BioConductor & Regular Expressions

GulnurUzun

2024-05-14

The libraries we will use in the study:

```
library(GEOquery)
library(ggplot2)
library(tidyr)
library(TxDb.Mmusculus.UCSC.mm9.knownGene)
library(biomaRt)
library(GenomicFeatures)
library(dplyr)
```

Assign the Centered Plot Title to centered.plot.title

For ggplot

centered.plot.title will used in next parts for title move the center.

```
centered.plot.title = theme(plot.title = element_text(hjust = 0.5))
```

Part a: Downloading the GSE115342 Dataset with the Geoquery Package

At this stage, the **GSE115342** dataset was downloaded using the **getGEO()** function from the Geoquery package. Next, the exprs function was used to extract the expression matrix. In the final step, the **dim()** function was employed to determine the dimensions of the expression matrix. According to our results, the matrix consists of **59,305 rows** (representing genes) and **12 columns** (representing sample groups).

```
gse = getGEO("GSE115342", AnnotGPL = TRUE)
gse
```

```
## $GSE115342_series_matrix.txt.gz
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 59305 features, 12 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM3175742 GSM3175743 ... GSM3175753 (12 total)
##   varLabels: title geo_accession ... treatment:ch1 (44 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: 4 5 ... 62972 (59305 total)
##   fvarLabels: ID COL ... SPOT_ID.1 (20 total)
##   fvarMetadata: Column Description labelDescription
## experimentData: use 'experimentData(object)'
##   pubMedIds: 31110277
## 31687435
## 31815184
## Annotation: GPL21810
```

```
#for reach numerical data
exp_data = exprs(gse[[1]])
head(exp_data) #first 6 rows in 12 sample groups
```

```
##    GSM3175742  GSM3175743  GSM3175744  GSM3175745 GSM3175746 GSM3175747
## 4 3267.271195 3213.865410 2982.575977 2831.704422 4297.18725 4504.76973
## 5   11.083762   15.734114   23.303580   19.798172   13.27260   20.02929
## 6    5.692115    7.692918    5.738932    7.306098   13.35453   14.09231
## 7  186.951488  181.276385  144.046990  172.811729  100.78696  129.50722
## 8 2049.208254 2167.279325 2147.740836 2172.501503  870.82899  797.26675
## 9  859.448143  861.837129  824.603239  842.584780  994.22146 1123.41089
##   GSM3175748 GSM3175749  GSM3175750 GSM3175751  GSM3175752  GSM3175753
## 4 7067.61545 7375.37182 3917.699844 4648.33260 4009.952174 5090.904378
## 5   14.60719  294.44233    9.572488   13.92412    9.399971    6.291990
## 6   14.70040   13.71533    8.021901    8.02426    9.451548    6.334297
## 7  207.47205  106.97183  214.953214  240.31419   96.736435  114.934933
## 8  808.74021  932.04205 1735.221290 1638.91857  920.905530  871.641699
## 9 1235.33525 1396.05946  862.728124  779.01058 1030.997852  939.815398
```

```
#for learn dimension of the data
dim(exp_data)
```

```
## [1] 59305    12
```

Part b: Checking the Sample Names and Separating them into Categories

At this stage, the goal was to extract sample indices from a dataset based on patterns corresponding to specific experimental categories. First, sample names were obtained from the phenotype data, and then four categories and their respective regular expression patterns were defined. For each category, the indices of samples matching the patterns were found and printed using **grep()** and **grepl()**. These indices were stored in a list named indices_list, categorized according to the experimental conditions. Finally, the indices for each category **(chow_cortex_ind, chow_liver_ind, lckd_cortex_ind, lckd_liver_ind)** were transferred to separate objects **(with for loop)**.

```
sample_names = phenoData(gse[[1]])$title #controlling sample names
sample_names
```

```
##  [1] "Ob_cortex_chow_7w_1"      "Ob_cortex_chow_7w_2"
##  [3] "Ob_cortex_LCKD_chow_7w_1" "Ob_cortex_LCKD_chow_7w_2"
##  [5] "Ob_liver_chow_7w_1"       "Ob_liver_chow_7w_2"
##  [7] "Ob_liver_LCKD_7w_1"       "Ob_liver_LCKD_7w_2"
##  [9] "Ob_cortex_chow_7w_3"      "Ob_cortex_LCKD_chow_7w_3"
## [11] "Ob_liver_chow_7w_3"       "Ob_liver_LCKD_7w_3"
```

```r
categories = c("chow_cortex", "chow_liver", "lckd_cortex", "lckd_liver")
patterns = c("Ob_cortex_chow_7w_(1|2|3)",
              "Ob_liver_chow_7w_(1|2|3)",
              "Ob_cortex_LCKD_chow_7w_(1|2|3)",
              "Ob_liver_LCKD_7w_(1|2|3)")

# Initialize empty lists to store indices
indices_list = list()

# Loop through each category and pattern
for (i in 1:length(categories)) {
  category = categories[i]
  pattern = patterns[i]

  # Find the indices
  number_indices = grep(pattern, sample_names)
  indices = which(grepl(pattern, sample_names))

  # Store the indices in the list
  indices_list[[category]] = indices

  # Print the number of samples and their indices
  print(paste("Number of samples for", gsub("_", "-", category), ":", length(number_indice
s)))
  print(paste("Indices of samples for", gsub("_", "-", category), ":", paste(indices, collaps
e = ", ")))
}
```

```
## [1] "Number of samples for chow-cortex : 3"
## [1] "Indices of samples for chow-cortex : 1, 2, 9"
## [1] "Number of samples for chow-liver : 3"
## [1] "Indices of samples for chow-liver : 5, 6, 11"
## [1] "Number of samples for lckd-cortex : 3"
## [1] "Indices of samples for lckd-cortex : 3, 4, 10"
## [1] "Number of samples for lckd-liver : 3"
## [1] "Indices of samples for lckd-liver : 7, 8, 12"
```

```r
# Assign the indices to the corresponding variables
chow_cortex_ind = indices_list[["chow_cortex"]]
chow_liver_ind = indices_list[["chow_liver"]]
lckd_cortex_ind = indices_list[["lckd_cortex"]]
lckd_liver_ind = indices_list[["lckd_liver"]]
```

Part c: Calculation of p-Values Using t-test

At this stage, **t-tests** were conducted for each gene to compare regular (chow) diet and LCKD conditions in cortex and liver samples, and the results were recorded as p-values. First, empty vectors were created to store the p-values. For each gene, a t-test was performed between the regular diet and LCKD conditions in cortex samples, and the p-value was recorded in the p_values_cortex vector. The same procedure was done for liver samples, with p-values recorded in the p_values_liver vector. As requested, a for loop was used to calculate the p-values for each gene. Finally, both p-value vectors were transformed using the **log10()** function, and the first few log p-values were displayed. The aim here is to identify significant differences in gene expression levels.

Using the log10 transformation makes the p-values more interpretable across a wide range. This transformation, particularly for small p-values, facilitates detailed examination by allowing a broader data distribution to be displayed on a single plot.

```
#Create empty vectors to store p-values
p_values_cortex = c()
p_values_liver = c()

#Perform t-test for each gene comparing mRNA levels of regular(chow)-diet and LCKD cases for
cortex and liver separately
for (i in 1:nrow(exp_data)) {
  # Perform t-test for cortex
  t_test_cortex = t.test(exp_data[i, chow_cortex_ind], exp_data[i, lckd_cortex_ind])
  # Store p-value
  p_values_cortex = c(p_values_cortex, t_test_cortex$p.value)

  # Perform t-test for liver
  t_test_liver = t.test(exp_data[i, chow_liver_ind], exp_data[i, lckd_liver_ind])
  # Store p-value
  p_values_liver = c(p_values_liver, t_test_liver$p.value)
}

# Convert p-values to log10 scale
log_p_values_cortex = -log10(p_values_cortex)
print(paste("logP Values for Cortex:", head(log_p_values_cortex)))
```

```
## [1] "logP Values for Cortex: 0.0108659692602695"
## [2] "logP Values for Cortex: 0.937721645952871"
## [3] "logP Values for Cortex: 0.038476933828671"
## [4] "logP Values for Cortex: 0.0985004491897129"
## [5] "logP Values for Cortex: 0.00373460281165677"
## [6] "logP Values for Cortex: 0.867986003042585"
```

```
log_p_values_liver = -log10(p_values_liver)
print(paste("logP Values for Liver:", head(log_p_values_liver)))
```

```
## [1] "logP Values for Liver: 1.07917302260636"
## [2] "logP Values for Liver: 0.358039250501056"
## [3] "logP Values for Liver: 0.0826666530845185"
## [4] "logP Values for Liver: 0.393483892316674"
## [5] "logP Values for Liver: 0.0531764209066963"
## [6] "logP Values for Liver: 0.393128475603098"
```

Create a new data frame for plot:

```
data_for_plot = data.frame(log_p_values_cortex, log_p_values_liver)
head(data_for_plot)
```

```
##   log_p_values_cortex log_p_values_liver
## 1         0.010865969         1.07917302
## 2         0.937721646         0.35803925
## 3         0.038476934         0.08266665
## 4         0.098500449         0.39348389
## 5         0.003734603         0.05317642
## 6         0.867986003         0.39312848
```

Subsequently, a scatter plot of the log10-transformed p-values for cortex and liver was created. Upon evaluating the plot, a negative correlation between log10(p-value) for cortex and log10(p-value) for liver was observed. To explain this: low p-values in one tissue (high significance) are generally associated with high p-values (low significance) in the other tissue. This is particularly evident when the log10(p-value) for cortex is between 0 and 2. The majority of the data points are concentrated at low values on both the log10(p-value) cortex and log10(p-value) liver axes. This indicates that many tests in both tissues have low p-values and are statistically significant.

The plot (Figure 1) also shows that the log10(p-values) for cortex and liver are quite high at some points. This indicates that some tests have high p-values in both tissues and are not significant. Additionally, some outlier points are noticeable in the plot. Notably, the points in the upper left and lower right corners are prominent outliers, representing tests that have high p-values (non-significant) in one tissue and low p-values (significant) in the other.

```
plot_1 = ggplot(data = data_for_plot)+
  aes(x = log_p_values_cortex,
      y = log_p_values_liver)+
  geom_point()+
  labs(title = "Relation between Log10 of p-values of Cortex and Liver",
       x = "Log10 of p-values for Cortex",
       y = "Log10 of p-values for Liver")+
  centered.plot.title
plot_1
```

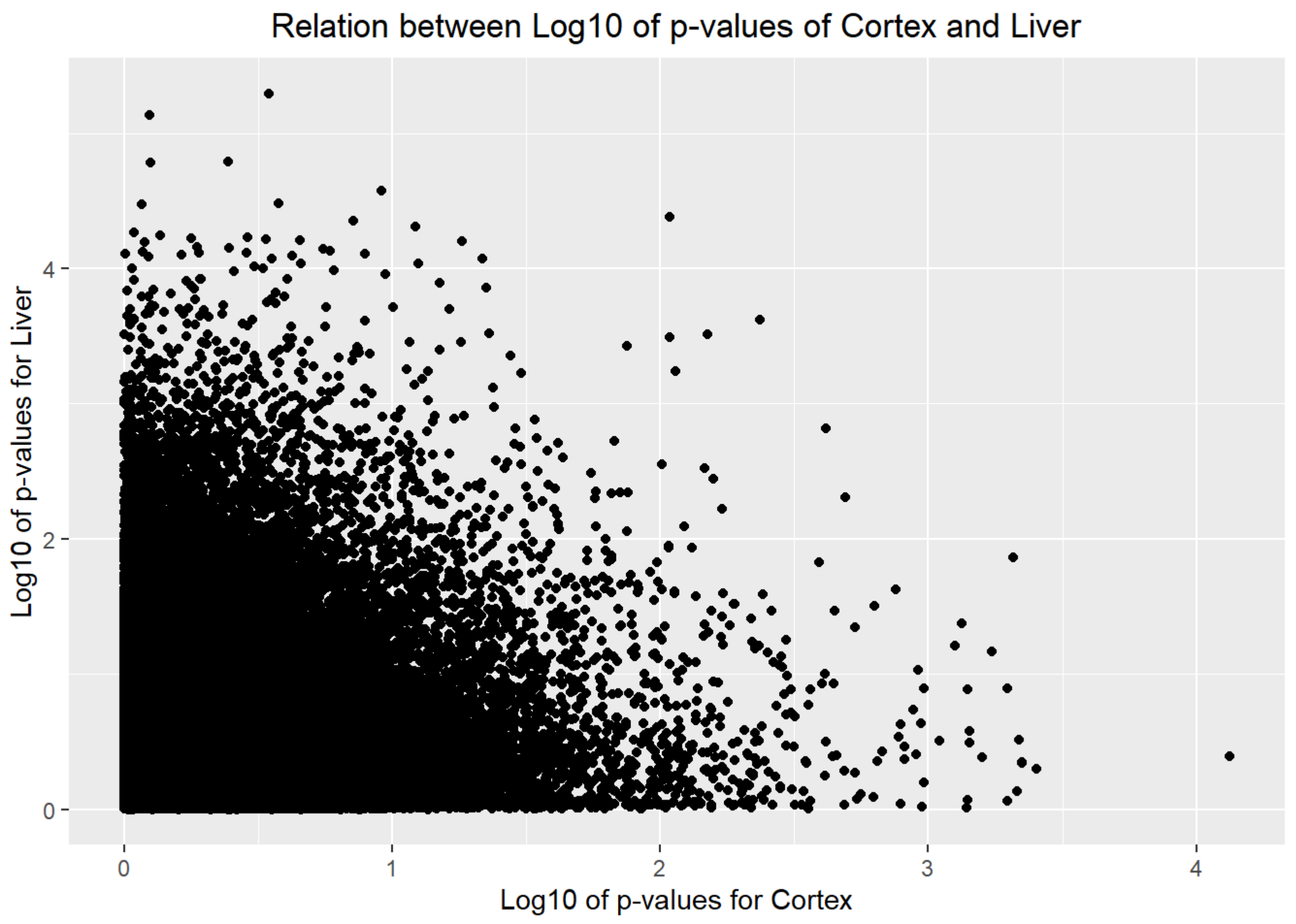

Figure 1. Relation between Log10 of p-values of Cortex and Liver

Part d: Achieving Genomic Interval from the Lowest p-value Data (with BioMart and Genomic Features Packages)

At this stage, the top five genes with the lowest p-values in the cortex data were identified, and the associated gene symbols and gene IDs were obtained using the given data. As suggested, the order() function was used to determine the indices of the lowest p-values. Subsequently, the gene symbols, ENSEMBL IDs, and chromosome information etc. were retrieved.

Indices of the top five genes with the lowest p-values are:

```
# Identify the indices of the five genes with the lowest p-values in the cortex data
lowest_p_values_indices = order(p_values_cortex)[1:5]
lowest_p_values_indices
```

```
## [1]  5543 45523 14736  7395 29195
```

The complete information for the relevant indices is provided below:

```
# Retrieve the data for the genes with the lowest p-values
lowest_genes = fData(gse[[1]])[lowest_p_values_indices, ]
lowest_genes[1:5, 1:5] #only for showed
```

```
##          ID COL ROW          NAME       SPOT_ID
## 5909   5909 174 320 A_55_P2055834
## 48389 48389  45 311 A_55_P2014041
## 15688 15688 145 113  A_51_P279050
## 7875   7875 168 324 A_55_P1965945 A_55_P1965945
## 31006 31006  98 309  A_66_P134497
```

The ENSEMBL IDs of the top five genes with the lowest p-values are:

```
# Extract ENSEMBL IDs and chromosomal locations of these genes
ids = lowest_genes$ENSEMBL_ID
ids
```

```
## [1] ""                   "ENSMUST00000113025" "ENSMUST00000028977"
## [4] "ENSMUST00000119898" "ENSMUST00000151176"
```

The chromosomes of the top five genes with the lowest p-values are: Chromosome 5, 2, and 1.

```
five_chromosome = lowest_genes$CHROMOSOMAL_LOCATION
five_chromosome
```

```
## [1] "chr5:145528663-145528604" "chr2:35035844-35035785"
## [3] "chr2:153157054-153157113" "chr1:39427887-39428178"
## [5] "chr1:175385660-175385719"
```

To obtain the Entrez IDs of these genes, the BioMart package was used. At this point, the database to be used was defined within the **mmusculus** variable.

```
# Use BioMart to convert ENSEMBL IDs to Entrez IDs
mmusculus = useMart("ensembl", dataset = "mmusculus_gene_ensembl")
entrez_ids = getBM(attributes = c('ensembl_gene_id', 'entrezgene_id'),
                   filters = 'ensembl_transcript_id',
                   values = ids,
                   mart = mmusculus)
entrez_ids
```

```
##      ensembl_gene_id entrezgene_id
## 1 ENSMUSG00000027475         16569
## 2 ENSMUSG00000026878         68365
## 3 ENSMUSG00000062896            NA
## 4 ENSMUSG00000037860        383619
```

Genes for which information was not available were displayed as NA, and these genes were removed from the relevant list using the **na.omit()** function. Subsequently, to define the genomic intervals of the genes for which information was available, the GenomicFeatures package was used. As requested, matches were defined in a single line using the meta data column in the **“TxDb.Mmusculus.UCSC.mm9.knownGene”** TxDb mouse database and the **%in%** operator. Finally, the desired information was stored and displayed in the **matched_genomic_intervals** variable.

```
# Filter out NA values
entrez_ids = na.omit(entrez_ids)

# Use the TxDb database to determine genomic ranges
txdb = TxDb.Mmusculus.UCSC.mm9.knownGene

# Retrieve genomic ranges based on Entrez IDs
genomic_intervals = genes(txdb, filter = list(gene_id = entrez_ids$entrezgene_id))

# Match genomic intervals to the retrieved Entrez IDs
matched_genomic_intervals = genomic_intervals[genomic_intervals$gene_id %in% entrez_ids$entre
zgene_id]
matched_genomic_intervals
```

```
## GRanges object with 3 ranges and 1 metadata column:
##          seqnames              ranges strand |     gene_id
##             <Rle>           <IRanges>  <Rle> | <character>
##    16569     chr2 153117152-153159125      + |       16569
##   383619     chr1 175350735-175396167      + |      383619
##    68365     chr2   35035725-35056640      - |       68365
##   -------
##   seqinfo: 35 sequences (1 circular) from mm9 genome
```

Part e: Conversion into a Data Frame and its Categorization

At this stage, the dataset was prepared for analysis and visualization. Operations such as preparing the data frame, transposing it, extracting sample names, and categorizing them were performed.

The expression data was converted into a dataframe and transposed (rows as samples, columns as genes). The as.data.frame() function was used to ensure that the output is a dataframe. Then, this dataframe was displayed using the head() function. Subsequently, a new column was added to the dataframe to replace sample names with 4 category names, and regular expressions were used to change the new category names with phenoData and “title”. (New category names: brain_lckd, liver_lckd, brain_chow, liver_chow). Next, the dataframe was transposed and assigned to another dataframe named transposed_df, which was then displayed for verification.

Following that, sample_titles were obtained from the gse object using phenoData and “title” information. These names were saved in sample_titles and printed. Then, an empty vector named category_names was created. Category names were determined by matching sample names with a pattern for each sample. The category names were added to the category_names vector based on the matching patterns, and the results were displayed. The obtained category_names were added as a new column to the transposed_df dataframe, and this column was converted to a factor variable.

```
exp_data_df = as.data.frame(exp_data)
head(exp_data_df)
```

```
##    GSM3175742  GSM3175743  GSM3175744  GSM3175745 GSM3175746 GSM3175747
## 4 3267.271195 3213.865410 2982.575977 2831.704422 4297.18725 4504.76973
## 5   11.083762   15.734114   23.303580   19.798172   13.27260   20.02929
## 6    5.692115    7.692918    5.738932    7.306098   13.35453   14.09231
## 7  186.951488  181.276385  144.046990  172.811729  100.78696  129.50722
## 8 2049.208254 2167.279325 2147.740836 2172.501503  870.82899  797.26675
## 9  859.448143  861.837129  824.603239  842.584780  994.22146 1123.41089
##   GSM3175748 GSM3175749  GSM3175750 GSM3175751  GSM3175752  GSM3175753
## 4 7067.61545 7375.37182 3917.699844 4648.33260 4009.952174 5090.904378
## 5   14.60719  294.44233    9.572488   13.92412    9.399971    6.291990
## 6   14.70040   13.71533    8.021901    8.02426    9.451548    6.334297
## 7  207.47205  106.97183  214.953214  240.31419   96.736435  114.934933
## 8  808.74021  932.04205 1735.221290 1638.91857  920.905530  871.641699
## 9 1235.33525 1396.05946  862.728124  779.01058 1030.997852  939.815398
```

```
# Transpose the dataframe
transposed_df = t(exp_data_df)
transposed_df = as.data.frame(transposed_df)
transposed_df[1:5,1:5] #for example
```

```
##                   4        5         6        7        8
## GSM3175742 3267.271 11.08376  5.692115 186.9515 2049.208
## GSM3175743 3213.865 15.73411  7.692918 181.2764 2167.279
## GSM3175744 2982.576 23.30358  5.738932 144.0470 2147.741
## GSM3175745 2831.704 19.79817  7.306098 172.8117 2172.502
## GSM3175746 4297.187 13.27260 13.354535 100.7870  870.829
```

```
# Use phenoData "title" field to get the sample names
sample_titles = gse[[1]]@phenoData$title
sample_titles
```

```
##  [1] "Ob_cortex_chow_7w_1"      "Ob_cortex_chow_7w_2"
##  [3] "Ob_cortex_LCKD_chow_7w_1" "Ob_cortex_LCKD_chow_7w_2"
##  [5] "Ob_liver_chow_7w_1"       "Ob_liver_chow_7w_2"
##  [7] "Ob_liver_LCKD_7w_1"       "Ob_liver_LCKD_7w_2"
##  [9] "Ob_cortex_chow_7w_3"      "Ob_cortex_LCKD_chow_7w_3"
## [11] "Ob_liver_chow_7w_3"       "Ob_liver_LCKD_7w_3"
```

```
# Initialize an empty vector to store the category names
category_names = c()

# Loop through each sample title and perform pattern matching
for (title in sample_titles) {
  if (grepl("Ob_cortex_chow_7w_(1|2|3)", title)) {
    category_names = c(category_names, "brain_chow")
  } else if (grepl("Ob_liver_chow_7w_(1|2|3)", title)) {
    category_names = c(category_names, "liver_chow")
  } else if (grepl("Ob_cortex_LCKD_chow_7w_(1|2|3)", title)) {
    category_names = c(category_names, "brain_lckd")
  } else if (grepl("Ob_liver_LCKD_7w_(1|2|3)", title)) {
    category_names = c(category_names, "liver_lckd")
  }
}
category_names
```

```
##  [1] "brain_chow" "brain_chow" "brain_lckd" "brain_lckd" "liver_chow"
##  [6] "liver_chow" "liver_lckd" "liver_lckd" "brain_chow" "brain_lckd"
## [11] "liver_chow" "liver_lckd"
```

```
# Add category_names as a column to transposed_df
transposed_df = as.data.frame(cbind(category_names, transposed_df))
# Convert the "category_names" column to a factor
transposed_df$category_names = factor(transposed_df$category_names)
```

```
# Rename the new column
colnames(transposed_df)[22592] = "Gm2a" #for easy next steps
transposed_df = as.data.frame(transposed_df)
```

Part f: Visualization of the Expression Level of the Gm2a Gene for each Category with Boxplot

At this stage, the aim was to extract the expression level of the Gm2a gene and compare its expression levels across different categories. To achieve this, the index of the Gm2a gene was first obtained, and its values were checked and verified.

```
which(gse[[1]]@featureData@data$GENE_SYMBOL == "Gm2a") #finding the index of the Gm2a gene
```

```
## [1] 22591
```

```
gse[[1]]@featureData@data$GENE_SYMBOL[22591]#for validation index information
```

```
## [1] "Gm2a"
```

IMPORTANT: Since the new column is added to the first column, we need to select the 22592nd column for the Gm2a gene from the transposed data.

```
#for controlling, involves same info
exp_data_df[22591,]
```

```
##       GSM3175742 GSM3175743 GSM3175744 GSM3175745 GSM3175746 GSM3175747
## 24011   582.6926   607.5126    605.588   618.7102   2030.561   2164.087
##       GSM3175748 GSM3175749 GSM3175750 GSM3175751 GSM3175752 GSM3175753
## 24011    811.158   1041.037   636.5474   602.4334   2387.728    863.032
```

```
transposed_df[,22592]
```

```
##  [1]  582.6926  607.5126  605.5880  618.7102 2030.5607 2164.0872  811.1580
##  [8] 1041.0367  636.5474  602.4334 2387.7280  863.0320
```

After confirming the relevant gene information, the plot_Gm2a variable was created. This variable stores the ggplot2 object that will be used to visualize the results as a graph. From the transposed_df dataframe, only the first column (category names) and the column of the Gm2a gene (22592) were selected and grouped according to the selected data. Then, a boxplot was created using ggplot2 (with geom_boxplot()), which contains boxplots showing the expression levels of the Gm2a gene for each category.

Based on the obtained graph, the categories “brain_chow” and “brain_lckd” represent brain tissue samples, while “liver_chow” and “liver_lckd” represent liver tissue samples. The boxplots visualize the central tendency, data spread, and potential outliers. Based on this data, we can make the following interpretations:

In brain tissue samples, the “brain_chow” group exhibits a higher expression of the Gm2a gene compared to the “brain_lckd” group. This suggests that a specific diet may influence gene expression in brain tissue. In liver tissue samples, the “liver_chow” group shows a higher expression of the Gm2a gene compared to the “liver_lckd” group. This indicates that a specific diet may affect gene expression in liver tissue as well. Additionally, both in brain and liver tissue, it is observed that the “lckd” group generally has lower Gm2a gene expression.

```
plot_Gm2a = transposed_df %>%
  select(c(1, 22592)) %>% #for select the first column/category information and Gm2a column
  group_by(category_names) %>% #grouping by the category
  ggplot(aes(x = category_names,
             y = Gm2a,
             color = category_names))+ #color scale by category
  geom_boxplot()+ #boxplot have added
  geom_jitter(alpha = 0.9)+
  labs(title = "Boxplot of Gm2a for Category Names",
     x = "Category Names",
     y = "Gm2a Expression Level")+
  centered.plot.title
plot_Gm2a
```

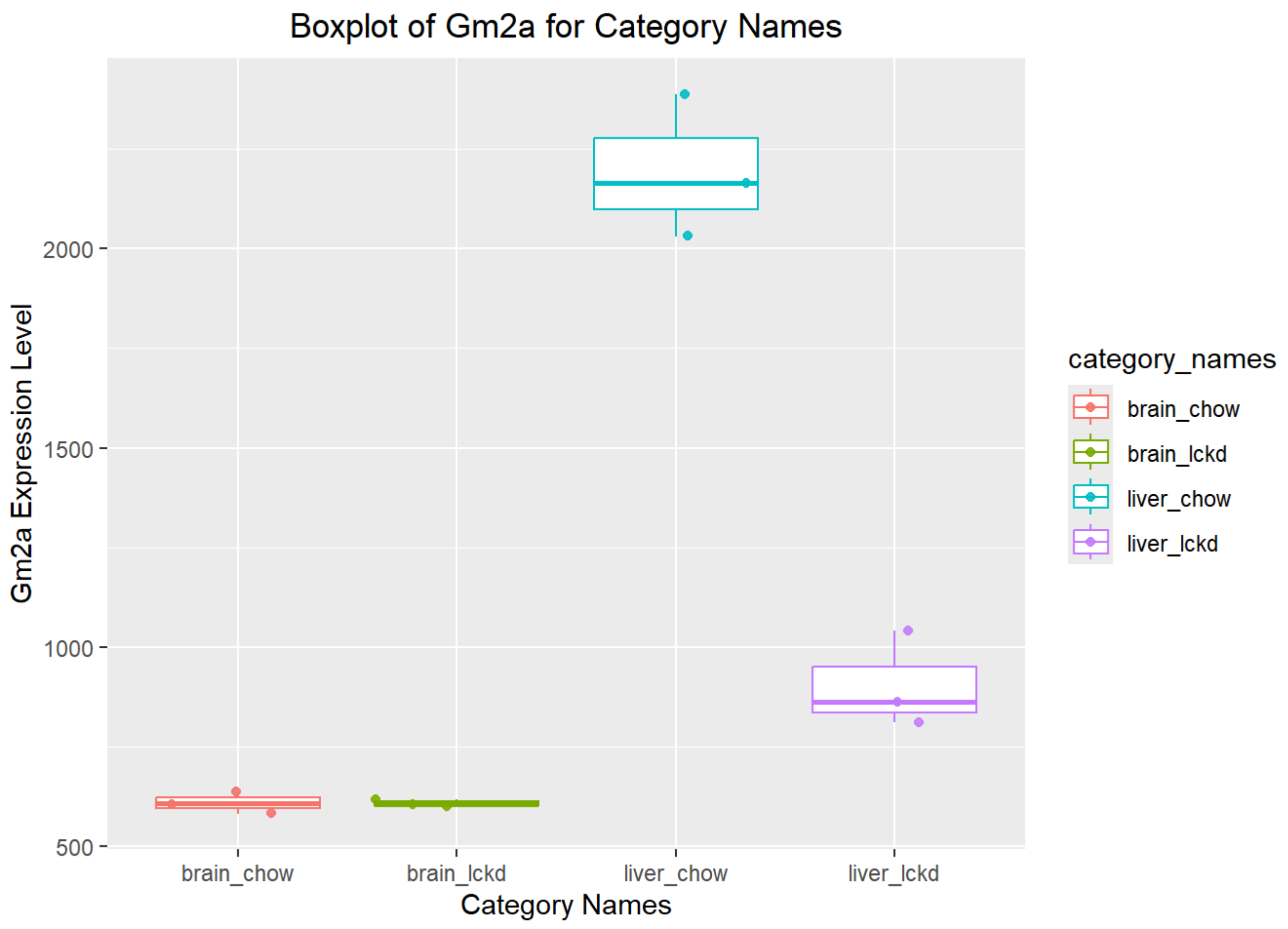

Figure 2. Boxplot of Gm2a for Category Names

Part g: Calculation of mRNA Expression Levels according to each Category for each Gene

At this stage, the average mRNA expression for each sample category for every gene in the dataset was calculated, and the results were stored in a new data frame. This new dataframe consists of approximately 50,000 rows, each corresponding to one of the four categories, and columns representing the genes. The row names were replaced with the respective category names, and the category column was removed from the dataframe. Finally, the transpose of this dataframe was taken to ensure that the rows represent genes and the columns represent categories.

```
# Calculate the mean mRNA expression levels for each category
mRNA_df = transposed_df %>%
  group_by(category_names) %>%
  summarise_all(mean, na.rm = TRUE)
```

```
# Set row names to the values of the 'category_names' column
rownames = mRNA_df$category_names

# Remove the 'category_names' column from the dataframe
mRNA_df_2 = mRNA_df %>%
  select(-category_names)

# Change row names with the 'rownames' vector
mRNA_df_2 = as.data.frame(mRNA_df_2)
rownames(mRNA_df_2) = rownames

# Transpose the dataframe
new_mRNA_df = t(mRNA_df_2)

# Convert to dataframe
new_mRNA_df = as.data.frame(new_mRNA_df)
head(new_mRNA_df)
```

```
##    brain_chow  brain_lckd liver_chow liver_lckd
## 4 3466.278816 3487.537667 4270.63638 6511.29722
## 5   12.130121   19.008623   14.23395  105.11384
## 6    7.135645    7.023097   12.29946   11.58334
## 7  194.393696  185.724303  109.01021  143.12627
## 8 1983.902956 1986.386969  863.00042  870.80799
## 9  861.337799  815.399531 1049.54340 1190.40337
```

Part h: Correlation of Gene Expression Levels between the Same Diet Type and Different Tissues

At this stage, the focus was on examining and visualizing the relationship between gene expression levels in different diet and tissue types. In the obtained graphs, mRNA levels between the “brain_lckd” and “liver_lckd” categories were compared with those of the “brain_chow” and “liver_chow” categories.

The scatter plot in Figure 3 illustrates the relationship between LCKD (Liver X Receptor Knockout Diet) and hepatic LCKD mRNA levels. The graph focuses on the distribution of mRNA levels between the two tissues. A significant portion of the data points are concentrated at lower mRNA levels (ranging from 0 to 200,000 on both the x and y axes). This indicates that low mRNA levels are common in both tissues. In contrast, data points are more sparsely distributed at higher mRNA levels (above 200,000 on the x-axis and above 300,000 on the y-axis). This suggests that high mRNA levels are rare in both tissues.

Although a clear linear relationship is not observed in the visual, there are instances where high mRNA levels increase simultaneously in both tissues. This implies that some genes are similarly expressed in both tissues. There is a dense clustering of data points within the 0-200,000 range on the x-axis and the 0-300,000 range on the y-axis. This indicates that mRNA levels in the brain and liver are generally lower. The density of points at the lower ends of the x and y axes suggests that mRNA levels are predominantly expressed at low levels.

```
#For this task, we'll use the dataframe obtained in part (g).
#Create a scatter plot between brain_lckd and liver_lckd categories.
lckd_brain_liver_plot = new_mRNA_df %>%
  select(brain_lckd, liver_lckd) %>%
  ggplot(aes(x = brain_lckd,
             y = liver_lckd)) +
  geom_point() + # Add points to the plot
  labs(x = "Brain LCKD mRNA Levels",
       y = "Liver LCKD mRNA Levels",
       title = "Scatter Plot of mRNA Levels between Brain LCKD and Liver LCKD") + # Set label
s and title
  centered.plot.title  # Center the plot title
lckd_brain_liver_plot
```

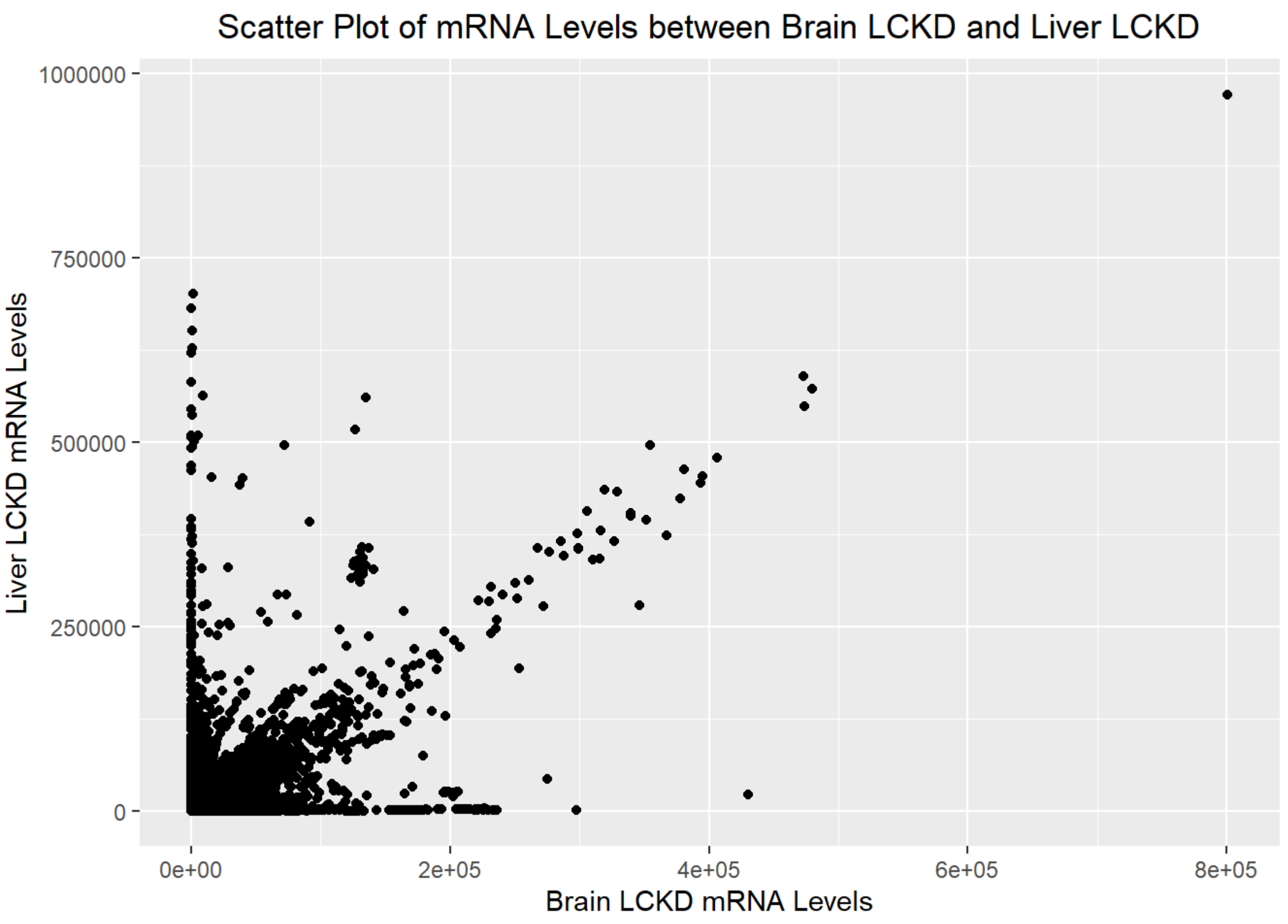

Figure 3. Scatter Plot of mRNA Levels between Brain LCKD and Liver LCKD

The scatter plot in Figure 4 features “Brain chow mRNA Levels” on the horizontal axis (x-axis) and “Liver chow mRNA Levels” on the vertical axis (y-axis). Each point represents a comparison of mRNA levels between brain and liver tissues. Many points are concentrated in the lower left corner of the graph, indicating that numerous samples have low mRNA levels in both brain and liver tissues. Conversely, samples with high mRNA levels are less common, showing that high mRNA levels are also rare in this graph.

Some points exhibit high values on the x-axis while showing low values on the y-axis, and vice versa. The graph also highlights several outliers with particularly high values on the y-axis, indicating samples with very high mRNA levels in liver tissue but relatively low levels in brain tissue. Similarly, a few outliers with high values on the x-axis are observed, suggesting samples with high mRNA levels in brain tissue but low levels in liver tissue.

The scatter plot shows points scattered without a specific pattern, indicating a wide variety in mRNA levels between brain and liver tissues and a lack of homogeneity in their distribution. This suggests that mRNA levels in brain and liver tissues are distributed across a broad range and that there is no clear correlation between the two tissues.

```
#another plot between brain_chow and liver_chow categories.
chow_brain_liver_plot = new_mRNA_df %>%
  select(brain_chow, liver_chow) %>%
  ggplot(aes(x = brain_chow,
             y = liver_chow)) +
  geom_point() + # Add points to the plot
  geom_jitter(alpha = 0.1) + # Add jitter to avoid overlapping points
  labs(x = "Brain chow mRNA Levels",
       y = "Liver chow mRNA Levels",
       title = "Scatter Plot of mRNA Levels between Brain chow and Liver chow") + # Set label
s and title
  centered.plot.title  # Center the plot title
chow_brain_liver_plot
```

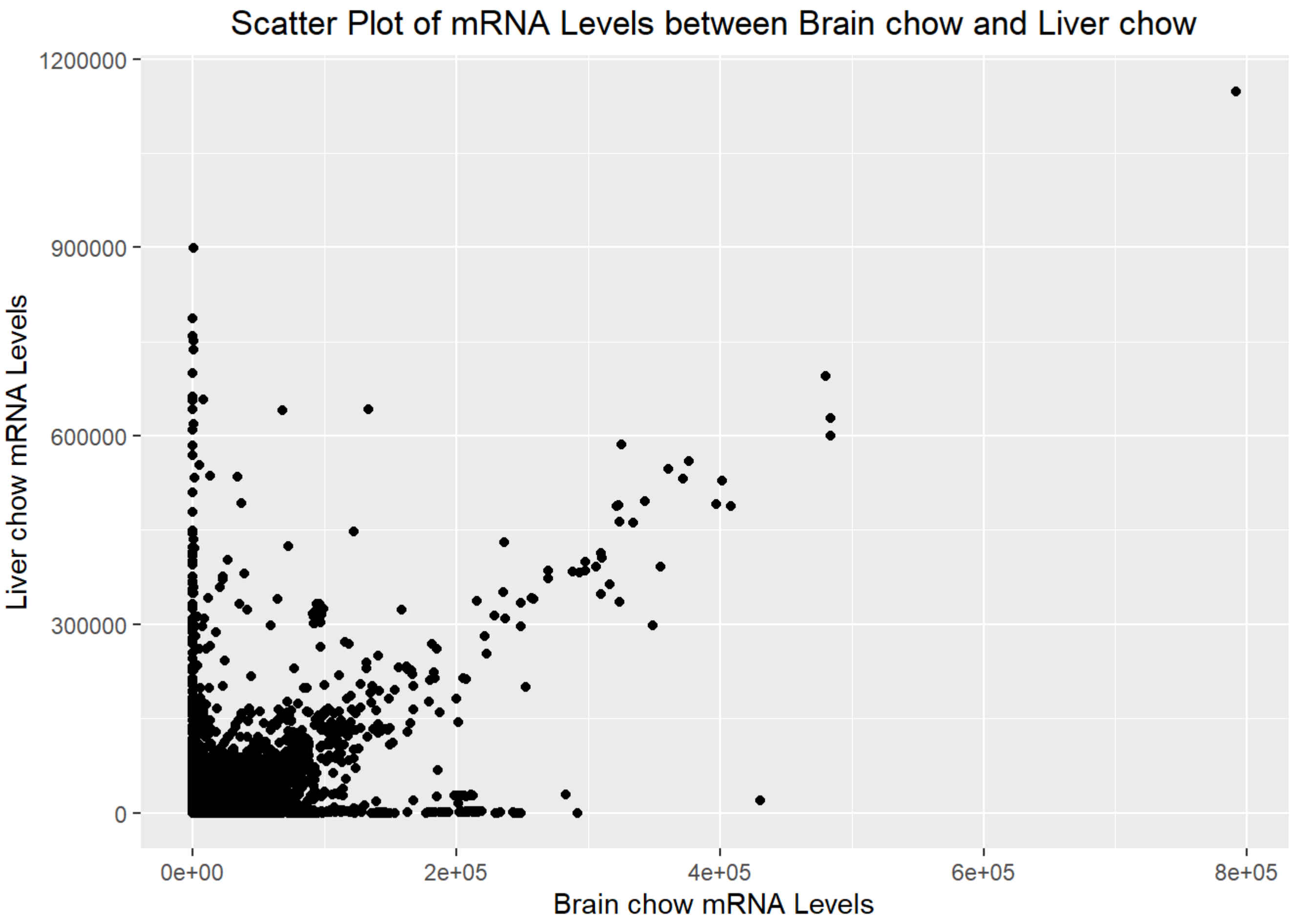

Figure 4. Scatter Plot of mRNA Levels between Brain chow and Liver chow

Part i: Converting Data to Tidy Format and Visualizing the Distribution of mRNA Levels for 4 Different Categories with Boxplot (>1000: mRNA expression level)

At this stage, the aim was to create a boxplot to compare the expression levels of all genes in the four categories. To achieve this, we first organized the data using the tidyR package: instead of four columns, each listing category names and their corresponding expression values for each gene, we wanted two columns, one for gene names and the other for their corresponding expression values. Using this organized version, we filtered rows with values greater than 1000, and then created a boxplot for each of the four categories to show the distribution of mRNA levels.

```
tidy_expr_data = new_mRNA_df %>%
  pivot_longer(cols = c("brain_chow", "brain_lckd", "liver_chow", "liver_lckd"),
               names_to = "Category", values_to = "Expression_Level") %>%
  filter(Expression_Level > 1000)
head(tidy_expr_data)
```

```
## # A tibble: 6 × 2
##   Category   Expression_Level
##   <chr>                 <dbl>
## 1 brain_chow            3466.
## 2 brain_lckd            3488.
## 3 liver_chow            4271.
## 4 liver_lckd            6511.
## 5 brain_chow            1984.
## 6 brain_lckd            1986.
```

Figure 5 presents a boxplot comparing mRNA levels across four different sample categories: “brain_chow,” “brain_lckd,” “liver_chow,” and “liver_lckd.” The data exhibit a wide distribution of mRNA expression levels and similar median values across all categories, suggesting no significant differences between them. While each category contains some outliers, the “liver_chow” category notably features outliers with exceptionally high expression levels. The similarity in the interquartile range across the categories indicates that the variance in mRNA levels is comparable among them. Overall, no substantial differences are observed between brain (brain_chow and brain_lckd) and liver (liver_chow and liver_lckd) samples or between the different diets (chow and lckd), indicating that mRNA expression levels are largely similar across these categories.

```
tidy_expr_plot = ggplot(tidy_expr_data,
       aes(x = Category,
           y = Expression_Level,
           color = Category)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.1) +
  theme_minimal() +
  labs(x = "Sample Category",
       y = "Expression Level",
       title = "Boxplot of mRNA levels across categories")+
  centered.plot.title
tidy_expr_plot
```

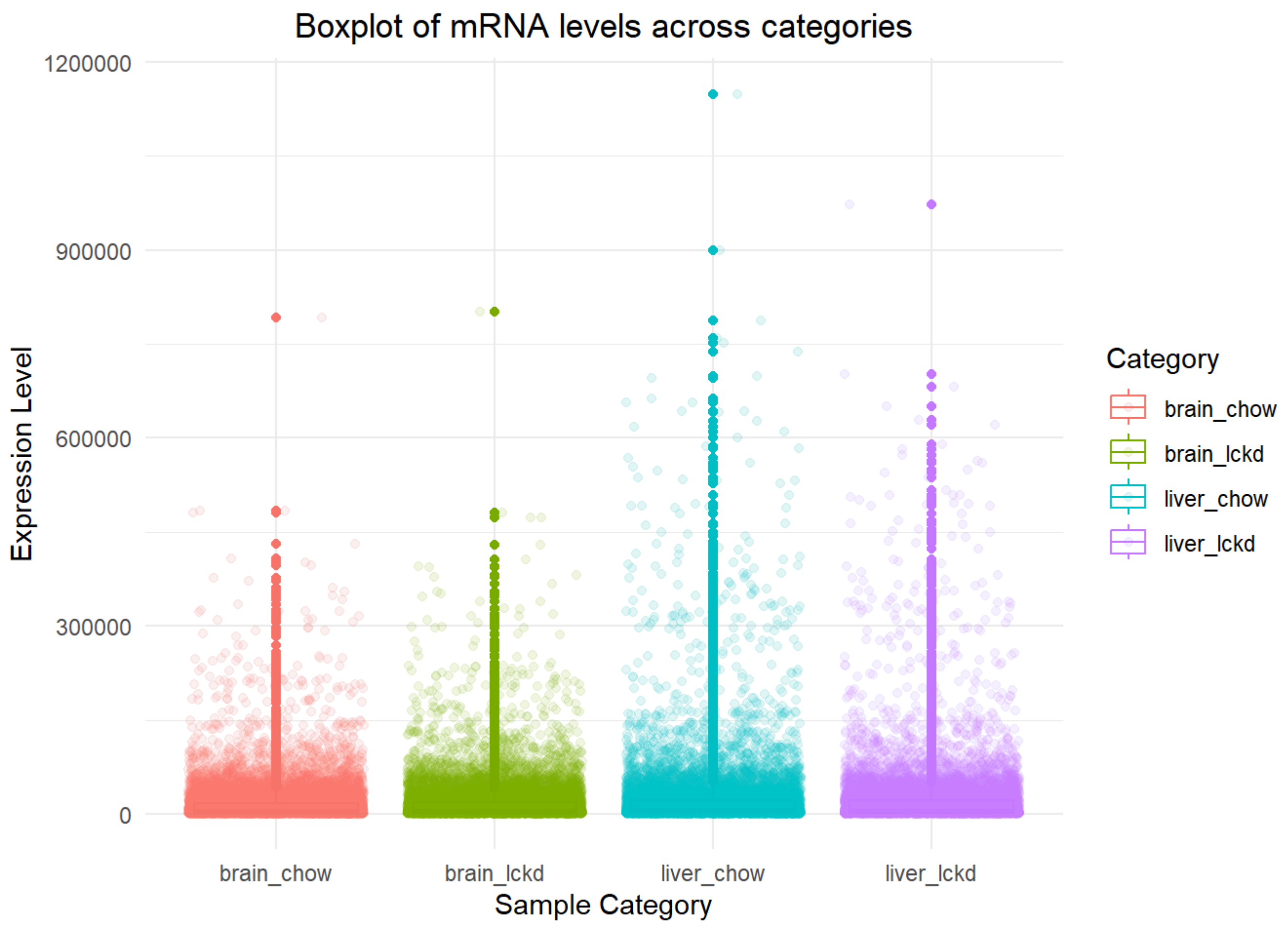

Figure 5. Boxplot of mRNA levels across categories