# Introduction to Proteomic Data & Quality Control

## Day 2

Miguel Casanova & Dany Mukesha
2025-11-23

# Introduction to Proteomic Data & Quality Control

# Learning Objectives

By the end of Day 2, you will be able to:

- Understand the structure of proteomic data matrices
- Identify common data quality issues
- Perform initial quality control checks
- Visualize data using PCA, boxplots, and heatmaps
- Conduct exploratory data analysis (EDA)

# Module 1: Introduction to Proteomic Data

# From Mass Spectrometry to Quantified Proteins

**Proteomics Workflow:**

1. **Sample Preparation**: Protein extraction, digestion into peptides
2. **LC-MS/MS**: Liquid chromatography coupled with tandem mass spectrometry
3. **Peptide Identification**: Match spectra to peptide sequences
4. **Protein Inference**: Aggregate peptides to proteins
5. **Quantification**: Measure protein abundance

**Quantification Methods:**

- Label-free quantification (LFQ)
- Isobaric labeling (TMT, iTRAQ)
- SILAC

# Structure of Proteomic Data Matrices

**Typical structure: Proteins × Samples**

- **Rows**: Quantified proteins (or protein groups)
- **Columns**: Biological or technical samples
- **Additional data**: Sample metadata and protein annotations

```
# Load data components
protein_matrix ← read.csv("data/protein_matrix.csv",
                          row.names = 1, check.names = FALSE)
sample_metadata ← read.csv("data/sample_metadata.csv")
protein_annotations ← read.csv("data/protein_annotations.csv")

# Check dimensions
cat("Matrix dimensions:", dim(protein_matrix))
cat("Number of proteins:", nrow(protein_matrix))
cat("Number of samples:", ncol(protein_matrix))
cat("Missing values:", sum(is.na(protein_matrix)))
```

# Understanding Your Data

**Essential first steps:**

- Explore sample metadata
- Check protein intensity distributions
- Identify missing value patterns
- Detect technical/biological variation

```
print(sample_metadata)

summary_stats ← data.frame(
  Sample = colnames(protein_matrix),
  Mean = apply(protein_matrix, 2, mean, na.rm = TRUE),
  Median = apply(protein_matrix, 2, median, na.rm = TRUE),
  SD = apply(protein_matrix, 2, sd, na.rm = TRUE),
  N_Missing = apply(protein_matrix, 2, function(x) sum(is.na(x)))
)

print(summary_stats)
```

# Exercise 2.1: Explore Your Data

Given a proteomic dataset, calculate:

1. Total number of proteins quantified
2. Average number of missing values per protein
3. Which sample has the most missing values?

Try it yourself before looking at the solution!

# Exercise 2.1 Solution

```
cat("1. Total proteins:", nrow(protein_matrix), "\n")

missing_per_protein ← apply(protein_matrix, 1,
                            function(x) sum(is.na(x)))
cat("2. Average missing per protein:",
    round(mean(missing_per_protein), 2), "\n")

missing_per_sample ← apply(protein_matrix, 2,
                           function(x) sum(is.na(x)))
worst_sample ← names(which.max(missing_per_sample))
cat("3. Sample with most missing:", worst_sample,
    "with", max(missing_per_sample), "missing values\n")
```

# Module 2: Initial Quality Control

# Missing Data Analysis

**Missing data is common in proteomics:**

- Low-abundance proteins
- Stochastic sampling
- Technical artifacts

```
# Calculate missingness
missing_per_protein ← apply(protein_matrix, 1,
                            function(x) sum(is.na(x)))
missing_per_sample ← apply(protein_matrix, 2,
                           function(x) sum(is.na(x)))

# Missing data percentage
missing_pct ← mean(is.na(protein_matrix)) * 100
cat("Total missing data:", round(missing_pct, 1), "%\n")
```

# Visualizing Missing Data

```
library(reshape2)
missing_df ← melt(is.na(protein_matrix))
colnames(missing_df) ← c("Protein", "Sample", "Missing

ggplot(missing_df, aes(x = Sample, y = Protein,
                       fill = Missing)) +
  geom_tile() +
  scale_fill_manual(values =
    c("TRUE" = "red", "FALSE" = "grey90")) +
  theme_minimal() +
  theme(axis.text.y = element_blank())
```

```
# Histogram of missing values
hist(missing_per_protein, breaks = 20,
     main = "Missing Values per Protein",
     xlab = "Number of Missing Values",
     col = "steelblue")
```

# Detecting Outliers and Extreme Values

**Boxplots:**

- Identify outlier samples
- Check intensity distributions
- Detect batch effects

```
library(tidyverse)
protein_long ← protein_matrix %>%
  as.data.frame() %>%
  rownames_to_column("protein_id") %>%
  pivot_longer(-protein_id,
               names_to = "sample_id",
               values_to = "abundance") %>%
  merge(sample_metadata, by = "sample_id")

ggplot(protein_long, aes(x = sample_id,
                         y = abundance,
                         fill = condition)) +
  geom_boxplot(outlier.size = 0.5) +
  theme(axis.text.x = element_text(angle = 45, hjust =
```

# Density Plots for Distribution Comparison

```
ggplot(protein_long, aes(x = abundance,
                         color = sample_id)) +
  geom_density() +
  theme_minimal() +
  labs(title = "Protein Abundance Distributions",
       x = "Log2 Abundance", y = "Density") +
  theme(legend.position = "none")
```

**Useful for:**

- Comparing sample distributions
- Detecting shifts in intensity
- Identifying normalization needs

# Batch Effects Detection with PCA

**Batch effects** = systematic non-biological variations

**PCA** projects high-dimensional data to reveal patterns

```
pca_data ← t(na.omit(protein_matrix))
pca_result ← prcomp(pca_data, scale. = TRUE)

# Create PCA data frame
pca_df ← data.frame(
  PC1 = pca_result$x[, 1],
  PC2 = pca_result$x[, 2],
  sample_id = rownames(pca_result$x)
) %>% merge(sample_metadata, by = "sample_id")

# Variance explained
var_explained ← summary(pca_result)$importance[2, 1:2] * 100
```

# PCA Visualization

**By Batch:**

```
ggplot(pca_df, aes(x = PC1, y = PC2,
                   color = batch,
                   shape = condition)) +
  geom_point(size = 4) +
  labs(x = paste0("PC1 (", round(var_explained[1], 1),
       y = paste0("PC2 (", round(var_explained[2], 1),
```

**By Condition:**

```
ggplot(pca_df, aes(x = PC1, y = PC2,
                   color = condition,
                   shape = batch)) +
  geom_point(size = 4) +
  labs(x = paste0("PC1 (", round(var_explained[1], 1),
       y = paste0("PC2 (", round(var_explained[2], 1),
```

# Sample Correlation Analysis

**Correlation heatmaps** show global sample similarity

```
library(pheatmap)

cor_matrix ← cor(protein_matrix,
                 use = "pairwise.complete.obs")

rownames(sample_metadata) ← sample_metadata$sample_id

pheatmap(cor_matrix,
         annotation_col = sample_metadata[,
                       c("condition", "batch")],
         main = "Sample-Sample Correlation",
         color = colorRampPalette(
           c("blue", "white", "red"))(50))
```

# Exercise 2.2: Quality Control Checks

Perform QC on the provided dataset:

1. Calculate the percentage of proteins with >50% missing values
2. Identify any outlier samples (median abundance far from others)
3. Check for batch effects using PCA

# Exercise 2.2 Solution

```r
# 1. Proteins with >50% missing
missing_pct ← apply(protein_matrix, 1,
                    function(x) sum(is.na(x)) / length(x))
high_missing ← sum(missing_pct > 0.5)
cat("Proteins with >50% missing:", high_missing,
    "(", round(high_missing / nrow(protein_matrix) * 100, 1), "%)\n")

# 2. Outlier samples
sample_medians ← apply(protein_matrix, 2, median, na.rm = TRUE)
median_overall ← median(sample_medians)
mad_overall ← mad(sample_medians)
outliers ← abs(sample_medians - median_overall) > 3 * mad_overall

if (any(outliers)) {
  cat("Outlier samples:", names(sample_medians)[outliers], "\n")
} else {
  cat("No outlier samples detected\n")
}

# 3. Batch effects - check PCA plots
```

# Module 3: Exploratory Data Analysis (EDA)

# Distribution of Intensities

Histogram:

```
ggplot(protein_long, aes(x = abundance)) +
  geom_histogram(bins = 50,
                 fill = "steelblue",
                 alpha = 0.7) +
  labs(title = "Protein Abundance Distribution")
```

Violin plots:

```
ggplot(protein_long,
       aes(x = condition, y = abundance,
           fill = condition)) +
  geom_violin() +
  geom_boxplot(width = 0.1,
               fill = "white",
               outlier.size = 0.5)
```

# Hierarchical Clustering

Groups samples/proteins by similarity

```r
# Filter proteins with many missing values
complete_proteins ← rowSums(is.na(protein_matrix)) <
                    ncol(protein_matrix) * 0.3
filtered_matrix ← protein_matrix[complete_proteins, ]

# Impute missing values
for (i in 1:nrow(filtered_matrix)) {
  missing_idx ← is.na(filtered_matrix[i, ])
  if (any(missing_idx)) {
    filtered_matrix[i, missing_idx] ←
      mean(filtered_matrix[i, ], na.rm = TRUE)
  }
}

# Clustering heatmap
pheatmap(filtered_matrix,
         scale = "row",
         annotation_col = sample_metadata[,
                          c("condition", "batch")],
```

# Sample Similarity Analysis

Distance matrix:

```
sample_dist ← dist(t(filtered_matrix))
sample_dist_matrix ← as.matrix(sample_dist)

pheatmap(sample_dist_matrix,
         annotation_col = sample_metadata[,
                          c("condition", "batch")],
         main = "Sample Distance Matrix")
```

MDS plot:

```
mds_result ← cmdscale(sample_dist, k = 2)
mds_df ← data.frame(
  MDS1 = mds_result[, 1],
  MDS2 = mds_result[, 2],
  sample_id = colnames(filtered_matrix)
) %>% merge(sample_metadata, by = "sample_id")

ggplot(mds_df, aes(x = MDS1, y = MDS2,
                   color = condition,
                   shape = batch)) +
  geom_point(size = 4)
```

# Coefficient of Variation Analysis

CV = (SD / Mean) × 100

- High CV: Technical variability or dynamic proteins
- Low CV: Stable proteins

```
calculate_cv ← function(x) {
  (sd(x, na.rm = TRUE) / mean(x, na.rm = TRUE)) * 100
}

cv_by_condition ← protein_long %>%
  group_by(protein_id, condition) %>%
  summarise(cv = calculate_cv(abundance),
            .groups = "drop")

ggplot(cv_by_condition, aes(x = cv, fill = condition)) +
  geom_histogram(bins = 30, alpha = 0.7,
                 position = "identity") +
  facet_wrap(~ condition, ncol = 1)
```

# Exercise 2.3: Complete EDA

Perform a complete exploratory analysis:

1. Create a report summarizing data quality
2. Identify the top 10 most variable proteins
3. Check if samples cluster by biological condition

# Exercise 2.3 Solution

```r
# 1. Data quality report
cat("=== DATA QUALITY REPORT ===\n")
cat("Dataset:", nrow(protein_matrix),
    "proteins x", ncol(protein_matrix), "samples\n")
cat("Missing values:", sum(is.na(protein_matrix)),
    "(", round(mean(is.na(protein_matrix)) * 100, 1), "%)\n")

# 2. Top variable proteins
protein_variance ← apply(filtered_matrix, 1,
                         var, na.rm = TRUE)
top10_variable ← names(sort(protein_variance,
                            decreasing = TRUE)[1:10])
cat("Top 10 variable proteins:\n")
print(top10_variable)

# 3. Condition clustering
cat("Check PCA and clustering plots for biological grouping\n")
```

# Creating Quality Control Reports

```
generate_qc_report ← function(data_matrix, metadata) {
  report ← list()

  report$n_proteins ← nrow(data_matrix)
  report$n_samples ← ncol(data_matrix)
  report$missing_pct ← mean(is.na(data_matrix)) * 100

  report$protein_stats ← data.frame(
    N_Complete = sum(rowSums(is.na(data_matrix)) == 0),
    N_Partial = sum(rowSums(is.na(data_matrix)) > 0 &
                    rowSums(is.na(data_matrix)) < ncol(data_matrix)),
    N_Mostly_Missing = sum(rowSums(is.na(data_matrix)) >
                           ncol(data_matrix) * 0.5)
  )

  return(report)
}

qc_report ← generate_qc_report(protein_matrix, sample_metadata)
```

# QC Report Output

```
cat("═══ QUALITY CONTROL SUMMARY ═══\n\n")
cat("Total Proteins:", qc_report$n_proteins, "\n")
cat("Total Samples:", qc_report$n_samples, "\n")
cat("Missing Data:", round(qc_report$missing_pct, 2), "%\n\n")

cat("Protein Completeness:\n")
cat("  Complete (no missing):",
    qc_report$protein_stats$N_Complete, "\n")
cat("  Partial missing:",
    qc_report$protein_stats$N_Partial, "\n")
cat("  Mostly missing (>50%):",
    qc_report$protein_stats$N_Mostly_Missing, "\n")
```

# Day 2 Summary

# What We Covered Today

- ✓ Structure of proteomic data matrices
- ✓ Common data quality issues (missing values, outliers, batch effects)
- ✓ Quality control visualization techniques
- ✓ Exploratory data analysis methods
- ✓ Sample correlation and clustering

# Key Takeaways

1. **Missing data** is common in proteomics - understand patterns before imputation
2. **Batch effects** can confound biological signals - always check with PCA
3. **Quality control** should be performed before any statistical analysis
4. **Visualization** is essential for understanding your data

# Homework

1. Apply QC pipeline to a new dataset
2. Practice identifying batch effects
3. Create custom QC visualizations

```
# Prepare for Day 3
install.packages(c("preprocessCore", "matrixStats"))
BiocManager::install(c("limma", "vsn", "sva"))
```

# Additional Resources

- Proteomics Data Analysis Best Practices
- Understanding PCA in Proteomics
- Kammers et al. (2015) "Detecting Significant Changes in Protein Abundance"

# Case Study: Real Proteomic Dataset

```
# Example workflow for your own data
# 1. Load data
my_data ← read.csv("your_protein_data.csv", row.names = 1)

# 2. Load metadata
my_metadata ← read.csv("your_sample_metadata.csv")

# 3. Initial QC
qc_report ← generate_qc_report(my_data, my_metadata)

# 4. Visualizations
# - PCA for batch effects
# - Correlation heatmap
# - Missing data patterns
# - Distribution boxplots
```

Questions?

See you tomorrow for Day 3!