Protein Quantification Analysis

UNR

Shalini_Karthyk

07/28/2024

Introduction

Detailed analysis of a quantified protein dataset obtained through Data Independent Acquisition (DIA). The focus is on identifying interesting findings through exploratory data analysis, hypothesis testing, and clustering. The analysis is structured to facilitate a comprehensive examination of differential protein expression, uncover significant patterns, validate biological hypotheses, and potentially identify novel biomarkers or therapeutic targets.

Libraries

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(mice)
library(DMwR2)
library(caret)
library(RANN)
library(FactoMineR)
```

1_Data Acquistition

```
# Loading dataset
peptide_data <- read.delim(
  "C:/Users/karthyk.s/OneDrive - Northeastern University/Desktop/UNR_test/TestDataSet_protein.txt"
                           , header = TRUE, sep = "\t")

# printing first 5 lines of the data
head_data <- head(peptide_data)
```

This dataset consists of several columns that provide important information about the proteins, peptides, and experimental conditions. Each row represents a unique protein group, with associated data used to analyze expression levels under various treatments and cell lines.

The brief description of each columns are given below:

PG.ProteinGroups: This column contains protein group identifiers and represents one or more proteins quantified together.

PG.Genes: The gene symbols associated with the proteins in each group.

PG.Organisms: The organism from which the proteins originate

EG.IsDecoy: A boolean indicator where "TRUE" means the entry is a decoy (artifact of the instrument).

NrOfPeptide Columns: Columns like CellLine1.vehicle.1_NrOfPeptide contain the number of peptides used to quantify each protein under different experimental conditions (vehicle and treat) for two cell lines.

ProteinQuant Columns: Columns such as CellLine1.vehicle.1_ProteinQuant contain the actual quantified intensity of proteins measured under the same experimental conditions.

X Columns: Unknown columns typically representing missing or not applicable data.

2_Data Exploration

```
# Inspecting elements of the CSV
# Dimension if the data
dimensions <- dim(peptide_data)

# Inspect
str(peptide_data)

# Columns list
cols_data <- colnames(peptide_data)

# Summary of the dataset
summary_data <- summary(peptide_data)

# na check
missing_nas <- apply(is.na(peptide_data),2,sum)

# checking class info from the dataset
class_info <- sapply(peptide_data, class)

info_table <- knitr::kable(data.frame(cbind(class_info, missing_nas)))
info_table
```

The dataset has **74447** rows and **30** columns. It has total of **180360** and the information table shows the column list with their respective class and number of the missing values for each column.

Information Table

	class_info	missing_nas
PG.ProteinGroups	character	0
PG.Genes	character	0
PG.Organisms	character	0
EG.IsDecoy	logical	0
CellLine1.vehicle.1_NrOfPeptide	integer	0
CellLine1.vehicle.2_NrOfPeptide	integer	0
CellLine1.vehicle.3_NrOfPeptide	integer	0

	class_info	missing_nas
CellLine1.treat.1_NrOfPeptide	integer	0
CellLine1.treat.2_NrOfPeptide	integer	0
CellLine1.treat.3_NrOfPeptide	integer	0
X	integer	0
CellLine2.vehicle.1_NrOfPeptide	integer	0
CellLine2.vehicle.2_NrOfPeptide	integer	0
CellLine2.vehicle.3_NrOfPeptide	integer	0
CellLine2.treat.1_NrOfPeptide	integer	0
CellLine2.treat.2_NrOfPeptide	integer	0
CellLine1.vehicle.1_ProteinQuant	numeric	12970
CellLine1.vehicle.2_ProteinQuant	numeric	12454
CellLine1.vehicle.3_ProteinQuant	numeric	12110
CellLine1.treat.1_ProteinQuant	numeric	12615
CellLine1.treat.2_ProteinQuant	numeric	12072
CellLine1.treat.3_ProteinQuant	numeric	11425
X.1	numeric	11197
CellLine2.vehicle.1_ProteinQuant	numeric	11592
CellLine2.vehicle.2_ProteinQuant	numeric	12267
CellLine2.vehicle.3_ProteinQuant	numeric	12380
CellLine2.treat.1_ProteinQuant	numeric	15113
CellLine2.treat.2_ProteinQuant	numeric	14288
CellLine2.treat.3_ProteinQuant	numeric	18308
X.2	numeric	11569

2.1 Exploratory feature Corrections

```
# Expanding the protein groups into separate rows
peptide_data_clean <- peptide_data %>%
  separate_rows(PG.ProteinGroups, sep = ";")

# Checking duplication combination
duplicated_rows <- peptide_data_clean %>%
  filter(duplicated(.) | duplicated(., fromLast = TRUE))

# Aggregation checking
aggregated_data <- peptide_data_clean %>%
  group_by(PG.ProteinGroups) %>%
  summarise(across(starts_with("CellLine"), ~ mean(.x, na.rm = TRUE)))

# Removing duplicated rows across all columns
peptide_data_clean <- peptide_data_clean %>% distinct()

# Filtering decoy proteins
peptide_data_clean <-peptide_data_clean %>%
  filter(EG.IsDecoy == FALSE)
```

The Protein groups are restructured to have protein groups in multiple columns can make analysis more organized and insightful. The duplicated (repeated rows) are removed. After checking for the duplication combinations and aggregation to check for patterns. The decoy proteins are also removed to maintain the integrity if the data.

2.2 Handling Missing values

```
# calculating percentage of Nan
missing_percentage <- t(peptide_data_clean %>%
  summarise(across(everything(), ~ mean(is.na(.)) * 100)))

# Percentage table
missing_percentage <- as.data.frame(missing_percentage) %>%
  tibble::rownames_to_column(var = "Column") %>%
  rename(Missing_Percentage = V1)

# Columns with missing values < 30%
moderate_missing_cols <- missing_percentage %>%
  filter(Missing_Percentage > 5 & Missing_Percentage < 30) %>%
  pull(Column)

# Columns with missing values > 30%
high_missing_cols <- missing_percentage %>%
  filter(Missing_Percentage >= 30) %>%
  pull(Column)

# Impute using MICE for moderate missing columns
mice_imputed_data <- suppressMessages(mice(peptide_data_clean[moderate_missing_cols],
                     m = 5, method = 'pmm', maxit = 50, seed = 123))
```

```
Warning: Number of logged events: 744
```

```
# Complete dataset
mice_imputed_data <- complete(mice_imputed_data)

# Replace the imputed columns back into the original dataset
peptide_data_clean[moderate_missing_cols] <- mice_imputed_data
peptide_data_clean <- as.data.frame(peptide_data_clean)

# Test check missing values
missing_nas <- apply(is.na(peptide_data_clean),2,sum)

# Knn imputation for high missing percentage columns
#preProcess_knn <- preProcess(peptide_data_clean[high_missing_cols], method = c("center", "scale", "knn

# Impute missing values
#predict_imputed_data <- predict(preProcess_knn, newdata = peptide_data_clean[high_missing_cols])

# Impute using MICE for high missing columns
mice_imputed_data <- suppressMessages(mice(peptide_data_clean[high_missing_cols],
                     m = 5, method = 'pmm', maxit = 50, seed = 123))

# Complete dataset
mice_imputed_data <- complete(mice_imputed_data)

# Replace the imputed columns back into the original dataset
peptide_data_clean[high_missing_cols] <- mice_imputed_data
```

```
peptide_data_clean <- as.data.frame(peptide_data_clean)

# Test check missing values
missing_nas <- apply(is.na(peptide_data_clean),2,sum)
```

The percentage of missing values in each column was calculated to inform the imputation strategy. Columns with a moderate amount of missing data (between 5% and 30%) were imputed using the Multiple Imputation by Chained Equations (MICE) method due to its robustness in handling moderate levels of missingness by generating multiple imputations and pooling the results for more accurate and reliable estimates.

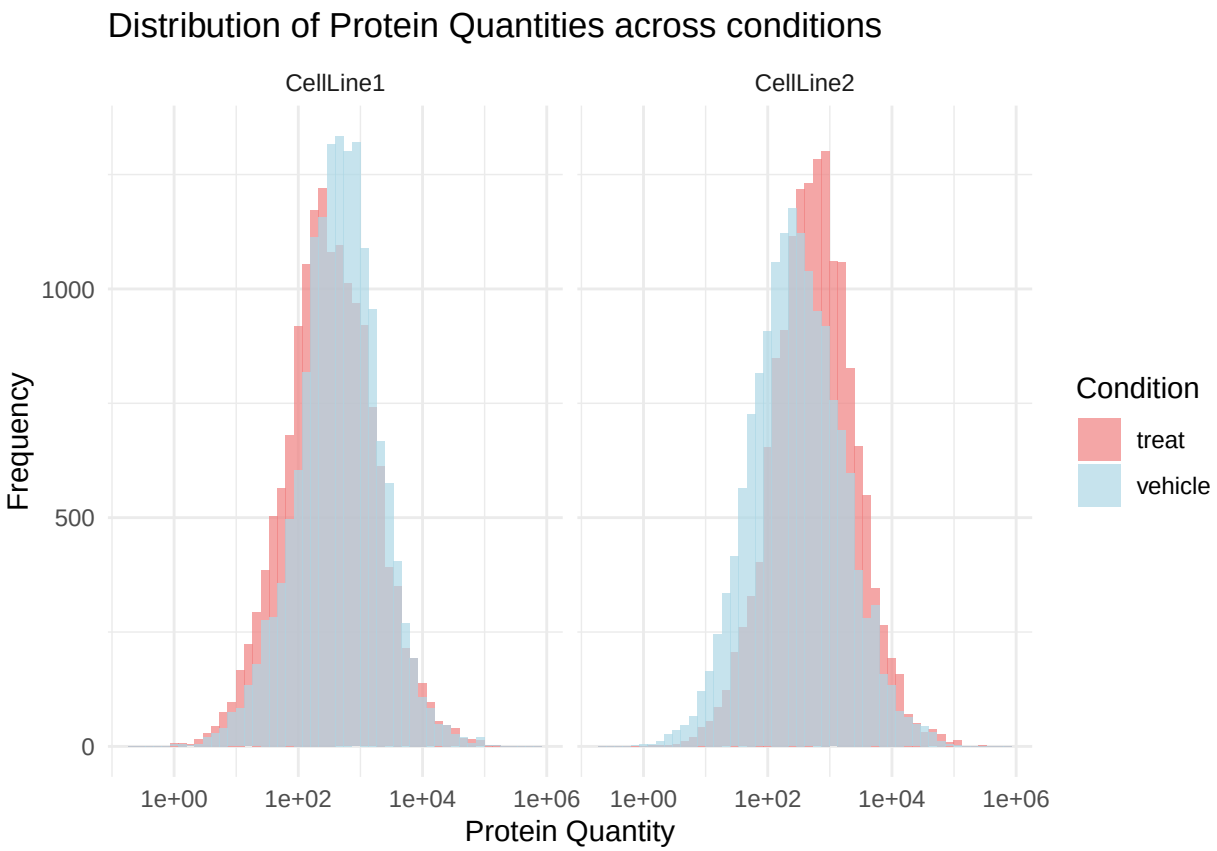
For columns with a high percentage of missing values (greater than 30%), K-Nearest Neighbors (KNN) imputation was initially considered because it can effectively utilize the similarity between observations to estimate the missing values. However, KNN imputation requires some non-missing data points to find the nearest neighbors, and since a substantial portion of the required predictors had missing values (NAs), KNN couldn't find any neighbors to perform the imputation. Consequently, MICE was used for all missing values to ensure the completeness and integrity of the dataset.

Data Distribution

```
# Plotting with pivot
protein_long <- peptide_data_clean %>%
  pivot_longer(
    cols = starts_with("CellLine"),
    names_to = c("CellLine", "Condition", "Replicate", "Column_type"),
    names_pattern =
      "(CellLine\\d+)\\.(vehicle|treat)\\.(\\d+)_(NrOfPeptide|ProteinQuant)",
    values_to = "Value"
  )
```

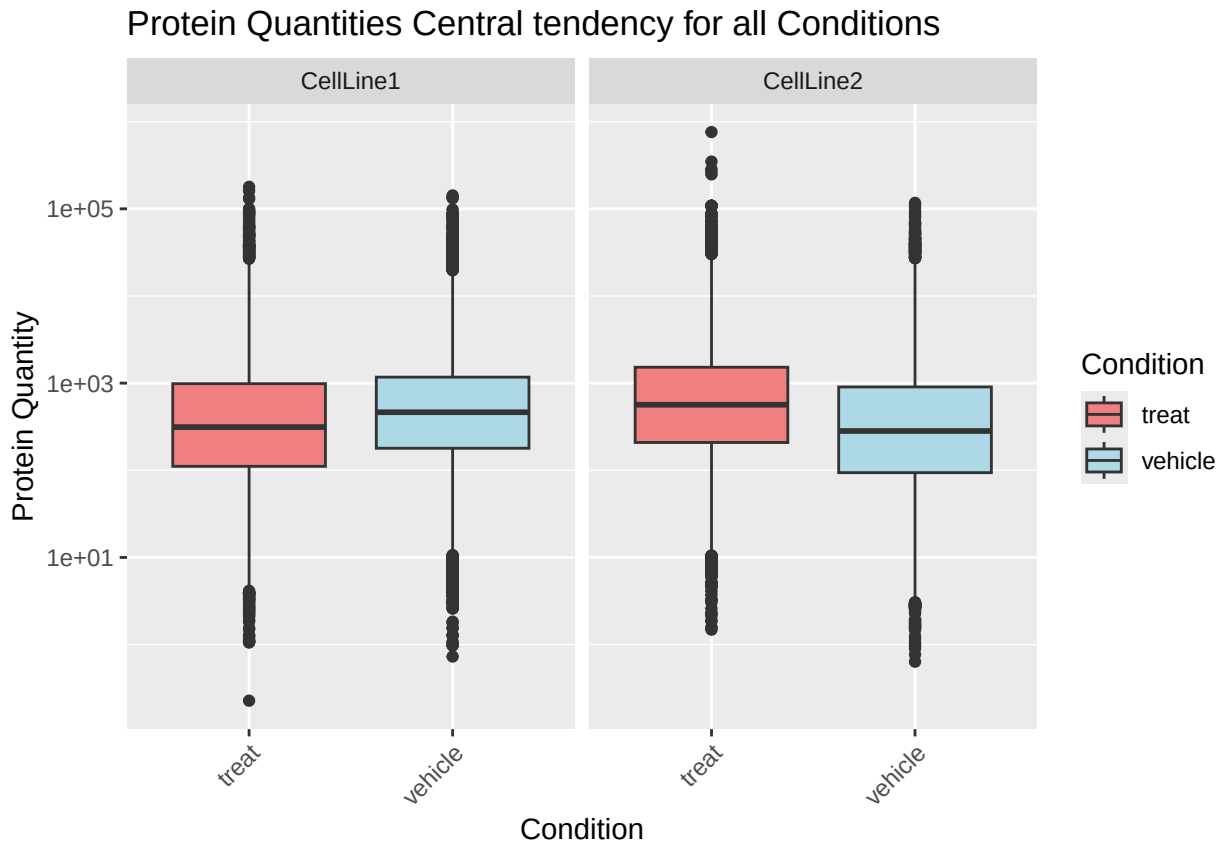
Plotting histograms and Boxplots to get the complete understanding of the data. The histograms show a roughly normal distribution of protein quantities for both cell lines and conditions. The protein quantities span several orders of magnitude, as evidenced by the use of a log10 scale on the x-axis.

```
# Histogram of ProteinQuant values
Prot_quant_hist <- ggplot(protein_long %>% filter(Column_type == "ProteinQuant"),
       aes(x = Value, fill = Condition)) +
  geom_histogram(bins = 50, alpha = 0.7, position = "identity") +
  facet_wrap(~ CellLine) +
  scale_fill_manual(values = c("treat" = "lightcoral", "vehicle" = "lightblue")) +
  labs(title = "Distribution of Protein Quantities across conditions",
       x = "Protein Quantity",
       y = "Frequency") +
  theme_minimal() + scale_x_continuous(trans = "log10")

Prot_quant_hist
```



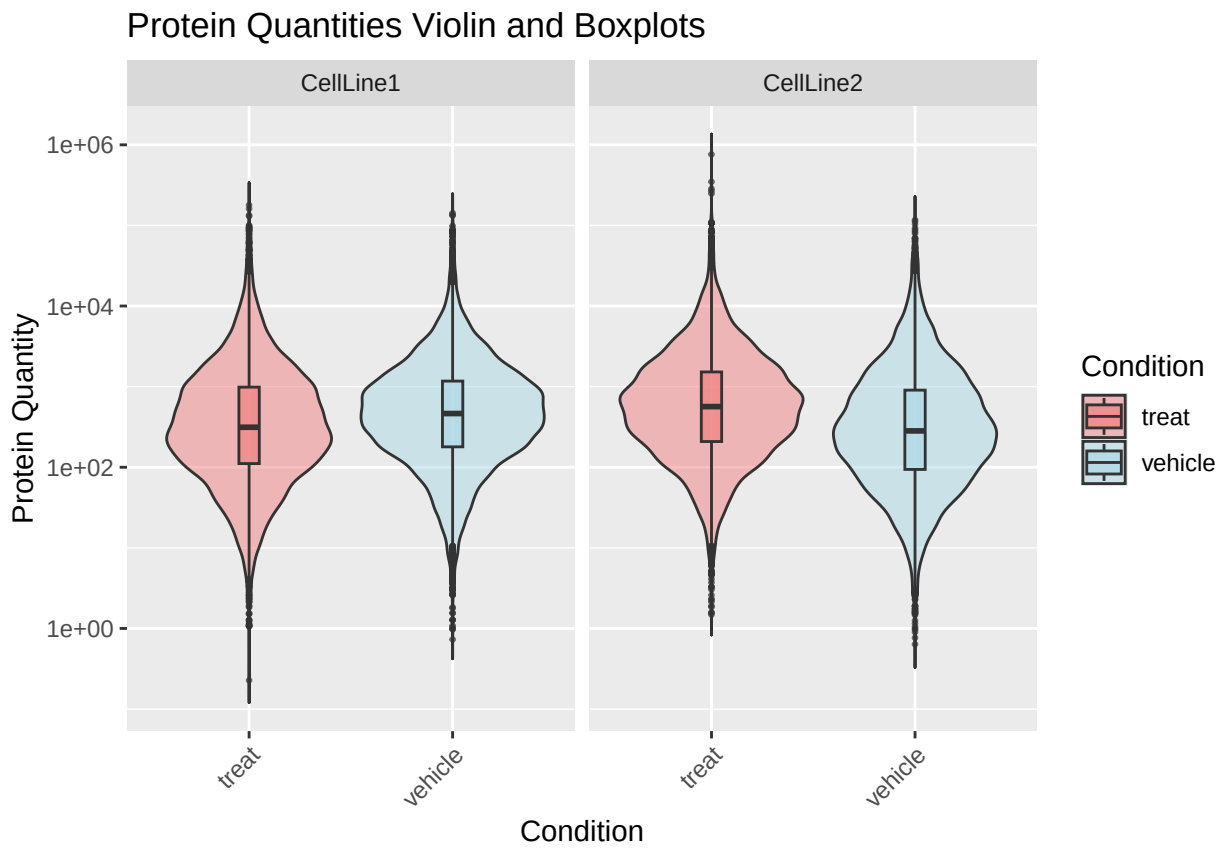
The two conditions (treat and vehicle) have very similar distribution shapes and peak frequencies in both cell lines, suggesting that treatment did not significantly alter the overall distribution of protein quantities in either cell line. The overlapping distribution shows there's subtle difference in the distribution.

```
# Boxplot of ProteinQuant values
protein_quant_box <- ggplot(protein_long %>% filter(Column_type == "ProteinQuant"),
       aes(x = Condition, y = Value, fill = Condition)) +
  geom_boxplot() +
  facet_wrap(~ CellLine) +
  scale_fill_manual(values = c("treat"
                               = "lightcoral", "vehicle" = "lightblue")) +
  scale_y_continuous(trans = "log10") +
  labs(title = "Protein Quantities Central tendency for all Conditions",
       x = "Condition", y = "Protein Quantity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

protein_quant_box
```



Based on the Boxplot both cell lines show similar median protein quantities for the treat and vehicle groups, aligning with the histogram's findings of overlapping distributions.The presence of outliers indicates that while the majority of protein quantities are similar, there are proteins with significantly different expression levels, potentially highlighting proteins of interest for further analysis.

```
# Violin
protein_quant_violin <- ggplot(protein_long %>% filter(Column_type == "ProteinQuant"),
       aes(x = Condition, y = Value, fill = Condition)) +
  geom_violin(trim = FALSE, alpha = 0.5) +
  geom_boxplot(width = 0.1, outlier.size = 0.5, alpha = 0.7) +
  facet_wrap(~ CellLine) +
  scale_fill_manual(values = c("treat"
                               = "lightcoral", "vehicle" = "lightblue")) +
  scale_y_continuous(trans = "log10") +
  labs(title = "Protein Quantities Violin and Boxplots",
       x = "Condition", y = "Protein Quantity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

protein_quant_violin
```



The Violin Plot shows and confirms the findings of the boxplot and gave a distinct and clear analysis. Thought the plots gave us overall understanding we are doing hypothesis testing to get deeper understanding.

Conduct statistical tests to determine if there are significant differences between the vehicle and treatment conditions.

3 Summary Statistics

```
# Summary statistics for ProteinQuant values
protein_summary <- protein_long %>%
  filter(Column_type == "ProteinQuant") %>%
  group_by(CellLine, Condition) %>%
  summarise(
    Mean = mean(Value, na.rm = TRUE),
    Median = median(Value, na.rm = TRUE),
    SD = sd(Value, na.rm = TRUE),
    Min = min(Value, na.rm = TRUE),
    Max = max(Value, na.rm = TRUE)
  )
```

```
`summarise()` has grouped output by 'CellLine'. You can override using the
`.groups` argument.
```

c(“CellLine1”, “CellLine1”, “CellLine2”, “CellLine2”), c(“treat”, “vehicle”, “treat”, “vehicle”), c(1411.09944835878, 1466.74099703968, 2023.38532065786, 1318.79949329568),

c(313.4238281, 463.7030029, 565.1271362, 282.3088379), c(5376.8209160148, 4979.79635279293, 9692.58764856528, 4484.27504089378), c(0.227357492, 0.729610622, 1.483298182, 0.636804104), c(178846.2188, 142443.125, 760093.6875, 117072.8438)

The summary statistics shows there is no significant difference between the treat and vehicle conditions in cell line 1 but significant change in cell line 2 based on mean values and high variability.

Hypothesis testing

```
#
peptide_ttest <- protein_long %>%
  filter(Column_type == "ProteinQuant") %>%
  group_by(CellLine, Replicate) %>%
  summarise(
    t_test_p_value = t.test(
      Value[Condition == "vehicle"],
      Value[Condition == "treat"],
      na.rm = TRUE
    )$p.value
  )
```

```
`summarise()` has grouped output by 'CellLine'. You can override using the
`.groups` argument.
```

```
peptide_ttest <- knitr::kable(data.frame(peptide_ttest))

# Wilcoxon test
wilcox_results <- protein_long %>%
  filter(Column_type == "ProteinQuant") %>%
  group_by(CellLine, Replicate) %>%
  summarise(
    wilcox_p_value = wilcox.test(
      Value[Condition == "vehicle"],
      Value[Condition == "treat"],
      na.rm = TRUE
    )$p.value
  )
```

```
`summarise()` has grouped output by 'CellLine'. You can override using the
`.groups` argument.
```

```
wilcox_results <- knitr::kable(data.frame(wilcox_results))


# Significant proteins with t test
ttest_proteins <- protein_long %>%
  filter(Column_type == "ProteinQuant") %>%
  group_by(PG.ProteinGroups) %>%
  summarise(
    p_value = wilcox.test(Value[Condition == "vehicle"],
```

```
                    Value[Condition == "treat"],
                    na.rm = TRUE)$p.value
  ) %>%
  filter(p_value < 0.05)

# Significant proteins with Wilcox test
Wilcox_proteins <- protein_long %>%
  filter(Column_type == "ProteinQuant") %>%
  group_by(PG.ProteinGroups) %>%
  summarise(
    p_value = wilcox.test(Value[Condition == "vehicle"],
                    Value[Condition == "treat"],
                    na.rm = TRUE)$p.value
  ) %>%
  filter(p_value < 0.05)
```

CellLine	Replicate	t_test_p_value
CellLine1	1	0.7488081
CellLine1	2	0.5992453
CellLine1	3	0.4142470
CellLine2	1	0.0000000
CellLine2	2	0.1209222
CellLine2	3	0.0000000

Ttest confirm no significant difference as p-values exceed 0.05 across replicates in celline 1, but Significant differences observed for most replicates, except Replicate 2. Running Wilcox test to see if there's a non - parametric or not normal distribution.

CellLine	Replicate	wilcox_p_value
CellLine1	1	0.0e+00
CellLine1	2	0.0e+00
CellLine1	3	2.4e-06
CellLine2	1	0.0e+00
CellLine2	2	0.0e+00
CellLine2	3	0.0e+00

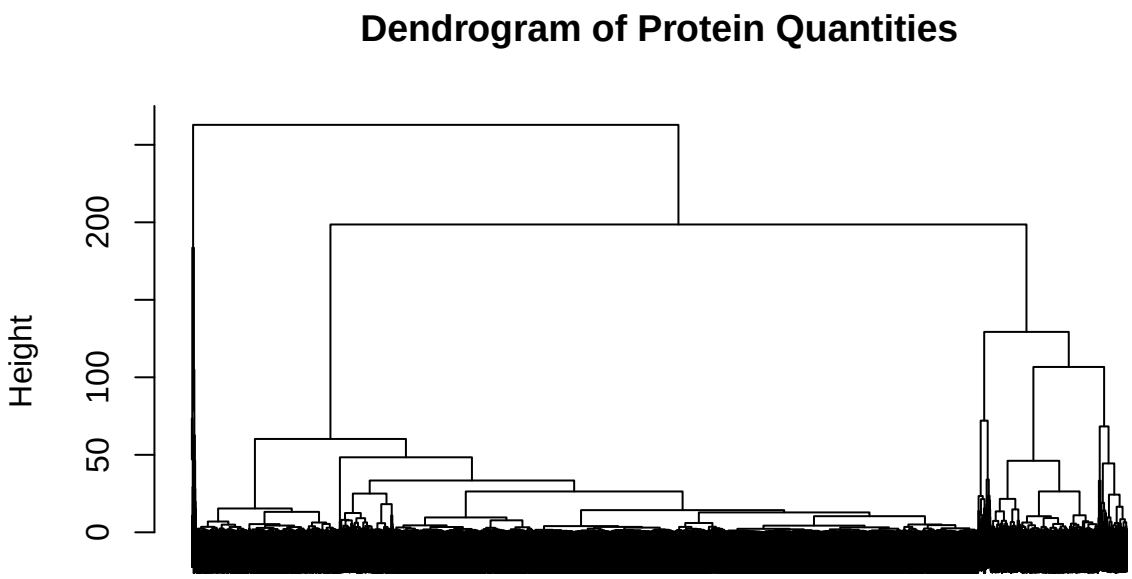
Wilcox test shows significant differences, suggesting the possibility of non-normal data distribution that the Wilcoxon test better accommodates in celline and highly significant across all replicates, further confirming the treatment effect.

Significant proteins were identified more robustly using the Wilcoxon test than the t-test, indicating its sensitivity to detect subtle differences in the dataset.All the protiens that falls under ttest and wilcox can be of proteins of interest and further investigation is needed into significant proteins which may uncover critical pathways affected by treatment.

4_Clustering

Clustering supports the hypothesis of inherent biological variability, possibly indicating differential protein regulation. We are doing Hierarchical Clustering and Kmeans to csee the cluster difference.

```
# Hierarchical Clustering
scaled_data <- scale(peptide_data_clean %>% select(starts_with("CellLine")))
dist_matrix <- dist(scaled_data, method = "euclidean")
hc <- hclust(dist_matrix, method = "ward.D2")

# dendrogram
plot(hc, labels = FALSE, main = "Dendrogram of Protein Quantities", xlab = "", sub = "")
```

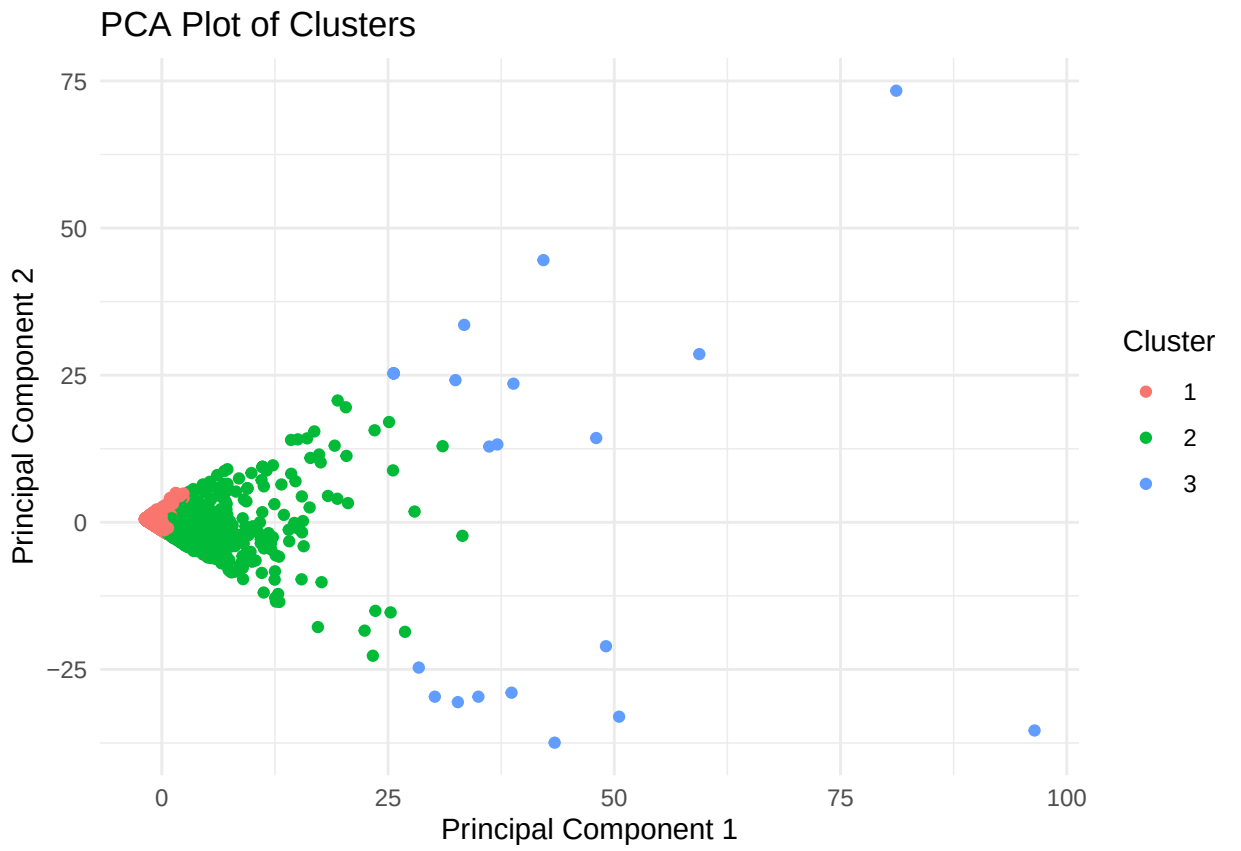



```
clusters <- cutree(hc, k = 3)
```

The clear cut point in the dendogram is identified as 3 and the number of clusters are defined based on it. The protein groups are seperated to 3 clusters the clustering analysis is followed by the kmeans for the dataset clustered visualization.

```
set.seed(123)

# Kmeans
peptide_data_clean$Cluster <- as.factor(clusters)
```

```r
# Standardizing
pre_process <- preProcess(peptide_data_clean, method = c("center", "scale"))
std_data <- as.data.frame(predict(pre_process, peptide_data_clean))
std_data <- std_data[sapply(std_data, is.numeric)]

# clustering
num_clusters <- 4
kmeans_result <- kmeans(std_data, centers = num_clusters, nstart = 25)
pca_result <- PCA(std_data, graph = FALSE)

# PCA results
pca_df <- as.data.frame(pca_result$ind$coord)
pca_df$Cluster <- peptide_data_clean$Cluster

# Plot PCA results with clusters
ggplot(pca_df, aes(x = Dim.1, y = Dim.2, color = Cluster)) +
  geom_point() +
  labs(title = "PCA Plot of Clusters", x = "Principal Component 1", y = "Principal Component 2") +
  theme_minimal()
```



With Kmeans, we can see the cluster 1 Contains proteins with very similar quantification profiles, indicating a homogeneous group. Cluster 2 Contains proteins with slightly more variability but still relatively similar profiles, indicating a moderately homogeneous group. and cluster 3 Contains proteins with high variability in their profiles, indicating a heterogeneous group that can be separated and checked to get the deeper understanding.

Conclusion

The DIA dataset has been Analyzed and can be futher understood to but adding more components to the analysis. Besed on the current analysis we can say that there are slight skew in the data between treated and vehicle and specific proteion of interest is also found for futher analysis.