VIDO BIOINFO Bioinformatics and Systems Biology Lab

Canadian Bioinformatics Hub

Canadian Bioinformatics Workshops

Introduction to R Programming for Bioinformatics

Day 2- Module 3A: Introduction to Bioconductor and Genomic Data

Mohamed Helmy, PhD
Principal Scientist and Adjunct Professor
Bioinformatics and Systems Biology Lab
VIDO, University of Saskatchewan

6-7 October 2025, VIDO, Saskatoon

Learning Objectives

- Learn what Bioconductor is
- Explore genomic data structures
- Practice loading and exploring datasets
- Prepare data for downstream bioinformatics

What is Bioconductor?

- Open-source project for computational biology and bioinformatics
- Built on R
- Provides thousands of packages for:
  - Genomics
  - Transcriptomics
  - Proteomics
  - Epigenomics
  - And many other types of biological data

- First released in 2001, now with a large global community

Why Bioconductor Matters

- **Standardized data structures → reproducibility**
- **Rich ecosystem of packages maintained by domain experts**
- **Integration with public data repositories (e.g., GEO, TCGA, Ensembl)**
- **Essential for omics analysis in biomedical research**

Installing Bioconductor Packages

- **Bioconductor uses *BiocManager* for installation**

- **Packages are version-matched with R**

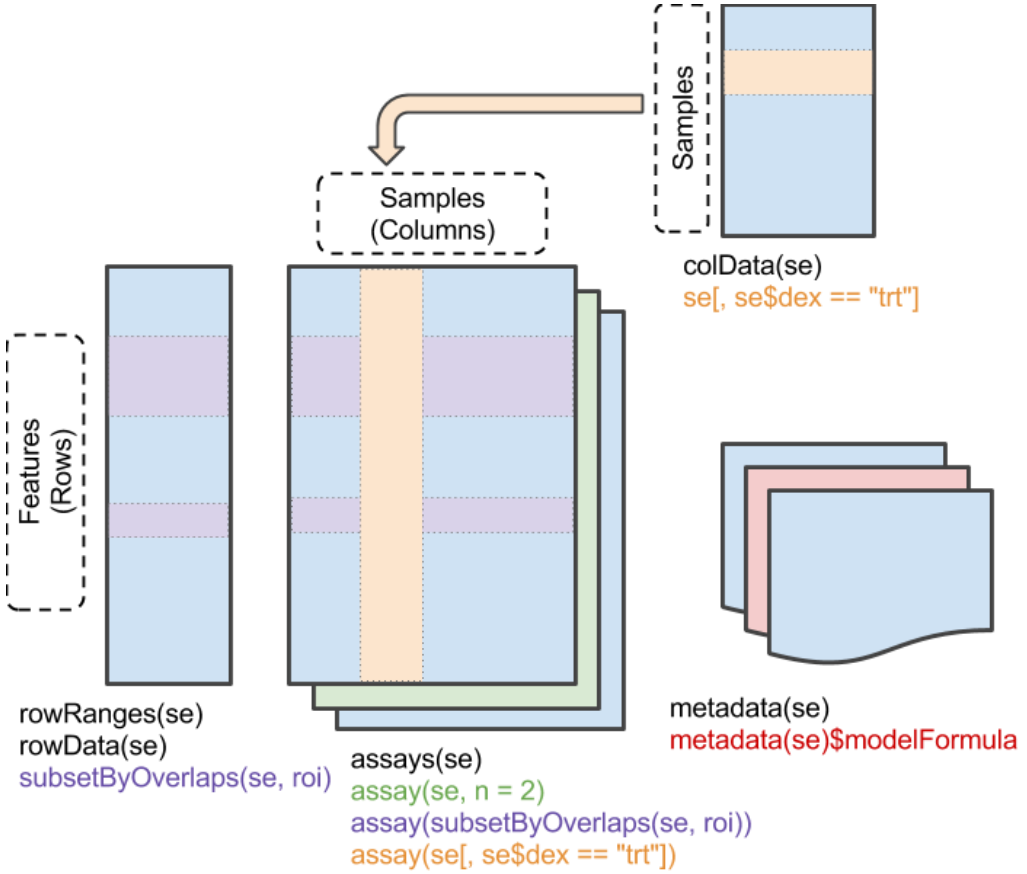
```
# Install BiocManager if not installed
install.packages("BiocManager")

# Install a package, e.g. GenomicRanges
BiocManager::install("GenomicRanges")

# Load a package
library(GenomicRanges)
```

Core Bioconductor Data Structures

- **ExpressionSet**
  - Legacy structure for microarray & expression data
  - Contains assay data + phenotypic data + feature data

- **SummarizedExperiment**
  - Modern replacement, widely used
  - Holds assay data (e.g., counts), row metadata (genes), and column metadata (samples/patients)

- **These structures ensure data + metadata stay linked**
  - Assays: numeric data (counts, intensities)
  - rowData / featureData: data about features (e.g. genes)
  - colData / phenoData: data about samples/patients
  - The slots ensure that subsetting keeps the assay–metadata linkage intact



- An **assay matrix** (features × samples)
- Linked **row metadata** (gene annotations)
- Linked **column metadata** (sample/clinical information)

Demo: SummarizedExperiment

- **Create simple SummarizedExperiment**
  - Rows = genes
  - Columns = samples
  - Metadata describes both

```
# Create simple SummarizedExperiment
counts <- matrix(rpois(20, 10), ncol=4)
colData <- DataFrame(condition=c("A","A","B","B"))
rowData <- DataFrame(gene=letters[1:5])

se <- SummarizedExperiment(assays=list(counts=counts),
                           colData=colData,
                           rowData=rowData)

se
```

```
class: SummarizedExperiment
dim: 5 4
assays(1): counts
rownames: NULL
rowData names(1): gene
colnames: NULL
colData names(1): condition
```

dim = 5 rows (genes) × 4 columns (samples)
assays(1) = the count matrix
rowData = gene names metadata
colData = sample conditions metadata

Demo2: The ALL Dataset

- ALL = Acute Lymphoblastic Leukemia expression data
- Packaged in Bioconductor (ALL package)
- 128 samples of ALL patients
- Metadata: age, gender, diagnosis date, etc.

```r
# Demo 2: ALL dataset
BiocManager::install("ALL")
library(ALL)
data(ALL)
ALL
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 128 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 01005 01010 ... LAL4 (128 total)
  varLabels: cod diagnosis ... date last seen (21 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
```

assayData: 12,625 features (genes) × 128 samples (patients).
phenoData: metadata about samples, such as: Patient code, Diagnosis date, Sex, Age, Immunophenotype (BT), Remission status, Relapse, Cytogenetics
featureData: empty here, but would contain gene annotations in some datasets.
Annotation: refers to the Affymetrix microarray chip used (HG-U95Av2).

Hands-on: Exploring ALL Metadata

Tasks for students:

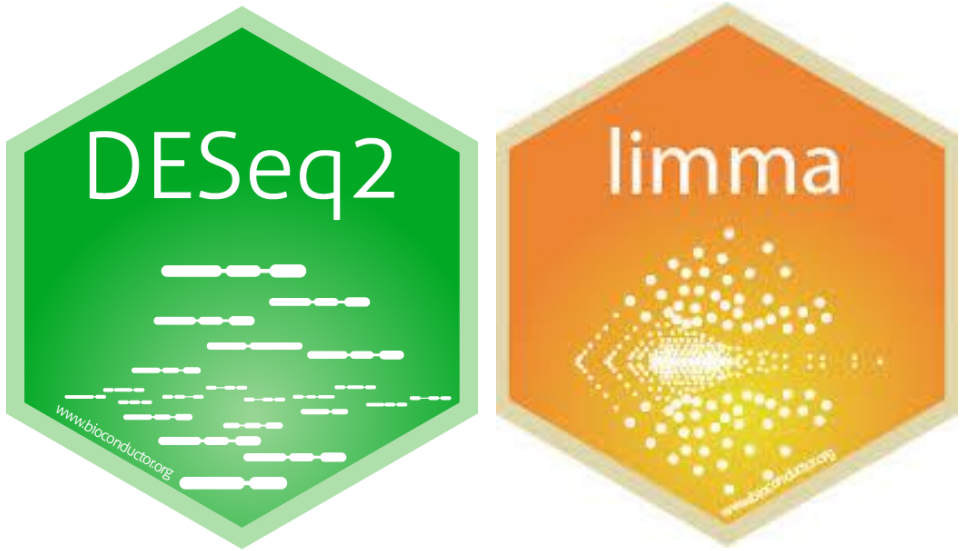
- **Inspect the dataset (ALL)**
  - Tip: type *ALL*
- **Extract column metadata:** pData(ALL)
  - Tip: use *pData()*
- **Count and plot how many patients by gender**
  - Tip: use *table()* on the meta data you got from the last step
- **Calculate average age at diagnosis**
  - Tip: use *mean()* and remove NA

```r
# Extract and preview sample (patient) metadata
meta <- pData(ALL)
head(meta)   # first 6 rows

# Gender distribution
table(meta$sex)

# Mean age (ignoring missing values)
amean(meta$age, na.rm = TRUE)
```

Preprocessing Data

Common tasks in genomic analysis:

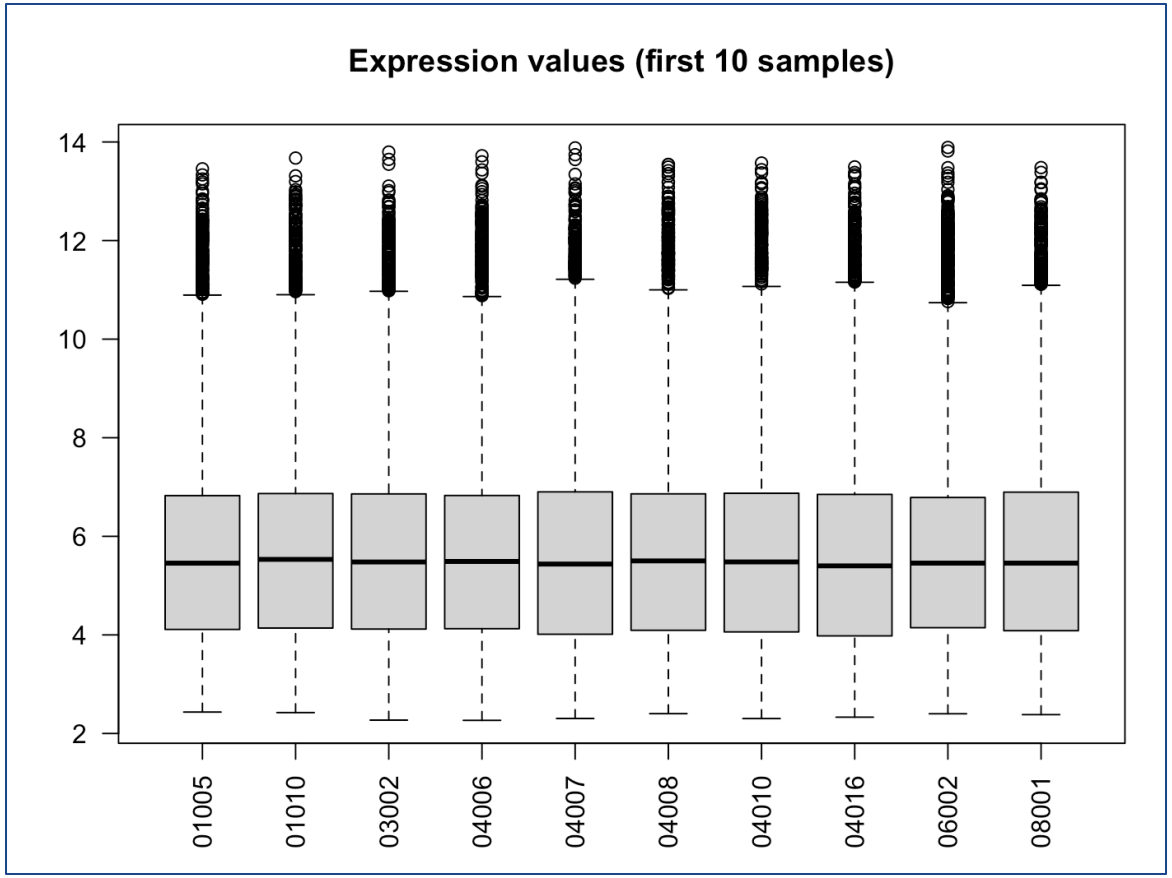
- **Filtering** low-quality samples or features
- **Normalization** of expression values
- **Transformation** (e.g., log2)
- **Batch correction** for technical artifacts



Many Bioconductor packages provide pipelines (e.g., limma, DESeq2)

Visualization in Bioconductor

- **Visual checks are essential before analysis**
- **Common plots:**
  - Boxplots for sample distributions
  - Heatmaps for expression matrices
  - PCA plots for sample clustering
- **Example (Boxplot of ALL expression data):**




```
# Visualization in Bioconductor
boxplot(exprs(ALL)[,1:10], las=2, main="Expression values (first 10 samples)")
```

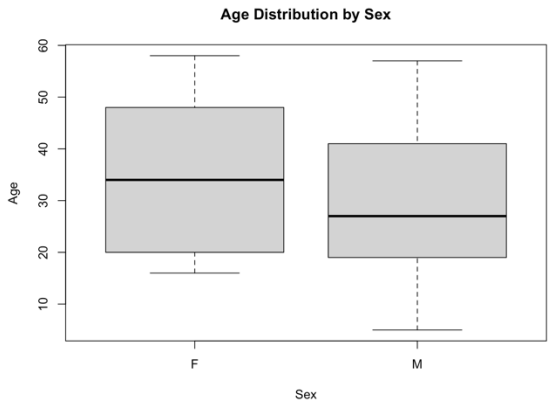
Hands-on Exercise

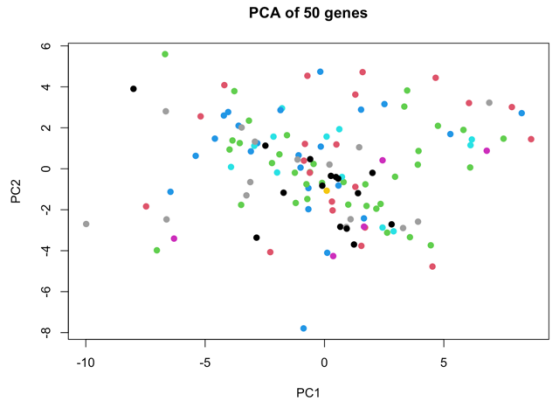
Tasks for Students:

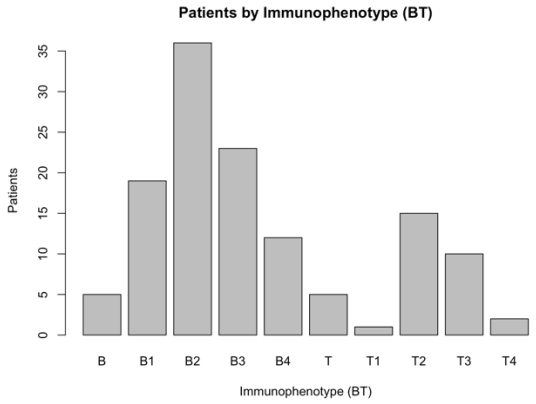
- **Subset the dataset: Select only patients younger than 20 years old.**
  - Tip: use *dim()*
- **Summarize by group: Plot how many patients are in each Immunophenotype group (BT).**
  - Tip: use *barplot(), table()*
- **Create a PCA plot of expression data for the first 50 genes across all patients.**
  - Tip: use *prcomp(), plot()*
- **Visualize gender differences: Create a boxplot of patient age split by sex.**
  - Tip: use *boxplot()*
- **(Challenge) Filter out patients with missing age values and re-run the PCA.**
  - Tip: explor the use of ! and *is.na()*

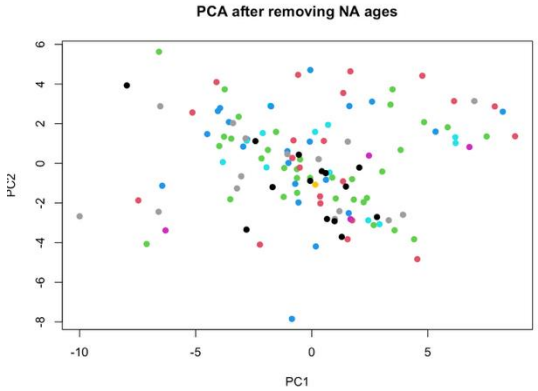
Hands-on Exercise

```r
# Subset patients < 20
young_patients <- ALL[, pData(ALL)$age < 20]
dim(young_patients)

# Count patients by Immunophenotype (BT)
table(pData(ALL)$BT)

# PCA on first 50 genes
expr <- exprs(ALL)[1:50, ]
pca <- prcomp(t(expr), scale. = TRUE)
plot(pca$x[,1:2], col = as.factor(pData(ALL)$BT),
     pch=19, main="PCA of 50 genes")


#Boxplot of Age by Sex
boxplot(age ~ sex, data = pData(ALL),
        main="Age Distribution by Sex", xlab="Sex", ylab="Age")

# Challenge (Filter missing age & re-run PCA)
ALL_clean <- ALL[, !is.na(pData(ALL)$age)]
expr_clean <- exprs(ALL_clean)[1:50, ]
pca_clean <- prcomp(t(expr_clean), scale. = TRUE)
plot(pca_clean$x[,1:2], col = as.factor(pData(ALL_clean)$BT),
     pch=19, main="PCA after removing NA ages")
```









THANK YOU

VIDO

VACCINE AND INFECTIOUS DISEASE ORGANIZATION

VIDO.ORG

UNIVERSITY OF SASKATCHEWAN