Lab 7

Maria Bassem_20011141

2024-05-01

```
library(ggplot2)
library(gridExtra)

## Warning: package 'gridExtra' was built under R version 4.3.3

# if (!require("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
# BiocManager::install("limma")
library(limma)

library(data.table)
```

Part 1: Data Wrangling (25 points)

Task 1.1: Data Acquisition (5 Points)

- Read the dataset CSV file into R. For the remainder of this lab, you are supposed to work with this dataframe and it will be referred to as 'your dataframe'.

- Notes about your data:

- Make sure you have 130 samples x 54673 genes
- You have two extra columns, one for the phenotype and one for the sample number
- Notice the orientation of your dataframe where rows are samples and columns are genes

```
my_dataframe <- fread("D:\\Third Year Computer\\Term 2\\Bio\\Labs\\Lab
7\\BrainCancerNA.csv")

my_dataframe <- as.data.frame(my_dataframe)
nrow(my_dataframe)

## [1] 130

ncol(my_dataframe)

## [1] 54547

head(my_dataframe[, 1:2])

##    V1       type
## 1 834 ependymoma
## 2 835 ependymoma
## 3 836 ependymoma
## 4 837 ependymoma
## 5 838 ependymoma
## 6 839 ependymoma
```

- Remove the sample id column and put it as rownames. Use the rownames() function

```
rownames(my_dataframe) <- my_dataframe[,1]
row_names <- rownames(my_dataframe)
print(row_names[1:5])

## [1] "834" "835" "836" "837" "838"

# Remove the first column from my_dataframe
my_dataframe <- my_dataframe[,-1]
print(dim(my_dataframe))

## [1]   130 54546
```

- Extract the expression data into a separate dataframe which will be called expression.data

- By this point you should have two dataframes one with expression data only and the second one with expression data + phenotypes.

```
expression.data <- my_dataframe
expression.data <- expression.data[,-1]
head(expression.data[1:5])

##     X1007_s_at X1053_at  X117_at  X121_at X1255_g_at
## 834   7.088426 6.112547 6.515819 7.015668   6.638210
## 835   7.898508 6.405272 6.845297 7.565933         NA
## 836   7.927454 6.930468 7.730750 7.399506   7.027887
## 837   7.015668 7.442440 6.338611 6.623962   6.564207
## 838   7.357802 7.137002 8.653812 7.088426         NA
## 839   7.040063 6.194321 6.918402 7.471887   6.060638

num_na_before <- sum(is.na(expression.data))
print(num_na_before)

## [1] 675281
```

Task 1.2: PCA Before QC (5 Points)

- Remove NAs by filling them with the means of their respective genes. (mean of gene across all samples)

```
library(tidyr)
col_means <- colMeans(expression.data, na.rm = TRUE)

filled_dataframe <- replace_na(expression.data, as.list(col_means))

head(filled_dataframe[1:3])
```

```
##     X1007_s_at X1053_at  X117_at
## 834   7.088426 6.112547 6.515819
## 835   7.898508 6.405272 6.845297
## 836   7.927454 6.930468 7.730750
## 837   7.015668 7.442440 6.338611
## 838   7.357802 7.137002 8.653812
## 839   7.040063 6.194321 6.918402
```

- Compute PCA using prcomp function. You are not allowed to use any other alternatives like princomp.
- You should think about how you should do your PCA. Should you use the genes as rows or as columns? How does the transformation matrix of each differ? Comment your findings.

```r
# Perform PCA using genes as rows [not preferred in prcomp]
pca_rows <- prcomp(t(filled_dataframe))
print(dim(pca_rows$x))

## [1] 54545   130

# Perform PCA using genes as columns [preferred in prcomp]
pca_columns <- prcomp(filled_dataframe)
print(dim(pca_columns$x))

## [1] 130 130
```

- **Using genes as rows (samples as columns)**:
  - If there are n samples and g genes (rows) in the dataframe, the dimensions of the transformed matrix will be g*k, where k is the number of principal components.
  - In our case, n = 130, g = 54545, k = 130 [PCs]. The dimension of transformation matrix = g*k = 54545*130
- **Using genes as columns:**
  - If there are n samples (rows) and g genes (columns) in the dataframe, the dimensions of the transformation matrix will be n*k, where k is the number of principal components.
  - In our case, n = 130, g = 54545, k = 130 [PCs]. The dimension of transformation matrix = n*k = 130*130

prcomp prefers to work with genes as columns and samples as rows. This way makes it easier to interpret the results of PCA as we try to reduce the dimensionality of the features (genes) not the samples.

- For the remainder of this lab, perform PCA using prcomp and the genes as columns. You should have a transformation matrix 130x130. We will work with this matrix so do not use the predict function.
- Make a dataframe of your principal components. This dataframe will be referred to as pcs.

```
# Create a dataframe of principal components
pcs <- as.data.frame(pca_columns$x)
print(dim(pcs))

## [1] 130 130
```

- Use the head function to view the data of your pcs

```
head(pcs[1:5])

##           PC1      PC2        PC3       PC4        PC5
## 834 -87.95760 10.01827 -43.181728 51.681932  -1.208282
## 835 -45.37387 19.11461  46.578166 12.875426 -29.005908
## 836 -38.32008 36.23841  12.946378 -4.893609 -11.580324
## 837  56.23692 21.16868  18.702611 27.928142  -4.900737
## 838 -35.27620 30.30553   9.106774 30.166660  -4.259258
## 839 -66.87661 16.78548 -17.107071 49.162971  -8.286427
```

Task 1.3: PCA Before QC [Visualization] (5 Points)

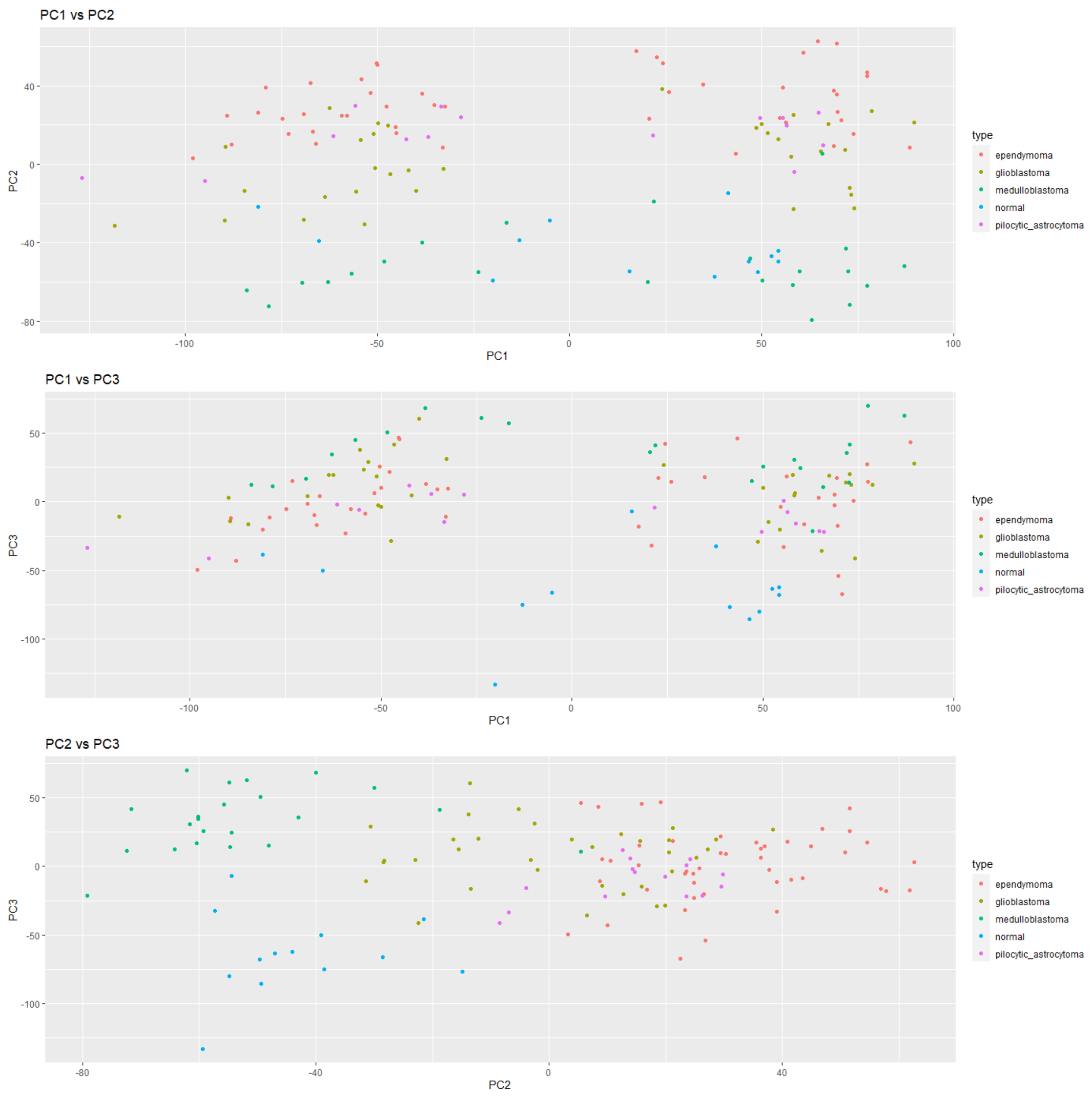
- Create plots comparing PC1 vs PC2, PC1 vs PC3, and PC2 vs PC3.

- You must use ggplot for each plot. Color according to the phenotype of each sample

```
pcs$type <- my_dataframe$type

# Create a scatter plot for PC1 vs PC2
#
pc1_pc2 <- ggplot(pcs, aes(x = PC1, y = PC2, color = type)) +
 geom_point() +
 labs(title = "PC1 vs PC2", x = "PC1", y = "PC2")

# Create a scatter plot for PC1 vs PC3
pc1_pc3 <- ggplot(pcs, aes(x = PC1, y = PC3, color = type)) +
  geom_point() +
  labs(title = "PC1 vs PC3", x = "PC1", y = "PC3")

# Create a scatter plot for PC2 vs PC3
pc2_pc3 <- ggplot(pcs, aes(x = PC2, y = PC3, color = type)) +
  geom_point() +
  labs(title = "PC2 vs PC3", x = "PC2", y = "PC3")
```

- You must view the three plots together in a figure of size 15x15. Use grid.arrange function to place them on 3 rows and 1 column. Think about how to change the figure size in markdown.

```
# Arrange the plots in a 3x1 grid
grid.arrange(pc1_pc2, pc1_pc3, pc2_pc3, ncol = 1)
```



Task 1.4: Data Cleaning (5 Points)

- For each gene, check for outliers (above/below three standard deviations)

- If outliers are present, remove them and place NAs instead.

```
col_stds <- apply(expression.data, 2, sd, na.rm = T)# 2 indicates that the
function should apply to each column
no_outliers.data <- expression.data
# Identify outliers (values above/below three standard deviations from the
mean) and replace them with NA
no_outliers.data[abs(no_outliers.data - col_means) > 3 * col_stds] <- NA
```

```
num_na_after <- sum(is.na(no_outliers.data))

print(num_na_before)

## [1] 675281

print(num_na_after)

## [1] 684246

print(num_na_after-num_na_before)

## [1] 8965
```

- Remove NAs by filling them with the means of their respective genes. (mean of gene across all samples)

```
# calculate the mean once more because the data was changed
col_outliers_means <- colMeans(no_outliers.data, na.rm = TRUE)
remove_outliers_data <- replace_na(no_outliers.data, as.list(col_means))
```

- Make sure that the data is in the right orientation (samples as rows and genes as columns)

```
# head(remove_outliers_data)
dim(remove_outliers_data)

## [1]   130 54545
```

- Perform normalization between genes using the quantile method. Use the normalizeBetweenArrays() function. Comment on what is this function and what the quantile method means.

```
normalized_data <- normalizeBetweenArrays(remove_outliers_data, method =
"quantile")
normalized_data <- as.data.frame(normalized_data)

head(normalized_data[1:5])

##     X1007_s_at X1053_at  X117_at  X121_at X1255_g_at
## 834   7.102080 6.134909 6.515459 7.037617   6.634232
## 835   7.852590 6.391484 6.876277 7.546383   7.132312
## 836   7.881815 6.903183 7.725809 7.366617   7.037617
## 837   7.024202 7.461060 6.367256 6.634232   6.568754
## 838   7.306970 7.121004 8.598797 7.112602   7.132312
## 839   7.051006 6.179602 6.943488 7.444840   6.081913

print(dim(normalized_data))

## [1]   130 54545
```

The normalizeBetweenArrays() function is used in bioinformatics to normalize gene expression data between different samples. Specifically, when the method parameter is set to “quantile” which ensures that the gene expression values have the same distribution across samples. It adjusts the expression values in such a way that the distribution of expression levels for each gene becomes consistent across all samples.

batch effect

This method addresses technical variations that may arise during the gene expression measurement process. Quantile normalization helps remove unwanted variations, making gene expression data more comparable and facilitating meaningful interpretations.

Task 1.5: Data Inspection (5 Points)

- Repeat tasks 1.2 and 1.3 but this is for the data after being cleaned.

Task 1.2: PCA After QC

- Remove NAs by filling them with the means of their respective genes. (mean of gene across all samples)

```
col_means_normalized_data <- colMeans(normalized_data, na.rm = TRUE)
filled_normalized_dataframe <- replace_na(normalized_data,
as.list(col_means_normalized_data))
head(filled_normalized_dataframe[1:5])

##     X1007_s_at X1053_at  X117_at  X121_at X1255_g_at
## 834   7.102080 6.134909 6.515459 7.037617   6.634232
## 835   7.852590 6.391484 6.876277 7.546383   7.132312
## 836   7.881815 6.903183 7.725809 7.366617   7.037617
## 837   7.024202 7.461060 6.367256 6.634232   6.568754
## 838   7.306970 7.121004 8.598797 7.112602   7.132312
## 839   7.051006 6.179602 6.943488 7.444840   6.081913
```

- Compute PCA using prcomp function. You are not allowed to use any other alternatives like princomp.
- You should think about how you should do your PCA. Should you use the genes as rows or as columns? How does the transformation matrix of each differ? Comment your findings.

```
# Perform PCA using genes as rows [not preferred in prcomp]
pca_norm_rows <- prcomp(t(filled_normalized_dataframe))
print(dim(pca_norm_rows$x))

## [1] 54545   130

# Perform PCA using genes as columns [preferred in prcomp]
pca_norm_columns <- prcomp(filled_normalized_dataframe)
print(dim(pca_norm_columns$x))

## [1] 130 130
```

- **Using genes as rows (samples as columns)**:
  - If there are n samples and g genes (rows) in the dataframe, the dimensions of the transformed matrix will be g*k, where k is the number of principal components.

  - In our case, n = 130, g = 54545, k = 130 [PCs]. The dimension of transformation matrix = g*k = 54545*130

- **Using genes as columns:**
  - If there are n samples (rows) and g genes (columns) in the dataframe, the dimensions of the transformation matrix will be n*k, where k is the number of principal components.
  - In our case, n = 130, g = 54545, k = 130 [PCs]. The dimension of transformation matrix = n*k = 130*130

prcomp prefers to work with genes as columns and samples as rows. This way makes it easier to interpret the results of PCA as we try to reduce the dimensionality of the features (genes) not the samples.

- For the remainder of this lab, perform PCA using prcomp and the genes as columns. You should have a transformation matrix 130x130. We will work with this matrix so do not use the predict function.
- Make a dataframe of your principal components. This dataframe will be referred to as pcs.

```
# Create a dataframe of principal components
pcs_norm <- as.data.frame(pca_norm_columns$x)
print(dim(pcs_norm))

## [1] 130 130
```

- Use the head function to view the data of your pcs

```
head(pcs_norm[1:5])

##            PC1       PC2       PC3        PC4         PC5
## 834 -86.64472 -12.05310  41.93098 -52.656262   0.9293421
## 835 -45.29489 -17.00631 -46.90271 -11.978154 -28.9314741
## 836 -38.18761 -35.31966 -14.09915   5.134287 -12.0786003
## 837  55.84369 -19.85633 -20.07638 -27.345551  -4.2277727
## 838 -35.08612 -29.73000 -10.49258 -29.981217  -3.2240553
## 839 -66.65246 -17.23718  15.69388 -49.625809  -6.2618599
```

Task 1.3: PCA Before QC [Visualization] (5 Points)

- Create plots comparing PC1 vs PC2, PC1 vs PC3, and PC2 vs PC3.

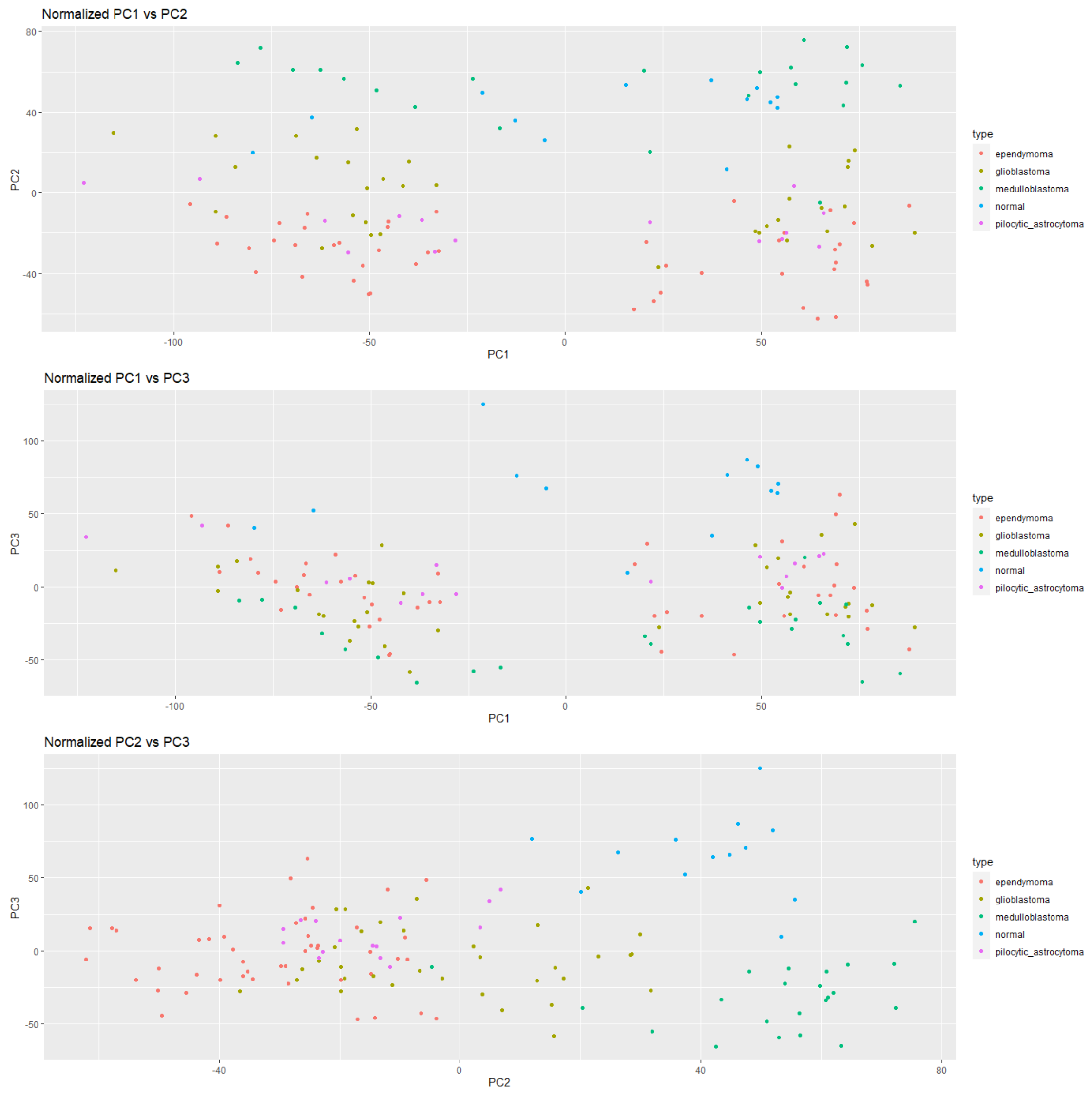
- You must use ggplot for each plot. Color according to the phenotype of each sample

```
pcs_norm$type <- my_dataframe$type
# Create a scatter plot for PC1 vs PC2
pc1_pc2_norm <- ggplot(pcs_norm, aes(x = PC1, y = PC2, color = type)) +
  geom_point() +
  labs(title = "Normalized PC1 vs PC2", x = "PC1", y = "PC2")

# Create a scatter plot for PC1 vs PC3
```

```r
pc1_pc3_norm <- ggplot(pcs_norm, aes(x = PC1, y = PC3, color = type)) +
  geom_point() +
  labs(title = "Normalized PC1 vs PC3", x = "PC1", y = "PC3")

# Create a scatter plot for PC2 vs PC3
pc2_pc3_norm <- ggplot(pcs_norm, aes(x = PC2, y = PC3, color = type)) +
  geom_point() +
  labs(title = "Normalized PC2 vs PC3", x = "PC2", y = "PC3")
```

- You must view the three plots together in a figure of size 15x15. Use grid.arrange function to place them on 3 rows and 1 column. Think about how to change the figure size in markdown.

```r
# Arrange the plots in a 3x1 grid
grid.arrange(pc1_pc2_norm, pc1_pc3_norm, pc2_pc3_norm, ncol = 1)
```



- Comment on changes in data before and after QC

- During the QC, we identify the outliers (above/below three standard deviations), missing values [NAs] in the data that may affect its quality and reliability. Next, we removed the outliers, filled the missing values using the means of genes.
- Second step was the "Normalization", which aims to remove variations between samples, ensuring that the data is comparable across all samples. It also ensures that the data follows the Gaussian distribution so we can apply regression analysis on this normalized data.
- Effect of QC on Principal Component Analysis (PCA):
    - QC and normalization had a significant impact on the results of PCA. Before QC and normalization, PCA captured some variations or biases in the data that were probably outliers, leading to misleading results.
    - After QC and normalization, PCA can more accurately capture biological variation, allowing for better interpretation of the principal components and identification of meaningful patterns or clusters in the data.

Part 2: Analysis (20 points)

Task 2.1: Regression Analysis (10 points)

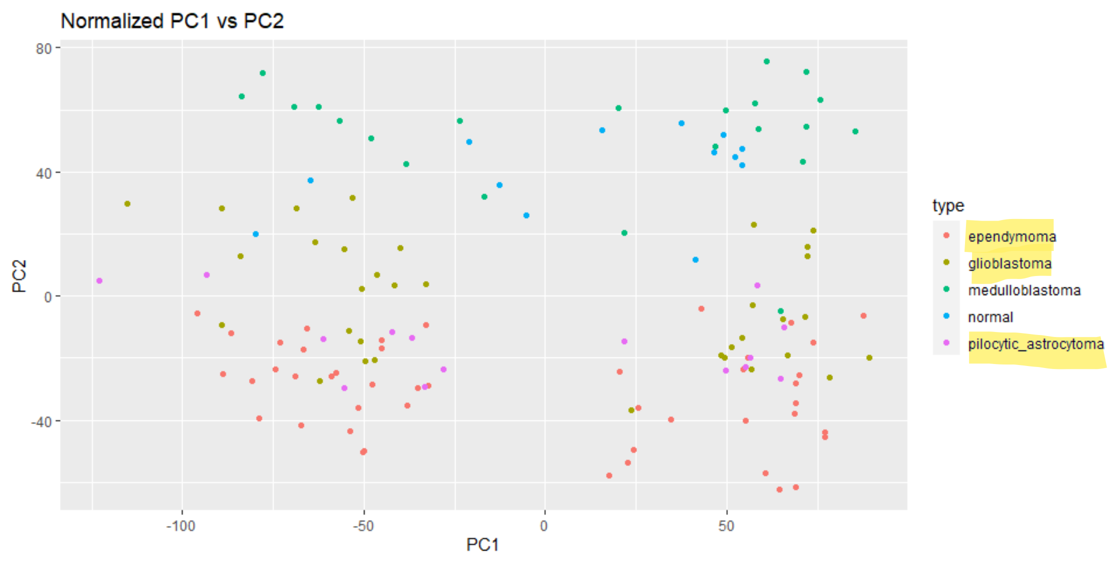
- Perform logistic regression analysis using glm function and family binomial

- You have to encode your phenotype before running the regression where positive for Tumor and negative for Normal.

```
# Encode 'Tumor' as positive (1) and 'Normal' as negative (0)
phenotype <- my_dataframe$type
encoded_phenotype <- ifelse(phenotype == "normal", 0, 1)
```

- Using only PC1 vs PC2 plots, check if there is any unexplained grouping by the PCA (after QC steps). Determine the PC that shows this variation and select it as a covariate in the model

```
pc1_pc2_norm
```



From the above plot, I notice that the there is a grouping between some samples from “pilocytic_astrocytoma”, “ependymoma” and “glioblastoma”. I think PC1 shows this variation, so I will choose it as the `selected_pc`.

```
selected_pc <- "PC1"
head(pcs_norm[selected_pc])

##             PC1
## 834 -86.64472
## 835 -45.29489
## 836 -38.18761
## 837  55.84369
## 838 -35.08612
## 839 -66.65246
```

- Run the regression model for class of tumor (positive/negative) against the gene expression: Class ~ Gene(i). There are almost 54000 genes, so specify only the 5000 genes found in this file [5000 genes]

```
# Read gene names from file
selected_genes <- readLines("D:\\Third Year Computer\\Term 2\\Bio\\Labs\\Lab 
7\\top_5000.txt")

final_selected_genes <- character()

# Initialize minimized_data dataframe
minimized_data <- data.frame(matrix(NA, nrow = nrow(normalized_data), ncol = 
0))
minimized_data$type <- encoded_phenotype
minimized_data$PC <- as.array(pcs_norm$PC1)
# Loop through each gene name and add corresponding column to minimized_data
for (gene_name in selected_genes) {
```

```r
  # print(gene_name)
  idx <- which(colnames(normalized_data) == gene_name)
  if (length(idx) > 0) {
    # print(idx)
    gene_column <- normalized_data[idx]
    minimized_data <- cbind(minimized_data, gene_column)
    final_selected_genes <- c(final_selected_genes, gene_name)
  } else {
    print(paste("Skipping gene", gene_name, "because it is not found in
normalized_data."))
  }
}
```

```
## [1] "Skipping gene X1553499_s_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553530_a_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553474_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553449_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553447_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553569_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553436_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553424_at because it is not found in
normalized_data."
## [1] "Skipping gene X1553508_at because it is not found in
normalized_data."
```

```r
# Print the first few rows of minimized_data
dim(minimized_data)
```

```
## [1]  130 4993
```

```r
pvalues_gene_df_glm <- data.frame()
p_values_list <- list()

# Loop over genes
for(gene in names(minimized_data[-(1:2)])) {
  # Specify formula
  formula <- paste("type ~", gene, " + PC")

  # Fit logistic regression model
  logistic_model <- glm(as.formula(formula), family = binomial, data =
minimized_data)

  # Extract p-value
  p_value <- summary(logistic_model)$coefficients[2, "Pr(>|z|)"]

  p_values_list[[gene]] <- p_value

  # Check significance based on threshold
  if (p_value < 0.05) {
    # Append significant associations to output data frame
     pvalues_gene_df_glm <- rbind(pvalues_gene_df_glm, data.frame(gene =
gene, p_value = p_value))

  }
}
```

- Determine all the significant Genes. (Use p-value threshold = 0.05)

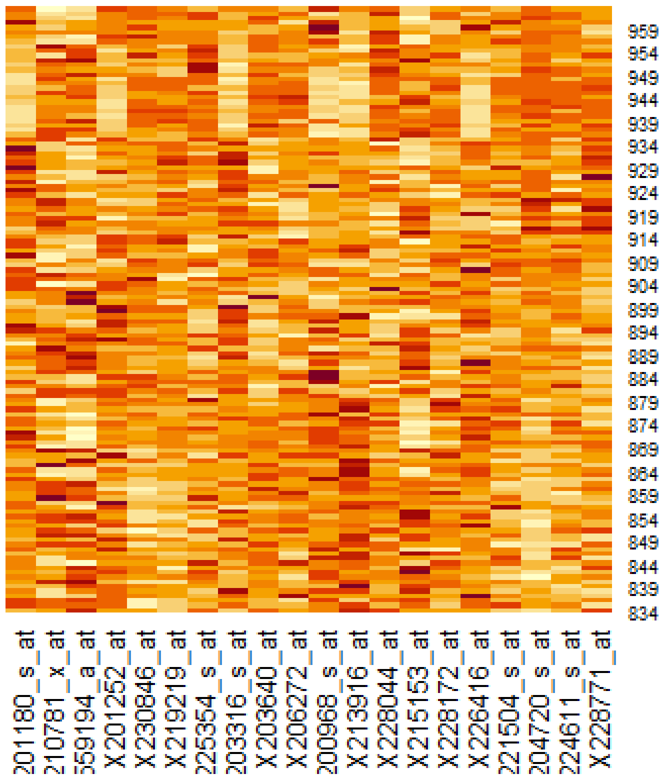
```
head(pvalues_gene_df_glm)

##             gene      p_value
## 1    X241365_at 1.578771e-04
## 2    X234991_at 2.052433e-02
## 3 X1558233_s_at 2.178969e-04
## 4  X201047_x_at 2.991113e-05
## 5  X215881_x_at 7.958936e-04
## 6    X240141_at 2.275530e-02
```

Task 2.2: Visualization (10 points)

- Plot the heatmap (using heatmap function in R or ggplot2) of the gene expression data of the top 20 significant genes

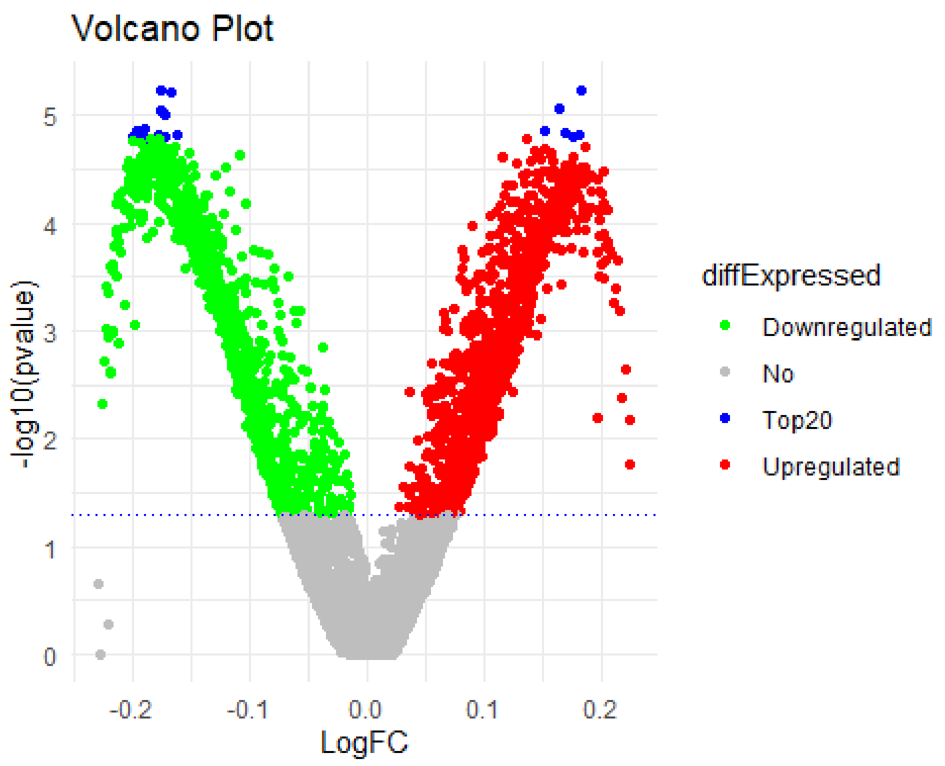
Reference: Heatmap in R

```
top_genes <- pvalues_gene_df_glm[order(pvalues_gene_df_glm$p_value), ][1:20,
"gene"]
head(top_genes)

## [1] "X201180_s_at"  "X210781_x_at"  "X1559194_a_at" "X201252_at"
## [5] "X230846_at"    "X219219_at"

top_gene_expression <- minimized_data[, top_genes]

heatmap(as.matrix(top_gene_expression), Rowv = NA, Colv = NA)
```



- Plot the volcano plot (-log10(p-value) on the y-axis and log fold change on x =-axis), and color the top 20 significant genes on that plot (using red for upregulation (cancer higher than control) and green for downregulation (opposite))

```
# Calculate the mean of cancer cells
mean_cancer_cells <- colMeans(minimized_data[minimized_data$type == 1, -
(1:2)]) # -1 means removing the type column

# Calculate the mean of normal cells
mean_normal_cells <- colMeans(minimized_data[minimized_data$type == 0, -
(1:2)])

logFC <- log2(mean_cancer_cells / mean_normal_cells)

p_values_arr <- as.numeric(unlist(p_values_list))

neg_log_p_values_arr <- -log10(p_values_arr)

volcano_data <- data.frame(
  logFC = logFC,
  neg_log_p_value = neg_log_p_values_arr
)

volcano_data$diffExpressed <- "No"

volcano_data$diffExpressed <- ifelse(volcano_data$logFC > 0 &
volcano_data$neg_log_p_value > -log10(0.05), "Upregulated",
ifelse(volcano_data$logFC < 0 & volcano_data$neg_log_p_value > -log10(0.05),
"Downregulated", "No"))
volcano_data[top_genes,]$diffExpressed <- "Top20"

# print(volcano_data[top_genes,])

ggplot(volcano_data, aes(x = logFC, y = neg_log_p_value, col=diffExpressed))
+
  scale_color_manual(values=c("Downregulated"="green", "Upregulated"="red",
"No"="gray", "Top20"="blue"))+
  geom_point() +
  geom_hline(yintercept = -log10(0.05), linetype = "dotted", color = "blue")
+
  theme_minimal()+
  xlab("LogFC") +
  ylab("-log10(pvalue)") +
  ggtitle("Volcano Plot")
```



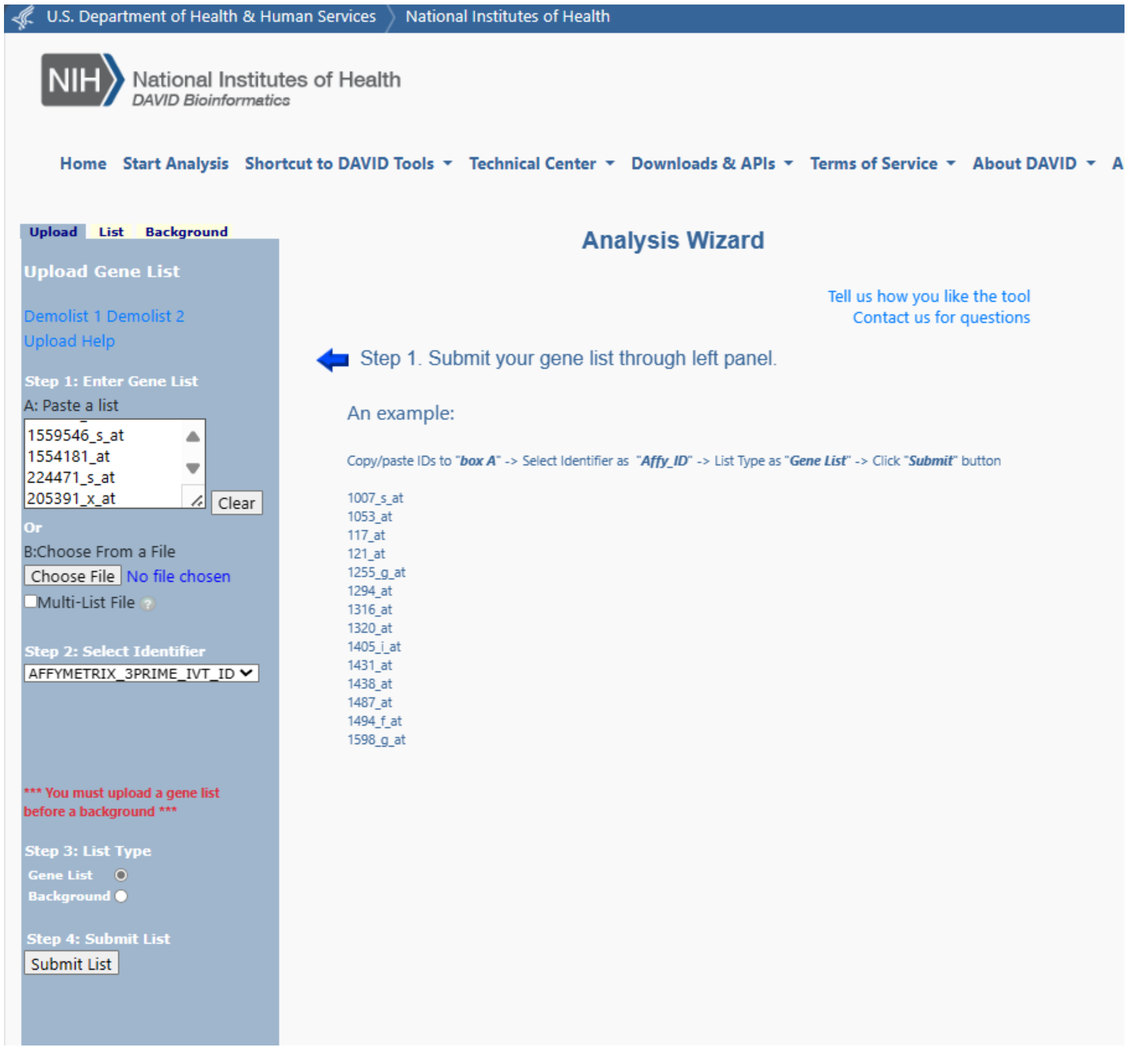
Part 4: Annotation (10 points)

- In the brain cancer dataset, all genes are in the Affymetrix format

- Use david tools to convert the top 20 significant gene names to normal gene names.

```
library(xlsx)

## Warning: package 'xlsx' was built under R version 4.3.3
```

```r
top20_gene_names_path <- "top20_gene_names.xlsx"
write.xlsx(colnames(top_gene_expression), top20_gene_names_path)

head(colnames(top_gene_expression))

## [1] "X201180_s_at"  "X210781_x_at"  "X1559194_a_at" "X201252_at"
## [5] "X230846_at"    "X219219_at"
```



Analysis Wizard

Step 1: Enter Gene List
A: Paste a list

```
1559546_s_at
1554181_at
224471_s_at
205391_x_at
```

Or
B:Choose From a File
Multi-List File

Step 2: Select Identifier
AFFYMETRIX_3PRIME_IVT_ID

*** You must upload a gene list before a background ***

Step 3: List Type
Gene List [✓]
Background

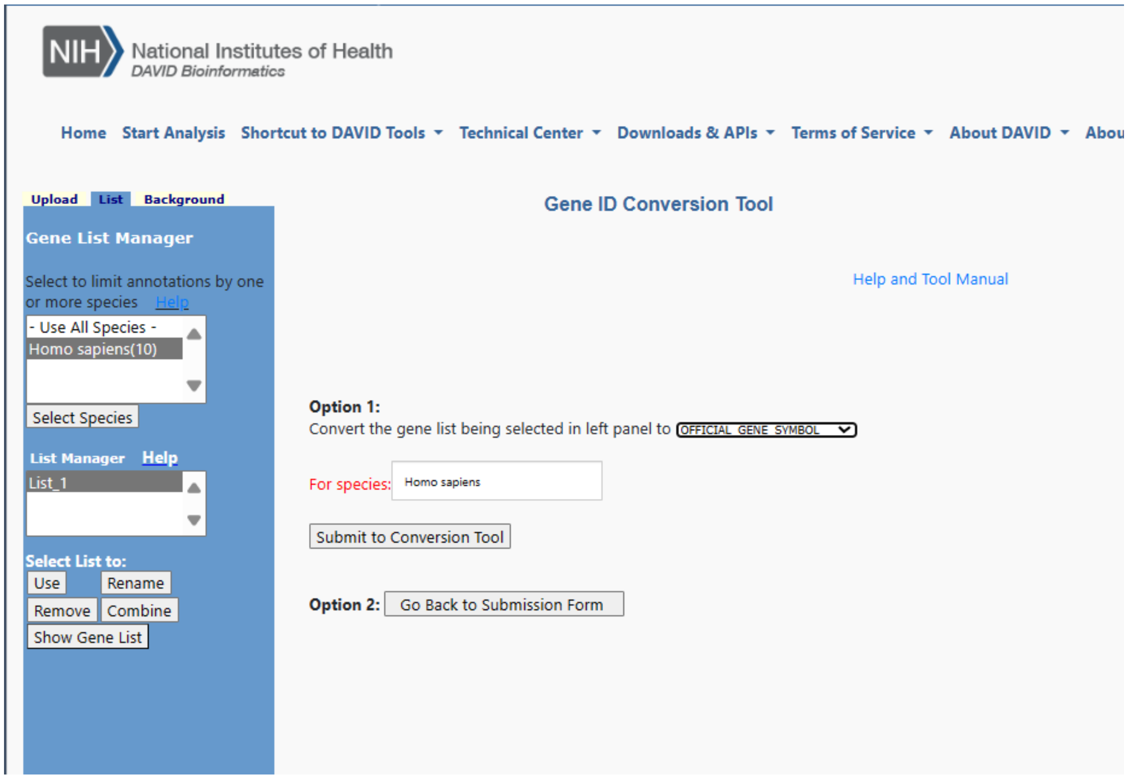
Step 4: Submit List

Step 1. Submit your gene list through left panel.

An example:

Copy/paste IDs to "*box A*" -> Select Identifier as "*Affy_ID*" -> List Type as "*Gene List*" -> Click "*Submit*" button

```
1007_s_at
1053_at
117_at
121_at
1255_g_at
1294_at
1316_at
1320_at
1405_i_at
1431_at
1438_at
1487_at
1494_f_at
1598_g_at
```



Gene ID Conversion Tool

Gene List Manager

Select to limit annotations by one or more species Help

- Use All Species -
Homo sapiens(10)

Select Species

List Manager Help

List_1

Select List to:

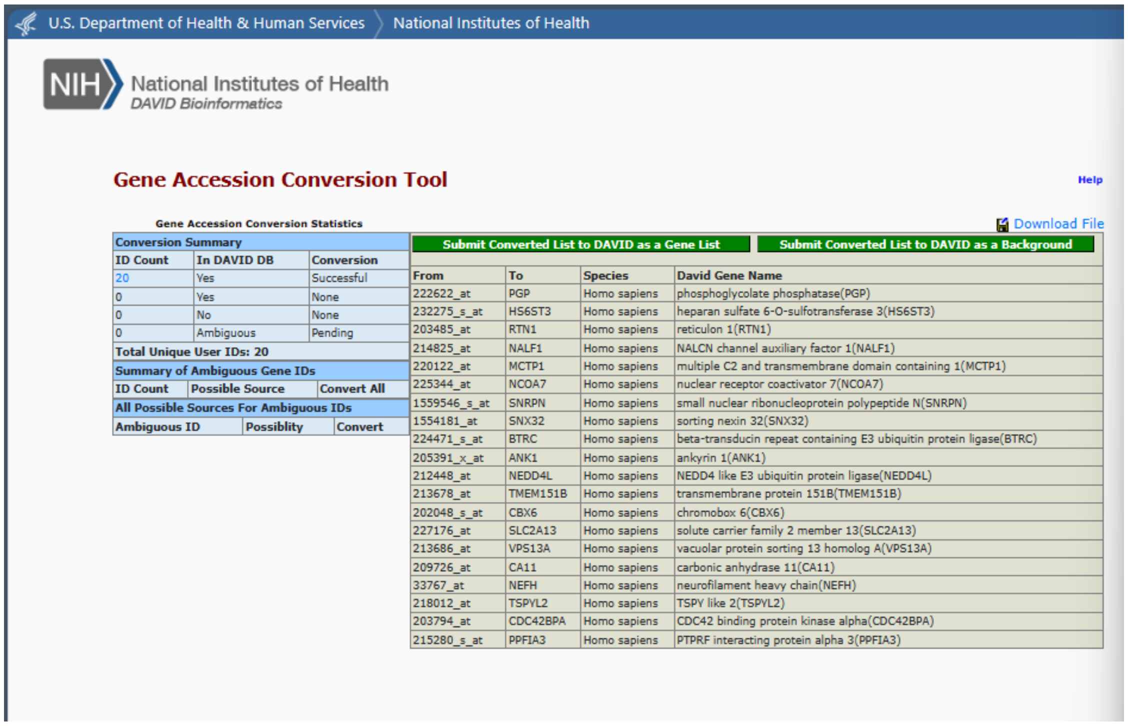
Use | Rename | Remove | Combine | Show Gene List

Option 1:
Convert the gene list being selected in left panel to OFFICIAL_GENE_SYMBOL

For species: Homo sapiens

Submit to Conversion Tool

Option 2: Go Back to Submission Form


U.S. Department of Health & Human Services
National Institutes of Health
NIH National Institutes of Health
DAVID Bioinformatics
Help
Download File
Submit Converted List to DAVID as a Gene List
Submit Converted List to DAVID as a Background

Gene Accession Conversion Tool

Gene Accession Conversion Statistics

Conversion Summary		
ID Count	In DAVID DB	Conversion
20	Yes	Successful
0	Yes	None
0	No	None
0	Ambiguous	Pending
Total Unique User IDs: 20		

Summary of Ambiguous Gene IDs		
ID Count	Possible Source	Convert All

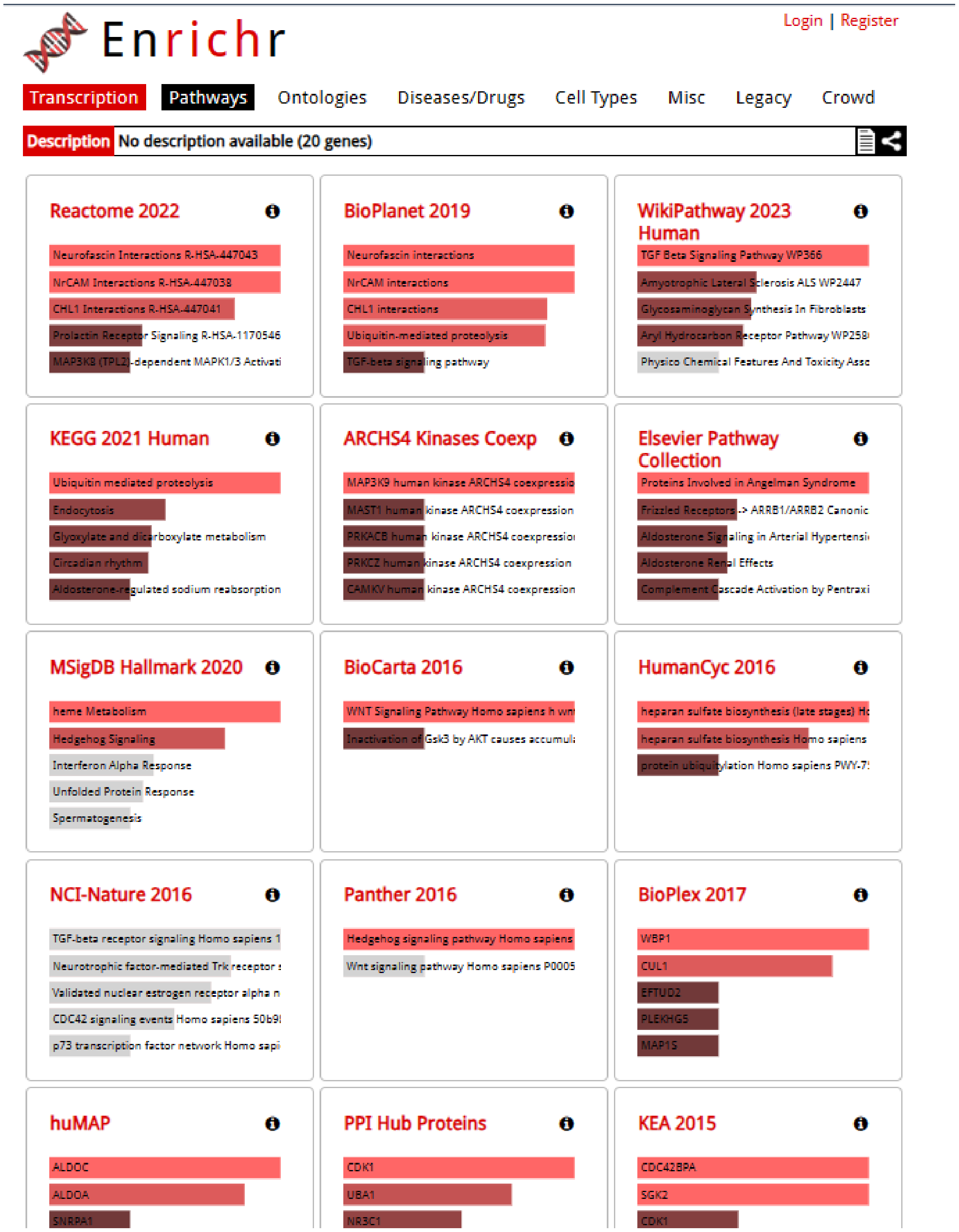
All Possible Sources For Ambiguous IDs		
Ambiguous ID	Possiblity	Convert

From	To	Species	David Gene Name
222622_at	PGP	Homo sapiens	phosphoglycolate phosphatase(PGP)
232275_s_at	HS6ST3	Homo sapiens	heparan sulfate 6-O-sulfotransferase 3(HS6ST3)
203485_at	RTN1	Homo sapiens	reticulon 1(RTN1)
214825_at	NALF1	Homo sapiens	NALCN channel auxiliary factor 1(NALF1)
220122_at	MCTP1	Homo sapiens	multiple C2 and transmembrane domain containing 1(MCTP1)
225344_at	NCOA7	Homo sapiens	nuclear receptor coactivator 7(NCOA7)
1559546_s_at	SNRPN	Homo sapiens	small nuclear ribonucleoprotein polypeptide N(SNRPN)
1554181_at	SNX32	Homo sapiens	sorting nexin 32(SNX32)
224471_s_at	BTRC	Homo sapiens	beta-transducin repeat containing E3 ubiquitin protein ligase(BTRC)
205391_x_at	ANK1	Homo sapiens	ankyrin 1(ANK1)
212448_at	NEDD4L	Homo sapiens	NEDD4 like E3 ubiquitin protein ligase(NEDD4L)
213678_at	TMEM151B	Homo sapiens	transmembrane protein 151B(TMEM151B)
202048_s_at	CBX6	Homo sapiens	chromobox 6(CBX6)
227176_at	SLC2A13	Homo sapiens	solute carrier family 2 member 13(SLC2A13)
213686_at	VPS13A	Homo sapiens	vacuolar protein sorting 13 homolog A(VPS13A)
209726_at	CA11	Homo sapiens	carbonic anhydrase 11(CA11)
33767_at	NEFH	Homo sapiens	neurofilament heavy chain(NEFH)
218012_at	TSPYL2	Homo sapiens	TSPY like 2(TSPYL2)
203794_at	CDC42BPA	Homo sapiens	CDC42 binding protein kinase alpha(CDC42BPA)
215280_s_at	PPFIA3	Homo sapiens	PTPRF interacting protein alpha 3(PPFIA3)

```r
normal_gene_names <- read.table("D:\\Third Year Computer\\Term 
2\\Bio\\Labs\\Lab 7\\david's.txt", header = TRUE, sep = "\t")

normal_gene_names <- normal_gene_names[1:2]
```

```
# Display the first few rows of the dataframe
head(normal_gene_names[2])

##        To
## 1     PGP
## 2 HS6ST3
## 3    RTN1
## 4   NALF1
## 5   MCTP1
## 6   NCOA7

gene_names_path <- "gene_names.xlsx"
write.xlsx(normal_gene_names[2], gene_names_path)
```

- Use the EnrichR tool to get enrichment results of which pathways the top 20 significant genes are present in.



No description available (20 genes)

Reactome 2022
- Neurofascin Interactions R-HSA-447043
- NrCAM Interactions R-HSA-447038
- CHL1 Interactions R-HSA-447041
- Prolactin Receptor Signaling R-HSA-1170546
- MAP3K8 (TPL2)-dependent MAPK1/3 Activati

BioPlanet 2019
- Neurofascin interactions
- NrCAM interactions
- CHL1 interactions
- Ubiquitin-mediated proteolysis
- TGF-beta signaling pathway

WikiPathway 2023 Human
- TGF Beta Signaling Pathway WP366
- Amyotrophic Lateral Sclerosis ALS WP2447
- Glycosaminoglycan Synthesis In Fibroblasts
- Aryl Hydrocarbon Receptor Pathway WP258
- Physico Chemical Features And Toxicity Assc

KEGG 2021 Human
- Ubiquitin mediated proteolysis
- Endocytosis
- Glyoxylate and dicarboxylate metabolism
- Circadian rhythm
- Aldosterone-regulated sodium reabsorption

ARCHS4 Kinases Coexp
- MAP3K9 human kinase ARCHS4 coexpressio
- MAST1 human kinase ARCHS4 coexpression
- PRKACB human kinase ARCHS4 coexpressio
- PRKCZ human kinase ARCHS4 coexpression
- CAMKV human kinase ARCHS4 coexpression

Elsevier Pathway Collection
- Proteins Involved in Angelman Syndrome
- Frizzled Receptors -> ARRB1/ARRB2 Canonic
- Aldosterone Signaling in Arterial Hypertensi
- Aldosterone Renal Effects
- Complement Cascade Activation by Pentraxi

MSigDB Hallmark 2020
- heme Metabolism
- Hedgehog Signaling
- Interferon Alpha Response
- Unfolded Protein Response
- Spermatogenesis

BioCarta 2016
- WNT Signaling Pathway Homo sapiens h wn
- Inactivation of Gsk3 by AKT causes accumul

HumanCyc 2016
- heparan sulfate biosynthesis (late stages) Hc
- heparan sulfate biosynthesis Homo sapiens
- protein ubiquitylation Homo sapiens PWY-7!

NCI-Nature 2016
- TGF-beta receptor signaling Homo sapiens 1
- Neurotrophic factor-mediated Trk receptor s
- Validated nuclear estrogen receptor alpha n
- CDC42 signaling events Homo sapiens 50b9
- p73 transcription factor network Homo sapi

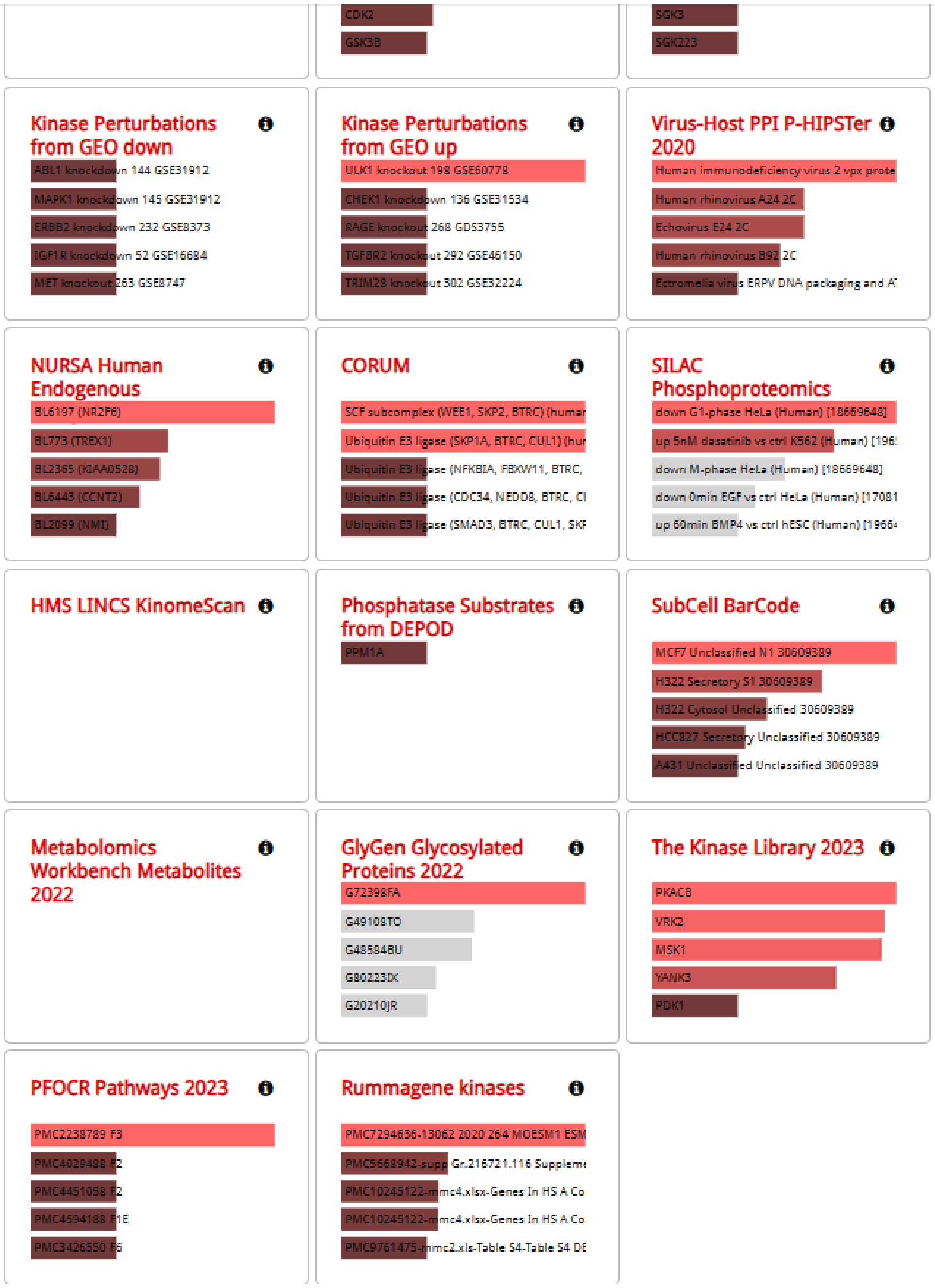
Panther 2016
- Hedgehog signaling pathway Homo sapiens
- Wnt signaling pathway Homo sapiens P0005

BioPlex 2017
- WBP1
- CUL1
- EFTUD2
- PLEKHG5
- MAP1S

huMAP
- ALDOC
- ALDOA
- SNRPA1

PPI Hub Proteins
- CDK1
- UBA1
- NR3C1

KEA 2015
- CDC42BPA
- SGK2
- CDK1



CDK2
GSK3B

SGK3
SGK223

Kinase Perturbations from GEO down

- ABL1 knockdown 144 GSE31912
- MAPK1 knockdown 145 GSE31912
- ERBB2 knockdown 232 GSE8373
- IGF1R knockdown 52 GSE16684
- MET knockout 263 GSE8747

Kinase Perturbations from GEO up

- ULK1 knockout 198 GSE60778
- CHEK1 knockdown 136 GSE31534
- RAGE knockout 268 GDS3755
- TGFBR2 knockout 292 GSE46150
- TRIM28 knockout 302 GSE32224

Virus-Host PPI P-HIPSTer 2020

- Human immunodeficiency virus 2 vpx prote
- Human rhinovirus A24 2C
- Echovirus E24 2C
- Human rhinovirus B92 2C
- Ectromelia virus ERPV DNA packaging and A

NURSA Human Endogenous

- BL6197 (NR2F6)
- BL773 (TREX1)
- BL2365 (KIAA0528)
- BL6443 (CCNT2)
- BL2099 (NMI)

CORUM

- SCF subcomplex (WEE1, SKP2, BTRC) (human
- Ubiquitin E3 ligase (SKP1A, BTRC, CUL1) (hur
- Ubiquitin E3 ligase (NFKBIA, FBXW11, BTRC,
- Ubiquitin E3 ligase (CDC34, NEDD8, BTRC, CI
- Ubiquitin E3 ligase (SMAD3, BTRC, CUL1, SKF

SILAC Phosphoproteomics

- down G1-phase HeLa (Human) [18669648]
- up 5nM dasatinib vs ctrl K562 (Human) [196
- down M-phase HeLa (Human) [18669648]
- down 0min EGF vs ctrl HeLa (Human) [17081
- up 60min BMP4 vs ctrl hESC (Human) [1966

HMS LINCS KinomeScan

Phosphatase Substrates from DEPOD

- PPM1A

SubCell BarCode

- MCF7 Unclassified N1 30609389
- H322 Secretory S1 30609389
- H322 Cytosol Unclassified 30609389
- HCC827 Secretory Unclassified 30609389
- A431 Unclassified Unclassified 30609389

Metabolomics Workbench Metabolites 2022

GlyGen Glycosylated Proteins 2022

- G72398FA
- G49108TO
- G48584BU
- G80223IX
- G20210JR

The Kinase Library 2023

- PKACB
- VRK2
- MSK1
- YANK3
- PDK1

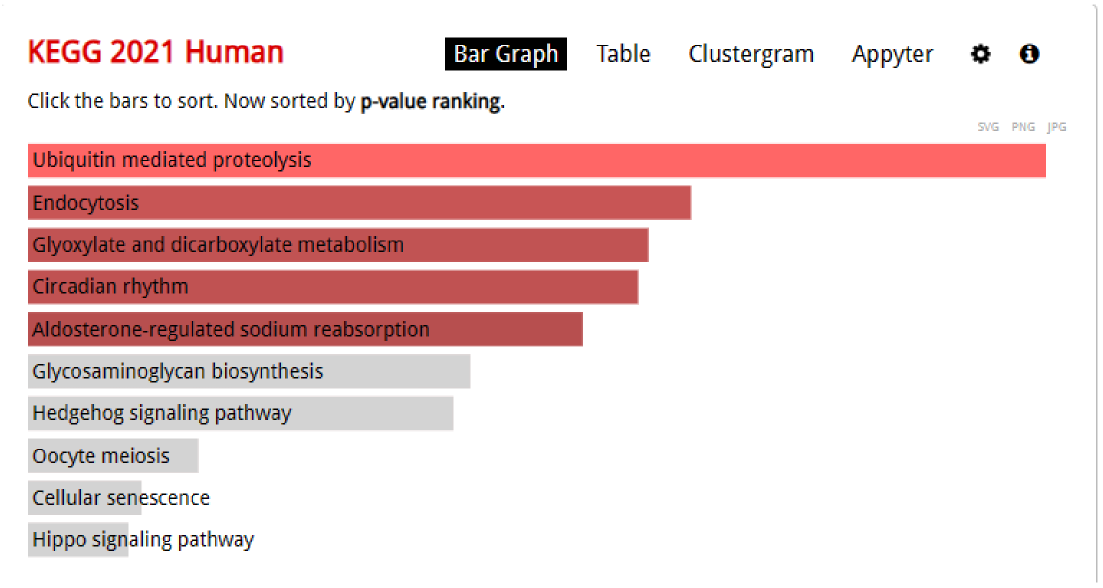
PFOCR Pathways 2023

- PMC2238789 F3
- PMC4029488 F2
- PMC4451058 F2
- PMC4594188 F1E
- PMC3426550 F6

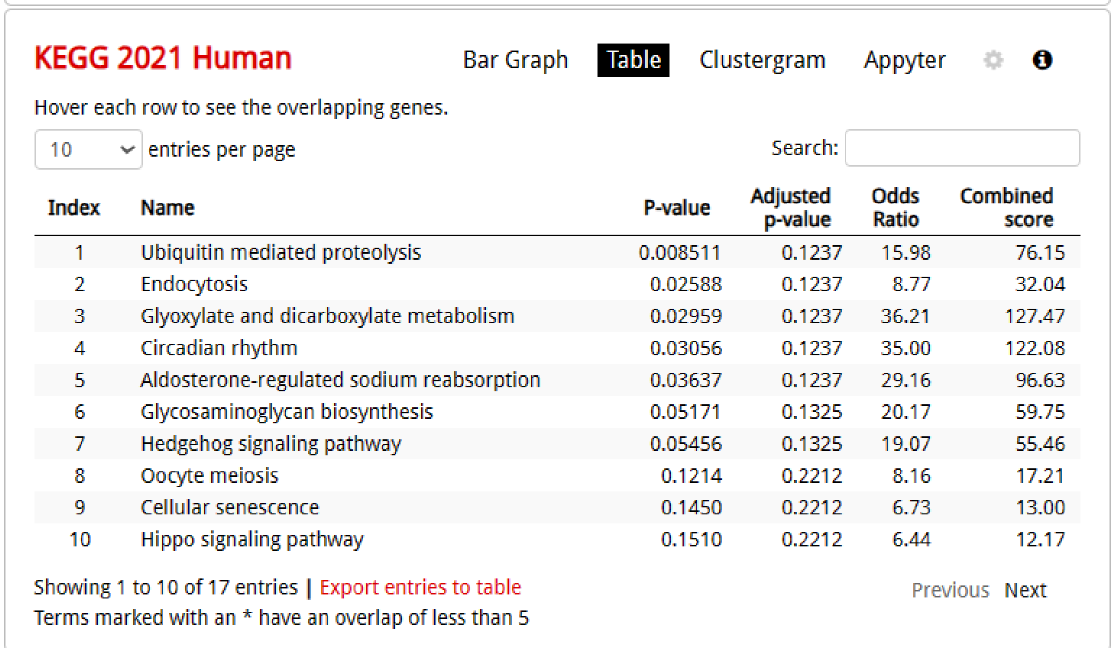
Rummagene kinases

- PMC7294636-13062 2020 264 MOESM1 ESM
- PMC5668942-supp Gr.216721.116 Suppleme
- PMC10245122-mmc4.xlsx-Genes In HS A Co
- PMC10245122-mmc4.xlsx-Genes In HS A Co
- PMC9761475-mmc2.xls-Table S4-Table S4 DE

- You are requested to extract Kegg pathways annotation in tables and graphs produced by EnrichR. Comment on the results based on the enrichment analysis p-values



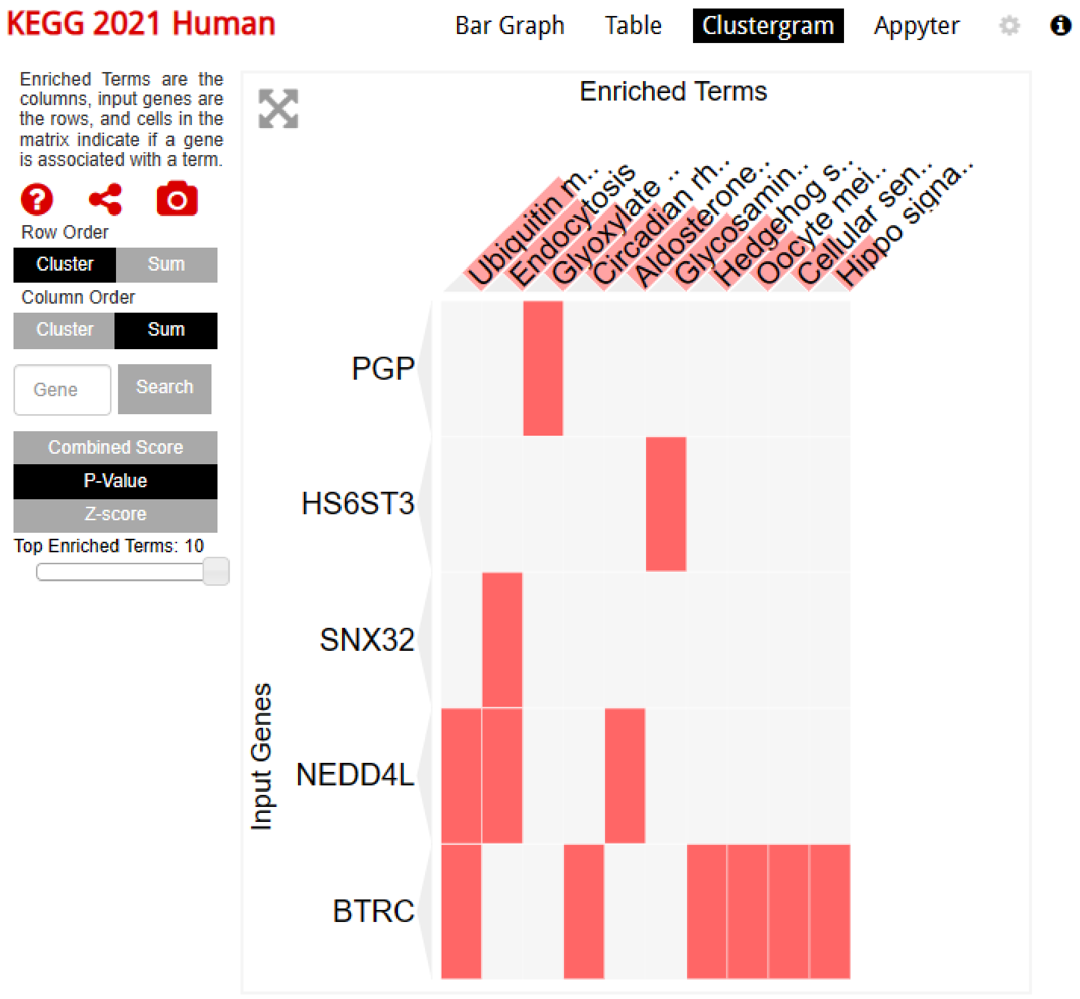
Click the bars to sort. Now sorted by **p-value ranking.**



Hover each row to see the overlapping genes.

Index	Name	P-value	Adjusted p-value	Odds Ratio	Combined score
1	Ubiquitin mediated proteolysis	0.008511	0.1237	15.98	76.15
2	Endocytosis	0.02588	0.1237	8.77	32.04
3	Glyoxylate and dicarboxylate metabolism	0.02959	0.1237	36.21	127.47
4	Circadian rhythm	0.03056	0.1237	35.00	122.08
5	Aldosterone-regulated sodium reabsorption	0.03637	0.1237	29.16	96.63
6	Glycosaminoglycan biosynthesis	0.05171	0.1325	20.17	59.75
7	Hedgehog signaling pathway	0.05456	0.1325	19.07	55.46
8	Oocyte meiosis	0.1214	0.2212	8.16	17.21
9	Cellular senescence	0.1450	0.2212	6.73	13.00
10	Hippo signaling pathway	0.1510	0.2212	6.44	12.17

Showing 1 to 10 of 17 entries | Export entries to table
Terms marked with an * have an overlap of less than 5



The analysis of KEGG pathways based on p-values reveals a gradient of statistical significance. Ubiquitin mediated proteolysis demonstrates the strongest enrichment (p-value: 0.008511), followed by endocytosis, glyoxylate and dicarboxylate metabolism, and circadian rhythm pathways.

Aldosterone-regulated sodium reabsorption, glycosaminoglycan biosynthesis, and the hedgehog signaling pathway exhibit moderate significance.

All the other pathways are less significant.

Overall, these results highlight potential biological pathways of interest, prioritizing those with lower p-values for further investigation. These results indicate the potential involvement of some of our genes in various KEGG pathways, with varying degrees of statistical significance based on the enrichment analysis p-values. The pathways with lower p-values are more strongly supported by the data, while those with higher p-values may have a weaker association.