8 Genes Expression Analysis

Your Name

2025-04-24

In this section, we designed qPCR primers for 8 selected genes to analyze their expression patterns. This part presents the workflow and steps used to evaluate their amplification efficiency.

I.Efficiency test of the 8 selected genes

##Summary of qPCR Primer Efficiency for 8 Genes

```
library(readr)
library(dplyr)
library(tibble)
library(knitr)

# Read and clean data
df1 <- read_csv("test one--qPCR primer efficiency.csv") %>%
  select(-`...10`) %>%
  mutate(Source = "test one", `Avg Ct` = suppressWarnings(as.numeric(`Avg Ct`)))

df2 <- read_csv("test two--qPCR primer efficiency.csv") %>%
  select(-`...10`) %>%
  mutate(Source = "test two", `Avg Ct` = suppressWarnings(as.numeric(`Avg Ct`)))

# Efficiency calculator
calculate_efficiency <- function(df, gene_list, source_label) {
  results <- lapply(gene_list, function(gene) {
    df_gene <- df %>% filter(`Gene name` == gene)
    if (nrow(df_gene) >= 3) {
      fit <- lm(`Avg Ct` ~ `Log (sample quantity)`, data = df_gene)
      slope <- coef(fit)[2]
      r2 <- summary(fit)$r.squared
      efficiency <- (10^(-1 / slope) - 1) * 100
      tibble(
        Gene = gene,
        Source = source_label,
        Slope = round(slope, 3),
        R_squared = round(r2, 4),
        Efficiency_percent = round(efficiency, 2)
      )
    } else {
      tibble(Gene = gene, Source = source_label,
             Slope = NA, R_squared = NA, Efficiency_percent = NA)
    }
```

```
  })
  bind_rows(results)
}

# Target genes
genes_one <- c("0082", "0109", "0208", "1580", "1654")
genes_two <- c("1597", "1610", "1624")

# Run and combine
table_one <- calculate_efficiency(df1, genes_one, "test one")
table_two <- calculate_efficiency(df2, genes_two, "test two")

efficiency_table <- bind_rows(table_one, table_two)

# Output as table
kable(efficiency_table, caption = "Efficiency Summary for 8 Selected Genes")
```

Table 1: Efficiency Summary for 8 Selected Genes

Gene	Source	Slope	R_squared	Efficiency_percent
0082	test one	-3.218	0.9772	104.53
0109	test one	-3.419	0.9677	96.10
0208	test one	-3.423	0.9610	95.95
1580	test one	-3.331	0.9603	99.62
1654	test one	-3.371	0.9687	97.99
1597	test two	-3.509	0.9887	92.74
1610	test two	-3.542	0.9509	91.57
1624	test two	-3.490	0.9775	93.43

Visualization of qPCR Primer Efficiency for 8 Genes

```
# Read data from relative paths
df1 <- read_csv("test one--qPCR primer efficiency.csv") %>%
  mutate(Source = "test one",
         `Avg Ct` = suppressWarnings(as.numeric(`Avg Ct`)))

df2 <- read_csv("test two--qPCR primer efficiency.csv") %>%
  mutate(Source = "test two",
         `Avg Ct` = suppressWarnings(as.numeric(`Avg Ct`)))

# Combine data
df_all <- bind_rows(df1, df2)

# Define plotting function
plot_gene <- function(df, gene_name) {
  df_gene <- df %>% filter(`Gene name` == gene_name)

  if (nrow(df_gene) < 2) return(ggplot() + labs(caption = paste("Gene", gene_name, "(insufficient data)

  fit <- lm(`Avg Ct` ~ `Log (sample quantity)`, data = df_gene)
```

```
  slope <- coef(fit)[2]
  r2 <- summary(fit)$r.squared
  efficiency <- (10^(-1 / slope) - 1) * 100

  ggplot(df_gene, aes(x = `Log (sample quantity)`, y = `Avg Ct`)) +
    geom_point(size = 3, color = "steelblue") +
    geom_smooth(method = "lm", se = FALSE, linetype = "dashed", color = "firebrick") +
    annotate("text", x = min(df_gene$`Log (sample quantity)`), y = max(df_gene$`Avg Ct`) - 1,
             label = paste0("Slope = ", round(slope, 3),
                            "\nR² = ", round(r2, 3),
                            "\nEfficiency = ", round(efficiency, 2), "%"),
             hjust = 0, size = 4) +
    labs(caption = paste("Gene", gene_name),
         x = "Log10(Sample Quantity)",
         y = "Average Ct") +
    theme_minimal(base_size = 12) +
    theme(plot.caption = element_text(hjust = 0.5, face = "bold"))
}
```

```
# Define genes from each source
genes_one <- c("0082", "0109", "0208", "1580", "1654")
genes_two <- c("1597", "1610", "1624")

# Generate plot list
plots <- list(
  `0082` = plot_gene(df1, "0082"),
  `0109` = plot_gene(df1, "0109"),
  `0208` = plot_gene(df1, "0208"),
  `1580` = plot_gene(df1, "1580"),
  `1654` = plot_gene(df1, "1654"),
  `1597` = plot_gene(df2, "1597"),
  `1610` = plot_gene(df2, "1610"),
  `1624` = plot_gene(df2, "1624")
)
```

```
library(ggpubr)
annotate_figure(
  ggarrange(
    plots[["0082"]], plots[["0109"]],
    plots[["0208"]], plots[["1580"]],
    plots[["1654"]], plots[["1597"]],
    plots[["1610"]], plots[["1624"]],
    ncol = 2, nrow = 4,
    labels = NULL,
    align = "hv"
  ),
  bottom = text_grob("Figure: 8 Genes qPCR Primer Efficiency Test",
                     size = 16, hjust = 0.5)
)
```

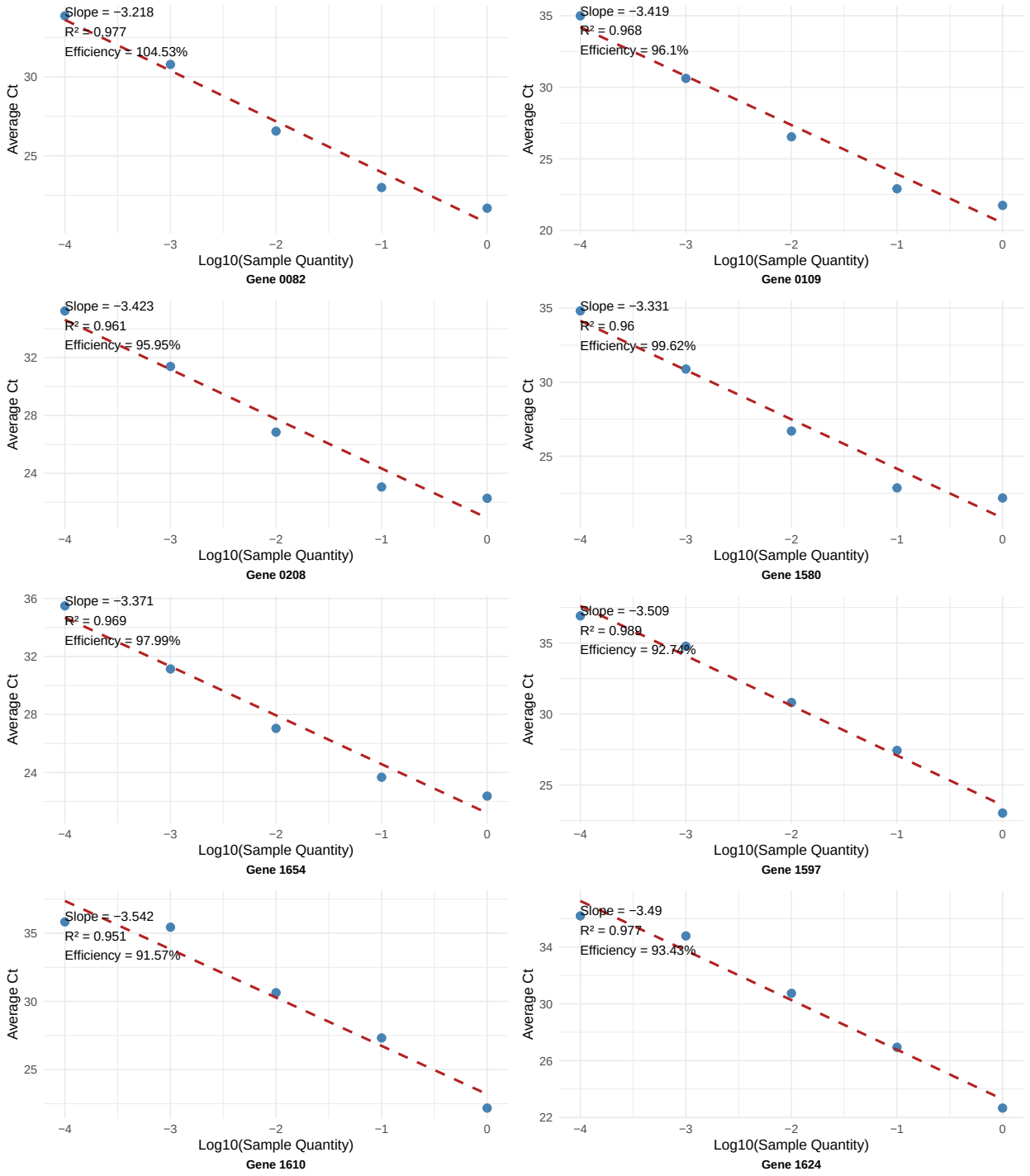

Figure: 8 Genes qPCR Primer Efficiency Test

Conclusion:

From the primer efficiency results, we observed that all eight gene primers had efficiencies ranging from 90% to 101%, indicating that they are suitable for subsequent RNA expression analysis.

In the next step, we used the Fusarium oxysporum strain 89-1A to infect cotton seedlings at the two-leaf

stage. Root samples were collected at five time points: 0 hours, 15 hours, 1 day, 2 days, and 3 days post-inoculation.

Total RNA was extracted using the E.Z.N.A.® Total RNA Kit, and gene expression analysis was performed on the collected samples.

qRT-PCR Analysis of 8 Selected Genes Across 5 Time Points

```
library(ggplot2)
library(dplyr)
library(readr)
library(ggpubr)
```

```
file_path <- "RNA expression data.csv"    ###read data
df_raw <- read_csv(file_path, col_names = FALSE)
plot_gene_expression <- function(df_raw, gene_id) {
  # Filter data for the selected gene
  df_gene <- df_raw %>%
    filter(X1 == gene_id) %>%
    mutate(
      time_point = trimws(gsub("-\\d+", "", X2)),  # Remove replicate ID (e.g., "-1", "-2")
      sample_Ct = as.numeric(X3),
      housekeeping_Ct = as.numeric(X4),
      delta_Ct = sample_Ct - housekeeping_Ct       # ΔCt = Sample Ct - Housekeeping Ct
    )

  # Calculate average ΔCt of 0h as baseline
  baseline <- df_gene %>%
    filter(time_point == "0 h") %>%
    summarise(mean_delta = mean(delta_Ct, na.rm = TRUE)) %>%
    pull()

  # Calculate ΔΔCt and relative expression (2^-ΔΔCt)
  df_gene <- df_gene %>%
    mutate(
      delta_delta_Ct = delta_Ct - baseline,
      relative_expression = 2^(-delta_delta_Ct)
    )

  # Compute mean and standard deviation for each time point
  summary_df <- df_gene %>%
    group_by(time_point) %>%
    summarise(
      mean_expression = mean(relative_expression),
      sd_expression = sd(relative_expression)
    )

  # Define time point order
  time_levels <- c("0 h", "15 h", "1 d", "2 d", "3 d")
  df_gene$time_point <- factor(df_gene$time_point, levels = time_levels)
  summary_df$time_point <- factor(summary_df$time_point, levels = time_levels)
```

```r
  # Morandi-style color palette
  morandi_colors <- c(
    "0 h"  = "#A7C3A8",
    "15 h" = "#D4A5A5",
    "1 d"  = "#B0A4C0",
    "2 d"  = "#E3CBA8",
    "3 d"  = "#A2B9B2"
  )

  # Create the expression plot
  p <- ggplot() +
    geom_col(data = summary_df, aes(x = time_point, y = mean_expression, fill = time_point),
             color = "black", alpha = 0.8, width = 0.6) +
    scale_fill_manual(values = morandi_colors) +


    # Add error bars
    geom_errorbar(data = summary_df, aes(x = time_point,
                                         ymin = mean_expression - sd_expression,
                                         ymax = mean_expression + sd_expression),
                  width = 0.2, color = "black", size = 1) +

    # Plot individual technical replicates
    geom_jitter(data = df_gene, aes(x = time_point, y = relative_expression),
                color = "black", size = 2.5, width = 0.1, alpha = 0.8) +

    # Add axis labels and title
    labs(
      title = bquote(italic(.(gene_id))),
      x = "Time Points",
      y = expression(2^-Delta*Delta*Ct)
    ) +

    # Clean visual theme
    theme_classic() +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
      axis.text.y = element_text(size = 10),
      plot.title = element_text(size = 15, face = "bold"),
      axis.title.x = element_text(size = 11),
      axis.title.y = element_text(size = 11)
    )

  return(p)
}
```

```r
# Define gene list and create plots
gene_ids <- c("1597", "1610", "1624", "0082", "1580", "0109")
gene_plots <- lapply(gene_ids, function(id) plot_gene_expression(df_raw, id))

# Display all plots in 2 rows × 3 columns layout
ggarrange(
  plotlist = gene_plots,
```

```
  ncol = 2, nrow = 3,
  common.legend = TRUE,
  legend = "bottom"
)
```