4))
combined_dataset$Expression5 <-
as.numeric(as.character(combined_dataset$Expression5))
combined_dataset$Expression6 <-
as.numeric(as.character(combined_dataset$Expression6))
combined_dataset$Expression7 <-
as.numeric(as.character(combined_dataset$Expression7))
combined_dataset$Expression8 <-
as.numeric(as.character(combined_dataset$Expression8))
combined_dataset$Expression9 <-
as.numeric(as.character(combined_dataset$Expression9))
combined_dataset$Expression10 <-
as.numeric(as.character(combined_dataset$Expression10))

combined_dataset$Expression11 <-
as.numeric(as.character(combined_dataset$Expression11))
combined_dataset$Expression12 <-
as.numeric(as.character(combined_dataset$Expression12))


# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2", "Expression3", "Expression4", "Expression5", "Expression6", "Expression7",
"Expression8", "Expression9", "Expression10", "Expression11", "Expression12")])

Vegfa_data2 <- combined_dataset[,c(1, 3, 4, 17)]
Vegfb_data2 <- subset(long_data_filtered,
                     GENE_SYMBOL %in% c("Vegfb"))

probe1 <- as.data.frame(subset(Vegfb_data2, noname == "22434"))
probe2 <- as.data.frame(subset(Vegfb_data2, noname == "42717"))


Vegfb_data2$Expression <- as.numeric(as.character(Vegfb_data2$Expression))
```

```r
combined_dataset <- cbind(probe1, probe2$Expression)

setnames(combined_dataset, "Expression", "Expression1")

combined_dataset <- dplyr::rename(combined_dataset, "Expression2" =
`probe2$Expression`)


combined_dataset$Expression1 <-
as.numeric(as.character(combined_dataset$Expression1))

combined_dataset$Expression2 <-
as.numeric(as.character(combined_dataset$Expression2))

# Calculate the average of each row
combined_dataset$Expression <- rowMeans(combined_dataset[, c("Expression1",
"Expression2")])

Vegfb_data2 <- combined_dataset[,c(1, 3, 4, 7)]
# Convert necessary columns to appropriate types
tnf_data2$Type <- as.factor(tnf_data2$Type)
tnf_data2$Measurement <- as.factor(tnf_data2$Measurement)
tnf_data2$Expression <- as.numeric(as.character(tnf_data2$Expression))

design <- model.matrix(~0 + Type, data = tnf_data2)

# Fit a linear model
fit <- lmFit(tnf_data2$Expression, design)

# Contrast matrix
contrast_matrix <- makeContrasts(
  MA10_2.vs.MA10_3 = TypeMA_10_2 - TypeMA_10_3,
  MA10_2.vs.MA10_4 = TypeMA_10_2 - TypeMA_10_4,
  MA10_2.vs.MA10_5 = TypeMA_10_2 - TypeMA_10_5,
  MA10_3.vs.MA10_4 = TypeMA_10_3 - TypeMA_10_4,
  MA10_3.vs.MA10_5 = TypeMA_10_3 - TypeMA_10_5,
  MA10_4.vs.MA10_5 = TypeMA_10_4 - TypeMA_10_5,
  levels = colnames(design)
)

# eBayes
fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

# Extract results for the "TNF" gene for each contrast
TNF_results1 <- topTable(fit, coef = "MA10_2.vs.MA10_3", number = Inf)
TNF_results1$adj.P.Val <- p.adjust(TNF_results1$P.Value, method = "bonferroni")
TNF_results1$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^3"

TNF_results2 <- topTable(fit, coef = "MA10_2.vs.MA10_4", number = Inf)
TNF_results2$adj.P.Val <- p.adjust(TNF_results2$P.Value, method = "bonferroni")
TNF_results2$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^4"
```

```
TNF_results3 <- topTable(fit, coef = "MA10_2.vs.MA10_5", number = Inf)
TNF_results3$adj.P.Val <- p.adjust(TNF_results3$P.Value, method = "bonferroni")
TNF_results3$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^5"

TNF_results4 <- topTable(fit, coef = "MA10_3.vs.MA10_4", number = Inf)
TNF_results4$adj.P.Val <- p.adjust(TNF_results4$P.Value, method = "bonferroni")
TNF_results4$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^4"

TNF_results5 <- topTable(fit, coef = "MA10_3.vs.MA10_5", number = Inf)
TNF_results5$adj.P.Val <- p.adjust(TNF_results5$P.Value, method = "bonferroni")
TNF_results5$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^5"

TNF_results6 <- topTable(fit, coef = "MA10_4.vs.MA10_5", number = Inf)
TNF_results6$adj.P.Val <- p.adjust(TNF_results6$P.Value, method = "bonferroni")
TNF_results6$Contrast <- "SARS MA15 10^4 vs SARS MA15 10^5"

TNF_results_combined <- rbind(TNF_results1, TNF_results2, TNF_results3, TNF_results4,
TNF_results5, TNF_results6)

# Print the combined results
print(TNF_results_combined)
##        logFC  AveExpr          t    P.Value  adj.P.Val         B
## 1 -0.9933370 9.597629 -2.5058643 0.01435376 0.01435376 -3.078344
## 2 -1.1156950 9.597629 -2.8145335 0.00621676 0.00621676 -2.433639
## 3 -0.8391065 9.597629 -2.1167912 0.03755030 0.03755030 -3.760520
## 4 -0.1223580 9.597629 -0.3086692 0.75841770 0.75841770 -4.623084
## 5  0.1542305 9.597629  0.3890731 0.69831006 0.69831006 -4.621388
## 6  0.2765885 9.597629  0.6977423 0.48746758 0.48746758 -4.611303
##                           Contrast
## 1 SARS MA15 10^2 vs SARS MA15 10^3
## 2 SARS MA15 10^2 vs SARS MA15 10^4
## 3 SARS MA15 10^2 vs SARS MA15 10^5
## 4 SARS MA15 10^3 vs SARS MA15 10^4
## 5 SARS MA15 10^3 vs SARS MA15 10^5
## 6 SARS MA15 10^4 vs SARS MA15 10^5
```

between each dose:

```
# Create a new variable for the interaction of Type and Measurement
tnf_data2$group <- interaction(tnf_data2$Type, tnf_data2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data2)

# Fit the model
fit <- lmFit(tnf_data2$Expression, design)

cont.matrix <- makeContrasts(
  # For group MA_10_2
  MA_10_2_D2_D1 = groupMA_10_2.D2 - groupMA_10_2.D1,
  MA_10_2_D4_D2 = groupMA_10_2.D4 - groupMA_10_2.D2,
  MA_10_2_D4_D1 = groupMA_10_2.D4 - groupMA_10_2.D1,
  MA_10_2_D7_D4 = groupMA_10_2.D7 - groupMA_10_2.D4,
  MA_10_2_D7_D2 = groupMA_10_2.D7 - groupMA_10_2.D2,
  MA_10_2_D7_D1 = groupMA_10_2.D7 - groupMA_10_2.D1,

  # For group MA_10_3
  MA_10_3_D2_D1 = groupMA_10_3.D2 - groupMA_10_3.D1,
  MA_10_3_D4_D2 = groupMA_10_3.D4 - groupMA_10_3.D2,
  MA_10_3_D4_D1 = groupMA_10_3.D4 - groupMA_10_3.D1,
  MA_10_3_D7_D4 = groupMA_10_3.D7 - groupMA_10_3.D4,
  MA_10_3_D7_D2 = groupMA_10_3.D7 - groupMA_10_3.D2,
  MA_10_3_D7_D1 = groupMA_10_3.D7 - groupMA_10_3.D1,

  # For group MA_10_4
  MA_10_4_D2_D1 = groupMA_10_4.D2 - groupMA_10_4.D1,
  MA_10_4_D4_D2 = groupMA_10_4.D4 - groupMA_10_4.D2,
  MA_10_4_D4_D1 = groupMA_10_4.D4 - groupMA_10_4.D1,
  MA_10_4_D7_D4 = groupMA_10_4.D7 - groupMA_10_4.D4,
  MA_10_4_D7_D2 = groupMA_10_4.D7 - groupMA_10_4.D2,
  MA_10_4_D7_D1 = groupMA_10_4.D7 - groupMA_10_4.D1,

  # For group MA_10_5
  MA_10_5_D2_D1 = groupMA_10_5.D2 - groupMA_10_5.D1,
  MA_10_5_D4_D2 = groupMA_10_5.D4 - groupMA_10_5.D2,
  MA_10_5_D4_D1 = groupMA_10_5.D4 - groupMA_10_5.D1,
  MA_10_5_D7_D4 = groupMA_10_5.D7 - groupMA_10_5.D4,
  MA_10_5_D7_D2 = groupMA_10_5.D7 - groupMA_10_5.D2,
  MA_10_5_D7_D1 = groupMA_10_5.D7 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2 <- eBayes(fit2)

# Get the top table
results <- topTable(fit2, number=Inf)
results$adj.P.Val <- p.adjust(results$P.Value, method = "bonferroni")

# Print the results
print(results)
##   MA_10_2_D2_D1 MA_10_2_D4_D2 MA_10_2_D4_D1 MA_10_2_D7_D4
MA_10_2_D7_D2
## 1       0.61989      0.609772      1.229662      0.311946      0.921718
##   MA_10_2_D7_D1 MA_10_3_D2_D1 MA_10_3_D4_D2 MA_10_3_D4_D1
MA_10_3_D7_D4
## 1      1.541608      3.684644     -1.045438      2.639206      0.050204
##   MA_10_3_D7_D2 MA_10_3_D7_D1 MA_10_4_D2_D1 MA_10_4_D4_D2
MA_10_4_D4_D1
## 1     -0.995234       2.68941      2.553522     -2.154048      0.399474
##   MA_10_4_D7_D4 MA_10_4_D7_D2 MA_10_4_D7_D1 MA_10_5_D2_D1
```

```
MA_10_5_D4_D2
## 1      0.776454     -1.377594      1.175928       0.74966     -1.460188
##   MA_10_5_D4_D1 MA_10_5_D7_D4 MA_10_5_D7_D2 MA_10_5_D7_D1  AveExpr
F
## 1     -0.710528     -0.150522      -1.61071      -0.86105 9.597629 7.961665
##        P.Value    adj.P.Val
## 1 7.537167e-09 7.537167e-09
results_long <- results %>%
  pivot_longer(cols = starts_with("MA"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_long)
## # A tibble: 24 × 6
##    AveExpr     F       P.Value     adj.P.Val Comparison      Value
##      <dbl> <dbl>         <dbl>         <dbl> <chr>           <dbl>
##  1    9.60  7.96 0.00000000754 0.00000000754 MA_10_2_D2_D1  0.620
##  2    9.60  7.96 0.00000000754 0.00000000754 MA_10_2_D4_D2  0.610
##  3    9.60  7.96 0.00000000754 0.00000000754 MA_10_2_D4_D1  1.23
##  4    9.60  7.96 0.00000000754 0.00000000754 MA_10_2_D7_D4  0.312
##  5    9.60  7.96 0.00000000754 0.00000000754 MA_10_2_D7_D2  0.922
##  6    9.60  7.96 0.00000000754 0.00000000754 MA_10_2_D7_D1  1.54
##  7    9.60  7.96 0.00000000754 0.00000000754 MA_10_3_D2_D1  3.68
##  8    9.60  7.96 0.00000000754 0.00000000754 MA_10_3_D4_D2 -1.05
##  9    9.60  7.96 0.00000000754 0.00000000754 MA_10_3_D4_D1  2.64
## 10    9.60  7.96 0.00000000754 0.00000000754 MA_10_3_D7_D4  0.0502
## # ℹ 14 more rows
```

Between different Time points:

Day 1:

```
tnf_data2$Measurement <- as.character(tnf_data2$Measurement)

# Subset the data to include only observations at time point D1
tnf_data2_D1 <- subset(tnf_data2,
                       Measurement %in% c("D1"))

tnf_data2_D1$group <- interaction(tnf_data2_D1$Type, tnf_data2_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data2_D1)

# Fit the model
fit <- lmFit(tnf_data2_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
    D1_MA_10_2_vs_MA_10_3 = groupMA_10_2.D1 - groupMA_10_3.D1,
    D1_MA_10_2_vs_MA_10_4 = groupMA_10_2.D1 - groupMA_10_4.D1,
    D1_MA_10_2_vs_MA_10_5 = groupMA_10_2.D1 - groupMA_10_5.D1,
  D1_MA_10_3_vs_MA_10_4 = groupMA_10_3.D1 - groupMA_10_4.D1,
  D1_MA_10_3_vs_MA_10_5 = groupMA_10_3.D1 - groupMA_10_5.D1,
  D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2_Day1 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day1 <- eBayes(fit2_Day1)

# Get the top table
results_D1 <- topTable(fit2_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##   D1_MA_10_2_vs_MA_10_3 D1_MA_10_2_vs_MA_10_4 D1_MA_10_2_vs_MA_10_5
## 1              0.412188             -0.931254             -1.892376
##   D1_MA_10_3_vs_MA_10_4 D1_MA_10_3_vs_MA_10_5 D1_MA_10_4_vs_MA_10_5
AveExpr
## 1             -1.343442             -2.304564             -0.961122 8.615664
##          F      P.Value    adj.P.Val
## 1 15.69025 5.038937e-05 5.038937e-05
results_D1_long <- results_D1 %>%
  pivot_longer(cols = starts_with("D1"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D1_long)
## # A tibble: 6 × 6
##   AveExpr     F   P.Value adj.P.Val Comparison            Value
##     <dbl> <dbl>     <dbl>     <dbl> <chr>                 <dbl>
## 1    8.62  15.7 0.0000504 0.0000504 D1_MA_10_2_vs_MA_10_3  0.412
## 2    8.62  15.7 0.0000504 0.0000504 D1_MA_10_2_vs_MA_10_4 -0.931
## 3    8.62  15.7 0.0000504 0.0000504 D1_MA_10_2_vs_MA_10_5 -1.89
## 4    8.62  15.7 0.0000504 0.0000504 D1_MA_10_3_vs_MA_10_4 -1.34
## 5    8.62  15.7 0.0000504 0.0000504 D1_MA_10_3_vs_MA_10_5 -2.30
## 6    8.62  15.7 0.0000504 0.0000504 D1_MA_10_4_vs_MA_10_5 -0.961
```

Day 2:

```
tnf_data2$Measurement <- as.character(tnf_data2$Measurement)

# Subset the data to include only observations at time point D1
tnf_data2_D2 <- subset(tnf_data2,
                       Measurement %in% c("D2"))

tnf_data2_D2$group <- interaction(tnf_data2_D2$Type, tnf_data2_D2$Measurement)

# Create the design matrix
```

```
design <- model.matrix(~0 + group, data = tnf_data2_D2)

# Fit the model
fit <- lmFit(tnf_data2_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
   D2_MA_10_2_vs_MA_10_3 = groupMA_10_2.D2 - groupMA_10_3.D2,
   D2_MA_10_2_vs_MA_10_4 = groupMA_10_2.D2 - groupMA_10_4.D2,
   D2_MA_10_2_vs_MA_10_5 = groupMA_10_2.D2 - groupMA_10_5.D2,
   D2_MA_10_3_vs_MA_10_4 = groupMA_10_3.D2 - groupMA_10_4.D2,
   D2_MA_10_3_vs_MA_10_5 = groupMA_10_3.D2 - groupMA_10_5.D2,
   D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
   levels = design
)

# Fit the contrasts
fit2_Day2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day2 <- eBayes(fit2_Day2)

# Get the top table
results_D2 <- topTable(fit2_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##   D2_MA_10_2_vs_MA_10_3 D2_MA_10_2_vs_MA_10_4 D2_MA_10_2_vs_MA_10_5
## 1             -2.652566             -2.864886             -2.022146
##   D2_MA_10_3_vs_MA_10_4 D2_MA_10_3_vs_MA_10_5 D2_MA_10_4_vs_MA_10_5
AveExpr
## 1              -0.21232               0.63042               0.84274 10.51759
##          F    P.Value  adj.P.Val
## 1 6.755965 0.00373215 0.00373215
results_D2_long <- results_D2 %>%
  pivot_longer(cols = starts_with("D2"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D2_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison            Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                 <dbl>
## 1    10.5  6.76 0.00373   0.00373 D2_MA_10_2_vs_MA_10_3 -2.65
## 2    10.5  6.76 0.00373   0.00373 D2_MA_10_2_vs_MA_10_4 -2.86
## 3    10.5  6.76 0.00373   0.00373 D2_MA_10_2_vs_MA_10_5 -2.02
## 4    10.5  6.76 0.00373   0.00373 D2_MA_10_3_vs_MA_10_4 -0.212
## 5    10.5  6.76 0.00373   0.00373 D2_MA_10_3_vs_MA_10_5  0.630
## 6    10.5  6.76 0.00373   0.00373 D2_MA_10_4_vs_MA_10_5  0.843
```

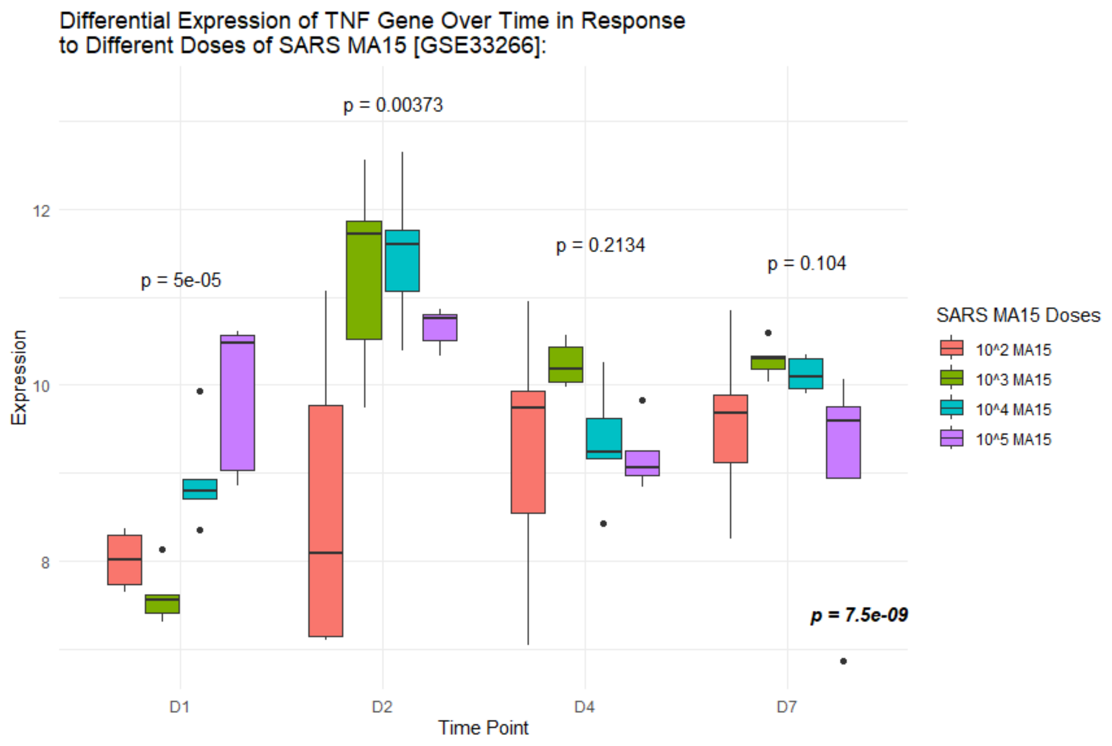
Day 4:

```
tnf_data2$Measurement <- as.character(tnf_data2$Measurement)

# Subset the data to include only observations at time point D1
tnf_data2_D4 <- subset(tnf_data2,
                       Measurement %in% c("D4"))

tnf_data2_D4$group <- interaction(tnf_data2_D4$Type, tnf_data2_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data2_D4)

# Fit the model
fit <- lmFit(tnf_data2_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D4_MA_10_2_vs_MA_10_3 = groupMA_10_2.D4 - groupMA_10_3.D4,
  D4_MA_10_2_vs_MA_10_4 = groupMA_10_2.D4 - groupMA_10_4.D4,
  D4_MA_10_2_vs_MA_10_5 = groupMA_10_2.D4 - groupMA_10_5.D4,
  D4_MA_10_3_vs_MA_10_4 = groupMA_10_3.D4 - groupMA_10_4.D4,
  D4_MA_10_3_vs_MA_10_5 = groupMA_10_3.D4 - groupMA_10_5.D4,
  D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
  levels = design
)

# Fit the contrasts
fit2_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day4 <- eBayes(fit2_Day4)

# Get the top table
results_D4 <- topTable(fit2_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##   D4_MA_10_2_vs_MA_10_3 D4_MA_10_2_vs_MA_10_4 D4_MA_10_2_vs_MA_10_5
## 1             -0.997356             -0.101066              0.047814
##   D4_MA_10_3_vs_MA_10_4 D4_MA_10_3_vs_MA_10_5 D4_MA_10_4_vs_MA_10_5
AveExpr
## 1               0.89629               1.04517               0.14888 9.505118
##         F   P.Value adj.P.Val
## 1 1.66951 0.2134447 0.2134447
results_D4_long <- results_D4 %>%
  pivot_longer(cols = starts_with("D4"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D4_long)
```

```
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison               Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                    <dbl>
## 1    9.51  1.67   0.213     0.213 D4_MA_10_2_vs_MA_10_3 -0.997
## 2    9.51  1.67   0.213     0.213 D4_MA_10_2_vs_MA_10_4 -0.101
## 3    9.51  1.67   0.213     0.213 D4_MA_10_2_vs_MA_10_5  0.0478
## 4    9.51  1.67   0.213     0.213 D4_MA_10_3_vs_MA_10_4  0.896
## 5    9.51  1.67   0.213     0.213 D4_MA_10_3_vs_MA_10_5  1.05
## 6    9.51  1.67   0.213     0.213 D4_MA_10_4_vs_MA_10_5  0.149
```

Day 7:

```
tnf_data2$Measurement <- as.character(tnf_data2$Measurement)

# Subset the data to include only observations at time point D1
tnf_data2_D7 <- subset(tnf_data2,
                       Measurement %in% c("D7"))

tnf_data2_D7$group <- interaction(tnf_data2_D7$Type, tnf_data2_D7$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = tnf_data2_D7)

# Fit the model
fit <- lmFit(tnf_data2_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D7_MA_10_2_vs_MA_10_3 = groupMA_10_2.D7 - groupMA_10_3.D7,
  D7_MA_10_2_vs_MA_10_4 = groupMA_10_2.D7 - groupMA_10_4.D7,
  D7_MA_10_2_vs_MA_10_5 = groupMA_10_2.D7 - groupMA_10_5.D7,
  D7_MA_10_3_vs_MA_10_4 = groupMA_10_3.D7 - groupMA_10_4.D7,
  D7_MA_10_3_vs_MA_10_5 = groupMA_10_3.D7 - groupMA_10_5.D7,
  D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
  levels = design
)

# Fit the contrasts
fit2_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day7 <- eBayes(fit2_Day7)

# Get the top table
results_D7 <- topTable(fit2_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##   D7_MA_10_2_vs_MA_10_3 D7_MA_10_2_vs_MA_10_4 D7_MA_10_2_vs_MA_10_5
## 1             -0.735614             -0.565574              0.510282
##   D7_MA_10_3_vs_MA_10_4 D7_MA_10_3_vs_MA_10_5 D7_MA_10_4_vs_MA_10_5
AveExpr
## 1               0.17004              1.245896              1.075856 9.752138
##          F  P.Value adj.P.Val
## 1 2.419937 0.103955  0.103955
results_D7_long <- results_D7 %>%
  pivot_longer(cols = starts_with("D7"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D7_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison             Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                  <dbl>
## 1    9.75  2.42   0.104     0.104 D7_MA_10_2_vs_MA_10_3 -0.736
## 2    9.75  2.42   0.104     0.104 D7_MA_10_2_vs_MA_10_4 -0.566
## 3    9.75  2.42   0.104     0.104 D7_MA_10_2_vs_MA_10_5  0.510
## 4    9.75  2.42   0.104     0.104 D7_MA_10_3_vs_MA_10_4  0.170
## 5    9.75  2.42   0.104     0.104 D7_MA_10_3_vs_MA_10_5  1.25
## 6    9.75  2.42   0.104     0.104 D7_MA_10_4_vs_MA_10_5  1.08
tnf_data_WT_MA15 <- tnf_data2

P.Value1 <- results$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "Differential Expression of TNF Gene Over Time in Response \nto Different
Doses of SARS MA15 [GSE33266]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^2 MA15", "10^3 MA15",
"10^4 MA15", "10^5 MA15")) +
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = Inf, y = 7.5, label = paste0("p = ", round(P.Value1, 10)), hjust =
1, vjust = 1, size = 4, fontface = "bold.italic") +
  annotate("text", x = 1.2, y = 11.3, label = paste0("p = ", round(P.Value2, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 2.3, y = 13.3, label = paste0("p = ", round(P.Value3, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 3.3, y = 11.7, label = paste0("p = ", round(P.Value4, 4)), hjust = 1,
vjust = 1) +
  annotate("text", x = 4.3, y = 11.5, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)

# Print the graph
print(p)
```



```
combined_results <- bind_rows(results_D1_long, results_D2_long, results_D4_long,
results_D7_long)

# Merge the data frames by columns
df2 <- merge(results_D1_long, results_D2_long, all = TRUE)
df2 <- merge(df2, results_D4_long, all = TRUE)
df2 <- merge(df2, results_D7_long, all = TRUE)

TNF_results_combined
```

logFC	AveExpr	t	P.Value	adj.P.Val	B	Contrast
-0.993	9.6	-2.51	0.0144	0.0144	-3.08	SARS MA15 10^2 vs SARS MA15 10^3
-1.12	9.6	-2.81	0.00622	0.00622	-2.43	SARS MA15 10^2 vs SARS MA15 10^4
-0.839	9.6	-2.12	0.0376	0.0376	-3.76	SARS MA15 10^2 vs SARS MA15 10^5
-0.122	9.6	-0.309	0.758	0.758	-4.62	SARS MA15 10^3 vs SARS MA15 10^4
0.154	9.6	0.389	0.698	0.698	-4.62	SARS MA15 10^3 vs SARS MA15 10^5
0.277	9.6	0.698	0.487	0.487	-4.61	SARS MA15 10^4 vs SARS MA15 10^5

```
##NFKB1:
# Convert necessary columns to appropriate types
```

```r
nfkb1_data2$Type <- as.factor(nfkb1_data2$Type)
nfkb1_data2$Measurement <- as.factor(nfkb1_data2$Measurement)
nfkb1_data2$Expression <- as.numeric(as.character(nfkb1_data2$Expression))

design <- model.matrix(~0 + Type, data = nfkb1_data2)

# Fit a linear model
fit <- lmFit(nfkb1_data2$Expression, design)

# Contrast matrix
contrast_matrix <- makeContrasts(
  MA10_2.vs.MA10_3 = TypeMA_10_2 - TypeMA_10_3,
  MA10_2.vs.MA10_4 = TypeMA_10_2 - TypeMA_10_4,
  MA10_2.vs.MA10_5 = TypeMA_10_2 - TypeMA_10_5,
  MA10_3.vs.MA10_4 = TypeMA_10_3 - TypeMA_10_4,
  MA10_3.vs.MA10_5 = TypeMA_10_3 - TypeMA_10_5,
  MA10_4.vs.MA10_5 = TypeMA_10_4 - TypeMA_10_5,
  levels = colnames(design)
)

# eBayes
fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

# Extract results for the "TNF" gene for each contrast
NFKB_results1 <- topTable(fit, coef = "MA10_2.vs.MA10_3", number = Inf)
NFKB_results1$adj.P.Val <- p.adjust(NFKB_results1$P.Value, method = "bonferroni")
NFKB_results1$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^3"

NFKB_results2 <- topTable(fit, coef = "MA10_2.vs.MA10_4", number = Inf)
NFKB_results2$adj.P.Val <- p.adjust(NFKB_results2$P.Value, method = "bonferroni")
NFKB_results2$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^4"

NFKB_results3 <- topTable(fit, coef = "MA10_2.vs.MA10_5", number = Inf)
NFKB_results3$adj.P.Val <- p.adjust(NFKB_results3$P.Value, method = "bonferroni")
NFKB_results3$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^5"

NFKB_results4 <- topTable(fit, coef = "MA10_3.vs.MA10_4", number = Inf)
NFKB_results4$adj.P.Val <- p.adjust(NFKB_results4$P.Value, method = "bonferroni")
NFKB_results4$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^4"

NFKB_results5 <- topTable(fit, coef = "MA10_3.vs.MA10_5", number = Inf)
NFKB_results5$adj.P.Val <- p.adjust(NFKB_results5$P.Value, method = "bonferroni")
NFKB_results5$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^5"

NFKB_results6 <- topTable(fit, coef = "MA10_4.vs.MA10_5", number = Inf)
NFKB_results6$adj.P.Val <- p.adjust(NFKB_results6$P.Value, method = "bonferroni")
NFKB_results6$Contrast <- "SARS MA15 10^4 vs SARS MA15 10^5"

NFKB_results_combined <- rbind(NFKB_results1, NFKB_results2, NFKB_results3,
NFKB_results4, NFKB_results5, NFKB_results6)

# Print the combined results
```

```
print(NFKB_results_combined)
##         logFC  AveExpr          t    P.Value  adj.P.Val         B
## 1 -0.105275769 11.55553 -1.2888266 0.20136772 0.20136772 -4.637436
## 2  0.009943846 11.55553  0.1217364 0.90342909 0.90342909 -5.048515
## 3  0.097363846 11.55553  1.1919658 0.23698338 0.23698338 -4.678616
## 4  0.115219615 11.55553  1.4105630 0.16245353 0.16245353 -4.565128
## 5  0.202639615 11.55553  2.4807925 0.01532007 0.01532007 -3.126834
## 6  0.087420000 11.55553  1.0702294 0.28790324 0.28790324 -4.708625
##                         Contrast
## 1 SARS MA15 10^2 vs SARS MA15 10^3
## 2 SARS MA15 10^2 vs SARS MA15 10^4
## 3 SARS MA15 10^2 vs SARS MA15 10^5
## 4 SARS MA15 10^3 vs SARS MA15 10^4
## 5 SARS MA15 10^3 vs SARS MA15 10^5
## 6 SARS MA15 10^4 vs SARS MA15 10^5
```

between each dose:

```
# Create a new variable for the interaction of Type and Measurement
nfkb1_data2$group <- interaction(nfkb1_data2$Type, nfkb1_data2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb1_data2)

# Fit the model
fit <- lmFit(nfkb1_data2$Expression, design)

cont.matrix <- makeContrasts(
  # For group MA_10_2
  MA_10_2_D2_D1 = groupMA_10_2.D2 - groupMA_10_2.D1,
  MA_10_2_D4_D2 = groupMA_10_2.D4 - groupMA_10_2.D2,
  MA_10_2_D4_D1 = groupMA_10_2.D4 - groupMA_10_2.D1,
  MA_10_2_D7_D4 = groupMA_10_2.D7 - groupMA_10_2.D4,
  MA_10_2_D7_D2 = groupMA_10_2.D7 - groupMA_10_2.D2,
  MA_10_2_D7_D1 = groupMA_10_2.D7 - groupMA_10_2.D1,

  # For group MA_10_3
  MA_10_3_D2_D1 = groupMA_10_3.D2 - groupMA_10_3.D1,
  MA_10_3_D4_D2 = groupMA_10_3.D4 - groupMA_10_3.D2,
  MA_10_3_D4_D1 = groupMA_10_3.D4 - groupMA_10_3.D1,
  MA_10_3_D7_D4 = groupMA_10_3.D7 - groupMA_10_3.D4,
  MA_10_3_D7_D2 = groupMA_10_3.D7 - groupMA_10_3.D2,
  MA_10_3_D7_D1 = groupMA_10_3.D7 - groupMA_10_3.D1,

  # For group MA_10_4
  MA_10_4_D2_D1 = groupMA_10_4.D2 - groupMA_10_4.D1,
  MA_10_4_D4_D2 = groupMA_10_4.D4 - groupMA_10_4.D2,
  MA_10_4_D4_D1 = groupMA_10_4.D4 - groupMA_10_4.D1,
  MA_10_4_D7_D4 = groupMA_10_4.D7 - groupMA_10_4.D4,
  MA_10_4_D7_D2 = groupMA_10_4.D7 - groupMA_10_4.D2,
  MA_10_4_D7_D1 = groupMA_10_4.D7 - groupMA_10_4.D1,

  # For group MA_10_5
  MA_10_5_D2_D1 = groupMA_10_5.D2 - groupMA_10_5.D1,
  MA_10_5_D4_D2 = groupMA_10_5.D4 - groupMA_10_5.D2,
  MA_10_5_D4_D1 = groupMA_10_5.D4 - groupMA_10_5.D1,
  MA_10_5_D7_D4 = groupMA_10_5.D7 - groupMA_10_5.D4,
  MA_10_5_D7_D2 = groupMA_10_5.D7 - groupMA_10_5.D2,
  MA_10_5_D7_D1 = groupMA_10_5.D7 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2 <- eBayes(fit2)

# Get the top table
results <- topTable(fit2, number=Inf)
results$adj.P.Val <- p.adjust(results$P.Value, method = "bonferroni")

# Print the results
print(results)
##   MA_10_2_D2_D1 MA_10_2_D4_D2 MA_10_2_D4_D1 MA_10_2_D7_D4
MA_10_2_D7_D2
## 1    0.06204308     0.2673646     0.3294077    -0.2675446      -0.00018
##   MA_10_2_D7_D1 MA_10_3_D2_D1 MA_10_3_D4_D2 MA_10_3_D4_D1
MA_10_3_D7_D4
## 1    0.06186308     0.3779646    -0.2777262     0.1002385   -0.03402462
##   MA_10_3_D7_D2 MA_10_3_D7_D1 MA_10_4_D2_D1 MA_10_4_D4_D2
MA_10_4_D4_D1
## 1    -0.3117508    0.06621385     0.6510215    -0.4265092     0.2245123
##   MA_10_4_D7_D4 MA_10_4_D7_D2 MA_10_4_D7_D1 MA_10_5_D2_D1
MA_10_5_D4_D2
## 1    0.03607692    -0.3904323     0.2605892   -0.03624615    -0.1673662
##   MA_10_5_D4_D1 MA_10_5_D7_D4 MA_10_5_D7_D2 MA_10_5_D7_D1  AveExpr
F
## 1    -0.2036123    0.05910154    -0.1082646    -0.1445108 11.55553 3.390731
##        P.Value    adj.P.Val
## 1 0.0007218288 0.0007218288
results_long <- results %>%
  pivot_longer(cols = starts_with("MA"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_long)
## # A tibble: 24 × 6
##    AveExpr     F  P.Value adj.P.Val Comparison        Value
##      <dbl> <dbl>    <dbl>     <dbl> <chr>             <dbl>
##  1    11.6  3.39 0.000722  0.000722 MA_10_2_D2_D1  0.0620
##  2    11.6  3.39 0.000722  0.000722 MA_10_2_D4_D2  0.267
##  3    11.6  3.39 0.000722  0.000722 MA_10_2_D4_D1  0.329
##  4    11.6  3.39 0.000722  0.000722 MA_10_2_D7_D4 -0.268
##  5    11.6  3.39 0.000722  0.000722 MA_10_2_D7_D2 -0.000180
##  6    11.6  3.39 0.000722  0.000722 MA_10_2_D7_D1  0.0619
##  7    11.6  3.39 0.000722  0.000722 MA_10_3_D2_D1  0.378
##  8    11.6  3.39 0.000722  0.000722 MA_10_3_D4_D2 -0.278
##  9    11.6  3.39 0.000722  0.000722 MA_10_3_D4_D1  0.100
## 10    11.6  3.39 0.000722  0.000722 MA_10_3_D7_D4 -0.0340
## # ℹ 14 more rows
```

Between different Time points:

Day 1:

```
nfkb1_data2$Measurement <- as.character(nfkb1_data2$Measurement)

# Subset the data to include only observations at time point D1
nfkb1_data2_D1 <- subset(nfkb1_data2,
                         Measurement %in% c("D1"))

nfkb1_data2_D1$group <- interaction(nfkb1_data2_D1$Type,
nfkb1_data2_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb1_data2_D1)

# Fit the model
fit <- lmFit(nfkb1_data2_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D1_MA_10_2_vs_MA_10_3 = groupMA_10_2.D1 - groupMA_10_3.D1,
  D1_MA_10_2_vs_MA_10_4 = groupMA_10_2.D1 - groupMA_10_4.D1,
  D1_MA_10_2_vs_MA_10_5 = groupMA_10_2.D1 - groupMA_10_5.D1,
  D1_MA_10_3_vs_MA_10_4 = groupMA_10_3.D1 - groupMA_10_4.D1,
  D1_MA_10_3_vs_MA_10_5 = groupMA_10_3.D1 - groupMA_10_5.D1,
  D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2_Day1 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day1 <- eBayes(fit2_Day1)

# Get the top table
results_D1 <- topTable(fit2_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##   D1_MA_10_2_vs_MA_10_3 D1_MA_10_2_vs_MA_10_4 D1_MA_10_2_vs_MA_10_5
## 1              -0.0825            0.1806462           -0.1120569
##   D1_MA_10_3_vs_MA_10_4 D1_MA_10_3_vs_MA_10_5 D1_MA_10_4_vs_MA_10_5
AveExpr
## 1             0.2631462           -0.02955692            -0.2927031 11.44619
##          F   P.Value adj.P.Val
## 1 2.362907 0.1096202 0.1096202
results_D1_long <- results_D1 %>%
  pivot_longer(cols = starts_with("D1"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D1_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison              Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                   <dbl>
## 1    11.4  2.36   0.110     0.110 D1_MA_10_2_vs_MA_10_3 -0.0825
## 2    11.4  2.36   0.110     0.110 D1_MA_10_2_vs_MA_10_4  0.181
## 3    11.4  2.36   0.110     0.110 D1_MA_10_2_vs_MA_10_5 -0.112
## 4    11.4  2.36   0.110     0.110 D1_MA_10_3_vs_MA_10_4  0.263
## 5    11.4  2.36   0.110     0.110 D1_MA_10_3_vs_MA_10_5 -0.0296
## 6    11.4  2.36   0.110     0.110 D1_MA_10_4_vs_MA_10_5 -0.293
```

Day 2:

```
nfkb1_data2$Measurement <- as.character(nfkb1_data2$Measurement)

# Subset the data to include only observations at time point D1
nfkb1_data2_D2 <- subset(nfkb1_data2,
                  Measurement %in% c("D2"))

nfkb1_data2_D2$group <- interaction(nfkb1_data2_D2$Type,
nfkb1_data2_D2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb1_data2_D2)

# Fit the model
fit <- lmFit(nfkb1_data2_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D2_MA_10_2_vs_MA_10_3 = groupMA_10_2.D2 - groupMA_10_3.D2,
  D2_MA_10_2_vs_MA_10_4 = groupMA_10_2.D2 - groupMA_10_4.D2,
  D2_MA_10_2_vs_MA_10_5 = groupMA_10_2.D2 - groupMA_10_5.D2,
  D2_MA_10_3_vs_MA_10_4 = groupMA_10_3.D2 - groupMA_10_4.D2,
  D2_MA_10_3_vs_MA_10_5 = groupMA_10_3.D2 - groupMA_10_5.D2,
  D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
  levels = design
)

# Fit the contrasts
fit2_Day2 <- contrasts.fit(fit, cont.matrix)
```

```
# Compute differential expression statistics
fit2_Day2 <- eBayes(fit2_Day2)

# Get the top table
results_D2 <- topTable(fit2_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##   D2_MA_10_2_vs_MA_10_3 D2_MA_10_2_vs_MA_10_4 D2_MA_10_2_vs_MA_10_5
## 1            -0.3984215            -0.4083323           -0.01376769
##   D2_MA_10_3_vs_MA_10_4 D2_MA_10_3_vs_MA_10_5 D2_MA_10_4_vs_MA_10_5
AveExpr
## 1          -0.009910769             0.3846538             0.3945646 11.70989
##          F    P.Value  adj.P.Val
## 1 2.719984 0.07898725 0.07898725
results_D2_long <- results_D2 %>%
  pivot_longer(cols = starts_with("D2"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D2_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison             Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                  <dbl>
## 1    11.7  2.72  0.0790    0.0790 D2_MA_10_2_vs_MA_10_3 -0.398
## 2    11.7  2.72  0.0790    0.0790 D2_MA_10_2_vs_MA_10_4 -0.408
## 3    11.7  2.72  0.0790    0.0790 D2_MA_10_2_vs_MA_10_5 -0.0138
## 4    11.7  2.72  0.0790    0.0790 D2_MA_10_3_vs_MA_10_4 -0.00991
## 5    11.7  2.72  0.0790    0.0790 D2_MA_10_3_vs_MA_10_5  0.385
## 6    11.7  2.72  0.0790    0.0790 D2_MA_10_4_vs_MA_10_5  0.395
```

Day 4:

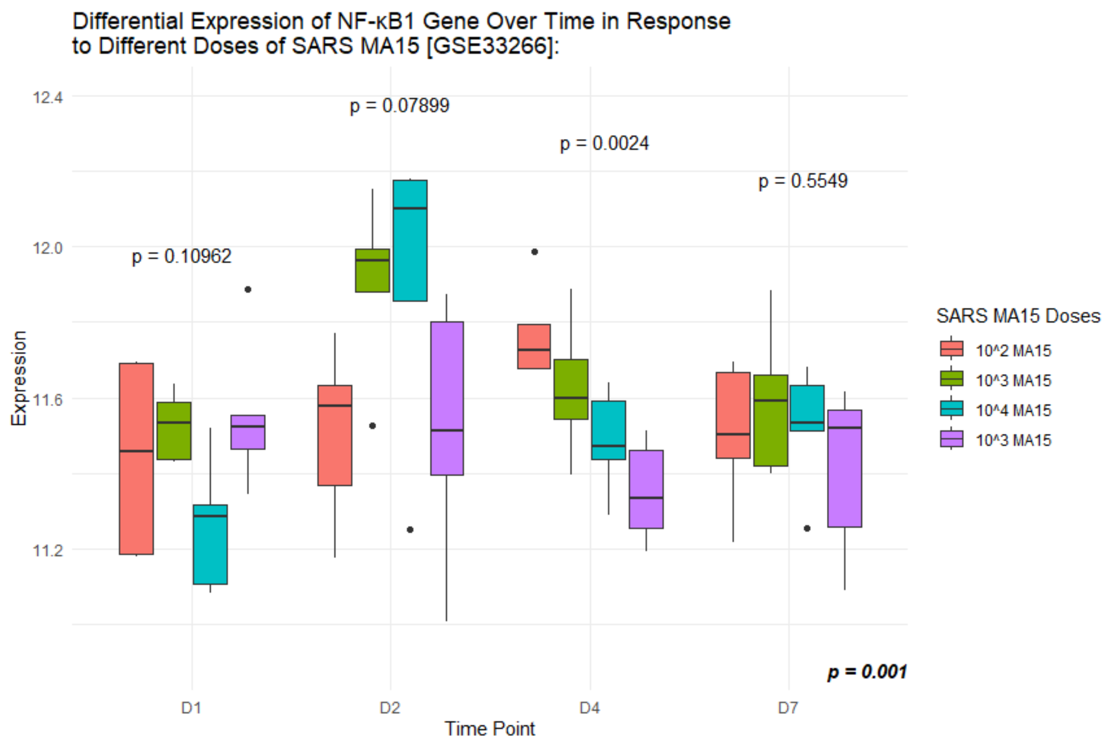
```
nfkb1_data2$Measurement <- as.character(nfkb1_data2$Measurement)

# Subset the data to include only observations at time point D1
nfkb1_data2_D4 <- subset(nfkb1_data2,
                         Measurement %in% c("D4"))

nfkb1_data2_D4$group <- interaction(nfkb1_data2_D4$Type,
nfkb1_data2_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb1_data2_D4)

# Fit the model
fit <- lmFit(nfkb1_data2_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
   D4_MA_10_2_vs_MA_10_3 = groupMA_10_2.D4 - groupMA_10_3.D4,
   D4_MA_10_2_vs_MA_10_4 = groupMA_10_2.D4 - groupMA_10_4.D4,
  D4_MA_10_2_vs_MA_10_5 = groupMA_10_2.D4 - groupMA_10_5.D4,
  D4_MA_10_3_vs_MA_10_4 = groupMA_10_3.D4 - groupMA_10_4.D4,
  D4_MA_10_3_vs_MA_10_5 = groupMA_10_3.D4 - groupMA_10_5.D4,
  D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
  levels = design
)

# Fit the contrasts
fit2_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day4 <- eBayes(fit2_Day4)

# Get the top table
results_D4 <- topTable(fit2_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##   D4_MA_10_2_vs_MA_10_3 D4_MA_10_2_vs_MA_10_4 D4_MA_10_2_vs_MA_10_5
## 1             0.1466692             0.2855415             0.4209631
##   D4_MA_10_3_vs_MA_10_4 D4_MA_10_3_vs_MA_10_5 D4_MA_10_4_vs_MA_10_5
AveExpr
## 1             0.1388723             0.2742938             0.1354215 11.55883
##          F     P.Value   adj.P.Val
## 1 7.474659 0.002393792 0.002393792
results_D4_long <- results_D4 %>%
  pivot_longer(cols = starts_with("D4"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D4_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison            Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                 <dbl>
## 1    11.6  7.47 0.00239   0.00239 D4_MA_10_2_vs_MA_10_3 0.147
## 2    11.6  7.47 0.00239   0.00239 D4_MA_10_2_vs_MA_10_4 0.286
## 3    11.6  7.47 0.00239   0.00239 D4_MA_10_2_vs_MA_10_5 0.421
## 4    11.6  7.47 0.00239   0.00239 D4_MA_10_3_vs_MA_10_4 0.139
## 5    11.6  7.47 0.00239   0.00239 D4_MA_10_3_vs_MA_10_5 0.274
## 6    11.6  7.47 0.00239   0.00239 D4_MA_10_4_vs_MA_10_5 0.135
```

Day 7:

```
nfkb1_data2$Measurement <- as.character(nfkb1_data2$Measurement)

# Subset the data to include only observations at time point D1
nfkb1_data2_D7 <- subset(nfkb1_data2,
                         Measurement %in% c("D7"))

nfkb1_data2_D7$group <- interaction(nfkb1_data2_D7$Type,
nfkb1_data2_D7$Measurement)
```

```
# Create the design matrix
design <- model.matrix(~0 + group, data = nfkb1_data2_D7)

# Fit the model
fit <- lmFit(nfkb1_data2_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D7_MA_10_2_vs_MA_10_3 = groupMA_10_2.D7 - groupMA_10_3.D7,
  D7_MA_10_2_vs_MA_10_4 = groupMA_10_2.D7 - groupMA_10_4.D7,
  D7_MA_10_2_vs_MA_10_5 = groupMA_10_2.D7 - groupMA_10_5.D7,
  D7_MA_10_3_vs_MA_10_4 = groupMA_10_3.D7 - groupMA_10_4.D7,
  D7_MA_10_3_vs_MA_10_5 = groupMA_10_3.D7 - groupMA_10_5.D7,
  D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
  levels = design
)

# Fit the contrasts
fit2_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day7 <- eBayes(fit2_Day7)

# Get the top table
results_D7 <- topTable(fit2_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##   D7_MA_10_2_vs_MA_10_3 D7_MA_10_2_vs_MA_10_4 D7_MA_10_2_vs_MA_10_5
## 1           -0.08685077              -0.01808            0.09431692
##   D7_MA_10_3_vs_MA_10_4 D7_MA_10_3_vs_MA_10_5 D7_MA_10_4_vs_MA_10_5
AveExpr
## 1            0.06877077             0.1811677             0.1123969 11.50723
##           F   P.Value adj.P.Val
## 1 0.7193628 0.5548641 0.5548641
results_D7_long <- results_D7 %>%
  pivot_longer(cols = starts_with("D7"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D7_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison             Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                  <dbl>
## 1    11.5 0.719   0.555     0.555 D7_MA_10_2_vs_MA_10_3 -0.0869
## 2    11.5 0.719   0.555     0.555 D7_MA_10_2_vs_MA_10_4 -0.0181
## 3    11.5 0.719   0.555     0.555 D7_MA_10_2_vs_MA_10_5  0.0943
## 4    11.5 0.719   0.555     0.555 D7_MA_10_3_vs_MA_10_4  0.0688
## 5    11.5 0.719   0.555     0.555 D7_MA_10_3_vs_MA_10_5  0.181
## 6    11.5 0.719   0.555     0.555 D7_MA_10_4_vs_MA_10_5  0.112
tnf_data_WT_MA15 <- nfkb1_data2
```

```r
P.Value1 <- results$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "Differential Expression of NF-κB1 Gene Over Time in Response \nto Different
Doses of SARS MA15 [GSE33266]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^2 MA15", "10^3 MA15",
"10^4 MA15", "10^3 MA15")) +
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = Inf, y = 10.9, label = paste0("p = ", round(P.Value1, 3)), hjust =
1, vjust = 1, size = 4, fontface = "bold.italic") +
  annotate("text", x = 1.2, y = 12, label = paste0("p = ", round(P.Value2, 5)), hjust = 1, vjust
= 1) +
  annotate("text", x = 2.3, y = 12.4, label = paste0("p = ", round(P.Value3, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 3.3, y = 12.3, label = paste0("p = ", round(P.Value4, 4)), hjust = 1,
vjust = 1) +
  annotate("text", x = 4.3, y = 12.2, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)

# Print the graph
print(p)
```



```
combined_results <- bind_rows(results_D1_long, results_D2_long, results_D4_long,
results_D7_long)

# Merge the data frames by columns
df2 <- merge(results_D1_long, results_D2_long, all = TRUE)
df2 <- merge(df2, results_D4_long, all = TRUE)
df2 <- merge(df2, results_D7_long, all = TRUE)

df2
```

AveExpr	F	P.Value	adj.P.Val	Comparison	Value
11.4	2.36	0.11	0.11	D1_MA_10_2_vs_MA_10_3	-0.0825
11.4	2.36	0.11	0.11	D1_MA_10_2_vs_MA_10_4	0.181
11.4	2.36	0.11	0.11	D1_MA_10_2_vs_MA_10_5	-0.112
11.4	2.36	0.11	0.11	D1_MA_10_3_vs_MA_10_4	0.263
11.4	2.36	0.11	0.11	D1_MA_10_3_vs_MA_10_5	-0.0296
11.4	2.36	0.11	0.11	D1_MA_10_4_vs_MA_10_5	-0.293
11.5	0.719	0.555	0.555	D7_MA_10_2_vs_MA_10_3	-0.0869
11.5	0.719	0.555	0.555	D7_MA_10_2_vs_MA_10_4	-0.0181
11.5	0.719	0.555	0.555	D7_MA_10_2_vs_MA_10_5	0.0943
11.5	0.719	0.555	0.555	D7_MA_10_3_vs_MA_10_4	0.0688
11.5	0.719	0.555	0.555	D7_MA_10_3_vs_MA_10_5	0.181
11.5	0.719	0.555	0.555	D7_MA_10_4_vs_MA_10_5	0.112
11.6	7.47	0.00239	0.00239	D4_MA_10_2_vs_MA_10_3	0.147
11.6	7.47	0.00239	0.00239	D4_MA_10_2_vs_MA_10_4	0.286

11.6	7.47	0.00239	0.00239	D4_MA_10_2_vs_MA_10_5	0.421
11.6	7.47	0.00239	0.00239	D4_MA_10_3_vs_MA_10_4	0.139
11.6	7.47	0.00239	0.00239	D4_MA_10_3_vs_MA_10_5	0.274
11.6	7.47	0.00239	0.00239	D4_MA_10_4_vs_MA_10_5	0.135
11.7	2.72	0.079	0.079	D2_MA_10_2_vs_MA_10_3	-0.398
11.7	2.72	0.079	0.079	D2_MA_10_2_vs_MA_10_4	-0.408
11.7	2.72	0.079	0.079	D2_MA_10_2_vs_MA_10_5	-0.0138
11.7	2.72	0.079	0.079	D2_MA_10_3_vs_MA_10_4	-0.00991
11.7	2.72	0.079	0.079	D2_MA_10_3_vs_MA_10_5	0.385
11.7	2.72	0.079	0.079	D2_MA_10_4_vs_MA_10_5	0.395

VEGFA:

```
# Convert necessary columns to appropriate types
Vegfa_data2$Type <- as.factor(Vegfa_data2$Type)
Vegfa_data2$Measurement <- as.factor(Vegfa_data2$Measurement)
Vegfa_data2$Expression <- as.numeric(as.character(Vegfa_data2$Expression))

design <- model.matrix(~0 + Type, data = Vegfa_data2)

# Fit a linear model
fit <- lmFit(Vegfa_data2$Expression, design)

# Contrast matrix
contrast_matrix <- makeContrasts(
  MA10_2.vs.MA10_3 = TypeMA_10_2 - TypeMA_10_3,
  MA10_2.vs.MA10_4 = TypeMA_10_2 - TypeMA_10_4,
  MA10_2.vs.MA10_5 = TypeMA_10_2 - TypeMA_10_5,
  MA10_3.vs.MA10_4 = TypeMA_10_3 - TypeMA_10_4,
  MA10_3.vs.MA10_5 = TypeMA_10_3 - TypeMA_10_5,
  MA10_4.vs.MA10_5 = TypeMA_10_4 - TypeMA_10_5,
  levels = colnames(design)
)

# eBayes
fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

# Extract results for the "TNF" gene for each contrast
Vegfa_results1 <- topTable(fit, coef = "MA10_2.vs.MA10_3", number = Inf)
Vegfa_results1$adj.P.Val <- p.adjust(Vegfa_results1$P.Value, method = "bonferroni")
Vegfa_results1$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^3"

Vegfa_results2 <- topTable(fit, coef = "MA10_2.vs.MA10_4", number = Inf)
Vegfa_results2$adj.P.Val <- p.adjust(Vegfa_results2$P.Value, method = "bonferroni")
Vegfa_results2$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^4"

Vegfa_results3 <- topTable(fit, coef = "MA10_2.vs.MA10_5", number = Inf)
Vegfa_results3$adj.P.Val <- p.adjust(Vegfa_results3$P.Value, method = "bonferroni")
Vegfa_results3$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^5"
```

```
Vegfa_results4 <- topTable(fit, coef = "MA10_3.vs.MA10_4", number = Inf)
Vegfa_results4$adj.P.Val <- p.adjust(Vegfa_results4$P.Value, method = "bonferroni")
Vegfa_results4$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^4"

Vegfa_results5 <- topTable(fit, coef = "MA10_3.vs.MA10_5", number = Inf)
Vegfa_results5$adj.P.Val <- p.adjust(Vegfa_results5$P.Value, method = "bonferroni")
Vegfa_results5$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^5"

Vegfa_results6 <- topTable(fit, coef = "MA10_4.vs.MA10_5", number = Inf)
Vegfa_results6$adj.P.Val <- p.adjust(Vegfa_results6$P.Value, method = "bonferroni")
Vegfa_results6$Contrast <- "SARS MA15 10^4 vs SARS MA15 10^5"

Vegfa_results_combined <- rbind(Vegfa_results1, Vegfa_results2, Vegfa_results3,
Vegfa_results4, Vegfa_results5, Vegfa_results6)

# Print the combined results
print(Vegfa_results_combined)
##         logFC AveExpr           t    P.Value  adj.P.Val         B
## 1  0.05236667 13.9586  0.42599740 0.67131381 0.67131381 -4.812520
## 2  0.04460417 13.9586  0.36285027 0.71772394 0.71772394 -4.822536
## 3  0.23940417 13.9586  1.94752805 0.05516517 0.05516517 -4.007687
## 4 -0.00776250 13.9586 -0.06314713 0.94981513 0.94981513 -4.848256
## 5  0.18703750 13.9586  1.52153065 0.13227670 0.13227670 -4.480571
## 6  0.19480000 13.9586  1.58467779 0.11719209 0.11719209 -4.424929
##                           Contrast
## 1 SARS MA15 10^2 vs SARS MA15 10^3
## 2 SARS MA15 10^2 vs SARS MA15 10^4
## 3 SARS MA15 10^2 vs SARS MA15 10^5
## 4 SARS MA15 10^3 vs SARS MA15 10^4
## 5 SARS MA15 10^3 vs SARS MA15 10^5
## 6 SARS MA15 10^4 vs SARS MA15 10^5
```

between each dose:

```
# Create a new variable for the interaction of Type and Measurement
Vegfa_data2$group <- interaction(Vegfa_data2$Type, Vegfa_data2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfa_data2)

# Fit the model
fit <- lmFit(Vegfa_data2$Expression, design)

cont.matrix <- makeContrasts(
  # For group MA_10_2
  MA_10_2_D2_D1 = groupMA_10_2.D2 - groupMA_10_2.D1,
  MA_10_2_D4_D2 = groupMA_10_2.D4 - groupMA_10_2.D2,
  MA_10_2_D4_D1 = groupMA_10_2.D4 - groupMA_10_2.D1,
  MA_10_2_D7_D4 = groupMA_10_2.D7 - groupMA_10_2.D4,
  MA_10_2_D7_D2 = groupMA_10_2.D7 - groupMA_10_2.D2,
  MA_10_2_D7_D1 = groupMA_10_2.D7 - groupMA_10_2.D1,
  # For group MA_10_3
  MA_10_3_D2_D1 = groupMA_10_3.D2 - groupMA_10_3.D1,
  MA_10_3_D4_D2 = groupMA_10_3.D4 - groupMA_10_3.D2,
  MA_10_3_D4_D1 = groupMA_10_3.D4 - groupMA_10_3.D1,
  MA_10_3_D7_D4 = groupMA_10_3.D7 - groupMA_10_3.D4,
  MA_10_3_D7_D2 = groupMA_10_3.D7 - groupMA_10_3.D2,
  MA_10_3_D7_D1 = groupMA_10_3.D7 - groupMA_10_3.D1,

  # For group MA_10_4
  MA_10_4_D2_D1 = groupMA_10_4.D2 - groupMA_10_4.D1,
  MA_10_4_D4_D2 = groupMA_10_4.D4 - groupMA_10_4.D2,
  MA_10_4_D4_D1 = groupMA_10_4.D4 - groupMA_10_4.D1,
  MA_10_4_D7_D4 = groupMA_10_4.D7 - groupMA_10_4.D4,
  MA_10_4_D7_D2 = groupMA_10_4.D7 - groupMA_10_4.D2,
  MA_10_4_D7_D1 = groupMA_10_4.D7 - groupMA_10_4.D1,

  # For group MA_10_5
  MA_10_5_D2_D1 = groupMA_10_5.D2 - groupMA_10_5.D1,
  MA_10_5_D4_D2 = groupMA_10_5.D4 - groupMA_10_5.D2,
  MA_10_5_D4_D1 = groupMA_10_5.D4 - groupMA_10_5.D1,
  MA_10_5_D7_D4 = groupMA_10_5.D7 - groupMA_10_5.D4,
  MA_10_5_D7_D2 = groupMA_10_5.D7 - groupMA_10_5.D2,
  MA_10_5_D7_D1 = groupMA_10_5.D7 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2 <- eBayes(fit2)

# Get the top table
results <- topTable(fit2, number=Inf)
results$adj.P.Val <- p.adjust(results$P.Value, method = "bonferroni")

# Print the results
print(results)
##   MA_10_2_D2_D1 MA_10_2_D4_D2 MA_10_2_D4_D1 MA_10_2_D7_D4
MA_10_2_D7_D2
## 1    -0.4164167     0.1931833    -0.2232333    -0.2544833       -0.0613
##   MA_10_2_D7_D1 MA_10_3_D2_D1 MA_10_3_D4_D2 MA_10_3_D4_D1
MA_10_3_D7_D4
## 1    -0.4777167    -0.3547667    -0.1576833      -0.51245    -0.3871667
##   MA_10_3_D7_D2 MA_10_3_D7_D1 MA_10_4_D2_D1 MA_10_4_D4_D2
MA_10_4_D4_D1
## 1      -0.54485    -0.8996167    -0.5054833      -0.02075    -0.5262333
##   MA_10_4_D7_D4 MA_10_4_D7_D2 MA_10_4_D7_D1 MA_10_5_D2_D1
MA_10_5_D4_D2
## 1    -0.2706333    -0.2913833    -0.7968667    -0.9021167       -0.0789
##   MA_10_5_D4_D1 MA_10_5_D7_D4 MA_10_5_D7_D2 MA_10_5_D7_D1 AveExpr
```

```
F
## 1    -0.9810167    0.01076667   -0.06813333      -0.97025 13.9586 11.52682
##        P.Value    adj.P.Val
## 1 7.063933e-12 7.063933e-12
results_long <- results %>%
  pivot_longer(cols = starts_with("MA"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_long)
## # A tibble: 24 × 6
##    AveExpr     F  P.Value adj.P.Val Comparison       Value
##      <dbl> <dbl>    <dbl>     <dbl> <chr>            <dbl>
##  1    14.0  11.5 7.06e-12  7.06e-12 MA_10_2_D2_D1 -0.416
##  2    14.0  11.5 7.06e-12  7.06e-12 MA_10_2_D4_D2  0.193
##  3    14.0  11.5 7.06e-12  7.06e-12 MA_10_2_D4_D1 -0.223
##  4    14.0  11.5 7.06e-12  7.06e-12 MA_10_2_D7_D4 -0.254
##  5    14.0  11.5 7.06e-12  7.06e-12 MA_10_2_D7_D2 -0.0613
##  6    14.0  11.5 7.06e-12  7.06e-12 MA_10_2_D7_D1 -0.478
##  7    14.0  11.5 7.06e-12  7.06e-12 MA_10_3_D2_D1 -0.355
##  8    14.0  11.5 7.06e-12  7.06e-12 MA_10_3_D4_D2 -0.158
##  9    14.0  11.5 7.06e-12  7.06e-12 MA_10_3_D4_D1 -0.512
## 10    14.0  11.5 7.06e-12  7.06e-12 MA_10_3_D7_D4 -0.387
## # ℹ 14 more rows
```

Between different Time points:

Day 1:

```
Vegfa_data2$Measurement <- as.character(Vegfa_data2$Measurement)

# Subset the data to include only observations at time point D1
Vegfa_data2_D1 <- subset(Vegfa_data2,
                         Measurement %in% c("D1"))

Vegfa_data2_D1$group <- interaction(Vegfa_data2_D1$Type,
Vegfa_data2_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfa_data2_D1)

# Fit the model
fit <- lmFit(Vegfa_data2_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D1_MA_10_2_vs_MA_10_3 = groupMA_10_2.D1 - groupMA_10_3.D1,
  D1_MA_10_2_vs_MA_10_4 = groupMA_10_2.D1 - groupMA_10_4.D1,
  D1_MA_10_2_vs_MA_10_5 = groupMA_10_2.D1 - groupMA_10_5.D1,
  D1_MA_10_3_vs_MA_10_4 = groupMA_10_3.D1 - groupMA_10_4.D1,
  D1_MA_10_3_vs_MA_10_5 = groupMA_10_3.D1 - groupMA_10_5.D1,
  D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2_Day1 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day1 <- eBayes(fit2_Day1)

# Get the top table
results_D1 <- topTable(fit2_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##   D1_MA_10_2_vs_MA_10_3 D1_MA_10_2_vs_MA_10_4 D1_MA_10_2_vs_MA_10_5
## 1                 -0.11               -0.1332               -0.1946
##   D1_MA_10_3_vs_MA_10_4 D1_MA_10_3_vs_MA_10_5 D1_MA_10_4_vs_MA_10_5
AveExpr
## 1               -0.0232               -0.0846               -0.0614 14.43148
##          F   P.Value adj.P.Val
## 1 1.124369 0.3687098 0.3687098
results_D1_long <- results_D1 %>%
  pivot_longer(cols = starts_with("D1"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D1_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison             Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                  <dbl>
## 1    14.4  1.12   0.369     0.369 D1_MA_10_2_vs_MA_10_3 -0.110
## 2    14.4  1.12   0.369     0.369 D1_MA_10_2_vs_MA_10_4 -0.133
## 3    14.4  1.12   0.369     0.369 D1_MA_10_2_vs_MA_10_5 -0.195
## 4    14.4  1.12   0.369     0.369 D1_MA_10_3_vs_MA_10_4 -0.0232
## 5    14.4  1.12   0.369     0.369 D1_MA_10_3_vs_MA_10_5 -0.0846
## 6    14.4  1.12   0.369     0.369 D1_MA_10_4_vs_MA_10_5 -0.0614
```

Day 2:

```
Vegfa_data2$Measurement <- as.character(Vegfa_data2$Measurement)

# Subset the data to include only observations at time point D1
Vegfa_data2_D2 <- subset(Vegfa_data2,
                         Measurement %in% c("D2"))

Vegfa_data2_D2$group <- interaction(Vegfa_data2_D2$Type,
Vegfa_data2_D2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfa_data2_D2)
```

```
# Fit the model
fit <- lmFit(Vegfa_data2_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D2_MA_10_2_vs_MA_10_3 = groupMA_10_2.D2 - groupMA_10_3.D2,
  D2_MA_10_2_vs_MA_10_4 = groupMA_10_2.D2 - groupMA_10_4.D2,
  D2_MA_10_2_vs_MA_10_5 = groupMA_10_2.D2 - groupMA_10_5.D2,
  D2_MA_10_3_vs_MA_10_4 = groupMA_10_3.D2 - groupMA_10_4.D2,
  D2_MA_10_3_vs_MA_10_5 = groupMA_10_3.D2 - groupMA_10_5.D2,
  D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
  levels = design
)

# Fit the contrasts
fit2_Day2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day2 <- eBayes(fit2_Day2)

# Get the top table
results_D2 <- topTable(fit2_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##   D2_MA_10_2_vs_MA_10_3 D2_MA_10_2_vs_MA_10_4 D2_MA_10_2_vs_MA_10_5
## 1              -0.17165           -0.04413333                0.2911
##   D2_MA_10_3_vs_MA_10_4 D2_MA_10_3_vs_MA_10_5 D2_MA_10_4_vs_MA_10_5
AveExpr
## 1             0.1275167               0.46275             0.3352333 13.88679
##          F    P.Value  adj.P.Val
## 1 2.523853 0.09444067 0.09444067
results_D2_long <- results_D2 %>%
  pivot_longer(cols = starts_with("D2"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D2_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison              Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                   <dbl>
## 1    13.9  2.52  0.0944    0.0944 D2_MA_10_2_vs_MA_10_3 -0.172
## 2    13.9  2.52  0.0944    0.0944 D2_MA_10_2_vs_MA_10_4 -0.0441
## 3    13.9  2.52  0.0944    0.0944 D2_MA_10_2_vs_MA_10_5  0.291
## 4    13.9  2.52  0.0944    0.0944 D2_MA_10_3_vs_MA_10_4  0.128
## 5    13.9  2.52  0.0944    0.0944 D2_MA_10_3_vs_MA_10_5  0.463
## 6    13.9  2.52  0.0944    0.0944 D2_MA_10_4_vs_MA_10_5  0.335
```

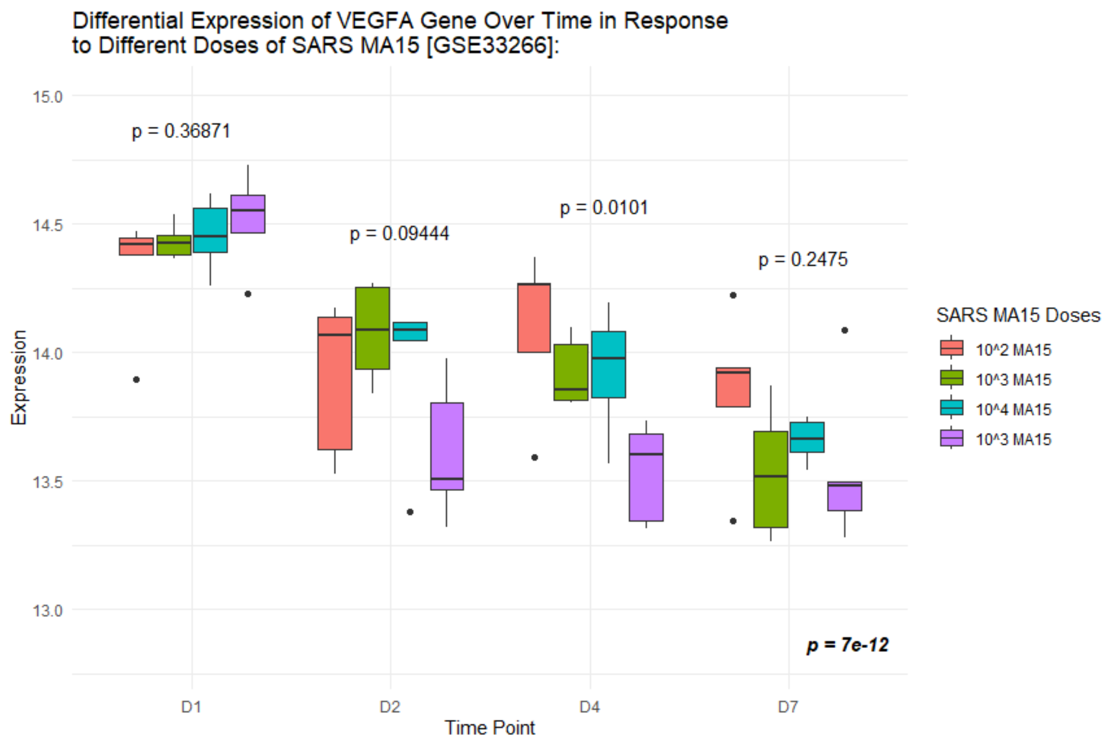
Day 4:

```
Vegfa_data2$Measurement <- as.character(Vegfa_data2$Measurement)
```

```
# Subset the data to include only observations at time point D1
Vegfa_data2_D4 <- subset(Vegfa_data2,
                          Measurement %in% c("D4"))

Vegfa_data2_D4$group <- interaction(Vegfa_data2_D4$Type,
Vegfa_data2_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfa_data2_D4)

# Fit the model
fit <- lmFit(Vegfa_data2_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
   D4_MA_10_2_vs_MA_10_3 = groupMA_10_2.D4 - groupMA_10_3.D4,
   D4_MA_10_2_vs_MA_10_4 = groupMA_10_2.D4 - groupMA_10_4.D4,
   D4_MA_10_2_vs_MA_10_5 = groupMA_10_2.D4 - groupMA_10_5.D4,
   D4_MA_10_3_vs_MA_10_4 = groupMA_10_3.D4 - groupMA_10_4.D4,
   D4_MA_10_3_vs_MA_10_5 = groupMA_10_3.D4 - groupMA_10_5.D4,
   D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
   levels = design
)

# Fit the contrasts
fit2_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day4 <- eBayes(fit2_Day4)

# Get the top table
results_D4 <- topTable(fit2_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##   D4_MA_10_2_vs_MA_10_3 D4_MA_10_2_vs_MA_10_4 D4_MA_10_2_vs_MA_10_5
## 1             0.1792167                0.1698             0.5631833
##   D4_MA_10_3_vs_MA_10_4 D4_MA_10_3_vs_MA_10_5 D4_MA_10_4_vs_MA_10_5
AveExpr
## 1          -0.009416667             0.3839667             0.3933833 13.87075
##          F    P.Value  adj.P.Val
## 1 5.284799 0.01005315 0.01005315
results_D4_long <- results_D4 %>%
  pivot_longer(cols = starts_with("D4"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D4_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison              Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                  <dbl>
## 1    13.9  5.28  0.0101    0.0101 D4_MA_10_2_vs_MA_10_3  0.179
## 2    13.9  5.28  0.0101    0.0101 D4_MA_10_2_vs_MA_10_4  0.170
## 3    13.9  5.28  0.0101    0.0101 D4_MA_10_2_vs_MA_10_5  0.563
## 4    13.9  5.28  0.0101    0.0101 D4_MA_10_3_vs_MA_10_4 -0.00942
## 5    13.9  5.28  0.0101    0.0101 D4_MA_10_3_vs_MA_10_5  0.384
## 6    13.9  5.28  0.0101    0.0101 D4_MA_10_4_vs_MA_10_5  0.393
```

Day 7:

```
Vegfa_data2$Measurement <- as.character(Vegfa_data2$Measurement)

# Subset the data to include only observations at time point D1
Vegfa_data2_D7 <- subset(Vegfa_data2,
                     Measurement %in% c("D7"))

Vegfa_data2_D7$group <- interaction(Vegfa_data2_D7$Type,
Vegfa_data2_D7$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfa_data2_D7)

# Fit the model
fit <- lmFit(Vegfa_data2_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D7_MA_10_2_vs_MA_10_3 = groupMA_10_2.D7 - groupMA_10_3.D7,
  D7_MA_10_2_vs_MA_10_4 = groupMA_10_2.D7 - groupMA_10_4.D7,
  D7_MA_10_2_vs_MA_10_5 = groupMA_10_2.D7 - groupMA_10_5.D7,
  D7_MA_10_3_vs_MA_10_4 = groupMA_10_3.D7 - groupMA_10_4.D7,
  D7_MA_10_3_vs_MA_10_5 = groupMA_10_3.D7 - groupMA_10_5.D7,
  D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
  levels = design
)

# Fit the contrasts
fit2_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day7 <- eBayes(fit2_Day7)

# Get the top table
results_D7 <- topTable(fit2_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##   D7_MA_10_2_vs_MA_10_3 D7_MA_10_2_vs_MA_10_4 D7_MA_10_2_vs_MA_10_5
## 1                0.3119               0.18595             0.2979333
##   D7_MA_10_3_vs_MA_10_4 D7_MA_10_3_vs_MA_10_5 D7_MA_10_4_vs_MA_10_5
AveExpr
## 1              -0.12595           -0.01396667             0.1119833 13.64537
##          F   P.Value adj.P.Val
## 1 1.520344 0.2475004 0.2475004
results_D7_long <- results_D7 %>%
  pivot_longer(cols = starts_with("D7"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D7_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison             Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                  <dbl>
## 1    13.6  1.52   0.248     0.248 D7_MA_10_2_vs_MA_10_3  0.312
## 2    13.6  1.52   0.248     0.248 D7_MA_10_2_vs_MA_10_4  0.186
## 3    13.6  1.52   0.248     0.248 D7_MA_10_2_vs_MA_10_5  0.298
## 4    13.6  1.52   0.248     0.248 D7_MA_10_3_vs_MA_10_4 -0.126
## 5    13.6  1.52   0.248     0.248 D7_MA_10_3_vs_MA_10_5 -0.0140
## 6    13.6  1.52   0.248     0.248 D7_MA_10_4_vs_MA_10_5  0.112
tnf_data_WT_MA15 <- Vegfa_data2

P.Value1 <- results$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "Differential Expression of VEGFA Gene Over Time in Response \nto Different
Doses of SARS MA15 [GSE33266]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^2 MA15", "10^3 MA15",
"10^4 MA15", "10^3 MA15")) +
  ylim(12.8, 15) +
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = 4.5, y = 12.9, label = paste0("p = ", round(P.Value1, 12)), hjust
= 1, vjust = 1, size = 4, fontface = "bold.italic") +
  annotate("text", x = 1.2, y = 14.9, label = paste0("p = ", round(P.Value2, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 2.3, y = 14.5, label = paste0("p = ", round(P.Value3, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 3.3, y = 14.6, label = paste0("p = ", round(P.Value4, 4)), hjust = 1,
vjust = 1) +
  annotate("text", x = 4.3, y = 14.4, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)

# Print the graph
print(p)
```



```
combined_results <- bind_rows(results_D1_long, results_D2_long, results_D4_long,
results_D7_long)

# Merge the data frames by columns
df2 <- merge(results_D1_long, results_D2_long, all = TRUE)
df2 <- merge(df2, results_D4_long, all = TRUE)
df2 <- merge(df2, results_D7_long, all = TRUE)

df2
```

AveExpr	F	P.Value	adj.P.Val	Comparison	Value
13.6	1.52	0.248	0.248	D7_MA_10_2_vs_MA_10_3	0.312
13.6	1.52	0.248	0.248	D7_MA_10_2_vs_MA_10_4	0.186
13.6	1.52	0.248	0.248	D7_MA_10_2_vs_MA_10_5	0.298
13.6	1.52	0.248	0.248	D7_MA_10_3_vs_MA_10_4	-0.126
13.6	1.52	0.248	0.248	D7_MA_10_3_vs_MA_10_5	-0.014
13.6	1.52	0.248	0.248	D7_MA_10_4_vs_MA_10_5	0.112
13.9	5.28	0.0101	0.0101	D4_MA_10_2_vs_MA_10_3	0.179
13.9	5.28	0.0101	0.0101	D4_MA_10_2_vs_MA_10_4	0.17
13.9	5.28	0.0101	0.0101	D4_MA_10_2_vs_MA_10_5	0.563
13.9	5.28	0.0101	0.0101	D4_MA_10_3_vs_MA_10_4	-0.00942
13.9	5.28	0.0101	0.0101	D4_MA_10_3_vs_MA_10_5	0.384
13.9	5.28	0.0101	0.0101	D4_MA_10_4_vs_MA_10_5	0.393

13.9	2.52	0.0944	0.0944	D2_MA_10_2_vs_MA_10_3	-0.172
13.9	2.52	0.0944	0.0944	D2_MA_10_2_vs_MA_10_4	-0.0441
13.9	2.52	0.0944	0.0944	D2_MA_10_2_vs_MA_10_5	0.291
13.9	2.52	0.0944	0.0944	D2_MA_10_3_vs_MA_10_4	0.128
13.9	2.52	0.0944	0.0944	D2_MA_10_3_vs_MA_10_5	0.463
13.9	2.52	0.0944	0.0944	D2_MA_10_4_vs_MA_10_5	0.335
14.4	1.12	0.369	0.369	D1_MA_10_2_vs_MA_10_3	-0.11
14.4	1.12	0.369	0.369	D1_MA_10_2_vs_MA_10_4	-0.133
14.4	1.12	0.369	0.369	D1_MA_10_2_vs_MA_10_5	-0.195
14.4	1.12	0.369	0.369	D1_MA_10_3_vs_MA_10_4	-0.0232
14.4	1.12	0.369	0.369	D1_MA_10_3_vs_MA_10_5	-0.0846
14.4	1.12	0.369	0.369	D1_MA_10_4_vs_MA_10_5	-0.0614

VEGFB:

```
# Convert necessary columns to appropriate types
Vegfb_data2$Type <- as.factor(Vegfb_data2$Type)
Vegfb_data2$Measurement <- as.factor(Vegfb_data2$Measurement)
Vegfb_data2$Expression <- as.numeric(as.character(Vegfb_data2$Expression))

design <- model.matrix(~0 + Type, data = Vegfb_data2)

# Fit a linear model
fit <- lmFit(Vegfb_data2$Expression, design)

# Contrast matrix
contrast_matrix <- makeContrasts(
  MA10_2.vs.MA10_3 = TypeMA_10_2 - TypeMA_10_3,
  MA10_2.vs.MA10_4 = TypeMA_10_2 - TypeMA_10_4,
  MA10_2.vs.MA10_5 = TypeMA_10_2 - TypeMA_10_5,
  MA10_3.vs.MA10_4 = TypeMA_10_3 - TypeMA_10_4,
  MA10_3.vs.MA10_5 = TypeMA_10_3 - TypeMA_10_5,
  MA10_4.vs.MA10_5 = TypeMA_10_4 - TypeMA_10_5,
  levels = colnames(design)
)

# eBayes
fit <- eBayes(contrasts.fit(fit, contrast = contrast_matrix))

# Extract results for the "TNF" gene for each contrast
Vegfb_results1 <- topTable(fit, coef = "MA10_2.vs.MA10_3", number = Inf)
Vegfb_results1$adj.P.Val <- p.adjust(Vegfb_results1$P.Value, method = "bonferroni")
Vegfb_results1$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^3"

Vegfb_results2 <- topTable(fit, coef = "MA10_2.vs.MA10_4", number = Inf)
Vegfb_results2$adj.P.Val <- p.adjust(Vegfb_results2$P.Value, method = "bonferroni")
Vegfb_results2$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^4"
```

```
Vegfb_results3 <- topTable(fit, coef = "MA10_2.vs.MA10_5", number = Inf)
Vegfb_results3$adj.P.Val <- p.adjust(Vegfb_results3$P.Value, method = "bonferroni")
Vegfb_results3$Contrast <- "SARS MA15 10^2 vs SARS MA15 10^5"

Vegfb_results4 <- topTable(fit, coef = "MA10_3.vs.MA10_4", number = Inf)
Vegfb_results4$adj.P.Val <- p.adjust(Vegfb_results4$P.Value, method = "bonferroni")
Vegfb_results4$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^4"

Vegfb_results5 <- topTable(fit, coef = "MA10_3.vs.MA10_5", number = Inf)
Vegfb_results5$adj.P.Val <- p.adjust(Vegfb_results5$P.Value, method = "bonferroni")
Vegfb_results5$Contrast <- "SARS MA15 10^3 vs SARS MA15 10^5"

Vegfb_results6 <- topTable(fit, coef = "MA10_4.vs.MA10_5", number = Inf)
Vegfb_results6$adj.P.Val <- p.adjust(Vegfb_results6$P.Value, method = "bonferroni")
Vegfb_results6$Contrast <- "SARS MA15 10^4 vs SARS MA15 10^5"

Vegfb_results_combined <- rbind(Vegfb_results1, Vegfb_results2, Vegfb_results3,
Vegfb_results4, Vegfb_results5, Vegfb_results6)

# Print the combined results
print(Vegfb_results_combined)
##       logFC  AveExpr        t      P.Value    adj.P.Val         B
## 1 0.2413750 11.50401 1.891843 0.0623212463 0.0623212463 -4.081972
## 2 0.3965300 11.50401 3.107914 0.0026503308 0.0026503308 -1.743823
## 3 0.5207725 11.50401 4.081699 0.0001094341 0.0001094341  1.008328
## 4 0.1551550 11.50401 1.216070 0.2277217707 0.2277217707 -4.669783
## 5 0.2793975 11.50401 2.189855 0.0316028131 0.0316028131 -3.644222
## 6 0.1242425 11.50401 0.973785 0.3332525663 0.3332525663 -4.712146
##                            Contrast
## 1 SARS MA15 10^2 vs SARS MA15 10^3
## 2 SARS MA15 10^2 vs SARS MA15 10^4
## 3 SARS MA15 10^2 vs SARS MA15 10^5
## 4 SARS MA15 10^3 vs SARS MA15 10^4
## 5 SARS MA15 10^3 vs SARS MA15 10^5
## 6 SARS MA15 10^4 vs SARS MA15 10^5
```

between each dose:

```
# Create a new variable for the interaction of Type and Measurement
Vegfb_data2$group <- interaction(Vegfb_data2$Type, Vegfb_data2$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfb_data2)

# Fit the model
fit <- lmFit(Vegfb_data2$Expression, design)

cont.matrix <- makeContrasts(
  # For group MA_10_2
  MA_10_2_D2_D1 = groupMA_10_2.D2 - groupMA_10_2.D1,
  MA_10_2_D4_D2 = groupMA_10_2.D4 - groupMA_10_2.D2,
  MA_10_2_D4_D1 = groupMA_10_2.D4 - groupMA_10_2.D1,
  MA_10_2_D7_D4 = groupMA_10_2.D7 - groupMA_10_2.D4,
  MA_10_2_D7_D2 = groupMA_10_2.D7 - groupMA_10_2.D2,
  MA_10_2_D7_D1 = groupMA_10_2.D7 - groupMA_10_2.D1,

  # For group MA_10_3
  MA_10_3_D2_D1 = groupMA_10_3.D2 - groupMA_10_3.D1,
  MA_10_3_D4_D2 = groupMA_10_3.D4 - groupMA_10_3.D2,
  MA_10_3_D4_D1 = groupMA_10_3.D4 - groupMA_10_3.D1,
  MA_10_3_D7_D4 = groupMA_10_3.D7 - groupMA_10_3.D4,
  MA_10_3_D7_D2 = groupMA_10_3.D7 - groupMA_10_3.D2,
  MA_10_3_D7_D1 = groupMA_10_3.D7 - groupMA_10_3.D1,

  # For group MA_10_4
  MA_10_4_D2_D1 = groupMA_10_4.D2 - groupMA_10_4.D1,
  MA_10_4_D4_D2 = groupMA_10_4.D4 - groupMA_10_4.D2,
  MA_10_4_D4_D1 = groupMA_10_4.D4 - groupMA_10_4.D1,
  MA_10_4_D7_D4 = groupMA_10_4.D7 - groupMA_10_4.D4,
  MA_10_4_D7_D2 = groupMA_10_4.D7 - groupMA_10_4.D2,
  MA_10_4_D7_D1 = groupMA_10_4.D7 - groupMA_10_4.D1,

  # For group MA_10_5
  MA_10_5_D2_D1 = groupMA_10_5.D2 - groupMA_10_5.D1,
  MA_10_5_D4_D2 = groupMA_10_5.D4 - groupMA_10_5.D2,
  MA_10_5_D4_D1 = groupMA_10_5.D4 - groupMA_10_5.D1,
  MA_10_5_D7_D4 = groupMA_10_5.D7 - groupMA_10_5.D4,
  MA_10_5_D7_D2 = groupMA_10_5.D7 - groupMA_10_5.D2,
  MA_10_5_D7_D1 = groupMA_10_5.D7 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2 <- eBayes(fit2)

# Get the top table
results <- topTable(fit2, number=Inf)
results$adj.P.Val <- p.adjust(results$P.Value, method = "bonferroni")

# Print the results
print(results)
##   MA_10_2_D2_D1 MA_10_2_D4_D2 MA_10_2_D4_D1 MA_10_2_D7_D4
MA_10_2_D7_D2
## 1        0.0206       -0.3937       -0.3731        0.0867        -0.307
##   MA_10_2_D7_D1 MA_10_3_D2_D1 MA_10_3_D4_D2 MA_10_3_D4_D1
MA_10_3_D7_D4
## 1       -0.2864       -0.5045       -0.2837       -0.7882        0.2833
##   MA_10_3_D7_D2 MA_10_3_D7_D1 MA_10_4_D2_D1 MA_10_4_D4_D2
MA_10_4_D4_D1
## 1        -4e-04       -0.5049      -1.15452       0.66922       -0.4853
##   MA_10_4_D7_D4 MA_10_4_D7_D2 MA_10_4_D7_D1 MA_10_5_D2_D1
```

```
MA_10_5_D4_D2
## 1       0.2217       0.89092       -0.2636      -1.06043       0.52253
##   MA_10_5_D4_D1 MA_10_5_D7_D4 MA_10_5_D7_D2 MA_10_5_D7_D1  AveExpr
F
## 1       -0.5379      -0.05736       0.46517      -0.59526 11.50401 12.63382
##        P.Value    adj.P.Val
## 1 1.055724e-12 1.055724e-12
results_long <- results %>%
  pivot_longer(cols = starts_with("MA"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_long)
## # A tibble: 24 × 6
##    AveExpr     F  P.Value adj.P.Val Comparison      Value
##      <dbl> <dbl>    <dbl>     <dbl> <chr>           <dbl>
##  1    11.5  12.6 1.06e-12  1.06e-12 MA_10_2_D2_D1  0.0206
##  2    11.5  12.6 1.06e-12  1.06e-12 MA_10_2_D4_D2 -0.394
##  3    11.5  12.6 1.06e-12  1.06e-12 MA_10_2_D4_D1 -0.373
##  4    11.5  12.6 1.06e-12  1.06e-12 MA_10_2_D7_D4  0.0867
##  5    11.5  12.6 1.06e-12  1.06e-12 MA_10_2_D7_D2 -0.307
##  6    11.5  12.6 1.06e-12  1.06e-12 MA_10_2_D7_D1 -0.286
##  7    11.5  12.6 1.06e-12  1.06e-12 MA_10_3_D2_D1 -0.504
##  8    11.5  12.6 1.06e-12  1.06e-12 MA_10_3_D4_D2 -0.284
##  9    11.5  12.6 1.06e-12  1.06e-12 MA_10_3_D4_D1 -0.788
## 10    11.5  12.6 1.06e-12  1.06e-12 MA_10_3_D7_D4  0.283
## # ℹ 14 more rows
```

Between different Time points:

Day 1:

```
Vegfb_data2$Measurement <- as.character(Vegfb_data2$Measurement)

# Subset the data to include only observations at time point D1
Vegfb_data2_D1 <- subset(Vegfb_data2,
                     Measurement %in% c("D1"))

Vegfb_data2_D1$group <- interaction(Vegfb_data2_D1$Type,
Vegfb_data2_D1$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfb_data2_D1)

# Fit the model
fit <- lmFit(Vegfb_data2_D1$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
   D1_MA_10_2_vs_MA_10_3 = groupMA_10_2.D1 - groupMA_10_3.D1,
   D1_MA_10_2_vs_MA_10_4 = groupMA_10_2.D1 - groupMA_10_4.D1,
  D1_MA_10_2_vs_MA_10_5 = groupMA_10_2.D1 - groupMA_10_5.D1,
  D1_MA_10_3_vs_MA_10_4 = groupMA_10_3.D1 - groupMA_10_4.D1,
  D1_MA_10_3_vs_MA_10_5 = groupMA_10_3.D1 - groupMA_10_5.D1,
  D1_MA_10_4_vs_MA_10_5 = groupMA_10_4.D1 - groupMA_10_5.D1,

  levels = design
)

# Fit the contrasts
fit2_Day1 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day1 <- eBayes(fit2_Day1)

# Get the top table
results_D1 <- topTable(fit2_Day1, number=Inf)
results_D1$adj.P.Val <- p.adjust(results_D1$P.Value, method = "bonferroni")

# Print the results
print(results_D1)
##   D1_MA_10_2_vs_MA_10_3 D1_MA_10_2_vs_MA_10_4 D1_MA_10_2_vs_MA_10_5
## 1               -0.0483                0.0804                0.1321
##   D1_MA_10_3_vs_MA_10_4 D1_MA_10_3_vs_MA_10_5 D1_MA_10_4_vs_MA_10_5
AveExpr
## 1                0.1287                0.1804                0.0517 11.91235
##          F  P.Value adj.P.Val
## 1 2.017607 0.152031  0.152031
results_D1_long <- results_D1 %>%
  pivot_longer(cols = starts_with("D1"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D1_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison              Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                   <dbl>
## 1    11.9  2.02   0.152     0.152 D1_MA_10_2_vs_MA_10_3 -0.0483
## 2    11.9  2.02   0.152     0.152 D1_MA_10_2_vs_MA_10_4  0.0804
## 3    11.9  2.02   0.152     0.152 D1_MA_10_2_vs_MA_10_5  0.132
## 4    11.9  2.02   0.152     0.152 D1_MA_10_3_vs_MA_10_4  0.129
## 5    11.9  2.02   0.152     0.152 D1_MA_10_3_vs_MA_10_5  0.180
## 6    11.9  2.02   0.152     0.152 D1_MA_10_4_vs_MA_10_5  0.0517
```

Day 2:

```
Vegfb_data2$Measurement <- as.character(Vegfb_data2$Measurement)

# Subset the data to include only observations at time point D1
Vegfb_data2_D2 <- subset(Vegfb_data2,
                         Measurement %in% c("D2"))

Vegfb_data2_D2$group <- interaction(Vegfb_data2_D2$Type,
Vegfb_data2_D2$Measurement)
```

```
# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfb_data2_D2)

# Fit the model
fit <- lmFit(Vegfb_data2_D2$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D2_MA_10_2_vs_MA_10_3 = groupMA_10_2.D2 - groupMA_10_3.D2,
  D2_MA_10_2_vs_MA_10_4 = groupMA_10_2.D2 - groupMA_10_4.D2,
  D2_MA_10_2_vs_MA_10_5 = groupMA_10_2.D2 - groupMA_10_5.D2,
  D2_MA_10_3_vs_MA_10_4 = groupMA_10_3.D2 - groupMA_10_4.D2,
  D2_MA_10_3_vs_MA_10_5 = groupMA_10_3.D2 - groupMA_10_5.D2,
  D2_MA_10_4_vs_MA_10_5 = groupMA_10_4.D2 - groupMA_10_5.D2,
  levels = design
)

# Fit the contrasts
fit2_Day2 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day2 <- eBayes(fit2_Day2)

# Get the top table
results_D2 <- topTable(fit2_Day2, number=Inf)
results_D2$adj.P.Val <- p.adjust(results_D2$P.Value, method = "bonferroni")

# Print the results
print(results_D2)
##   D2_MA_10_2_vs_MA_10_3 D2_MA_10_2_vs_MA_10_4 D2_MA_10_2_vs_MA_10_5
## 1                0.4768               1.25552               1.21313
##   D2_MA_10_3_vs_MA_10_4 D2_MA_10_3_vs_MA_10_5 D2_MA_10_4_vs_MA_10_5
AveExpr
## 1               0.77872               0.73633              -0.04239 11.23764
##         F      P.Value    adj.P.Val
## 1 10.8638 0.0003879043 0.0003879043
results_D2_long <- results_D2 %>%
  pivot_longer(cols = starts_with("D2"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D2_long)
## # A tibble: 6 × 6
##   AveExpr     F  P.Value adj.P.Val Comparison               Value
##     <dbl> <dbl>    <dbl>     <dbl> <chr>                    <dbl>
## 1    11.2  10.9 0.000388  0.000388 D2_MA_10_2_vs_MA_10_3  0.477
## 2    11.2  10.9 0.000388  0.000388 D2_MA_10_2_vs_MA_10_4  1.26
## 3    11.2  10.9 0.000388  0.000388 D2_MA_10_2_vs_MA_10_5  1.21
## 4    11.2  10.9 0.000388  0.000388 D2_MA_10_3_vs_MA_10_4  0.779
## 5    11.2  10.9 0.000388  0.000388 D2_MA_10_3_vs_MA_10_5  0.736
## 6    11.2  10.9 0.000388  0.000388 D2_MA_10_4_vs_MA_10_5 -0.0424
```

Day 4:

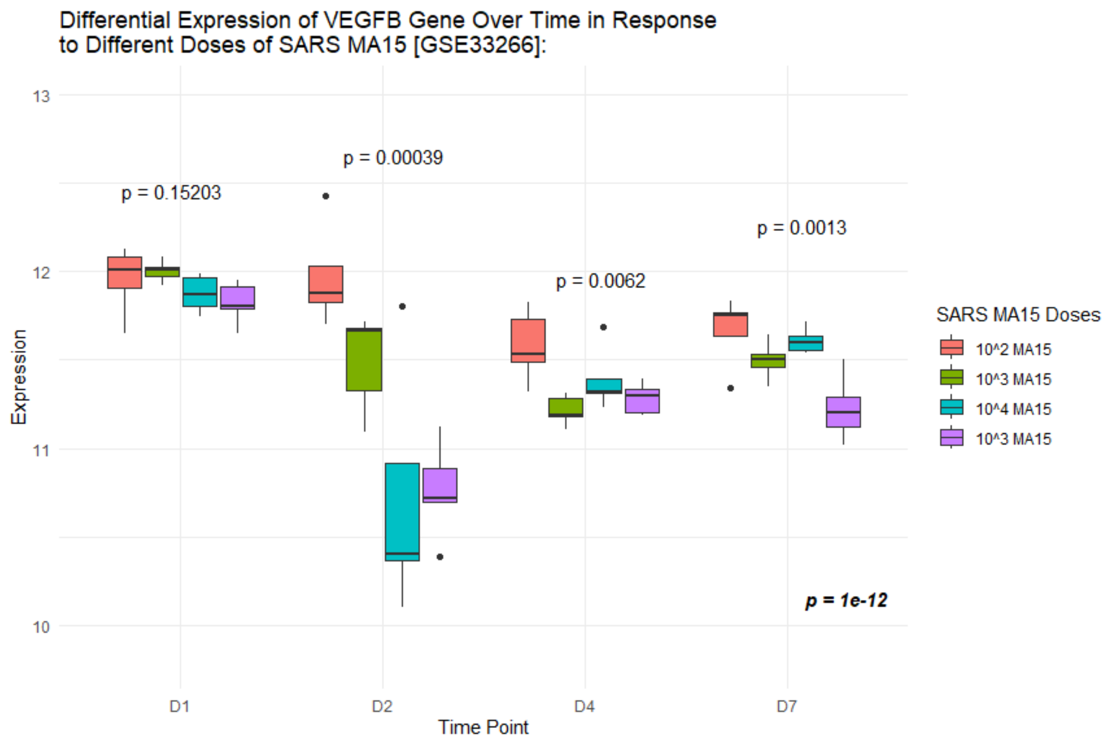
```
Vegfb_data2$Measurement <- as.character(Vegfb_data2$Measurement)

# Subset the data to include only observations at time point D1
Vegfb_data2_D4 <- subset(Vegfb_data2,
                         Measurement %in% c("D4"))

Vegfb_data2_D4$group <- interaction(Vegfb_data2_D4$Type,
Vegfb_data2_D4$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfb_data2_D4)

# Fit the model
fit <- lmFit(Vegfb_data2_D4$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
    D4_MA_10_2_vs_MA_10_3 = groupMA_10_2.D4 - groupMA_10_3.D4,
    D4_MA_10_2_vs_MA_10_4 = groupMA_10_2.D4 - groupMA_10_4.D4,
    D4_MA_10_2_vs_MA_10_5 = groupMA_10_2.D4 - groupMA_10_5.D4,
    D4_MA_10_3_vs_MA_10_4 = groupMA_10_3.D4 - groupMA_10_4.D4,
    D4_MA_10_3_vs_MA_10_5 = groupMA_10_3.D4 - groupMA_10_5.D4,
    D4_MA_10_4_vs_MA_10_5 = groupMA_10_4.D4 - groupMA_10_5.D4,
    levels = design
)

# Fit the contrasts
fit2_Day4 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day4 <- eBayes(fit2_Day4)

# Get the top table
results_D4 <- topTable(fit2_Day4, number=Inf)
results_D4$adj.P.Val <- p.adjust(results_D4$P.Value, method = "bonferroni")

# Print the results
print(results_D4)
##   D4_MA_10_2_vs_MA_10_3 D4_MA_10_2_vs_MA_10_4 D4_MA_10_2_vs_MA_10_5
## 1                0.3668                0.1926                0.2969
##   D4_MA_10_3_vs_MA_10_4 D4_MA_10_3_vs_MA_10_5 D4_MA_10_4_vs_MA_10_5
AveExpr
## 1               -0.1742               -0.0699                0.1043 11.36623
##          F     P.Value   adj.P.Val
## 1 5.970226 0.006243326 0.006243326
results_D4_long <- results_D4 %>%
  pivot_longer(cols = starts_with("D4"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
```

```
print(results_D4_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison              Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                   <dbl>
## 1    11.4  5.97 0.00624   0.00624 D4_MA_10_2_vs_MA_10_3  0.367
## 2    11.4  5.97 0.00624   0.00624 D4_MA_10_2_vs_MA_10_4  0.193
## 3    11.4  5.97 0.00624   0.00624 D4_MA_10_2_vs_MA_10_5  0.297
## 4    11.4  5.97 0.00624   0.00624 D4_MA_10_3_vs_MA_10_4 -0.174
## 5    11.4  5.97 0.00624   0.00624 D4_MA_10_3_vs_MA_10_5 -0.0699
## 6    11.4  5.97 0.00624   0.00624 D4_MA_10_4_vs_MA_10_5  0.104
```

Day 7:

```
Vegfb_data2$Measurement <- as.character(Vegfb_data2$Measurement)

# Subset the data to include only observations at time point D1
Vegfb_data2_D7 <- subset(Vegfb_data2,
                    Measurement %in% c("D7"))

Vegfb_data2_D7$group <- interaction(Vegfb_data2_D7$Type,
Vegfb_data2_D7$Measurement)

# Create the design matrix
design <- model.matrix(~0 + group, data = Vegfb_data2_D7)

# Fit the model
fit <- lmFit(Vegfb_data2_D7$Expression, design)

# Define the contrasts
cont.matrix <- makeContrasts(
  D7_MA_10_2_vs_MA_10_3 = groupMA_10_2.D7 - groupMA_10_3.D7,
  D7_MA_10_2_vs_MA_10_4 = groupMA_10_2.D7 - groupMA_10_4.D7,
  D7_MA_10_2_vs_MA_10_5 = groupMA_10_2.D7 - groupMA_10_5.D7,
  D7_MA_10_3_vs_MA_10_4 = groupMA_10_3.D7 - groupMA_10_4.D7,
  D7_MA_10_3_vs_MA_10_5 = groupMA_10_3.D7 - groupMA_10_5.D7,
  D7_MA_10_4_vs_MA_10_5 = groupMA_10_4.D7 - groupMA_10_5.D7,
  levels = design
)

# Fit the contrasts
fit2_Day7 <- contrasts.fit(fit, cont.matrix)

# Compute differential expression statistics
fit2_Day7 <- eBayes(fit2_Day7)

# Get the top table
results_D7 <- topTable(fit2_Day7, number=Inf)
results_D7$adj.P.Val <- p.adjust(results_D7$P.Value, method = "bonferroni")

# Print the results
print(results_D7)
##   D7_MA_10_2_vs_MA_10_3 D7_MA_10_2_vs_MA_10_4 D7_MA_10_2_vs_MA_10_5
## 1                0.1702                0.0576               0.44096
##   D7_MA_10_3_vs_MA_10_4 D7_MA_10_3_vs_MA_10_5 D7_MA_10_4_vs_MA_10_5
```

```
AveExpr
## 1              -0.1126              0.27076              0.38336 11.49981
##          F     P.Value   adj.P.Val
## 1 8.582347 0.001261469 0.001261469
results_D7_long <- results_D7 %>%
  pivot_longer(cols = starts_with("D7"),
               names_to = "Comparison",
               values_to = "Value")

# Print the reshaped data
print(results_D7_long)
## # A tibble: 6 × 6
##   AveExpr     F P.Value adj.P.Val Comparison              Value
##     <dbl> <dbl>   <dbl>     <dbl> <chr>                   <dbl>
## 1    11.5  8.58 0.00126   0.00126 D7_MA_10_2_vs_MA_10_3  0.170
## 2    11.5  8.58 0.00126   0.00126 D7_MA_10_2_vs_MA_10_4  0.0576
## 3    11.5  8.58 0.00126   0.00126 D7_MA_10_2_vs_MA_10_5  0.441
## 4    11.5  8.58 0.00126   0.00126 D7_MA_10_3_vs_MA_10_4 -0.113
## 5    11.5  8.58 0.00126   0.00126 D7_MA_10_3_vs_MA_10_5  0.271
## 6    11.5  8.58 0.00126   0.00126 D7_MA_10_4_vs_MA_10_5  0.383
tnf_data_WT_MA15 <- Vegfb_data2

P.Value1 <- results$P.Value

# Get the p-value for the comparison of interest
P.Value2 <- results_D1$P.Value

P.Value3 <- results_D2$P.Value

P.Value4 <- results_D4$P.Value

P.Value5 <- results_D7$P.Value

# Create the line chart
p <- ggplot(data = tnf_data_WT_MA15, aes(x = Measurement, y = Expression, fill = Type))
+
  geom_boxplot() +
  labs(title = "Differential Expression of VEGFB Gene Over Time in Response \nto Different
Doses of SARS MA15 [GSE33266]:",
       x = "Time Point",
       y = "Expression") +
  scale_fill_discrete(name = "SARS MA15 Doses", labels = c("10^2 MA15", "10^3 MA15",
"10^4 MA15", "10^3 MA15")) +
  ylim(9.8, 13) +
  theme_minimal()

# Add the p-value to the graph
p <- p + annotate("text", x = 4.5, y = 10.2, label = paste0("p = ", round(P.Value1, 12)), hjust
= 1, vjust = 1, size = 4, fontface = "bold.italic") +
  annotate("text", x = 1.2, y = 12.5, label = paste0("p = ", round(P.Value2, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 2.3, y = 12.7, label = paste0("p = ", round(P.Value3, 5)), hjust = 1,
vjust = 1) +
  annotate("text", x = 3.3, y = 12, label = paste0("p = ", round(P.Value4, 4)), hjust = 1, vjust
= 1) +
  annotate("text", x = 4.3, y = 12.3, label = paste0("p = ", round(P.Value5, 4)), hjust = 1,
vjust = 1)

# Print the graph
print(p)
```



```
combined_results <- bind_rows(results_D1_long, results_D2_long, results_D4_long,
results_D7_long)

# Merge the data frames by columns
df2 <- merge(results_D1_long, results_D2_long, all = TRUE)
df2 <- merge(df2, results_D4_long, all = TRUE)
df2 <- merge(df2, results_D7_long, all = TRUE)

df2
```

AveExpr	F	P.Value	adj.P.Val	Comparison	Value
11.2	10.9	0.000388	0.000388	D2_MA_10_2_vs_MA_10_3	0.477
11.2	10.9	0.000388	0.000388	D2_MA_10_2_vs_MA_10_4	1.26
11.2	10.9	0.000388	0.000388	D2_MA_10_2_vs_MA_10_5	1.21
11.2	10.9	0.000388	0.000388	D2_MA_10_3_vs_MA_10_4	0.779
11.2	10.9	0.000388	0.000388	D2_MA_10_3_vs_MA_10_5	0.736
11.2	10.9	0.000388	0.000388	D2_MA_10_4_vs_MA_10_5	-0.0424
11.4	5.97	0.00624	0.00624	D4_MA_10_2_vs_MA_10_3	0.367
11.4	5.97	0.00624	0.00624	D4_MA_10_2_vs_MA_10_4	0.193
11.4	5.97	0.00624	0.00624	D4_MA_10_2_vs_MA_10_5	0.297

11.4	5.97	0.00624	0.00624	D4_MA_10_3_vs_MA_10_4	-0.174
11.4	5.97	0.00624	0.00624	D4_MA_10_3_vs_MA_10_5	-0.0699
11.4	5.97	0.00624	0.00624	D4_MA_10_4_vs_MA_10_5	0.104
11.5	8.58	0.00126	0.00126	D7_MA_10_2_vs_MA_10_3	0.17
11.5	8.58	0.00126	0.00126	D7_MA_10_2_vs_MA_10_4	0.0576
11.5	8.58	0.00126	0.00126	D7_MA_10_2_vs_MA_10_5	0.441
11.5	8.58	0.00126	0.00126	D7_MA_10_3_vs_MA_10_4	-0.113
11.5	8.58	0.00126	0.00126	D7_MA_10_3_vs_MA_10_5	0.271
11.5	8.58	0.00126	0.00126	D7_MA_10_4_vs_MA_10_5	0.383
11.9	2.02	0.152	0.152	D1_MA_10_2_vs_MA_10_3	-0.0483
11.9	2.02	0.152	0.152	D1_MA_10_2_vs_MA_10_4	0.0804
11.9	2.02	0.152	0.152	D1_MA_10_2_vs_MA_10_5	0.132
11.9	2.02	0.152	0.152	D1_MA_10_3_vs_MA_10_4	0.129
11.9	2.02	0.152	0.152	D1_MA_10_3_vs_MA_10_5	0.18
11.9	2.02	0.152	0.152	D1_MA_10_4_vs_MA_10_5	0.0517