Data Analysis in Genomics,Transcriptiomics and Preoteomics

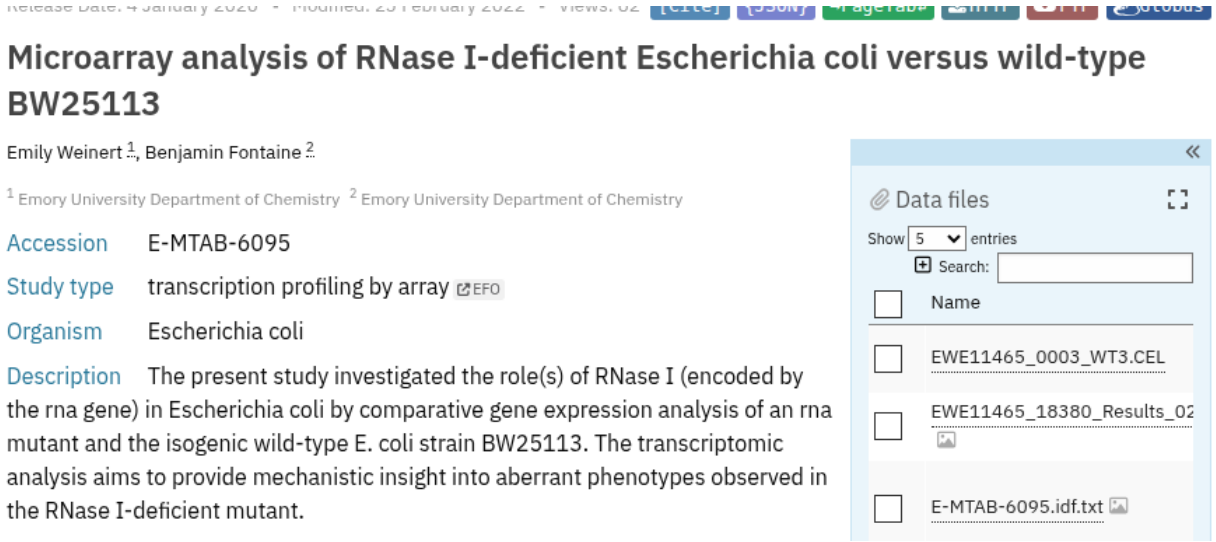
Results log

Microarray Data Analysis

Data Collection (Array express,SRA,GEO)

Data is downloaded from Array-express

ACCESSION:E-MTAB-6095



Microarray analysis of RNase I-deficient Escherichia coli versus wild-type BW25113

Emily Weinert [1], Benjamin Fontaine [2]

[1] Emory University Department of Chemistry [2] Emory University Department of Chemistry

Accession E-MTAB-6095

Study type transcription profiling by array EFO

Organism Escherichia coli

Description The present study investigated the role(s) of RNase I (encoded by the rna gene) in Escherichia coli by comparative gene expression analysis of an rna mutant and the isogenic wild-type E. coli strain BW25113. The transcriptomic analysis aims to provide mechanistic insight into aberrant phenotypes observed in the RNase I-deficient mutant.

Data files

Show 5 entries

Search:

Name

EWE11465_0003_WT3.CEL

EWE11465_18380_Results_02

E-MTAB-6095.idf.txt

Samples they used for study

Source Name	organism	strain	genotype	genotype	Label	Assay Name	Raw	Processe
drna1	Escherichia coli	BW25113	Δrna	Δrna	biotin	EWE11465_0004_drna1		
drna2	Escherichia coli	BW25113	Δrna	Δrna	biotin	EWE11465_0005_drna2		
drna3	Escherichia coli	BW25113	Δrna	Δrna	biotin	EWE11465_0006_drna3		
WT1	Escherichia coli	BW25113	wild type genotype	wild type genotype	biotin	EWE11465_0001_WT1		
WT2	Escherichia coli	BW25113	wild type genotype	wild type genotype	biotin	EWE11465_0002_WT2		
WT3	Escherichia coli	BW25113	wild type genotype	wild type genotype	biotin	EWE11465_0003_WT3		

SDRF file format(Sample and Data Relationship Format)

SDRF (Sample and Data Relationship Format) files are tab-delimited text files that provide structured metadata for microarray and other high-throughput functional genomics experiments While SDRF files don't contain the raw microarray data itself, they are essential for interpreting and analyzing it correctly.

Cel files contains raw data values

Reading celfiles and EDA of raw data

```
library(oligo)

# stores the names of cel files

celFiles = list.celfiles()

# read all the cell files into a FeatureSet object

rawData = read.celfiles(celFiles)

# get pm & mm info

mmdata = data.frame(mm(rawData))

pmdata = data.frame(pm(rawData))
```

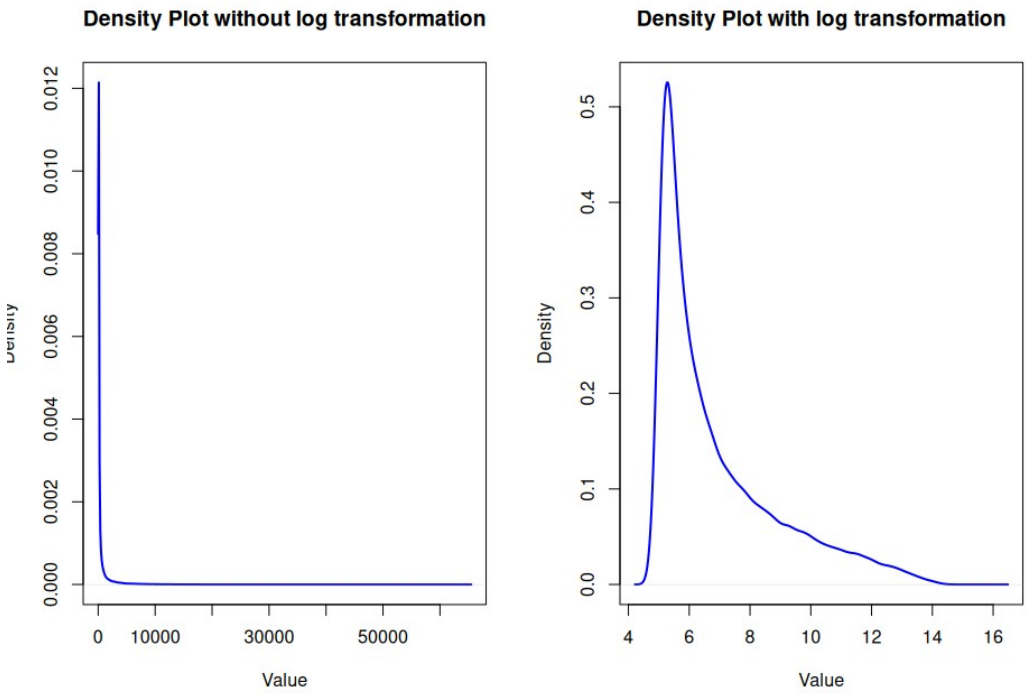
```
dim(pmdata)
dim(mmdata)
summary(mmdata)
summary(pmdata)
```

EDA of Raw data

Log Transformation vs without log transformation

plot(density(pmdata$EWE11465_0001_WT1.CEL), main="Density Plot without log transformation", xlab="Value", ylab="Density", col="blue",lwd=2)

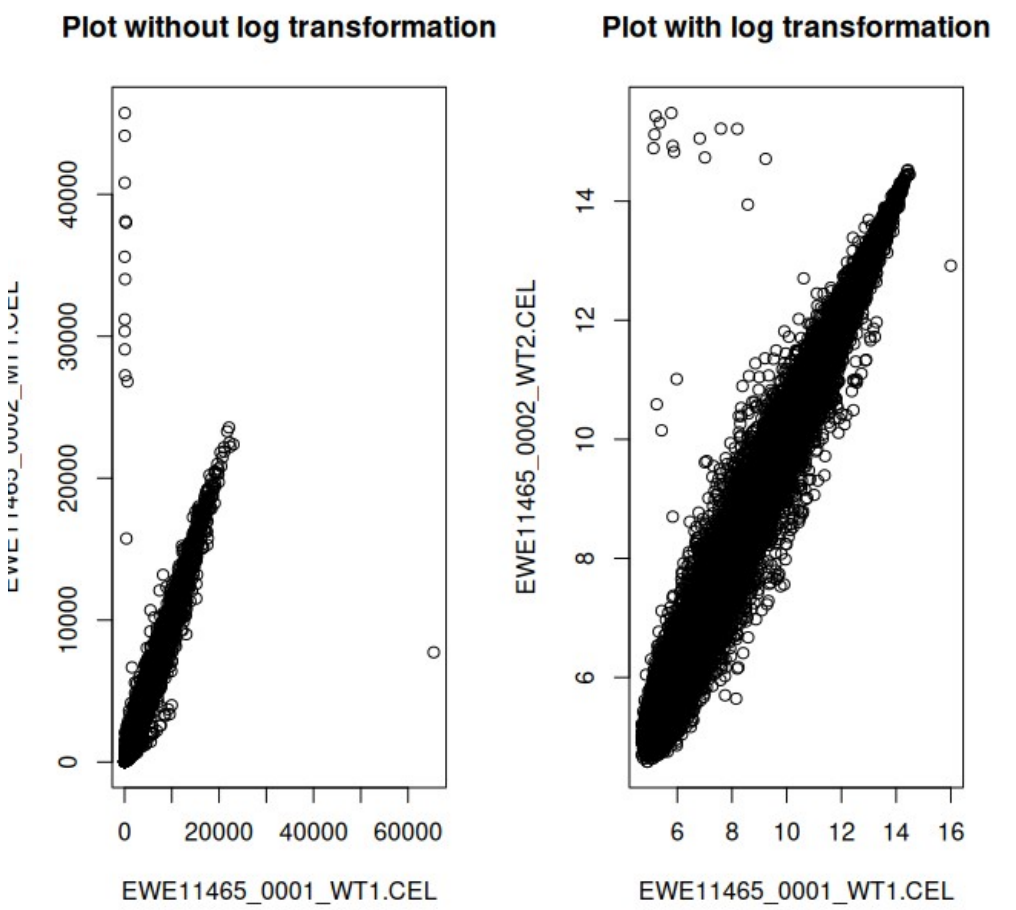
plot(density(log2(pmdata$EWE11465_0001_WT1.CEL)), main="Density Plot with log transformation", xlab="Value", ylab="Density", col="blue",lwd=2)



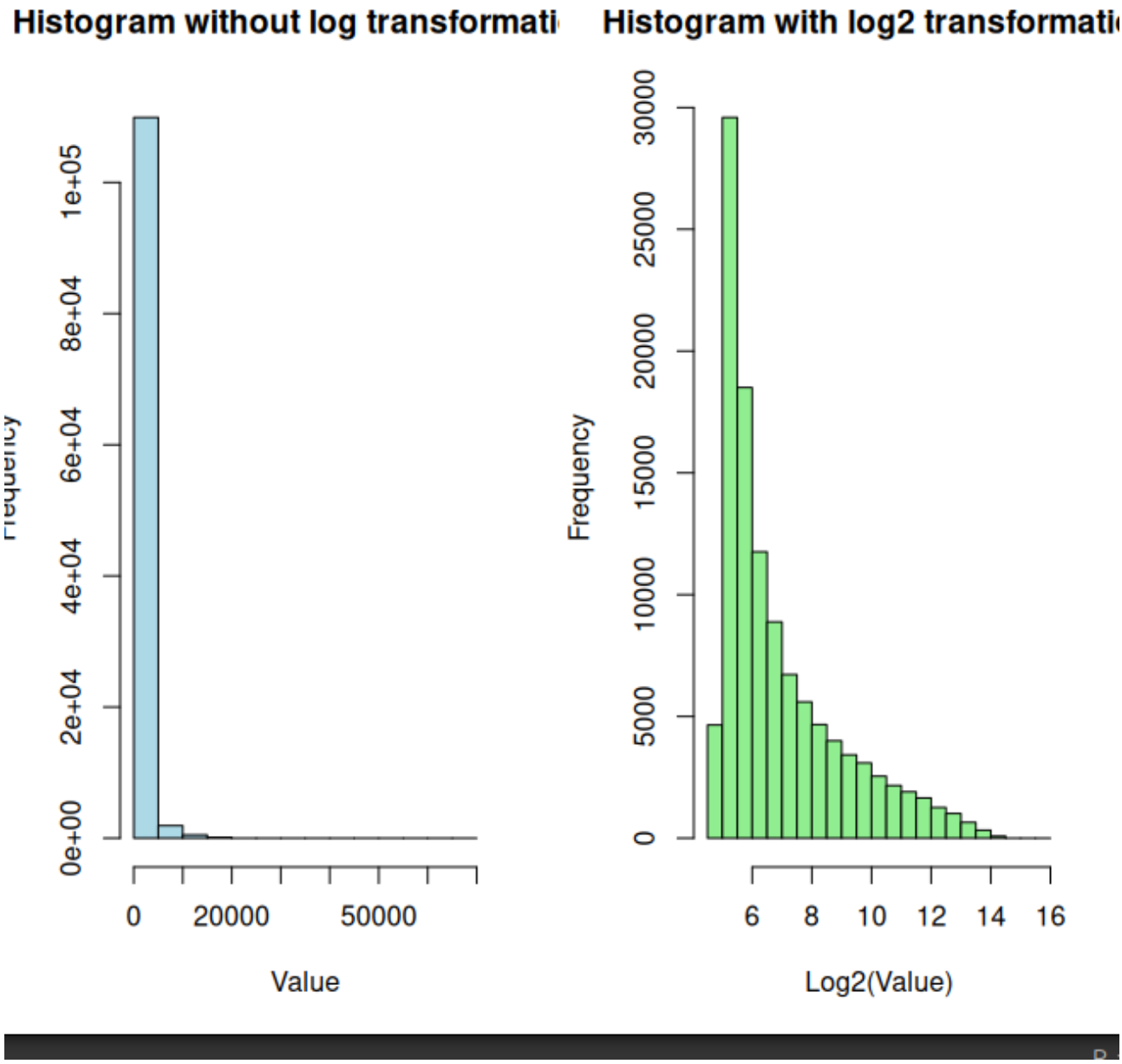
The plot on the left shows raw microarray intensities, which are highly skewed and difficult to interpret due to extreme values. After log transformation (right plot), the data distribution becomes more normalized and symmetric, making it suitable for meaningful statistical

analysis and visualization. Log transformation stabilizes variance and enhances interpretability of the expression values.

```
par( mfrow = c( 1, 2 ), oma = c( 0, 0, 2, 0 ) )
plot(pmdata$EWE11465_0001_WT1.CEL,
pmdata$EWE11465_0002_WT2.CEL,
    main="Plot without log transformation",
    xlab='EWE11465_0001_WT1.CEL', ylab='EWE11465_0002_MT1.CEL')

plot(log2(pmdata$EWE11465_0001_WT1.CEL),
log2(pmdata$EWE11465_0002_WT2.CEL),
    main="Plot with log transformation",
    xlab='EWE11465_0001_WT1.CEL', ylab='EWE11465_0002_WT2.CEL')
```



Log2 transformation of microarray data stabilizes variance and makes the data distribution more symmetric, enabling more reliable statistical analysis. It reduces the impact of extreme values seen in raw intensities and better reveals biological differences across samples.

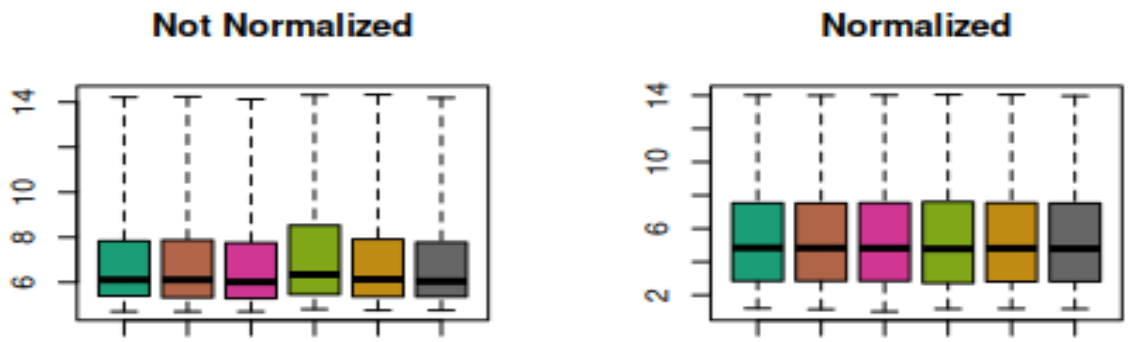


The histogram on the left shows a highly skewed distribution of raw intensity values, with most data clustered near zero and few high values, making interpretation difficult. After log2 transformation (right), the data become more evenly spread and approximately normal, which makes patterns clearer and supports more robust statistical analysis.

Normalization

RMA normalaization

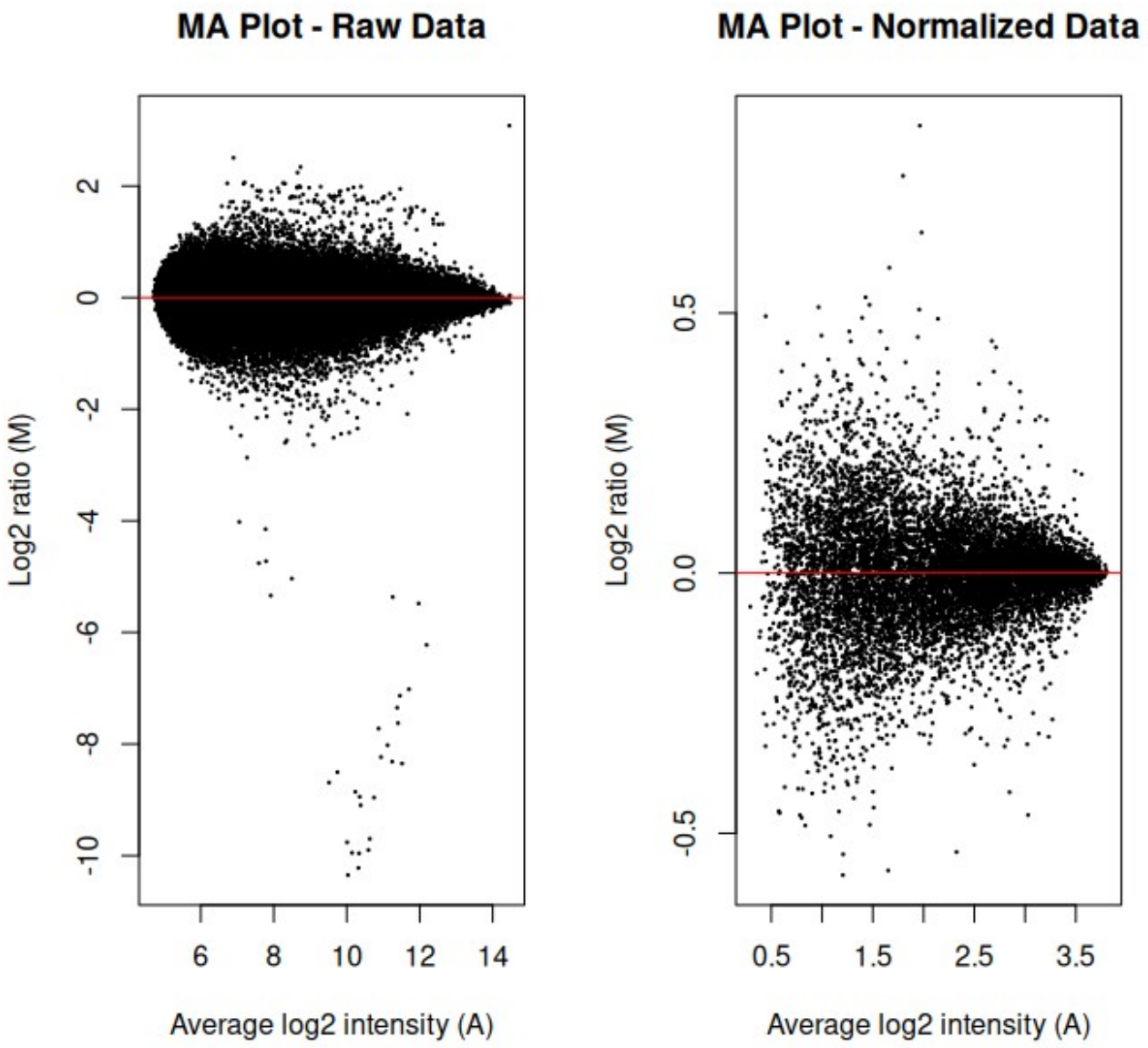
```
#box plot of the raw exp data


oligo::boxplot(raw_data, target = "core",
            main = "Boxplot of log2-intensitites for the raw data")
#box plot of the raw exp data


oligo::boxplot(rma(raw_data), target = "core",
            main = "Boxplot of log2-intensitites for the raw data")
```



The boxplots show that before normalization, sample distributions are uneven and inconsistent, indicating technical variation. After normalization, the distributions are aligned and similar across samples, which corrects for such variation and ensures comparability for downstream analysis

MA plot

```
library(oligo)
celFiles = list.celfiles()
rawData = read.celfiles(celFiles)
normData = rma(rawData)
exprs_raw = exprs(rawData)
exprs_norm = exprs(normData)
par(mfrow = c(1, 2), oma = c(0, 0, 2, 0))
# MA plot for raw data comparing sample 1 and 2
M_raw <- log2(exprs_raw[,1]) - log2(exprs_raw[,2])
A_raw <- (log2(exprs_raw[,1]) + log2(exprs_raw[,2])) / 2
plot(A_raw, M_raw, pch=20, cex=0.3, main="MA Plot - Raw Data",
    xlab="Average log2 intensity (A)", ylab="Log2 ratio (M)")
abline(h=0, col="red")
# MA plot for normalized data comparing sample 1 and 2
M_norm <- log2(exprs_norm[,1]) - log2(exprs_norm[,2])
A_norm <- (log2(exprs_norm[,1]) + log2(exprs_norm[,2])) / 2
plot(A_norm, M_norm, pch=20, cex=0.3, main="MA Plot - Normalized Data",
    xlab="Average log2 intensity (A)", ylab="Log2 ratio (M)")
abline(h=0, col="red")
```



The MA plot for raw data (left) shows high variability and bias, especially at lower intensities, making differences between samples difficult to evaluate. After normalization (right), the data points are centered and the variance is stabilized, which improves accuracy in detecting true expression differences and reduces technical artifacts. Normalization ensures that most log2 ratios are near zero, reflecting a balanced and comparable measurement across samples.

Differential gene expression

volcano plot

```
library(limma)
exprsData = exprs(normData)
group = factor(c("WT", "WT", "WT", "TRT", "TRT", "TRT"))
design = model.matrix(~ 0 + group)
colnames(design) = levels(group) # creates a table, run design & check
fit = lmFit(exprsData, design)
contrast.matrix = makeContrasts(TRTvsWT = TRT - WT, levels=design)
fit2 = contrasts.fit(fit, contrast.matrix)
fit2 = eBayes(fit2)
results = topTable(fit2, adjust="fdr", number=10)
library(annotate)
library(ecoli2.db)
library(AnnotationDbi)  # for mapIds()
# Map Gene Symbols (probe IDs -> symbols)
results$GeneSymbol = unname(mapIds(ecoli2.db,
                                   keys = rownames(results),
                                   column = "SYMBOL",
                                   keytype = "PROBEID",
                                   multiVals = "first"))
# Map Gene Names (probe IDs -> full names)
results$GeneName = unname(mapIds(ecoli2.db,
                                 keys = rownames(results),
                                 column = "GENENAME",
                                 keytype = "PROBEID",
                                 multiVals = "first"))
# Check first few rows
```

```
print(head(results))
library(ggplot2)

# volcano plot with normData
exprsData = exprs(normData)
# Create design matrix for differential expression
# Adjust the group factor based on your actual sample names
# This example assumes 3 WT and 3 treatment samples based on typical naming
group = factor(c("WT", "WT", "WT", "TRT", "TRT", "TRT"))
design = model.matrix(~ 0 + group)
colnames(design) = levels(group)
# Fit linear model
fit = lmFit(exprsData, design)
# Create contrasts (TRT vs WT)
contrast.matrix = makeContrasts(TRTvsWT = TRT - WT, levels = design)
fit2 = contrasts.fit(fit, contrast.matrix)
fit2 = eBayes(fit2)
# Get results
results = topTable(fit2, adjust = "fdr", number = Inf)  # Get all results
# Prepare data for volcano plot
volcano_data <- data.frame(
  logFC = results$logFC,
  p_value = results$P.Value,
  probe_id = rownames(results)
)
# Calculate -log10 p-values
volcano_data$neg_log10_p <- -log10(volcano_data$p_value)
# Define significance thresholds
```

```
fc_threshold <- 1.0  # |log2FC| > 1
p_threshold <- 0.05  # p-value < 0.05
# Identify significant genes
volcano_data$significant <- ifelse(
  abs(volcano_data$logFC) > fc_threshold & volcano_data$p_value < p_threshold,
  "Significant",
  "Not significant"
)
# Create the volcano plot using base R
par(mar = c(5, 5, 4, 2))  # Set margins
# Create empty plot
plot(volcano_data$logFC, volcano_data$neg_log10_p,
     type = "n",
     xlab = "log2 Fold Change",
     ylab = "-log10(p-value)",
     main = "Volcano Plot - Normalized Data",
     cex.lab = 1.2,
     cex.main = 1.5,
     xlim = c(-max(abs(volcano_data$logFC)), max(abs(volcano_data$logFC))))
# Add points for non-significant genes
points(volcano_data$logFC[volcano_data$significant == "Not significant"],
       volcano_data$neg_log10_p[volcano_data$significant == "Not significant"],
       col = "gray60", pch = 16, cex = 0.6)
# Add points for significant genes
points(volcano_data$logFC[volcano_data$significant == "Significant"],
       volcano_data$neg_log10_p[volcano_data$significant == "Significant"],
       col = "red", pch = 16, cex = 0.8)
# Add threshold lines
```

```
abline(h = -log10(p_threshold), col = "blue", lty = 2, lwd = 2)
abline(v = c(-fc_threshold, fc_threshold), col = "blue", lty = 2, lwd = 2)
# Add legend
legend("topright",
       legend = c("Significant", "Not significant"),
       col = c("red", "gray60"),
       pch = 16,
       bty = "n")
# Add counts of significant genes
sig_count <- sum(volcano_data$significant == "Significant")
total_count <- nrow(volcano_data)
text(par("usr")[1], par("usr")[4],
     paste("Significant:", sig_count, "/", total_count),
     pos = 4, col = "darkred", cex = 0.9)
```