- R codes used in this study (shown for human data, and for expression data in the bovine species the same code form was used with different accession numbers and platforms).

- کدهای R به‌کار رفته در این پژوهش (برای داده انسانی نشان داده شده است و برای داده بیان در گونه گاو، همین شکل از کدها با شماره دسترسی و پلتفرم متفاوت استفاده شد).

```
setwd("F:/cow")

# بارگذاری کتابخانه
library(GEOquery)
# دریافت داده‌ها
geo_id <- "GSE51874"
gset <- getGEO(geo_id, GSEMatrix = TRUE)
# استخراج داده‌ها و اطلاعات ژن‌ها
data <- exprs(gset [[1]])
gene_data <- fData(gset [[1]]) # اطلاعات مربوط به ژن‌ها
# نمایش برخی از اطلاعات ژن‌ها
head(gene_data)
# بررسی و پاک‌سازی داده‌ها
data [is.na(data)] <- 0 # جایگزینی مقادیر NA
data [data < 0] <- 0  # جایگزینی مقادیر منفی
# تبدیل به مقیاس log2
data_log <- log2(data + 1)
# بررسی نتایج
summary(data_log)

# 4. Annotation برای تبدیل نام IDها به نام ژن‌ها
# فایل Annotation باید دانلود و خوانده شود
# مثال فایل فرضی: annotation.csv
# فرض می‌کنیم این فایل دارای دو ستون: "ID" و "GeneSymbol" است.
# بارگذاری کرده‌اید annotation فرض بر این است که شما داده‌های آنوتیشن را به نام
# تخصیص دهید data_log به ردیف‌های GeneSymbol نام ژن‌ها را از ستون
annotation <- read.csv("F:/cow/GPL6104-11576.csv", header = TRUE, stringsAsFactors = FALSE)
rownames(data_log) <- annotation$Symbol [match(rownames(data_log), annotation$ID)]
# بررسی صحت نام ردیف‌ها
head(rownames(data_log))
head(annotation$ID)
gene_list <- c("DGAT1", "DGAT2L3", "MOGAT3", "DGAT2L6", "DGAT2L4", "MOGAT1", "DGAT2", "MOGAT2")
gene_list %in% rownames(data_log)
grep("GAT", annotation$Symbol, value = TRUE)


# 5. فیلتر ژن‌های خانواده GAT
gene_list <- c("DGAT1", "DGAT2L3", "MOGAT3", "DGAT2L6","DGAT2L4","MOGAT1","DGAT2","MOGAT2")
selected_genes <- data_log [rownames(data_log) %in% gene_list,]
if (nrow(selected_genes) == 0) {
  cat("No matching genes found. Check annotation or gene list.\n")
} else {
  print(selected_genes)
}
# 6. طراحی ماتریس گروه‌ها
```

```
# فرض کنید ۱۰ نمونه برای گروه "Control" و بقیه برای "Treatment" است
group <- factor(c(rep("Control", 10), rep("Treatment", ncol(data_log) - 10)))
design <- model.matrix(~group)
keep <- apply(data_log, 1, var) > 0
data_log <- data_log [keep,]

library(limma)
# 7. تحلیل بیان دیفرانسیلی
fit <- lmFit(data_log, design)
fit <- eBayes(fit)
# مشاهده چند نام اول
head(gene_list)
# 8. نتایج تحلیل
top_table <- topTable(fit, coef = 2, number = Inf, sort.by = "p")
head(top_table)
# 9. استخراج نتایج ژن‌های GAT
gat_results <- top_table [top_table$ID %in% gene_list,]
print(gat_results)

# کتابخانه‌های مورد نیاز
library(ggplot2)
# لیست ژن‌ها
genes_of_interest <- c("DGAT1", "DGAT2L3", "MOGAT3", "DGAT2L6", "DGAT2L4", "MOGAT1",
"DGAT2", "MOGAT2")
# حلقه برای رسم Boxplot برای هر ژن
for (gene in genes_of_interest) {
 # داده‌های ژن خاص
 gene_data <- data.frame(
   Group = rep(group, each = 1),
   Expression = c(data_log [gene, group == "Control"] ,
                  data_log [gene, group == "Treatment"])
)

 # ایجاد Boxplot
 plot <- ggplot(gene_data, aes(x = Group, y = Expression)) +
   geom_boxplot(aes(fill = Group), alpha = 0.7) +
   theme_minimal() +
   labs(x = "Group", y = "Expression Level", title = paste("Boxplot of", gene)) +
   scale_fill_manual(values = c("Control" = "skyblue", "Treatment" = "orange"))

 # نمایش نمودار
 print(plot)

 # ذخیره نمودار به فایل
 ggsave(filename = paste0("Boxplot_", gene, ".png") , plot = plot, width = 6, height = 4)
}

library(ggplot2)
pca <- prcomp(t(data_log), scale. = TRUE)
pca_data <- data.frame(PC1 = pca$x [,1] , PC2 = pca$x [,2] , Group = group)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group)) +
 geom_point(size = 3) +
 theme_minimal() +
 labs(title = "PCA of Gene Expression Data")
str(data_log)     # بررسی ساختار داده‌ها
dim(data_log)     # ابعاد ماتریس
length(group)     # طول گروه‌ها
```

```
table(group)      # توزیع گروه‌ها

print(ggplot(pca_data, aes(x = PC1, y = PC2, color = Group)) +
       geom_point(size = 3) +
       theme_minimal() +
       labs(title = "PCA of Gene Expression Data"))
# بررسی و آماده‌سازی داده
if (any(is.na(data_log))) {
  data_log <- na.omit(data_log)
}
if (any(is.infinite(data_log))) {
  data_log <- data_log [rowSums(is.infinite(data_log)) == 0,]
}
# اجرای PCA
pca <- prcomp(t(data_log), scale. = TRUE)
# ساخت داده برای پلات
pca_data <- data.frame(PC1 = pca$x [, 1] ,
                       PC2 = pca$x [, 2] ,
                       Group = group)

# محاسبه درصد واریانس برای هر مولفه اصلی
var_explained <- (pca$sdev^2 / sum(pca$sdev^2)) * 100
# ایجاد Scree Plot
scree_data <- data.frame(PC = 1:length(var_explained), Variance = var_explained)
ggplot(scree_data, aes(x = PC, y = Variance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_line(aes(x = PC, y = Variance), color = "red", size = 1) +
  geom_point(size = 2, color = "red") +
  theme_minimal() +
  labs(title = "Scree Plot of PCA",
       x = "Principal Component",
       y = "Percentage of Variance Explained")

# محاسبه درصد واریانس برای هر مولفه اصلی
var_explained <- (pca$sdev^2 / sum(pca$sdev^2)) * 100
# ایجاد Scree Plot
scree_data <- data.frame(PC = 1:length(var_explained), Variance = var_explained)
ggplot(scree_data, aes(x = PC, y = Variance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_line(aes(x = PC, y = Variance), color = "red", size = 1) +
  geom_point(size = 2, color = "red") +
  theme_minimal() +
  labs(title = "Scree Plot of PCA",
       x = "Principal Component",
       y = "Percentage of Variance Explained")

# اجرای k-means روی ۲ مولفه اصلی اول
set.seed(123)  # برای تکرارپذیری
kmeans_result <- kmeans(pca$x [, 1:2] , centers = 2)  # دو خوشه
# اضافه کردن خوشه‌ها به داده‌ها
pca_data$Cluster <- as.factor(kmeans_result$cluster)
# رسم داده‌ها با خوشه‌ها
ggplot(pca_data, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point(size = 3) +
  theme_minimal() +
  labs(title = "PCA with K-means Clustering",
       x = "Principal Component 1",
       y = "Principal Component 2")
```

```
# نصب و بارگذاری بسته‌های مورد نیاز
if (! require("igraph")) install.packages("igraph")
library(igraph)
save(data_log,top_table,group, file = "all.RData")
# پاکسازی محیط کار
rm(list = ls())
# بارگذاری مجدد داده‌ها
load("all.RData")

# کتابخانه‌های مورد نیاز
library(igraph)
gene_list <- c("DGAT1", "DGAT2L3", "MOGAT3",
"DGAT2L6","DGAT2L4","MOGAT1","DGAT2","MOGAT2")
# 1. فیلتر کردن DEGها
deg_genes <- top_table$ID [top_table$adj.P.Val < 0.05]  # ژن‌های معنی‌دار
all_genes <- unique(c(gene_list, deg_genes))  # ترکیب ژن‌های منتخب و DEG
# 2. استخراج داده‌های ژن‌های مورد نظر
selected_data <- data_log [rownames(data_log) %in% all_genes,]
# 3. ایجاد ماتریس همبستگی
cor_matrix <- cor(t(selected_data), method = "pearson")
# 4. تنظیم آستانه برای ارتباطات شبکه
threshold <- 0.3  # آستانه همبستگی
adj_matrix <- (abs(cor_matrix) > threshold) * 1  # ایجاد ماتریس اتصال
# 5. ساخت گراف از ماتریس اتصال
gene_network <- graph.adjacency(adj_matrix, mode = "undirected", diag = FALSE)
# 6. افزودن ویژگی‌ها به گره‌ها
V(gene_network)$type <- ifelse(V(gene_network)$name %in% deg_genes, "DEG", "Selected")
# 7. رسم شبکه
plot(gene_network,
    vertex.color = ifelse(V(gene_network)$type == "DEG", "red", "blue") ,
    vertex.size = 8,
    vertex.label.cex = 0.8,
    main = "Gene Network")
library(ggraph)
library(tidygraph)
graph_tbl <- as_tbl_graph(gene_network)
ggraph(graph_tbl, layout = "fr") +
  geom_edge_link(alpha = 0.8) +
  geom_node_point(aes(color = type), size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_minimal()

library(ggraph)
library(tidygraph)
# تبدیل گراف به قالب tbl_graph
graph_tbl <- as_tbl_graph(gene_network)
# استفاده از الگوریتم layout مناسب (مانند 'fr' یا 'kk')
ggraph(graph_tbl, layout = "fr") +  # استفاده می‌کنیم Fruchterman-Reingold از طرح‌بندی
  geom_edge_link(alpha = 0.6, color = "gray") +  # خطوط اتصال بین گره‌ها
  geom_node_point(aes(color = type), size = 5) +  # گره‌ها با رنگ‌بندی نوع ژن
  geom_node_text(aes(label = name), repel = TRUE, size = 3) +  # نام گره‌ها
  scale_color_manual(values = c("DEG" = "red", "Selected" = "blue")) +  # تنظیم رنگ
  theme_minimal() +  # تم ساده
  labs(title = "Gene Network", color = "Gene Type")
```

```
library(igraph)
library(ggraph)
library(tidygraph)
# اجرای الگوریتم خوشه‌بندی Louvain
clusters <- cluster_louvain(gene_network)
V(gene_network)$cluster <- clusters$membership  # اضافه کردن اطلاعات خوشه به گره‌ها
# تبدیل گراف به tbl_graph
graph_tbl <- as_tbl_graph(gene_network)
# نمایش گراف با رنگ‌بندی خوشه‌ها
ggraph(graph_tbl, layout = "fr") +  # طرح‌بندی Fruchterman-Reingold
  geom_edge_link(alpha = 0.6, color = "gray") +  # نمایش لبه‌ها
  geom_node_point(aes(color = as.factor(cluster)) , size = 5) +  # رنگ‌بندی گره‌ها براساس خوشه
  geom_node_text(aes(label = name), repel = TRUE, size = 3) +  # نمایش نام گره‌ها
  theme_minimal() +  # استفاده از تم ساده
  labs(title = "Gene Network with Clustering", color = "Cluster") +
  scale_color_brewer(palette = "Set3")  # تنظیم پالت رنگ
```