Dictyostelium discoideum complete life cycle proteomics

R code

R version 4.3.3

Imputation

#MinProb imputation

```
install.packages("BiocManager")
BiocManager::install("MSnbase")
install.packages("imputeLCMD")

library(MSnbase)
library(tidyverse)
library(imputeLCMD)
library(MSnbase)
library(tidyverse)

data <- read.csv("Dublicates_corrected_data.csv")
abundance_cols <- c(
  "Abundances_F1_Vegetative", "Abundances_F1_Aggregation",
  "Abundances_F1_Mound", "Abundances_F1_Culmination",
  "Abundances_F1_Fruitingbody", "Abundances_F2_Vegetative",
  "Abundances_F2_Aggregation", "Abundances_F2_Mound",
  "Abundances_F2_Culmination", "Abundances_F2_Fruitingbody"
)
# log2-transform abundance matrix
abundance_matrix <- as.matrix(data[, abundance_cols])
log_abundance <- log2(abundance_matrix)
log_abundance[!is.finite(log_abundance)] <- NA
rownames(log_abundance) <- make.names(1:nrow(log_abundance), unique = TRUE)
msnset <- MSnSet(exprs = log_abundance)
# Impute missing values using MinProb
msnset_imputed <- impute(msnset, method = "MinProb")
# Back-transform to original scale
imputed_matrix <- 2^exprs(msnset_imputed)
data[, abundance_cols] <- imputed_matrix
```

```
write.csv(data, "Imputed_data.csv", row.names = FALSE)
```

Correlation between biological duplicates
#Spearman correlation between biological duplicates for each developmental stage.
#Raw, non-imputed data for which both replicates were available were used for the analysis.
#distribution and correlation test for each stage #Correlation between biological duplicates (per stage)

```
library(ggplot2)
library(dplyr)
lfq <- read.csv("Consolidated_raw_data.csv")
stages <- list(
 V = c("Abundances_F1_Vegetative",   "Abundances_F2_Vegetative"),
 A = c("Abundances_F1_Aggregation",  "Abundances_F2_Aggregation"),
 M = c("Abundances_F1_Mound",        "Abundances_F2_Mound"),
 C = c("Abundances_F1_Culmination",  "Abundances_F2_Culmination"),
 F = c("Abundances_F1_Fruitingbody", "Abundances_F2_Fruitingbody")
)

dir.create("Stagewise_Correlation_Plots", showWarnings = FALSE)
for (s in names(stages)) {
 pair <- stages[[s]]

 df <- lfq %>%
  select(Accession, all_of(pair)) %>%
  filter(!is.na(.data[[pair[1]]]) & !is.na(.data[[pair[2]]])) %>%
  mutate(
   F1_log2 = log2(.data[[pair[1]]] + 1),
   F2_log2 = log2(.data[[pair[2]]] + 1)
  )
 n_proteins <- nrow(df)
 if (n_proteins > 2) {
  cor_test <- cor.test(df$F1_log2, df$F2_log2, method = "spearman")
  rho <- cor_test$estimate
  pval <- cor_test$p.value
  if (pval < 2.2e-16) pval <- "<2.2e−16"
  lm_fit <- lm(F2_log2 ~ F1_log2, data = df)
  slope <- coef(lm_fit)[2]
  intercept <- coef(lm_fit)[1]
  p <- ggplot(df, aes(x = F1_log2, y = F2_log2)) +
   geom_point(alpha = 0.6, color = "steelblue") +
   geom_smooth(method = "lm", se = FALSE, color = "black",
               linetype = "dashed", linewidth = 0.8) +
      labs(
        title = paste0("Stage ", s, " (n = ", n_proteins, ")"),
        subtitle = paste0("Spearman ρ = ", round(rho, 3),
                          ", p ", pval,
                          " | slope = ", round(slope, 3)),
        x = paste0("log₂(F1_", s, " abundance)"),
        y = paste0("log₂(F2_", s, " abundance)")
      ) +
      theme_minimal(base_family = "sans", base_size = 14) +
      theme(
        panel.border = element_rect(color = "black", fill = NA, linewidth = 0.5),
        panel.grid.minor = element_blank(),
        plot.title = element_text(face = "bold")
      )
    tiff_file <- paste0("Stagewise_Correlation_Plots/Spearman_", s, "_log2.tiff")
    svg_file  <- paste0("Stagewise_Correlation_Plots/Spearman_", s, "_log2.svg")
    ggsave(filename = tiff_file, plot = p,
           width = 3, height = 3, dpi = 300, compression = "lzw")
    ggsave(filename = svg_file, plot = p, width = 3, height = 3)

  }
}
```

Venn plot

```
library(VennDiagram)
library(grid)
data <- read.csv("Consolidated_raw_data.csv", stringsAsFactors = FALSE)
stages <- list(
  Vegetative = grep("Vegetative", colnames(data), value = TRUE),
  Aggregation = grep("Aggregation", colnames(data), value = TRUE),
  Mound = grep("Mound", colnames(data), value = TRUE),
  Culmination = grep("Culmination", colnames(data), value = TRUE),
  Fruitingbody = grep("Fruitingbody", colnames(data), value = TRUE)
)

accessions <- data$Accession
expressed_proteins <- lapply(stages, function(cols) {
  apply(data[, cols], 1, function(x) any(!is.na(x)))
})
stage_lists <- lapply(expressed_proteins, function(logical_vec) accessions[logical_vec])
colors <- c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd")
```

```
tiff("Venn_Proteins_5Stages.tiff", width = 9, height = 9, units = "in", res = 300)

venn.plot <- venn.diagram(
  x = stage_lists,
  category.names = c("Vegetative", "Aggregation", "Mound", "Culmination", "Fruiting body"),
  filename = NULL,
  fill = colors,
  alpha = 0.6,
  lwd = 1.2,
  cex = 1.3,
  cat.cex = 1.4,
  cat.fontface = "bold",
  fontfamily = "sans",
  cat.fontfamily = "sans",
  fontface = "bold",
  margin = 0.1,
  cat.pos = c(-8, 15, 0, -8, 18),
  cat.dist = c(0.06, 0.06, 0.06, 0.05, 0.06),
  euler.d = FALSE,
  scaled = FALSE
)
grid.draw(venn.plot)
dev.off()

#to save Venn classification

protein_details_list <- lapply(names(stage_lists), function(stage) {
  data.frame(Accession = stage_lists[[stage]], Stage = stage)
})
all_proteins <- do.call(rbind, protein_details_list)
protein_sets <- lapply(stage_lists, unique)
intersections <- list(
  "Vegetative" = protein_sets$Vegetative,
  "Aggregation" = protein_sets$Aggregation,
  "Mound" = protein_sets$Mound,
  "Culmination" = protein_sets$Culmination,
  "Fruitingbody" = protein_sets$Fruitingbody,

  # Two-way
  "Vegetative_Aggregation" = intersect(protein_sets$Vegetative, protein_sets$Aggregation),
  "Vegetative_Mound" = intersect(protein_sets$Vegetative, protein_sets$Mound),
  "Vegetative_Culmination" = intersect(protein_sets$Vegetative, protein_sets$Culmination),
  "Vegetative_Fruitingbody" = intersect(protein_sets$Vegetative, protein_sets$Fruitingbody),

  # Three-way
  "Vegetative_Aggregation_Mound" = Reduce(intersect, list(protein_sets$Vegetative,
protein_sets$Aggregation, protein_sets$Mound)),
  "Vegetative_Aggregation_Culmination" = Reduce(intersect, list(protein_sets$Vegetative,
protein_sets$Aggregation, protein_sets$Culmination)),
  "Vegetative_Aggregation_Fruitingbody" = Reduce(intersect, list(protein_sets$Vegetative,
protein_sets$Aggregation, protein_sets$Fruitingbody)),

  # Four-way
  "Vegetative_Aggregation_Mound_Culmination" = Reduce(intersect,
list(protein_sets$Vegetative, protein_sets$Aggregation, protein_sets$Mound,
protein_sets$Culmination)),
  # Five-way intersection
  "All_Stages" = Reduce(intersect, protein_sets)
)

#
venn_classification <- list()
for (protein in unique(all_proteins$Accession)) {
  classification <- character(0)
  for (name in names(intersections)) {
    if (protein %in% intersections[[name]]) {
      classification <- c(classification, name)
    }
  }
  if (length(classification) > 0) {
    venn_classification[[protein]] <- classification
  }
}

#Unique and common protein list only
unique_proteins <- list()
for (stage in names(stage_lists)) {
  current_stage_proteins <- protein_sets[[stage]]
  other_stage_proteins <- unlist(protein_sets[names(protein_sets) != stage])
  unique_to_stage <- setdiff(current_stage_proteins, other_stage_proteins)
  unique_proteins[[stage]] <- unique_to_stage
}
common_proteins <- Reduce(intersect, protein_sets)
unique_proteins_df <- data.frame(
  Stage = rep(names(unique_proteins), sapply(unique_proteins, length)),
  Accession = unlist(unique_proteins),
  stringsAsFactors = FALSE
)
common_proteins_df <- data.frame(
  Stage = rep("All_Stages", length(common_proteins)),
  Accession = common_proteins,
  stringsAsFactors = FALSE
)
write.csv(unique_proteins_df, "Unique_Proteins_Per_Stage_200625.csv", row.names =
FALSE)
"Common_Proteins_Among_All_Stages_200625.csv", row.names = FALSE)
head(unique_proteins_df)
head(common_proteins_df)
venn_classification_df <- data.frame(
  Accession = names(venn_classification),
  Classification = sapply(venn_classification, function(x) paste(x, collapse = ", ")),
  stringsAsFactors = FALSE
)
write.csv(venn_classification_df, "Venn_Protein_Classification.csv", row.names = FALSE)
#to merge the gene ID and other information
minprob_data <- read.csv("Imputed_data.csv", stringsAsFactors = FALSE)
unique_proteins_data <- read.csv("Unique_Proteins_Per_Stage_200625.csv",
stringsAsFactors = FALSE)
common_proteins_data <- read.csv("Common_Proteins_Among_All_Stages_200625.csv",
stringsAsFactors = FALSE)
merged_unique <- merge(unique_proteins_data, minprob_data, by.x = "Accession", by.y =
"Accession", all.x = TRUE)
Continuous expression proteome <- merge(common_proteins_data, minprob_data, by.x =
"Accession", by.y = "Accession", all.x = TRUE)
write.csv(merged_unique, "Merged_Unique_Proteins_with_Minprob_200625.csv",
row.names = FALSE)
proteome.csv", row.names = FALSE)
```

PCA on raw (non-imputed) proteomics data

```
library(ggplot2)
library(dplyr)

pca_data <- read.csv("Consolidated_raw_data.csv")
abundance_columns <- c("Abundances_F1_Vegetative", "Abundances_F2_Vegetative",
                "Abundances_F1_Aggregation", "Abundances_F2_Aggregation",
                   "Abundances_F1_Mound", "Abundances_F2_Mound",
                   "Abundances_F1_Culmination", "Abundances_F2_Culmination",
                   "Abundances_F1_Fruitingbody", "Abundances_F2_Fruitingbody")

pca_filtered <- pca_data %>%
  select(all_of(abundance_columns)) %>%
  filter(if_all(everything(), ~ !is.na(.)))
cat("Proteins included in PCA (complete across all replicates/stages):",
    nrow(pca_filtered), "\n")
#PCA matrix
pca_matrix <- t(pca_filtered)
pca_result <- prcomp(pca_matrix, center = TRUE, scale. = TRUE)
pca_scores <- as.data.frame(pca_result$x)
pca_scores$Stage <- c("V1", "V2", "A1", "A2", "M1", "M2", "C1", "C2", "F1", "F2")
stage_colors <- c("V1" = "#1f77b4", "V2" = "#1f77b4",
              "A1" = "#ff7f0e", "A2" = "#ff7f0e",
              "M1" = "#2ca02c", "M2" = "#2ca02c",
              "C1" = "#d62728", "C2" = "#d62728",
              "F1" = "#9467bd", "F2" = "#9467bd")

pc_var <- summary(pca_result)$importance["Proportion of Variance", 1:2] * 100
x_lab <- paste0("PC1: ", round(pc_var[1], 1), "% variance")
y_lab <- paste0("PC2: ", round(pc_var[2], 1), "% variance")
pca_plot <- ggplot(pca_scores, aes(x = PC1, y = PC2, color = Stage)) +
  geom_point(size = 4) +
  scale_color_manual(values = stage_colors) +
  scale_x_continuous(
    name = x_lab,
    limits = c(-50, 50),
    breaks = seq(-50, 50, by = 25)
  ) +
  scale_y_continuous(
    name = y_lab,
    limits = c(-30, 30),
    breaks = seq(-30, 30, by = 10)
  ) +
  theme_minimal() +
  theme(
    axis.title = element_text(size = 12, family = "sans"),
    axis.text = element_text(size = 10, family = "sans"),
    legend.position = "none",
    plot.title = element_blank(),
    text = element_text(family = "sans")
```

```
 )

ggsave("PCA_plot_RawData_noLabels.tiff", plot = pca_plot,
       device = "tiff", width = 6, height = 5, dpi = 300)


#PCA plot (with stage labels)
pca_plot_label <- ggplot(pca_scores, aes(x = PC1, y = PC2, color = Stage, label = Stage)) +
  geom_point(size = 4) +
  geom_text(vjust = -1.2, size = 4, fontface = "bold", family = "sans") +
  scale_color_manual(values = stage_colors) +
  scale_x_continuous(
    name = x_lab,
    limits = c(-50, 50),
    breaks = seq(-50, 50, by = 25)
  ) +
  scale_y_continuous(
    name = y_lab,
    limits = c(-30, 30),
    breaks = seq(-30, 30, by = 10)
  ) +
  theme_minimal() +
  theme(
    axis.title = element_text(size = 12, family = "sans"),
    axis.text = element_text(size = 10, family = "sans"),
    legend.position = "none",
    text = element_text(family = "sans")
  )
ggsave("PCA_plot_RawData_withLabels.tiff", plot = pca_plot_label,
       device = "tiff", width = 6, height = 5, dpi = 300)
```

Heatmap

```
install.packages("pheatmap")
install.packages("RColorBrewer")

library(pheatmap)
library(RColorBrewer)
lfq_data <- read.csv("Imputed_data.csv")
abundance_data <- lfq_data[, c("V1_MinC", "V2_MinC", "A1_MinC", "A2_MinC",
                               "M1_MinC", "M2_MinC", "C1_MinC", "C2_MinC",
                               "F1_MinC", "F2_MinC")]
```

```
combined_data <- data.frame(
  Vegetative   = rowMeans(abundance_data[, c("V1_MinC", "V2_MinC")], na.rm = TRUE),
  Aggregation  = rowMeans(abundance_data[, c("A1_MinC", "A2_MinC")], na.rm = TRUE),
  Mound        = rowMeans(abundance_data[, c("M1_MinC", "M2_MinC")], na.rm = TRUE),
  Culmination  = rowMeans(abundance_data[, c("C1_MinC", "C2_MinC")], na.rm = TRUE),
  Fruitingbody = rowMeans(abundance_data[, c("F1_MinC", "F2_MinC")], na.rm = TRUE)
)
# Z-score normalization
zscore_data <- t(apply(combined_data, 1, function(x) {
  if (sd(x, na.rm = TRUE) == 0) return(rep(0, length(x)))
  (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
}))
#Row clustering and cluster annotation
row_dist <- dist(zscore_data)
row_hclust <- hclust(row_dist, method = "ward.D")
clusters <- cutree(row_hclust, k = 5)
rownames(zscore_data) <- paste0("Prot_", seq_len(nrow(zscore_data)))
annotation_row <- data.frame(ClusterGroup = as.factor(clusters))
rownames(annotation_row) <- rownames(zscore_data)
cluster_palette <- brewer.pal(5, "Set1")
names(cluster_palette) <- levels(annotation_row$ClusterGroup)
annotation_colors <- list(ClusterGroup = cluster_palette)
custom_breaks <- c(-Inf, -1.5, -0.5, -0.2, 0.2, 0.5, 1.5, Inf)
custom_colors <- c("#9ABDDC", "#F5FBFF", "lightyellow", "white", "lightyellow", "#FFFBC8",
"#FF6242")

tiff("heatmapzscore_clustered_with_labeltree_200625.tiff", width = 3000, height = 4000, res =
300, compression = "lzw")
pheatmap(zscore_data,
        cluster_rows = row_hclust,
        cluster_cols = FALSE,
        color = custom_colors,
        breaks = custom_breaks,
        show_rownames = FALSE,
        show_colnames = TRUE,
        fontsize = 12,
        treeheight_row = 80,
        annotation_row = annotation_row,
        annotation_colors = annotation_colors,
        main = "")
dev.off()
#heatmap without cluster lines and labels
# heatmap without annotation colors and column labels
```

```
tiff("heatmapzscore.tiff", width = 3000, height = 4000, res = 300, compression = "lzw")
pheatmap(zscore_data,
        cluster_rows = row_hclust,
        cluster_cols = FALSE,
        color = custom_colors,
        breaks = custom_breaks,
        show_rownames = FALSE,
        show_colnames = FALSE,
        fontsize = 12,
        treeheight_row = 80,
        main = "")
dev.off()
```

Bootstrap dendogram for protein clustering

```
install.packages("readr")
install.packages("pvclust", dependencies = TRUE)

library(readr)
library(pvclust)
lfq_data <- read.csv("Imputed_data.csv", row.names = 1)
abundance_data <- lfq_data[, c("V1_MinC", "V2_MinC",
                          "A1_MinC", "A2_MinC",
                          "M1_MinC", "M2_MinC",
                          "C1_MinC", "C2_MinC",
                          "F1_MinC", "F2_MinC")]
stage_avg <- data.frame(
  V = rowMeans(abundance_data[, c("V1_MinC", "V2_MinC")], na.rm = TRUE),
  A = rowMeans(abundance_data[, c("A1_MinC", "A2_MinC")], na.rm = TRUE),
  M = rowMeans(abundance_data[, c("M1_MinC", "M2_MinC")], na.rm = TRUE),
  C = rowMeans(abundance_data[, c("C1_MinC", "C2_MinC")], na.rm = TRUE),
  F = rowMeans(abundance_data[, c("F1_MinC", "F2_MinC")], na.rm = TRUE)
)
spearman_dist <- function(x) {
  as.dist(1 - cor(x, method = "spearman"))
}
result <- pvclust(stage_avg,
                method.hclust = "average",
                method.dist = spearman_dist,
                nboot = 1000)
tiff("pvclust_dendrogram.tiff", width = 4000, height = 2000, res = 600)
plot(result)
pvrect(result, alpha = 0.95)
```

```
dev.off()

# dendrogram without labels, bootstrap numbers, or p-values
tiff("pvclust_horizontal_tree.tiff",
    width = 4000, height = 2000, res = 600, compression = "lzw")
plot(result,
    main = "",
    labels = FALSE,
    print.num = FALSE,
    print.pv = FALSE,
    cex = 1,
    horiz = TRUE)  # horizontal tree
dev.off()
```

Pairwise comparisons of protein expression between development stages

```
library(limma)
library(dplyr)
library(tibble)  # To use rownames_to_column()
exprs_matrix <- read.csv("Imputed_data.csv")
exprs_matrix$Accession <- as.character(exprs_matrix$Accession)
exprs_matrix_log <- exprs_matrix %>%
  mutate_at(vars(V1_MinC:F2_MinC), ~ log2(. + 1))
exprs_matrix_log <- exprs_matrix_log %>%
  column_to_rownames(var = "Accession")
exprs_matrix_numeric <- exprs_matrix_log %>%
  select(starts_with("V1_MinC"):starts_with("F2_MinC"))
stage_names <- c("Vegetative", "Vegetative", "Aggregation", "Aggregation",
              "Mound", "Mound", "Culmination", "Culmination",
              "FruitingBody", "FruitingBody")
stages <- factor(stage_names, levels = c("Vegetative", "Aggregation", "Mound",
"Culmination", "FruitingBody"))
design <- model.matrix(~ 0 + stages)
colnames(design) <- levels(stages)
fit <- lmFit(exprs_matrix_numeric, design)

# contrasts for pairwise comparisons
contrast_matrix <- makeContrasts(
  Agg_vs_Veg = Aggregation - Vegetative,
  Mound_vs_Agg = Mound - Aggregation,
  Culm_vs_Mound = Culmination - Mound,
  Fruit_vs_Culm = FruitingBody - Culmination,
  Veg_vs_Fruit = Vegetative - FruitingBody,
  Fruit_vs_Veg = FruitingBody - Vegetative,
  levels = design
)

fit2 <- contrasts.fit(fit, contrast_matrix)
fit2 <- eBayes(fit2)
results_full <- topTable(fit2, coef = NULL, adjust = "BH", number = Inf)
results_full$Accession <- rownames(results_full)
results_full <- left_join(results_full, exprs_matrix %>%
                          select(Accession, Description, UniProt_ID, DDB_ID, DDB_GID,
Protein_names, Gene_Names),
write.csv(results_full, "Pairwise_230625.csv", row.names = TRUE)
results_list <- list()
for (contrast in colnames(contrast_matrix)) {
  results_list[[contrast]] <- topTable(fit2, coef = contrast, adjust = "BH", number = Inf, sort.by =
"logFC") %>%
    dplyr::mutate(Comparison = contrast)
  results_list[[contrast]] <- left_join(results_list[[contrast]],
                                        exprs_matrix %>% select(Accession, Description, UniProt_ID,
DDB_ID, DDB_GID, Protein_names, Gene_Names),
                                        by = "Accession")
  write.csv(results_list[[contrast]], paste0("Pairwise", contrast, ".csv"), row.names = FALSE)
}
```

Paiwise data counting

```
library(readr)
library(dplyr)
# File list
files <- c(
  "Pairwise_Agg_vs_Veg.csv",
  "Pairwise_Culm_vs_Mound.csv",
  "Pairwise_Fruit_vs_Culm.csv",
  "Pairwise_Mound_vs_Agg.csv",
  "Pairwise_Veg_vs_Fruit.csv"
)
summary_table <- data.frame()
for (file in files) {
  df <- read_csv(file, show_col_types = FALSE)
  logfc <- df[[2]]
  adj_p <- df$adj.P.Val
  up_all        <- sum(logfc >= 1, na.rm = TRUE)
  down_all      <- sum(logfc <= -1, na.rm = TRUE)
  unchanged     <- sum(logfc > -1 & logfc < 1, na.rm = TRUE)
  sig_up        <- sum(logfc >= 1 & adj_p < 0.05, na.rm = TRUE)
  sig_down      <- sum(logfc <= -1 & adj_p < 0.05, na.rm = TRUE)
  sig_unchanged <- sum(logfc > -1 & logfc < 1 & adj_p < 0.05, na.rm = TRUE)
  comparison <- gsub("Pairwise_|.csv", "", file)
  summary_table <- rbind(summary_table, data.frame(
    Comparison = comparison,
    Up_1_or_more = up_all,
    Down_1_or_less = down_all,
    Unchanged = unchanged,
    Sig_Up = sig_up,
    Sig_Down = sig_down,
    Sig_Unchanged = sig_unchanged
  ))
}
print(summary_table)
write_csv(summary_table, "logFC_significance_summary.csv")
```

pairwise comparison plot

```
library(readr)
library(dplyr)
# File list
files <- c(
  "Pairwise_Agg_vs_Veg.csv",
  "Pairwise_Culm_vs_Mound.csv",
  "Pairwise_Fruit_vs_Culm.csv",
  "Pairwise_Mound_vs_Agg.csv",
  "Pairwise_Veg_vs_Fruit.csv"
)
# Process each file
for (file in files) {
  message("Processing: ", file)
  df <- read_csv(file, show_col_types = FALSE)
  df <- df %>%
    mutate(
      Avg_OP = rowMeans(select(., 15, 16), na.rm = TRUE),
      Avg_QR = rowMeans(select(., 17, 18), na.rm = TRUE),
      SD_OP = apply(select(., 15, 16), 1, sd, na.rm = TRUE),
      SD_QR = apply(select(., 17, 18), 1, sd, na.rm = TRUE)
    )
  out_file <- sub(".csv$", "_withAvg_SD.csv", file)
  write_csv(df, out_file)
}

library(readr)
library(dplyr)
library(ggplot2)
library(tidyr)
library(forcats)

file_info <- list(
  list(file = "Pairwise_Agg_vs_Veg.csv",
       group1 = c("A1_MinC", "A2_MinC"), g1_label = "Agg (A)",
       group2 = c("V1_MinC", "V2_MinC"), g2_label = "Veg (V)",
       title = "Agg vs Veg", output = "Agg_vs_Veg_TopBottom10.tiff"),

  list(file = "Pairwise_Culm_vs_Mound.csv",
       group1 = c("C1_MinC", "C2_MinC"), g1_label = "Culm (C)",
       group2 = c("M1_MinC", "M2_MinC"), g2_label = "Mound (M)",
       title = "Culm vs Mound", output = "Culm_vs_Mound_TopBottom10.tiff"),

  list(file = "Pairwise_Fruit_vs_Culm.csv",
       group1 = c("F1_MinC", "F2_MinC"), g1_label = "Fruit (F)",
       group2 = c("C1_MinC", "C2_MinC"), g2_label = "Culm (C)",
       title = "Fruit vs Culm", output = "Fruit_vs_Culm_TopBottom10.tiff"),

  list(file = "Pairwise_Mound_vs_Agg.csv",
       group1 = c("M1_MinC", "M2_MinC"), g1_label = "Mound (M)",
       group2 = c("A1_MinC", "A2_MinC"), g2_label = "Agg (A)",
       title = "Mound vs Agg", output = "Mound_vs_Agg_TopBottom10_log10.tiff"),

  list(file = "Pairwise_Veg_vs_Fruit.csv",
       group1 = c("V1_MinC", "V2_MinC"), g1_label = "Veg (V)",
       group2 = c("F1_MinC", "F2_MinC"), g2_label = "Fruit (F)",
       title = "Veg vs Fruit", output = "Veg_vs_Fruit_TopBottom10_log10.tiff")
)
custom_colors <- c(
  "Veg (V)" = "#1f77b4",
  "Agg (A)" = "#ff7f0e",
  "Mound (M)" = "#2ca02c",
  "Culm (C)" = "#d62728",
  "Fruit (F)" = "#9467bd"
)
```

```
all_plot_data <- list()
for (info in file_info) {
  df <- read_csv(info$file, show_col_types = FALSE)

  df <- df %>%
    filter(!is.na(logFC)) %>%
    mutate(
      Avg_G1 = rowMeans(select(., all_of(info$group1)), na.rm = TRUE),
      Avg_G2 = rowMeans(select(., all_of(info$group2)), na.rm = TRUE),
      SD_G1 = apply(select(., all_of(info$group1)), 1, sd, na.rm = TRUE),
      SD_G2 = apply(select(., all_of(info$group2)), 1, sd, na.rm = TRUE)
    )
  top10 <- df %>%
    arrange(desc(logFC)) %>%
    slice_head(n = 10) %>%
    arrange(desc((Avg_G1 + Avg_G2)/2))
  bottom10 <- df %>%
    arrange(logFC) %>%
    slice_head(n = 10) %>%
    arrange(desc((Avg_G1 + Avg_G2)/2))
  spacer <- top10[1, ]
  spacer[1, ] <- NA
  spacer$DDB_GID <- " "

  plot_data <- bind_rows(top10, spacer, bottom10)
  plot_data$PlotLabel <- ifelse(is.na(plot_data$Gene_Names), " ", plot_data$Gene_Names)

  plot_long <- plot_data %>%
    select(PlotLabel, Avg_G1, Avg_G2, SD_G1, SD_G2) %>%
    pivot_longer(cols = c(Avg_G1, Avg_G2),
                 names_to = "Condition", values_to = "Avg") %>%
    mutate(
      SD = ifelse(Condition == "Avg_G1", SD_G1, SD_G2),
      Condition = recode(Condition,
                         "Avg_G1" = info$g1_label,
                         "Avg_G2" = info$g2_label),
      Condition = factor(Condition, levels = c(info$g1_label, info$g2_label))
    )
  plot_long <- plot_long %>%
    mutate(
      Avg_log = log10(Avg),
      SD_log  = SD / (Avg * log(10))  # delta method: SE of log10(Avg)
    )
  all_plot_data[[length(all_plot_data) + 1]] <- plot_long
  output_csv_file <- sub("\\.tiff$", ".csv", info$output)
  write_csv(plot_long, output_csv_file)
  ordered_labels <- c(top10$Gene_Names, " ", bottom10$Gene_Names)
  plot_long$PlotLabel <- factor(plot_long$PlotLabel, levels = rev(ordered_labels))

  pd <- position_dodge(width = 0.8)
  p <- ggplot(plot_long, aes(x = Avg_log, y = PlotLabel, fill = Condition)) +
    geom_bar(stat = "identity", position = pd, width = 0.7, alpha = 0.8, na.rm = TRUE) +
    geom_errorbar(aes(xmin = Avg_log - SD_log, xmax = Avg_log + SD_log),
                  position = pd, width = 0.3, color = "black", size = 1, na.rm = TRUE) +
    labs(x = NULL, y = NULL, title = NULL) +
    scale_x_continuous(limits = c(0, 10)) +
    scale_fill_manual(values = custom_colors) +
    theme_minimal(base_size = 50) +
    theme(
      text = element_text(color = "black", family = "sans"),
      axis.text = element_text(color = "black", family = "sans"),
      axis.title = element_text(color = "black", family = "sans"),
      plot.title = element_text(color = "black", family = "sans"),
      legend.text = element_text(color = "black", family = "sans"),
      legend.title = element_text(color = "black", family = "sans"),
      axis.ticks = element_blank(),
      legend.position = "none"
    )
  ggsave(info$output, plot = p, width = 18, height = 18, dpi = 300)
}
final_data <- bind_rows(all_plot_data)
write.csv(final_data, "All_Plot_Data.csv", row.names = TRUE)
```

Differential expression analysis

#Each developmental stage compared against the average of the remaining stages

```
library(limma)
library(dplyr)
library(tibble)
exprs_matrix <- read.csv("Imputed_data.csv")
exprs_matrix$Accession <- as.character(exprs_matrix$Accession)
exprs_matrix_log <- exprs_matrix %>%
  mutate_at(vars(V1_MinC:F2_MinC), ~ log2(. + 1))
```

```
exprs_matrix_log <- exprs_matrix_log %>%
  column_to_rownames(var = "Accession")

exprs_matrix_numeric <- exprs_matrix_log %>%
  select(starts_with("V1_MinC"):starts_with("F2_MinC"))
#design matrix for Limma
stage_names <- c("Vegetative", "Vegetative", "Aggregation", "Aggregation",
               "Mound", "Mound", "Culmination", "Culmination",
               "FruitingBody", "FruitingBody")
stages <- factor(stage_names, levels = c("Vegetative", "Aggregation", "Mound",
                                  "Culmination", "FruitingBody"))
design <- model.matrix(~ 0 + stages)
colnames(design) <- levels(stages)
# Fit the linear model using Limma
fit <- lmFit(exprs_matrix_numeric, design)
# contrast matrix
contrast_matrix <- makeContrasts(
  Veg_vs_AllOthers = Vegetative - (Aggregation + Mound + Culmination + FruitingBody)/4,
  Agg_vs_AllOthers = Aggregation - (Vegetative + Mound + Culmination + FruitingBody)/4,
  Mound_vs_AllOthers = Mound - (Vegetative + Aggregation + Culmination +
FruitingBody)/4,
+ FruitingBody)/4,
Mound + Culmination)/4,
)
fit2 <- contrasts.fit(fit, contrast_matrix)
fit2 <- eBayes(fit2)
results_full <- topTable(fit2, coef = NULL, adjust = "BH", number = Inf)
results_full$Accession <- rownames(results_full)
results_full <- left_join(results_full, exprs_matrix %>%
                      select(Accession, Description, UniProt_ID, DDB_ID, DDB_GID,
Protein_names, Gene_Names),
write.csv(results_full, "Differential_Expression_each_stage_to_rest_all.csv", row.names =
TRUE)
```

Hierarchical clustering-> 9 clusters

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(gridExtra)
library(tibble)
library(tidyverse)
```

```
df <- read.csv("Differential_Expression_each_stage_to_rest_all.csv")
logFC_cols <- c("Veg_vs_AllOthers", "Agg_vs_AllOthers", "Mound_vs_AllOthers",
"Culm_vs_AllOthers", "Fruit_vs_AllOthers")
logFC_data <- df[, logFC_cols]
colnames(logFC_data) <- c("V", "A", "M", "C", "F")
rownames(logFC_data) <- df$Accession
clusters <- cutree(hclust(dist(logFC_data), method = "ward.D"), k = 9)
prot_clusters <- setNames(clusters, rownames(logFC_data))
logFC_data_long <- logFC_data %>%
  rownames_to_column("Protein") %>%
  pivot_longer(cols = V:F, names_to = "Stage", values_to = "Expression") %>%
  mutate(Cluster = factor(prot_clusters[Protein], levels = 1:9))
logFC_data_long$Stage <- factor(logFC_data_long$Stage, levels = c("V", "A", "M", "C", "F"))
avg_expression <- logFC_data_long %>%
  group_by(Cluster, Stage) %>%
  summarise(Avg_Expression = mean(Expression), .groups = "drop")
y_limits <- range(logFC_data_long$Expression, na.rm = TRUE)
plots <- lapply(1:9, function(i) {
  cluster_data <- filter(logFC_data_long, Cluster == i)
  cluster_avg <- filter(avg_expression, Cluster == i)
  n_proteins <- n_distinct(cluster_data$Protein)
  ggplot() +
    geom_line(data = cluster_data, aes(x = Stage, y = Expression, group = Protein), color =
"gray80", alpha = 0.7) +
group = Cluster), color = "darkred", linewidth = 1.3)+
    geom_point(data = cluster_avg, aes(x = Stage, y = Avg_Expression), color = "darkred", size
= 2.5) +
    labs(title = paste0("Cluster ", i, " (n=", n_proteins, ")"), x = "Stage", y = "Expression") +
    theme_minimal() +
    coord_cartesian(ylim = y_limits) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(hjust = 0.5),
          axis.title = element_text(size = 10))
})

#sans text plot
tiff("cluster_expression_line_plots_k9_sans.tiff", width = 3600, height = 3600, res = 300,
compression = "lzw")
  p + theme(text = element_text(family = "sans"))
}), ncol = 3)
dev.off()
#no text plot
tiff("cluster_expression_line_plots_notext.tiff", width = 3600, height = 3600, res = 300,
compression = "lzw")
grid.arrange(grobs = lapply(plots, function(p) {
  p + labs(title = NULL, x = NULL, y = NULL) +
    theme(
      axis.text.x = element_blank(),
      axis.text.y = element_blank(),
      axis.ticks = element_blank(),
      axis.title.x = element_blank(),
      axis.title.y = element_blank(),
      plot.title = element_blank()
    )
}), ncol = 3)
dev.off()
#save the clusters
for (cl in 1:9) {
  cluster_proteins <- names(prot_clusters[prot_clusters == cl])
  cluster_data <- df %>% filter(Accession %in% cluster_proteins)
  write.csv(cluster_data, file = paste0("Clusterk9_", cl, "_proteins.csv"), row.names = FALSE)
}
```

GO plot for 9 hierarchical cluster

```
#Go ontology(topGO) for 9 hierarchical cluster
if (!requireNamespace("GO.db", quietly = TRUE)) {
  BiocManager::install("GO.db")
}
if (!requireNamespace("topGO", quietly = TRUE)) {
  BiocManager::install("topGO")
}
library(GO.db)
library(topGO)
#  gene-to-GO mapping
gene_to_go_list <- readRDS("gene_to_go_list1.RDS")
bg_data <- read.csv("Imputed_data.csv")
bg_genes <- unique(na.omit(bg_data$DDB_GID))
gene_to_go_list <- gene_to_go_list[names(gene_to_go_list) %in% bg_genes]
all_genes <- unique(names(gene_to_go_list))
comb_GO <- data.frame()
for (cluster_num in 1:9) {
  cat("### Processing Cluster", cluster_num, "\n")
  cluster_file <- paste0("Clusterk9_", cluster_num, "_proteins.csv")
  cluster_data <- read.csv(cluster_file)
  cluster_data <- cluster_data[!is.na(cluster_data$DDB_GID), ]
  cluster_genes <- unique(cluster_data$DDB_GID)
  cluster_genes_with_annot <- intersect(cluster_genes, names(gene_to_go_list))
  cat("Total genes in cluster file:", length(cluster_genes), "\n")
  cat("Genes with GO annotation:", length(cluster_genes_with_annot), "\n")
  gene_vector <- factor(as.integer(all_genes %in% cluster_genes_with_annot))
  names(gene_vector) <- all_genes
  if (length(levels(gene_vector)) < 2) {
    cat("Skipping Cluster", cluster_num, ": gene_vector has only 1 level.\n\n")
    next
  }

   GOdata <- new("topGOdata",
              ontology = "BP",  # Biological Process
              allGenes = gene_vector,
              annot = annFUN.gene2GO,
              gene2GO = gene_to_go_list,
              nodeSize = 5)
  result <- runTest(GOdata, algorithm = "weight01", statistic = "fisher")
  top_res <- GenTable(GOdata, Fis = result, topNodes = 100)
  top_res$Term <- Term(GOTERM[top_res$GO.ID])
  top_res$Enrichment <- top_res$Significant / top_res$Expected
  top_res <- top_res[top_res$Enrichment > 1, ]
  top_res$cluster <- paste0("cluster", cluster_num)
  comb_GO <- rbind(comb_GO, top_res)
  write.csv(top_res, paste0("GO_results_clusterk9_BP_", cluster_num, ".csv"), row.names =
FALSE)
}
write.csv(comb_GO, "Combined_GO_Results_ Hierarchicalclustersk1-9_BP_.csv",
row.names = FALSE)
```

#**GO plot**

```
library(ggplot2)
library(dplyr)
library(readr)
library(stringr)
library(purrr)
clusters <- 1:9
go_data_combined <- map_dfr(clusters, function(cluster_num) {
  file <- paste0("GO_results_clusterk9_BP_", cluster_num, ".csv")
  df <- read_csv(file, show_col_types = FALSE)
  df$Cluster <- paste0("Cluster ", cluster_num)
  return(df)
})
```

```
top_go <- go_data_combined %>%
  group_by(Cluster) %>%
  arrange(Fis, .by_group = TRUE) %>%
  slice_head(n = 10) %>%
  ungroup() %>%
  mutate(Term = str_trunc(Term, 60),
         Term = factor(Term, levels = rev(unique(Term))))
tiff("GO_dotplot_clusters_k9_top10_BP.tiff", width = 16000, height = 9000, res = 300,
compression = "lzw")
ggplot(top_go, aes(x = -log10(Fis), y = Term,
                   color = Cluster, size = Enrichment)) +
  geom_point(alpha = 3.85) +
  geom_vline(xintercept = -log10(0.01), linetype = "dashed", color = "gray40") +
  facet_wrap(~ Cluster, scales = "free_y") +
  scale_color_brewer(palette = "Set1") +
  theme_minimal(base_size = 50) +
  theme(
    strip.text = element_blank(),
    strip.background = element_blank(),
    panel.spacing = unit(5.0, "lines"),
    axis.text.y = element_text(size = 34, color = "black"),
    axis.text.x = element_text(color = "black"),
    axis.title = element_text(color = "black"),
    legend.text = element_text(color = "black"),
    legend.title = element_text(color = "black"),
    text = element_text(family = "sans", color = "black")
  )
dev.off()

# Plot
tiff("GO_dotplot_clusters_k9_top10_BP.tiff", width = 6600, height = 4500, res = 300,
compression = "lzw")
ggplot(top_go, aes(x = -log10(Fis), y = Term,
                   color = Cluster, size = Enrichment)) +
  geom_point(alpha = 0.85) +
  geom_vline(xintercept = -log10(0.01), linetype = "dashed", color = "gray40") +
  facet_wrap(~ Cluster, scales = "free_y") +
  scale_color_brewer(palette = "Set1") +
  theme_minimal(base_size = 16) +
  theme(
    strip.text = element_text(face = "bold", color = "black"),
    strip.background = element_blank(),          # ← removes gray strip background
    axis.text.y = element_text(size = 14, color = "black"),
    axis.text.x = element_text(color = "black"),
    axis.title = element_text(color = "black"),
    legend.text = element_text(color = "black"),
    legend.title = element_text(color = "black"),
    text = element_text(family = "sans", color = "black")
  )
dev.off()
```

Continuous expression and stable proteome

#Two way venn-Continuous expression proteins intersect with not differentially expressed proteins

```
#No_Change_Allothers.csv
#Clusterk9_9_proteins.csv
#Continuous expression proteome_with_Minprob.csv

install.packages("VennDiagram")

library(VennDiagram)
library(readr)
library(dplyr)

fileA <- read_csv("Differential_Expression_each_stage_to_rest_all.CSV", show_col_types =
FALSE)
A <- unique(fileA %>% filter(adj.P.Val > 0.01) %>% pull(Accession))  # Non-significant
B <- unique(fileB$Accession)  # VennCommon
sets_list <- list(
  `Non-significant (adj.P.Val > 0.01)` = A,
  VennCommon = B
)

tiff("Venn_Updated_Accession_Comparison_0.01.tiff", width = 2000, height = 2000, res = 300)
venn.plot <- venn.diagram(
  x = sets_list,
  filename = NULL,
  fill = c("orange", "lightgreen"),
  alpha = 0.5,
  cex = 1.2,
  cat.cex = 1.2,
  cat.pos = c(-20, 20),
  margin = 0.1,
  lwd = 2
)
grid.draw(venn.plot)
dev.off()
A_only <- setdiff(A, B)
B_only <- setdiff(B, A)
AB <- intersect(A, B)
cat("Venn Overlap Summary (2 sets):\n")
cat("Non-significant only:", length(A_only), "\n")
cat("VennCommon only:", length(B_only), "\n")
cat("Intersection (A ∩ B):", length(AB), "\n")
write_csv(data.frame(Accession = AB), " Continuous expression proteome.csv")
```

Continuous expression proteome kegg pathway

#**kegg enrichment**

```
library(dplyr)
library(readr)
library(stringr)

# Input files
input_kegg_file <- "Continuous expression proteome.csv"
input_ddb_file <- "Continuous expression proteome_DDB_IDs.txt"
bg_ddb_file <- "All_Kegg_DDB_IDs.txt"
bg_kegg_annot <- "All_Kegg.csv"

background_genes <- read_lines(bg_ddb_file)
background_size <- length(background_genes)

bg_kegg_counts <- read_csv(bg_kegg_annot, show_col_types = FALSE) %>%
  select(Kegg.ID, total_bg_in_term = Annotated)

kegg_data <- read_csv(input_kegg_file, show_col_types = FALSE)
input_genes <- read_lines(input_ddb_file)
input_size <- length(input_genes)

kegg_data <- left_join(kegg_data, bg_kegg_counts, by = "Kegg.ID")

kegg_data <- kegg_data %>%
  rowwise() %>%
  mutate(
    a = Annotated,
    b = input_size - a,
    c = total_bg_in_term - a,
    d = background_size - a - b - c,
    fisher_p = tryCatch({
      fisher.test(matrix(c(a, b, c, d), nrow = 2))$p.value
    }, error = function(e) { NA }),
    enrichment_score = (a / input_size) / (total_bg_in_term / background_size)
  ) %>%
  ungroup()

write_csv(kegg_data, "Continuous expression proteome_FisherEnriched.csv")
```

#**kegg plot**

```
library(ggplot2)
library(readr)
library(dplyr)
library(stringr)

df <- read_csv("Continuous expression proteome_FisherEnriched.csv", show_col_types =
FALSE)
df <- df %>%
  filter(!is.na(fisher_p), !is.na(enrichment_score), is.finite(fisher_p)) %>%
  mutate(
    fisher_p = as.numeric(fisher_p),
    enrichment_score = as.numeric(enrichment_score)
  )
top_terms <- df %>%
  slice_min(order_by = fisher_p, n = 15, with_ties = FALSE)

# Step 4: Clean term names
top_terms$Term <- str_trunc(top_terms$Term, 60)
top_terms$Term <- str_to_sentence(top_terms$Term)  # <- THIS LINE
top_terms$Term <- factor(top_terms$Term, levels = rev(unique(top_terms$Term)))

tiff("Continuous expression proteome_Top15_BubblePlot.tiff", width = 1800, height = 1500,
res = 300, compression = "lzw")

ggplot(top_terms, aes(x = -log10(fisher_p), y = Term, size = enrichment_score)) +
  geom_point(color = "black", alpha = 0.85) +
  geom_vline(xintercept = -log10(0.01), linetype = "dashed", color = "gray40") +
  scale_size_continuous(name = "Enrichment") +
  theme_minimal(base_size = 12) +
  labs(x = "-log10(P-value)", y = NULL) +
  theme(
    axis.text.y = element_text(size = 10, family = "sans", color = "black"),
    axis.text.x = element_text(size = 10, family = "sans", color = "black"),
    legend.text = element_text(size = 10, family = "sans", color = "black"),
    legend.title = element_text(size = 10, family = "sans", color = "black")
  )

dev.off()
```

Continuous expression proteome go ontology

```
if (!requireNamespace("GO.db", quietly = TRUE)) BiocManager::install("GO.db")
if (!requireNamespace("topGO", quietly = TRUE)) BiocManager::install("topGO")

library(topGO)
library(GO.db)
library(tidyverse)


minprob_data <- read_csv("Imputed_data.csv", show_col_types = FALSE)
bg_genes <- unique(na.omit(Imputed_data$DDB_GID))
gene_to_go_list <- readRDS("gene_to_go_list1.RDS")
gene_to_go_list <- gene_to_go_list[names(gene_to_go_list) %in% bg_genes]
all_genes <- unique(names(gene_to_go_list))
Continuous expression proteome_data <- read_csv("Continuous expression proteome.csv",
show_col_types = FALSE)
gene_ids <- unique(na.omit(Continuous expression proteome_data$DDB_GID))

run_GO_analysis <- function(gene_list, set_name) {
  gene_list <- intersect(gene_list, all_genes)

  if (length(gene_list) == 0) {
    cat("Skipping:", set_name, "— No overlapping GO-annotated genes.\n")
    return(NULL)
  }

  gene_vector <- factor(as.integer(all_genes %in% gene_list))
  names(gene_vector) <- all_genes
  if (length(levels(gene_vector)) < 2) {
    cat("Skipping:", set_name, "— gene_vector has only 1 level.\n")
    return(NULL)
  }

  GOdata <- new("topGOdata",
              ontology = "BP",
              allGenes = gene_vector,
              annot = annFUN.gene2GO,
              gene2GO = gene_to_go_list,
              nodeSize = 5)

  result <- runTest(GOdata, algorithm = "weight01", statistic = "fisher")

  top_res <- GenTable(GOdata,
                  Fis = result,
                  topNodes = 100)

  top_res$Term <- Term(GOTERM[top_res$GO.ID])
  top_res$Stage <- set_name
  top_res$Enrichment <- as.numeric(top_res$Significant) / as.numeric(top_res$Expected)

  write_csv(top_res, paste0("GO_results_BP_", set_name, ".csv"))
  return(top_res)
}

go_Continuous expression proteome <- run_GO_analysis(gene_ids, "Continuous expression
proteome")
```

#**GO Plot**

```
library(ggplot2)
library(readr)
library(dplyr)
library(stringr)

go_data <- read_csv("GO_results_BP_Continuous expression proteome.CSV",
show_col_types = FALSE)
go_data <- go_data %>%
  filter(is.finite(Fis)) %>%
  mutate(
    Fis = as.numeric(Fis),
    Enrichment = as.numeric(Enrichment)
  ) %>%
  slice_min(order_by = Fis, n = 15)


library(stringr)

go_data$Term <- str_trunc(go_data$Term, 60)
go_data$Term <- str_to_sentence(go_data$Term)
go_data$Term <- factor(go_data$Term, levels = rev(go_data$Term))

tiff("GO_Top15_BP_CommonProteins_BubblePlot.tiff", width = 1800, height = 1500, res = 300,
compression = "lzw")
ggplot(go_data, aes(x = -log10(Fis), y = Term, size = Enrichment)) +
  geom_point(color = "#000000", alpha = 0.85) +
  geom_vline(xintercept = -log10(0.01), linetype = "dashed", color = "gray40") +
  theme_minimal(base_size = 12) +
  labs(x = "-log10(P-value)", y = NULL, size = "Enrichment") +
  theme(
    axis.text.y = element_text(size = 10, family = "sans", color = "black"),
    axis.text.x = element_text(size = 10, family = "sans", color = "black"),
    legend.text = element_text(size = 10, family = "sans", color = "black"),
    legend.title = element_text(size = 10, family = "sans", color = "black"),
    legend.position = "right"
  )
dev.off()
```

Three way venn-Comparison of hierarchical cluster 9 with the continuous expressed and stable proteomes

#Comparison of hierarchical cluster 9 with the continuously expressed proteome and the not differentially expressed proteins (logFC not significantly changed).

```
install.packages("VennDiagram")

library(VennDiagram)
library(readr)
library(dplyr)
```

```
# Read the data files
fileA <- read_csv("Differential_Expression_each_stage_to_rest_all.CSV", show_col_types =
FALSE)
fileB <- read_csv("Clusterk9_9_proteins.csv", show_col_types = FALSE)
fileC <- read_csv("Continuous expression proteome.csv", show_col_types = FALSE)

A <- unique(fileA %>% filter(adj.P.Val > 0.01) %>% pull(Accession))  # Replaces No_Change
B <- unique(fileB$Accession)  # k9
C <- unique(fileC$Accession)  # VennCommon

sets_list <- list(
  `Non-significant (adj.P.Val > 0.01)` = A,
  k9 = B,
  VennCommon = C
)

tiff("Venn_Updated_Accession_Comparison_0.0111.tiff", width = 2000, height = 2000, res =
300)
  x = sets_list,
  filename = NULL,
  fill = c("orange", "lightblue", "lightgreen"),
  alpha = 0.5,
  cex = 1.2,
  cat.cex = 1.2,
  cat.pos = c(-20, 20, 180),
  margin = 0.1,
  lwd = 2
)
grid.draw(venn.plot)
dev.off()

A_only <- setdiff(A, union(B, C))
B_only <- setdiff(B, union(A, C))
C_only <- setdiff(C, union(A, B))
AB <- intersect(A, B)
AC <- intersect(A, C)
BC <- intersect(B, C)
ABC <- Reduce(intersect, list(A, B, C))

cat("Venn Overlap Summary:\n")
cat("Non-significant only:", length(A_only), "\n")
cat("k9 only:", length(B_only), "\n")
cat("VennCommon only:", length(C_only), "\n")
cat("Non-significant ∩ k9:", length(AB), "\n")
cat("Non-significant ∩ VennCommon:", length(AC), "\n")
cat("k9 ∩ VennCommon:", length(BC), "\n")
cat("All three (Non-significant ∩ k9 ∩ VennCommon):", length(ABC), "\n")

write_csv(data.frame(Accession = ABC), "Common_Accession_Intersection_2.csv")
library(readr)
library(dplyr)
intersection <- read_csv("Common_Accession_Intersection_0.05.CSV", show_col_types =
FALSE)
show_col_types = FALSE)

merged_data <- intersection %>%
  left_join(full_data, by = "Accession")

write_csv(merged_data, "Continuous expression proteome_Accession_Full_Info.csv")
```

Validation of LC-MS/MS protein abundances using published data

LC-MS/MS protein abundances compared to published Western blot quantification data

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

df <- read_csv("Westernblot_quantification_published.csv")

df_norm <- df %>%
  group_by(Protein, Experiment_Type) %>%
  mutate(
    Protein_Expression_norm = (Protein_Expression - min(Protein_Expression)) /
      (max(Protein_Expression) - min(Protein_Expression))
  ) %>%
  ungroup()

p <- ggplot(df_norm, aes(x = `Time(h)`, y = Protein_Expression_norm,
                    color = Experiment_Type, group = Experiment_Type)) +
  geom_line(size = 1.2) +
  geom_point(size = 3) +
  facet_wrap(~ Protein, scales = "free_y", ncol = 5) +  # 5 panels per row
  theme_minimal(base_size = 18) +
  theme(
    axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    legend.position = "none",
    strip.text = element_blank(),
    panel.spacing.y = unit(1.5, "cm")
  )

# Save as TIFF
ggsave("Westernblot_quantification_published.tiff",
       plot = p, width = 12, height = 8, dpi = 300)
```

#Time interpolated
#distribution and correlations

```
library(readr)
library(dplyr)
library(ggplot2)
library(purrr)
library(tidyr)

df <- read_csv("Westernblot_quantification_Protein_Time_interpolated.csv")

p1 <- ggplot(df, aes(x = Our_norm)) +
  geom_histogram(bins = 20, fill = "steelblue", alpha = 0.7, color = "black") +
  labs(title = "Distribution of Our (LC-MS/MS) Normalized Values", x = "Our_norm", y =
"Count") +
  theme_minimal(base_size = 14)

p2 <- ggplot(df, aes(x = Pub_norm)) +
  geom_histogram(bins = 20, fill = "orange", alpha = 0.7, color = "black") +
  labs(title = "Distribution of Published (Western blot) Normalized Values", x = "Pub_norm", y
= "Count") +
  theme_minimal(base_size = 14)

ggsave("Distribution_Our_norm.tiff", plot = p1, width = 7, height = 5, dpi = 300)
ggsave("Distribution_Pub_norm.tiff", plot = p2, width = 7, height = 5, dpi = 300)

# Shapiro-Wilk test for normality
our_shapiro <- if(nrow(df) > 3) shapiro.test(df$Our_norm) else NA
```

```
pub_shapiro <- if(nrow(df) > 3) shapiro.test(df$Pub_norm) else NA

cat("\n--- Normality tests ---\n")
cat("Our_norm Shapiro p-value:", our_shapiro$p.value, "\n")
cat("Pub_norm Shapiro p-value:", pub_shapiro$p.value, "\n")

if (our_shapiro$p.value > 0.05 & pub_shapiro$p.value > 0.05) {
  cat("both Our_norm and Pub_norm appear normally distributed.\n")
} else {
  cat("At least one of the datasets deviates from normality.\n")
}

correlation_results <- df %>%
  group_by(Protein) %>%
  summarise(
    Pearson_r = suppressWarnings(cor(Our_norm, Pub_norm, method = "pearson")),
    Spearman_rho = suppressWarnings(cor(Our_norm, Pub_norm, method = "spearman")),
    .groups = "drop"
  )

correlation_results <- correlation_results %>%
  rowwise() %>%
  mutate(
    Pearson_p = tryCatch(cor.test(df$Our_norm[df$Protein == Protein],
                                  df$Pub_norm[df$Protein == Protein],
                                  method = "pearson")$p.value, error = function(e) NA),
    Spearman_p = tryCatch(cor.test(df$Our_norm[df$Protein == Protein],
                                   df$Pub_norm[df$Protein == Protein],
                                   method = "spearman")$p.value, error = function(e) NA)
  )

pearson_all <- cor.test(df$Our_norm, df$Pub_norm, method = "pearson")
spearman_all <- cor.test(df$Our_norm, df$Pub_norm, method = "spearman")

overall_summary <- data.frame(
  Pearson_r = pearson_all$estimate,
  Pearson_p = pearson_all$p.value,
  Spearman_rho = spearman_all$estimate,
  Spearman_p = spearman_all$p.value
)

cat("\n--- Overall Correlation Summary ---\n")
print(overall_summary)
```

```
write_csv(correlation_results, "Protein_Pearson_Spearman_PerProtein_Interpolated.csv")
write_csv(overall_summary, "Overall_Pearson_Spearman_Interpolated.csv")
p_combined <- ggplot(df, aes(x = Pub_norm, y = Our_norm)) +
  geom_point(size = 3, color = "steelblue", alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "red", linetype = "dashed") +
  annotate("text", x = 0.1, y = 0.95,
           label = paste0("r = ", round(overall_summary$Pearson_r, 2),
                          "\nρ = ", round(overall_summary$Spearman_rho, 2),
                          "\np = ", signif(overall_summary$Spearman_p, 2)),
           hjust = 0, vjust = 1, size = 5) +
  labs(
    title = "Overall Correlation Between LC–MS/MS and Western Blot (Normalized)",
    x = "Published (Western blot) Normalized Intensity",
    y = "Our (LC–MS/MS) Normalized Intensity"
  ) +
  theme_minimal(base_size = 14)

ggsave("Overall_Correlation_Interpolated.tiff", plot = p_combined, width = 8, height = 6, dpi
= 300)

p_combined
```

#Time adjusted
#distribution and correlations

```
# --- Load packages ---
library(readr)
library(dplyr)
library(ggplot2)
library(purrr)
library(tidyr)

df <- read_csv("Westernblot_quantification_Protein_Time_adjusted.csv")

p1 <- ggplot(df, aes(x = Our_norm)) +
  geom_histogram(bins = 20, fill = "steelblue", alpha = 0.7, color = "black") +
  labs(title = "Distribution of Our (LC-MS/MS) Normalized Values", x = "Our_norm", y =
"Count") +
  theme_minimal(base_size = 14)

p2 <- ggplot(df, aes(x = Pub_norm)) +
  geom_histogram(bins = 20, fill = "orange", alpha = 0.7, color = "black") +
  labs(title = "Distribution of Published (Western blot) Normalized Values", x = "Pub_norm", y
= "Count") +
  theme_minimal(base_size = 14)

ggsave("Distribution_Our_norm_timechange.tiff", plot = p1, width = 7, height = 5, dpi = 300)
ggsave("Distribution_Pub_norm_timechange.tiff", plot = p2, width = 7, height = 5, dpi = 300)

# Shapiro-Wilk test for normality
our_shapiro <- if(nrow(df) > 3) shapiro.test(df$Our_norm) else NA
pub_shapiro <- if(nrow(df) > 3) shapiro.test(df$Pub_norm) else NA

cat("\n--- Normality tests ---\n")
cat("Our_norm Shapiro p-value:", our_shapiro$p.value, "\n")
cat("Pub_norm Shapiro p-value:", pub_shapiro$p.value, "\n")

if (our_shapiro$p.value > 0.05 & pub_shapiro$p.value > 0.05) {
  cat("Both Our_norm and Pub_norm appear normally distributed.\n")
} else {
  cat(" At least one of the datasets deviates from normality.\n")
}

correlation_results <- df %>%
  group_by(Protein) %>%
  summarise(
    Pearson_r = suppressWarnings(cor(Our_norm, Pub_norm, method = "pearson")),
    Spearman_rho = suppressWarnings(cor(Our_norm, Pub_norm, method = "spearman")),
    .groups = "drop"
  )
correlation_results <- correlation_results %>%
  rowwise() %>%
  mutate(
    Pearson_p = tryCatch(cor.test(df$Our_norm[df$Protein == Protein],
                              df$Pub_norm[df$Protein == Protein],
                              method = "pearson")$p.value, error = function(e) NA),
    Spearman_p = tryCatch(cor.test(df$Our_norm[df$Protein == Protein],
                               df$Pub_norm[df$Protein == Protein],
                               method = "spearman")$p.value, error = function(e) NA)
  )

pearson_all <- cor.test(df$Our_norm, df$Pub_norm, method = "pearson")
spearman_all <- cor.test(df$Our_norm, df$Pub_norm, method = "spearman")
```

```
overall_summary <- data.frame(
  Pearson_r = pearson_all$estimate,
  Pearson_p = pearson_all$p.value,
  Spearman_rho = spearman_all$estimate,
  Spearman_p = spearman_all$p.value
)

cat("\n--- Overall Correlation Summary ---\n")
print(overall_summary)

write_csv(correlation_results, "Protein_Pearson_Spearman_PerProtein_Timechange.csv")
write_csv(overall_summary, "Overall_Pearson_Spearman_Timechange.csv")

p_combined <- ggplot(df, aes(x = Pub_norm, y = Our_norm)) +
  geom_point(size = 3, color = "steelblue", alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "red", linetype = "dashed") +
  annotate("text", x = 0.1, y = 0.95,
           label = paste0("r = ", round(overall_summary$Pearson_r, 2),
                          "\nρ = ", round(overall_summary$Spearman_rho, 2),
                          "\np = ", signif(overall_summary$Spearman_p, 2)),
           hjust = 0, vjust = 1, size = 5) +
  labs(
    title = "Overall Correlation Between LC–MS/MS and Western Blot (Normalized)",
    x = "Published (Western blot) Normalized Intensity",
    y = "Our (LC–MS/MS) Normalized Intensity"
  ) +
  theme_minimal(base_size = 14)

ggsave("Overall_Correlation_Time_adjusted.tiff", plot = p_combined, width = 8, height = 6,
dpi = 300)

p_combined
```

This study data (LC-MS/MS protein abundances) compared to Edelbroek et al. (2024) proteomics data

# This study vs Edelbroek et al

```
install.packages("lmodel2")

library(fields)
library(ggplot2)
library(dplyr)
library(scales)
library(lmodel2)  # <-- added for ranged major axis regression

merged <- read.csv("Correlation_ED_SB_log2.csv")

Thisstudy_cols  <-
c("wt_0h_as","wt_0h_bs","wt_8h_as","wt_8h_bs","wt_10h_as","wt_10h_bs")
<- c("wt_0h_a","wt_0h_b","wt_0h_c",
          "wt_10h_a","wt_10h_b","wt_10h_c")

merged <- merged %>% filter(!is.na(GENE_ID) & GENE_ID != "")

Thisstudy_means <- merged %>%
  transmute(
    GENE_ID,
    `0h`  = rowMeans(across(c(wt_0h_as, wt_0h_bs)), na.rm = TRUE),
    `8h`  = rowMeans(across(c(wt_8h_as, wt_8h_bs)), na.rm = TRUE),
    `10h` = rowMeans(across(c(wt_10h_as, wt_10h_bs)), na.rm = TRUE)
  )

edelbroek_means <- merged %>%
  transmute(
    GENE_ID,
    `0h`  = rowMeans(across(c(wt_0h_a, wt_0h_b, wt_0h_c)), na.rm = TRUE),
    `8h`  = rowMeans(across(c(wt_8h_a, wt_8h_b, wt_8h_c)), na.rm = TRUE),
    `10h` = rowMeans(across(c(wt_10h_a, wt_10h_b, wt_10h_c)), na.rm = TRUE)
  )

merged_means <- inner_join(Thisstudy_means, edelbroek_means, by = "GENE_ID",
                           suffix = c(".thisstudy", ".edelbroek"))
cat("Common genes used for correlation:", nrow(merged_means), "\n")

for (tp in c("0h","8h","10h")) {
  merged_means[[paste0(tp, ".thisstudy.lin")]] <-
    (2^merged_means[[paste0(tp, ".thisstudy")]]) /
    sum(2^merged_means[[paste0(tp, ".thisstudy")]], na.rm = TRUE) * 1e6

  merged_means[[paste0(tp, ".edelbroek.lin")]] <-
    (2^merged_means[[paste0(tp, ".edelbroek")]]) /
    sum(2^merged_means[[paste0(tp, ".edelbroek")]], na.rm = TRUE) * 1e6
}
```

```
set.seed(123)
sample_genes <- sample(merged_means$GENE_ID, min(500, nrow(merged_means)))  # up
to 500 random genes

x_all <- unlist(lapply(sample_genes, function(g)
  2^merged_means[merged_means$GENE_ID == g, c("0h.thisstudy", "8h.thisstudy",
"10h.thisstudy")]
y_all <- unlist(lapply(sample_genes, function(g)
  2^merged_means[merged_means$GENE_ID == g, c("0h.edelbroek", "8h.edelbroek",
"10h.edelbroek")]

n_x <- min(length(x_all), 5000)
n_y <- min(length(y_all), 5000)

shapiro_x <- shapiro.test(sample(x_all, n_x))
shapiro_y <- shapiro.test(sample(y_all, n_y))

cat("\nShapiro-Wilk p-values:\n")
cat("This study:", shapiro_x$p.value, "\n")
cat("Edelbroek et al.:", shapiro_y$p.value, "\n\n")

if (shapiro_x$p.value < 0.05 | shapiro_y$p.value < 0.05) {
  cat("→ Data not normally distributed → Spearman is safer.\n\n")
} else {
  cat("→ Data approximately normal → Pearson is acceptable.\n\n")
}


library(ggplot2)

x_lin <- 2^unlist(merged_means[, c("0h.thisstudy", "8h.thisstudy", "10h.thisstudy")])
y_lin <- 2^unlist(merged_means[, c("0h.edelbroek", "8h.edelbroek", "10h.edelbroek")])

df <- data.frame(
  value = c(x_lin, y_lin),
  dataset = rep(c("This study", "Edelbroek et al."), each = length(x_lin))
)

p <- ggplot(df, aes(x = value, fill = dataset)) +
  geom_histogram(bins = 100, alpha = 0.6, position = "identity") +
  scale_x_log10() +
  theme_minimal(base_size = 10) +
  labs(
    title = "Distribution of linear CPM protein abundances",
    x = "Protein abundance (log10 scale)",
    y = "Count",
    fill = "Dataset"
  )

ggsave("Protein_Distribution_ThisStudy_vs_Edelbroek.svg", p,
       width = 4, height = 3, units = "in")

gene_corrs <- merged_means
gene_corrs$spearman_r <- apply(merged_means, 1, function(row) {
  xvals <- as.numeric(row[c("0h.thisstudy", "8h.thisstudy", "10h.thisstudy")])
  yvals <- as.numeric(row[c("0h.edelbroek", "8h.edelbroek", "10h.edelbroek")])
  if (all(is.finite(xvals)) && all(is.finite(yvals))) {
    cor(xvals, yvals, method = "spearman")
  } else {
    NA
  }
})

gene_corrs <- gene_corrs[, c("GENE_ID", "spearman_r")]

# Save to CSV
write.csv(gene_corrs, "PerGene_Spearman_Correlation_ThisStudy_vs_Edelbroek.csv",
row.names = FALSE)

for (tp in c("0h","8h","10h")) {

  x <- merged_means[[paste0(tp, ".thisstudy.lin")]]
  y <- merged_means[[paste0(tp, ".edelbroek.lin")]]

  ok <- complete.cases(x, y)
  x <- x[ok]; y <- y[ok]

  if (length(x) > 10) {
    r <- cor(x, y, method = "spearman")

    model <- lmodel2(log10(y) ~ log10(x),
                 range.y = "interval", range.x = "interval")
    slope <- model$regression.results[4, 3]     # RMA slope
    intercept <- model$regression.results[4, 2] # intercept
    plotdat <- data.frame(x = x, y = y)

    library(MASS)
    dens <- kde2d(log10(x), log10(y), n = 200)
    plotdat$density <- fields::interp.surface(dens, cbind(log10(x), log10(y)))

    lims <- range(c(x, y), na.rm = TRUE)

    p <- ggplot(plotdat, aes(x = x, y = y, color = density)) +
      geom_point(size = 1.1, alpha = 0.8) +
      scale_color_viridis_c(option = "C", direction = 1, name = "Density") +  # <-- nice legend
      geom_abline(slope = 1, intercept = 0, color = "red", linewidth = 0.5) +
      geom_abline(slope = slope, intercept = intercept,
                linetype = "dashed", color = "black", linewidth = 0.5) +
      scale_x_log10(labels = label_number(scale_cut = cut_short_scale())) +
      scale_y_log10(labels = label_number(scale_cut = cut_short_scale())) +
      annotation_logticks(sides = "bl") +
      coord_cartesian(xlim = lims, ylim = lims) +
      theme_bw(base_size = 9) +
      theme(
        axis.title = element_text(size = 9),
        axis.text  = element_text(size = 8),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 9),
        legend.position = "right",
        legend.key.height = unit(1, "cm"),
        panel.grid.minor = element_blank()
      ) +
      labs(
        x = "This study (normalized protein abundance, CPM)",
        y = "Edelbroek et al. (normalized protein abundance, CPM)",
        title = paste0(tp, " | Spearman ρ = ", round(r, 3),
                  " | slope = ", round(slope, 3))
      )

    ggsave(
      paste0("Correlation_", tp, "_ThisStudy_vs_Edelbroek_lmodel2_densityLegend.svg"),
      p, width = 3.2, height = 3.0, units = "in"
    )

    cat(tp, "Spearman:", round(r, 3), "| slope:", round(slope, 3), "\n")
  }
```

```
}
```